IMM / Informatics and Mathematical Modeling
Master thesis

Business Modeling

Alla Morozova

Kgs. Lyngby 2003

DTU

PREFACE

The author of this project would like to thank the master thesis project supervisor Mr.Tom Østerby, the docent of Institute of Informatics and Mathematical Modeling at DTU, for his great support during the project work and particularly for his help with business enterprise and enterprise solution terminology and ontology work, and Mr.Lars Hammer, the architect of Microsoft Business Solutions, who provided with materials about the REA model and Navision Jamaica system and brought practical and innovative spirit into discussions of the project.

Alla Alexandrovna Morozova, 30 June 2003

CONTENT

1

1 Introduction

This thesis is focused on the topics of enterprise modeling and enterprise solution modeling. The following is a short abstract of the work done and the documentation of the work.

Abstract

This thesis covers the following aspects:

1. The E/R notation and the REA model notation in UML for accounting and a retail shop enterprise,
2. Analysis of enterprises in general and enterprise solution,
3. Construction of enterprise solutions using Microsoft architecture and Navision Jamaica,
4. In the end some suggestions are presented for improving modeling enterprises and enterprise solutions.

Rationale

Present-day business life requires software applications to perform management solutions for enterprises. These software applications are named *enterprise solutions*. Enterprise solution improves an enterprise's infrastructure and helps to run a business.

Rapid development of information technologies creates a shift from software engineering to modeling engineering. Now the process of analysis and modeling of the enterprise domain plays a key role in construction of enterprise solutions.

This thesis examines some recent techniques for business modeling and strategies for constructing enterprise solutions. The goal of this project is to indicate possibilities for enterprise modeling. Subjects of conceptual modeling and ontologies are emphasized as having big potential for enterprise modeling and future enterprise solutions.

Report structure

The report is structured as follows:

- Chapter 1 provides an introduction to the report along with brief description of the content;
- Chapter 2 examines the E/R notation and REA notation for enterprise modeling. A REA model of a retail shop enterprise is taken as a study example. Possibilities to represent REA model in UML notation are suggested;
- Chapter 3 explains artifacts of business enterprise and enterprise solution. General aspects of enterprise functioning are described in order to establish a base for modeling an enterprise solution. The chapter defines the role of enterprise solution in enterprise functioning. The chapter shortly covers requirements for

enterprise solution and its architecture. The standard IEEE Std. 1471-2000 is discussed and proposed to be used as a conceptual framework for enterprise solution architecture;

- Chapter 4 describes the Microsoft enterprise strategy. The chapter gives a description of Microsoft enterprise solution architecture as a key aspect in enterprise strategy. The Microsoft enterprise solution architecture is compared with the standard IEEE Std. 1471-2000. The chapter briefly introduces the Microsoft .Net platform as the implementation of the Microsoft enterprise strategy;
- Chapter 5 contains a description of Navision Jamaica model. The conceptual model of Jamaica is presented. The strategy of Jamaica is provided in comparison with Microsoft enterprise solution strategy and the ideas given in the standard IEEE Std. 1471-2000. Finally the possibilities for future Jamaica development are suggested;
- Chapter 6 considers future enterprise solution in respect with business modeling and covers UML notation, conceptual modeling and business ontology supporting enterprise solution. UML notation is suggested as a language suitable for modeling enterprise solution. Conceptual modeling is described in regards to enterprise conceptual modeling and application modeling; the conceptual model of an enterprise is suggested. The chapter considers the subject of ontology and proposes methods to construct a business ontology;
- Chapter 7 gives a conclusion for the report and proposes directions for future work;
- Appendix A lists a bibliography for the report;
- Appendix B gives a term dictionary for the report. The term dictionary is structured into the following groups: REA model, Double-entry accounting, Business enterprise, Enterprise solution strategy, Microsoft .Net, Navision Jamaica modeling concepts;
- Appendix C represents the terms and definitions developed in cooperation with Tom Østerby. The terms and definitions are the structured refinement of some of the terms given in Appendix B. Appendix C is structured as follows: Business terms, Navision Jamaica terms. The terms and definitions can be used for enterprise conceptual modeling and business ontology;
- Appendix D contains the E/R diagram of a retail shop enterprise;
- Appendix E encloses UML diagrams: class diagrams and activity diagrams for a retail shop enterprise described in chapter 2; class diagrams for the Jamaica modeling concepts represented in chapter 5;
- Appendix F includes Sowa’s diamond introduced in chapter 6.

3 Enterprise modeling

Enterprise modeling also named as b*usiness modeling* is a process of describing how a business is organized and run using formal notations. Enterprise modeling is required for both business purposes and constructing information systems to support an enterprise:

- engineering and reengineering of business processes and organization,
- certification of an enterprise,
- enterprise information system development,
- development of a software system as a part of an enterprise information system.

When developing an information system or a software system, the enterprise modeling is used for understanding an application domain and analyzing requirements for the system.

There are a number of modeling techniques to describe an enterprise domain. Chapter 2 covers one of the techniques called *Resource-Event-Agent (REA) diagram notation*. The REA notation is based on *Entity-Relationship (E/R) diagram notation*, which is shortly described in the chapter. Possibilities to represent REA model in UML notation are also examined.

Chapter 2.1 gives an introduction to the E/R notation and modeling. The chapter aims to provide a basis for understanding REA notation. The chapter covers an extension of E/R model *Object Role Modeling* (ORM) methodology as a tool for conceptual modeling.

In chapter 2.2, REA diagram notation is introduced. The REA model of a Retail shop enterprise is taken as a study example. The example is not a real case of a particular retail store; it represents a simplification of a retail enterprise capturing its basic business activities.

Chapter 2.3 suggests possibilities to represent REA model in UML notation.

In chapter 2.4, some parts of the REA model of a Retail Enterprise introduced in chapter 2.2 are presented in UML notation. The chapter demonstrates on the example of a Retail shop enterprise how REA notation can be represented in UML for the purpose of domain analysis.

3.1 E/R notation and modeling

Entity-Relationships (E/R) diagrams notation was introduced in 1976 by Peter Pin-Shan Chen. The E/R notation was created for the purpose of database design. The model is very popular and has been widely used in database modeling.

The E/R modeling has been considered as a stage in a database design preceding the relational database modeling. The E/R model gives data structures representation:

- what information have to be stored,
- the relationships between informational elements and
- constraints on the data structure.

The E/R model has to be converted into relational schemas for the stage of database implementation.

The advantage of modeling databases in E/R notations is so that the model is more flexible and expressive comparing with the relational model. The relational model has only one concept – the relation. The E/R model has several complementary concepts.

There are three levels of abstraction of E/R notation. These levels are steps in the E/R modeling process:

- Conceptual level includes the essential data elements and their connections;
- Information level presents semantics of data structure;
- Data level specifies constraints for data structure.

The elements of the E/R notation are explained in section 2.1.1.

Section 2.1.2 describes the levels of E/R notations.

Section 2.1.3 contains the description of an extension of the E/R model – the Object-Role Model.

3.1.1 E/R notation

The E/R notation is based on the set theory and the relational theory.

The E/R notation contains graphical representation of elements of three basic types:

1. Entity sets,
2. Attribute sets,
3. Relationship sets.

An *entity* represents an object from the real world. A collection of similar entities forms an *entity set*. They are *members* of the entity set. An entity set is presented as a rectangle on an E/R diagram.

An entity set has associated *attribute sets* which are properties of the entities of that set. Attribute sets have their names and are shown by ovals on an E/R diagram. Attribute sets can have values of the types:

- atomic,
- structured, as a tuple formed with a fixed number of atomic components,
- a set of values of one type: atomic or structured.

For each entity set E, a finite set of entities (e_1, e_2, ..., e_n) forms an instance of this entity set. Each of these entities has particular values for each associated attribute.

A *relationship set* is a connection between two or more entity sets. It is presented as a diamond on an E/R diagram. A *relationship* is an instance of a relationship set. It is a connection between two or more entities.

A relationship set R connects n entity sets E_1, E_2, ..., E_n. The relationship set R has an instance that consists of a finite set of lists (e_1, e_2, ..., e_n), where each entity e_i belongs to a current instance of an entity set E_i. Each of these lists of n entities is connected by a relationship set R. This set of lists is a relationship set for the current instance of R.

An *E/R diagram* is a graph representing elements as entity sets, attributes and relationship sets. The elements are nodes of the graph. Edges connect an entity set to its attributes and connect a relationship set to its entity sets.

A relationship set can connect any number of entity sets. A binary relationship set connects two entity sets. A binary relationship set can connect any member of one of its entity sets to any number of members of the other entity set. There are restrictions on the multiplicity of a relationship set *R* between entity sets E_1 and E_2:

- if each member of E_1 is connected to at most one member of E_2, the *R* is *many-to-one* (m-to-1) relationship set from E_1 to E_2;
- if *R* is many-one from E_1 to E_2 and also many-one from E_2 to E_1, then *R* is *one-to-one* (1-to-1) relationship set;
- if *R* is neither many-one from E_1 to E_2 or from E_2 to E_1, then *R* is *many-to-many* (m-to-n) relationship set.

The multiplicity can be shown as labels on edges on an E/R diagram.

If an entity set appears *n* times in a single relationship set, the relationship set is drawn by *n* edges. Each edge represents a different *role* that an entity set plays in the relationship set. A role explains the meaning of the entity set in the relationship set. A role is representing as a label on an edge between an entity set and a relationship set.

An entity set *E* can contain members that have special properties not associated with all members of the set. The members with special properties can be combined in a separate class *E'*. *E'* is a *subclass* of *E*, *E* is a *superclass* of *E'*. An entity set is connected to its subclasses using a relationship set of type *isa*, represented by a triangle. One side of the triangle is attached to a subclass. The opposite point is attached to the superclass. Every *isa* relationship set is one-to-one.

3.1.2 E/R modeling

The E/R notation allows modeling a static representation of data structures. The notation does not define operations on data, so it is impossible to model behavior of entity sets.

The E/R model can be used on different levels of abstraction.

The *conceptual level modeling* is the most abstract. The conceptual model is used to present general ideas for thinking. The model is concerned with static elements like entities and relationships between them.

The *information level modeling* uses the ideas introduced at the conceptual level. The information model aims to represent entities and relationships into information structures. The model is concerned with semantics of the data structure. It includes roles of entity sets describing a meaning of entity sets in relationship sets.

The model also can include attributes and values connected to the entity sets and relationship sets.

The *data level modeling* is intended to represents information model elements into the data to be processed by a computer. The data model includes constraints on value sets, attribute sets, entity sets and relationship sets.

3.1.3 E/R model extensions

The initial E/R model was extended, and purpose of use the model was shifted from database design to domain modeling. An example of extension is *ORM (Object-Role Modeling) methodology*.

Later in the thesis, in section 4.2.3.3, the ORM model is mentioned in connection with Visual Studio. Net development environment for applications. ORM model is used to generate a data model, and later the model is used in database design.

ORM was developed by Terry Halpin as a tool for conceptual modeling of information systems. The intention was to provide an instrument for analyzing an information system at the level easily understood for non-technical domain experts. ORM supports modeling of an information system at the conceptual level, using natural language and graphical notations.

The ORM methodology considers a domain as a set of objects (entities or values). Similar objects are collected into an *object type*. Objects perform different functions or play *roles* (parts in relationships). The approach is called *fact-oriented modeling*. It presents data as a collection of elementary facts. These facts can not be divided into sub-facts without loosing information.

A *fact* is a relationship between objects. A fact says that an object has a property. One or more objects participate in a relationship. Similar facts (relationships) compose a *fact type*.

In ORM, it is possible to have relationships with one role and many roles analogously to the sentences in natural language. A fact in ORM is a predicate; it can be viewed as a sentence of natural language consisting from objects. A sentence having particular objects is an instance of a fact type.

Instances of fact types can be combined into ORM sample populations and supplement ORM conceptual schema. Sample populations can be used for checking and understanding constraints.

ORM gives a possibility to group facts into different structures. But that structures are not parts of the ORM conceptual schema, they are not conceptual issues. What mattes in the ORM conceptual modeling is fact orientation. The conceptual level is a level of discussion between a domain expert and a modeler. A domain expert must be able to understand easily a conceptual schema and not be bothered by technical details.

ORM has a number of advantages:

- ORM simplifies conceptual modeling. It uses natural language and simple graphical notations for modeling and validation the models. ORM can be used for capturing business rules.
- ORM simplifies query formalization. ORM allows making queries at the conceptual level without knowledge about data structure representation. The queries are formulated as conditional paths through an ORM diagram.
- In contrast to E/R model, ORM model does not contain attributes. It helps to minimize the effect of changes in the model to conceptual schema and queries.
- ORM methodology has a mechanism for representing in other models. It includes mapping procedures to transform the ORM model into E/R model, UML model and relational model.

3.2 REA model of a retail shop enterprise

The E/R notation has been traditionally used for data structures representation in database design. A specific use of the E/R notation is REA (Resource-Event-Agent) model presented in following.

In REA model the E/R diagram notation is used to map a real business world to a formal model. REA model was first introduced to model an accounting domain. Later it was extended for modeling an enterprise domain.

This chapter gives a short introduction to the REA accounting model and describes the extension of REA model for enterprise modeling. An instantiation of the REA model for a retail shop enterprise is represented in the chapter.

3.2.1 REA model of accounting

The REA accounting model was proposed by McCarthy in 1979. Purpose of the model is to provide a generalized framework for domain understanding of accounting. The REA model can be a base for database construction.

The REA model was developed from an analysis of traditional accounting, but aims to avoid double-entry principle elements. The double-entry bookkeeping implies artifacts for manual recording the financial data which are not necessary for accounting system. It is an extensive way to capture the economic activities. The basic principle of double-entry bookkeeping is to document all economic events in a ledger divided into two parts - debit and credit –fixing information about income and expenditures of an enterprise. The ledger has sections called accounts for every type of business transaction. The money value of each transaction should be entered in the ledger once on each side of the ledger. The financial situation of an enterprise can be derived from the ledger by summation all entries from credit side and debit side for every account. Basic definitions and principles of double-entry accounting are placed in the Appendix B.

The REA model tends to capture the essence of accounting rather than bookkeeping. It deals mainly with business activities that include monetary transactions caused by those activities. The double-entry elements can be presented in the model as an extension.

The REA model specifies possible entity sets as Resources, Events and Agents which REA abbreviation stands for:

- **Resource** represents any economic object that is under control of an enterprise. A resource is limited and has value.
- **Event** represents any economic activity.
- **Agent** represents a person or an enterprise participating in economical activities.

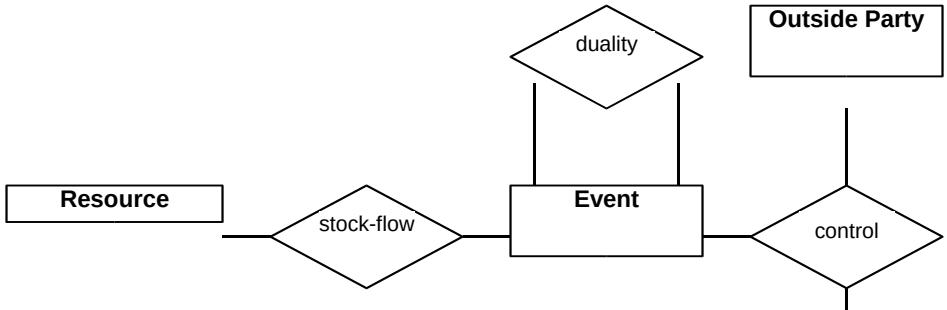
The REA model supports two types of relations between entity sets: generalization and association.

Generalization defines relation between entity sets of the same type. It relates a subset of entities to a generalized superset. Thus subset of Agents can be generalized to a **Unit** set. Agents and Units can be generalized to a **Party**. Units are always inside parties. They are organization departments or divisions working for an enterprise.

The *association* shows relation between entity sets of different or the same types. Entity sets can be assigned roles playing in the relations. There are four possible association relationships:

- **stock-flow** - a binary relationship connects a Resource with role ‘stock’ and an Event with role ‘flow’. The Event causes a change in the Resource;
- **duality** - a binary relationship connects two interplaying Events. The Events mutually depend on each other. One Event increments the related resource set and has a role ‘increment’. Another mirror event decrements the related resource set and has a role ‘decrement’;
- **control** – a three-way relationship connects a Resource, a Party belonging to an enterprise and having ‘inside’ role, and a Party not belonging to an enterprise and having ‘outside’ role. The Parties has a power to use or dispose the Resource;
- **responsibility** - a binary relationship connects a ‘superior’ Unit and a ‘subordinate’ Unit. It describes a hierarchical structure of the Units.

The REA initial model for accounting is represented in the Fig. 2.2.1. The model shows that data associated with each economic event are recorded and stored in connection to resources and agents linked to the event.



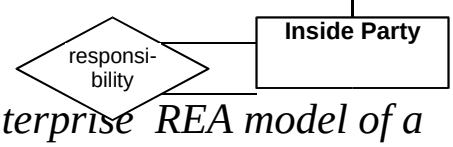


Figure 2.2.1. *REA model for accounting.*

3.2.2 REA model of enterprise

The REA model can be extended for enterprise-wide modeling. REA notation can represent business activities not directly concerned with accounting. A topic of business enterprise and enterprise functioning is explained in details in chapter 3.1.

REA model proposes a structure of a business enterprise. The structuring idea is that there are two related events involved in any economic process. They are connected by duality relationships. Fig. 2.2.2 illustrates the extension of the initial REA accounting model for enterprise modeling and shows two basic 'mirror' economic activities (i.e. events within the REA model) being essential for an enterprise:

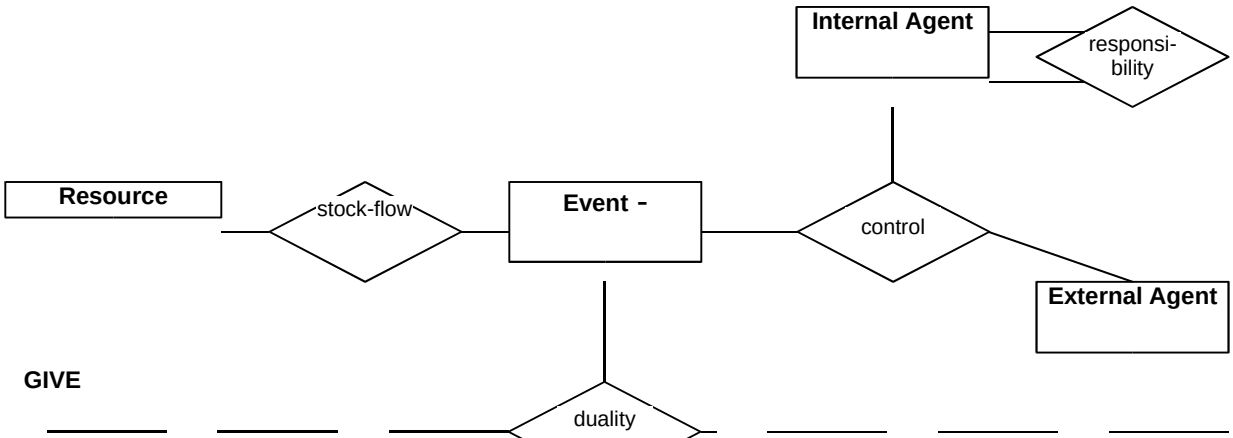
1. **Give** – resource outflows (decrements),
2. **Take** – resource inflows (increments).

All other activities are extensions or supplementary of these two. The duality represents that an economic process is the exchange of resources, where one recourse is incremented and another is decremented. The model also presents internal and external agents involved in every event. The model establishes recursive responsibility relationship between inside agents (units).

The REA model's structure is consolidated in **REA axioms** and establishes the REA modeling principles stated in [5], p.12:

1. "At least one inflow event and one outflow event exist for each economic resource, conversely inflow and outflow events must affect identifiable resources.
2. All events affecting an outflow must be eventually paired in duality relationships with events effecting an inflow and vice-versa.
3. Each exchange needs an instance of both the inside and outside subsets."

The axioms can be used for consistency checking and validation of REA-based models.



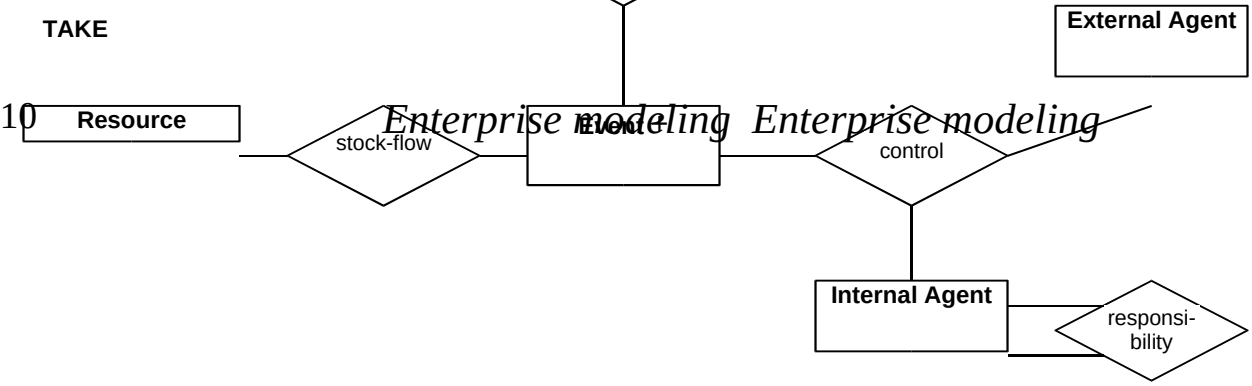


Figure 2.2.2. *REA model for accounting extended for entire enterprise.*

The REA framework supposes *three-levels of abstraction* to present business activities of an enterprise.

The highest level represents business **processes** and flows of consumed and created resources involved in the exchange. The sequence of the processes forms an enterprise value chain. Value chain is explained in chapter 3.1.1.

The middle level specifies every process form the highest level. Each process is the basic REA model presented by Give -Take pair of operations. The process modeled via REA template is also called a cycle.

The lowest level shows a sequence of **tasks** composing an event. Tasks do not require REA model principles and serve to represent detailed activities of a process.

In spite of semantic and structure orientation, the REA model is limited to represent enterprise accounting infrastructure. Firstly the shortage of REA framework is that it's a static model. It does not provide facilities to model things in time. Thus it's impossible to solve problems as periodical accounting, for instance, periodical allocation of expired resources.

Also it's problematical to model explicitly not tangible resources, in particular **claims**. Claims are temporary imbalances that exist between sales and cash receipts, for example, accounts payable and accounts receivable, which are very important terms in accounting.

Another weakness is insufficient expressiveness of E-R diagram notation to show part-whole relationships.

Entity sets and relationship sets combined in a graph form the REA modeling specification - the **conceptual schema**. It represents a global view of an enterprise, both static and dynamic. The conceptual schema expresses semantics of the REA model. The conceptual schema closely corresponds to real business world phenomena.

3.2.3 REA model of retail shop enterprise

The REA model for enterprise can be instantiated for a particular enterprise domain.

A source for the model is the E/R diagram for a retail shop enterprise represented in Appendix D. The model was introduced by McCarthy in [8], Figure 6, p.675. The scheme is made in terms of REA accounting model and shows primary activities of a retail shop which are relevant for accounting.

The *retail shop enterprise* is a business enterprise to sell goods to customers. Customers buy goods for their own needs but not for reselling to anybody else.

The source E-R diagram for a retail shop enterprise is a very early attempt of REA modeling. The diagram presents a general model for an accounting domain of a retail enterprise on a high abstraction level. It stresses only the basic business processes going in the retail shop and does not consider any supplementary activities as documents/paper issues confirming the business transactions.

The considering model does not specify, weather it is a web-based sale or not, what are the types of products the shop sales to customers, what the particular stuff is involved in the business activities, how the products are obtained from a wholesaler, how the enterprise keeps track on their customers and employees, how the customer service works. All the omitted details, as well as organization document flow, dependent on scope and profile of a retail shop do not make sense on the core model and can be specified afterwards.

The detailed description of basic business activities in the considered retail shop model is placed later in the section.

The diagram contains the entity set corresponding to objects from accounting domain of the retail shop enterprise:

Resources: Inventory, Cash, Equipment;

Events: Capital Transaction, Cash Disbursement, Cash Receipt, General & Administrative Service, Personnel Service, Equipment Acquisition, Sale, Purchase, Order;

Agents: Customer, Stockholder, Vendor, Employee.

There is a short explanations of the 16 terms named above:

- **Customer** – a person or enterprise buying products from the retail enterprise.
- **Stockholder** - a person or enterprise owns the resources of the retail enterprise.
- **Vendor** - a person or enterprise selling products or services or equipment for the retail enterprise.
- **Employee** - a person hired and paid by the retail enterprise and doing personal service.
- **Inventory** – products to be sold by the retail enterprise to customers and amount of these products presented at the stock of the enterprise.
- **Cash** – a resource having monetary value available for the enterprise.
- **Equipment** – a set of things needed for the enterprise to perform its main business to sell goods to customers but not for sale in the frame of the main business.

- **Capital Transaction** – an event of cash transaction between a stockholder and the enterprise as a result of financial activity.
- **Cash Disbursement** – an event of cash transaction from the enterprise to a vendor or employee as a payment for goods or services or to a stockholder as a result of financial activity.
- **Cash Receipt** – an event of cash transaction from a customer to the enterprise for goods sold or from a stockholder to the enterprise as a result of financial activity.
- **General & Administrative Service** – A management activity event required for running the enterprise.
- **Personnel Service** – an event where an employee provides service to the enterprise.
- **Equipment Acquisition** – an event where the enterprise buys equipment from a vendor.
- **Sale** – an event where the enterprise sell goods to a customer.
- **Purchase –** an event where the enterprise purchases goods from a vendor.
- **Order** - an event where a customer makes a request to the enterprise for goods he wants to buy.

The source diagram also contains relationship sets. Fig. 2.2.3.1 presents the relationship sets and indicates entity sets involved in each relationship set, types of roles played by the entity sets and names of relationship sets. Every relationship set is named individually though the source diagram presented in Appendix D contains repetitive names for different relationship sets.

'Payment for', 'payment of' and 'fills' are *duality relations*. 'Allocate cost of', 'flow of', 'line item' are *stock-flow relations.* 'Received from', 'made to', 'supplier of', 'employed in' are *control relations*.

Relationship set's name	Entity set 1 Name	Role type	Relationship name given in the source diagram from Appendix D	Entity set 2 Name	Role type	Relationship set's multiplicity
Rel.1	Order	*exchange transaction*	received from	Customer	*outside party*	(n-to-1)
Rel.2	Order	*flow*	line item	Inventory	*stock*	(m-to-n)
Rel.3	Order	*decrement*	fills	Sale	*increment*	(1-to-n)
Rel.4	Sale	*exchange transaction*	made to	Customer	*outside party*	(n-to-1)
Rel.5	Sale	*flow*	line item	Inventory	*stock*	(n-to-m)
Rel.6	Purchase	*exchange transaction*	supplier of	Vendor	*outside party*	(n-to-1)
Rel.7	Purchase	*flow*	line item	Inventory	*stock*	(m-to-n)
Rel.8	Cash Receipt	*increment*	payment for	Sale	*decrement*	(1-to-n)
Rel.9	Cash Receipt	*increment*	payment for	Capital Transaction	*decrement*	(1-to-1)
Rel.10	Cash Receipt	*flow*	flow of	Cash	*stock*	(n-to-1)
Rel.11	Stockholder	*outside party*	partner to	Capital Transaction	*exchange transaction*	(1-to-n)
Rel.12	Cash Disbursement	*decrement*	payment of	Capital Transaction	*increment*	(1-to-1)
Rel.13	Cash Disbursement	*decrement*	payment for	Purchase	*increment*	(1-to-1)
Rel.14	Cash Disbursement	*decrement*	payment for	General & Administrative Service	*increment*	(m-to-n)
Rel.15	Cash Disbursement	*decrement*	payment for	Equipment Acquisition	*increment*	(m-to-n)
Rel.16	Cash Disbursement	*decrement*	payment for	Personnel Service	*increment*	(1-to-n)
Rel.17	Cash Disbursement	*flow*	flow of	Cash	*stock*	(n-to-1)
Rel.18	Employee	*outside party*	employed in	Personnel Service	*exchange transaction*	(1-to-n)
Rel.19	Vendor	*outside party*	supplier of	Equipment Acquisition	*exchange transaction*	(1-to-n)
Rel.20	Vendor	*outside party*	supplier of	General & Administrative Service	*exchange transaction*	(1-to-n)
Rel.21	General & Administrative Service	*flow*	allocate cost of	Equipment	*stock*	(1-to-n)
Rel.22	Equipment Acquisition	*flow*	flow of	Equipment	*stock*	(1-to-1)

As mentioned above, the source E-R diagram is rather old instantiation of a REA model. Here are several critical points named and some necessary extensions for the model are done.

The source E-R diagram misses some information. Thus it does not specify inside parties involved in the relations. It's logically to suppose that Stockholder and Employee are inside parties, and Customer and Vendor are outside parties.

The model also misses responsibility relations. There no entities representing units.

Figure 2.2.3.1. *Relationship sets in the REA model for a Retail Shop Enterprise.*

inverse directions in the complete description of the domain.

Below there is an attempt to name the roles of entity sets participated in associations from Fig. 2.2.3.1 (all associations are considered as bidirectional):

1: Order is received from Customer / Customer issues Order

2: Order specifies item in Inventory / Inventory is reserved by Order

3: Order is part of Sale / Sale has some Orders

4: Sale is made to Customer / Customer is the destination for Sale

5: Sale removes item in Inventory / Inventory is decremented by Sale

6: Vendor is supplier of Purchase / Purchase is done by Vendor

7: Purchase adds items in Inventory / Inventory is decremented by Purchase

8: Cash Receipt is payment for Sale / Sale pays to Cash Receipt

9: Cash Receipt is payment for Capital Transaction / Capital Transaction pays to Cash Receipt

10: Cash Receipt is inflow of Cash / Cash is incremented by Cash Receipt

11: Stockholder is agent to Capital Transaction / Capital Transaction is done by Stockholder

12: Cash Disbursement is payment of Capital Transaction / Capital Transaction is paid by Cash Disbursement

13: Cash Disbursement is payment for Purchase / Purchase is paid by Cash Disbursement

14: Cash Disbursement is payment for General & Administrative Service / General & Administrative Service is paid by Cash Disbursement

15: Cash Disbursement is payment for Equipment Acquisition / Equipment Acquisition is paid by Cash Disbursement

16: Cash Disbursement is payment for Personnel Service / Personnel Service is paid by Cash Disbursement

17: Cash Disbursement is outflow of Cash / Cash is decremented by Cash Disbursement

18: Employee is an agent of Personnel Service / Personnel Service is done by Employee

19: Vendor is an agent of Equipment Acquisition / Equipment Acquisition is done by Vendor

20: Vendor is an agent of General & Administrative Service / General & Administrative Service is done by Vendor

21: General & Administrative Service is allocation cost of Equipment / Equipment cost is allocated by General & Administrative Service

22: Equipment Acquisition is inflow of Equipment / Equipment is bought by Equipment Acquisition

The generalization relations also do not mentioned in the source. Here generalization relationship sets are proposed:

Resource > Cash, Inventory, Equipment;

Event > Order, Cash Receipt, Capital Transaction 9, Sale 8, Cash Disbursement, Purchase 13, Capital Transaction 12, General & Administrative Service 14, Equipment Acquisition 15, Personnel Service;

Agent > Customer, Vendor, Employee, Stockholder.

It seems easier to come to understanding the Retail shop REA model by looking at its parts. Though the highest level of three-level REA architecture is not presented, one can derive processes from the source diagram. The diagram can be divided into several pieces, which contain entities and relations representing a set of logically connected business transactions.

These pieces of diagrams or cycles are processes because they carry the duality nature of economic events. Fig. 2.2.3.2 presents significant meaning of every cycle and agents, resources, events connected to a cycle.

Note, that inside party agents are not named here due to the incompleteness of the considering E/R model. An inside party agent can be is a clerk employed at the retail shop.

Payroll Cycle and Expenditure Cycle – Services does not present a service as resource due to its intangibility.

Cycles' descriptions are presented below in the section. The descriptions contain:

- brief literate explanation of the cycles;
- REA diagrams. The diagrams are parts of the source diagram from Appendix D. The relationship sets have their original names from the source diagram and unique names introduced in the table 2.2.3.1;
- tables commenting the diagrams. The tables contain relationship sets and entity sets. The tables show relationship sets' names and types, entity sets' names, types and roles in the relationships.

Cycle name	Cycle significance	Agents	Resources	Events
Revenue cycle	Products are exchanged for cash	Customer	Inventory, Cash	Order, Sale, Cash Receipt
Expenditure cycle – Inventory	Cash is exchanged for products	Vendor	Inventory, Cash	Purchase, Cash Disbursement
Expenditure cycle – Equipment	Cash is exchanged for equipment	Vendor	Equipment, Cash	Equipment Acquisition, Cash Disbursement, General & Administrative Service
Expenditure cycle – Services	Cash is exchanged for services	Vendor	Cash	Cash Disbursement, General & Administrative Service

Payroll cycle	Cash is exchanged for labour	Employee	Cash	Personnel Service, Cash Disbursement
Finance cycle	Capital flow is directed between a stockholder and the shop cash account	Stockholder	Cash	Cash Receipt, Cash Disbursement, Capital Transaction

Figure 2.2.3.2. *Business processes cycles in the REA model for a Retail Shop Enterprise.*

3.2.3.1 Revenue cycle

The Revenue cycle is presented in Fig.2.2.3.1 and implies the following business activities. Customers place orders for products to the retail shop. As far as the ordered items are available at the shop, the products are taken out from the store and delivered to customers. Customers pay for the products with cash. Cash comes from customers to the retail enterprise and amount of cash available at the retail enterprise increases.

Placing an order can be considered as an event initiated the sale. This event does not correspond to the basic REA pattern.

The sale occurs only if a payment for the sale takes place, which expresses the duality nature of these two business operations.

A customer can buy the ordered products selectively, i.e. not pay for all the items at the same time or pay once for several shipments. Products also can be delivered to the customer in a selective sequence, for example in a case of a prepaid purchase.

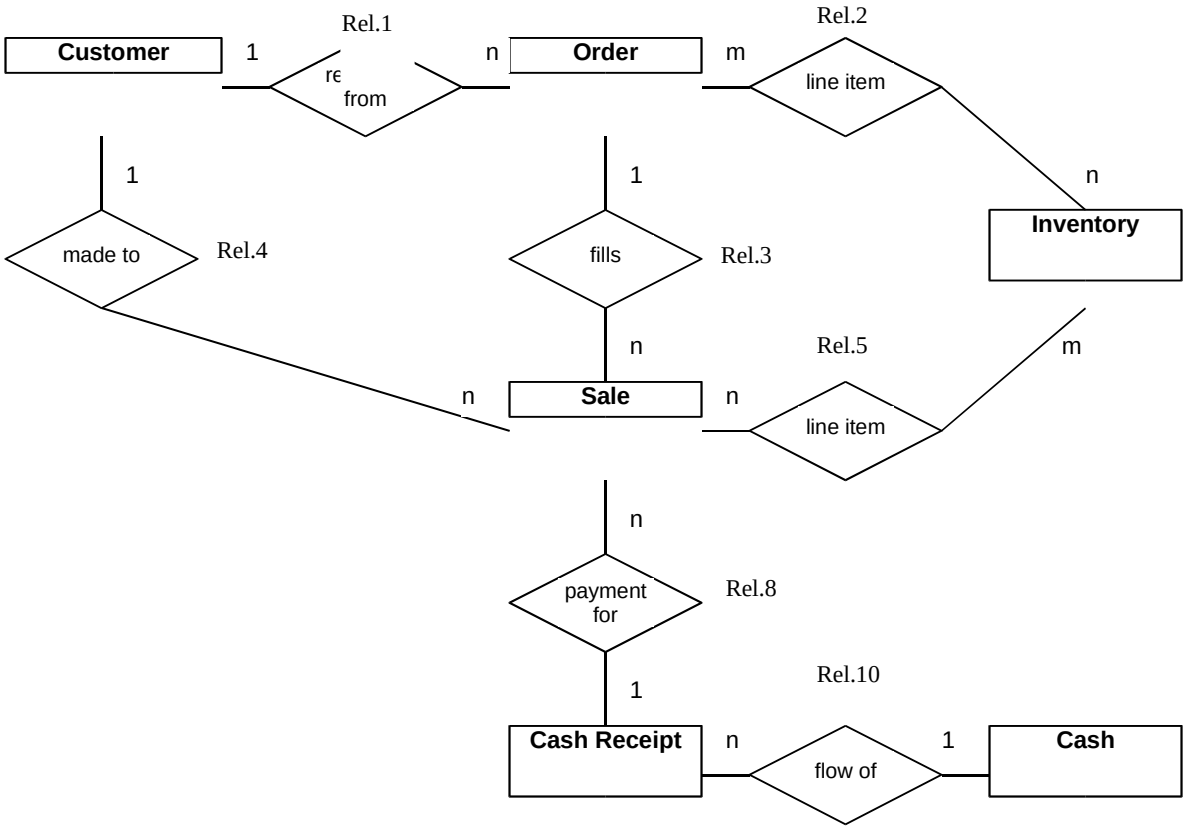


Figure 2.2.3.1 (a) *Revenue cycle from the Retail shop E/R model.*

Relationship sets		**Entity sets**		
Name	**Type**	**Name**	**Type**	**Role**
Rel.1	Control	Customer	Agent	issues Order
		Order	Event	is received from Customer
Rel.2	Stock-flow	Order	Event	specifies item in Inventory
		Inventory	Resource	is received by Order
Rel.3	Duality	Order	Event	is part of Sale
		Sale	Event	has some Orders
Rel.4	Control	Sale	Event	is made to Customer
		Customer	Agent	is the destination for Sale
Rel.5	Stock-flow	Sale	Event	removes item in Inventory
		Inventory	Resource	is decremented by Sale
Rel.8	Duality	Sale	Event	pays to Cash Receipt
		Cash receipt	Event	is payment for Sale
Rel.10	Stock-flow	Cash receipt	Event	is inflow of Cash
		Cash	Resource	Is incremented by Cash Receipt

B.3.2

Figure 2.2.3.1 (b) *Explanations to the Revenue cycle diagram.*

3.2.3.2 Expenditure cycle – Inventory

The Expenditure cycle – Inventory is shown in Fig. 2.2.3.2. Products for the shop are purchased from vendors. A purchase is initiated by the retail shop. As it was mentioned earlier, the inside parties are not shown at the source model, so the shop unit entity does not present at the cycle diagram.

When the products are received they are stored at the shop and the quantity on hand for each purchased products is increased. A purchase event initiates a cash disbursement from the shop to a vendor. The amount of cash available at the shop in decreased.

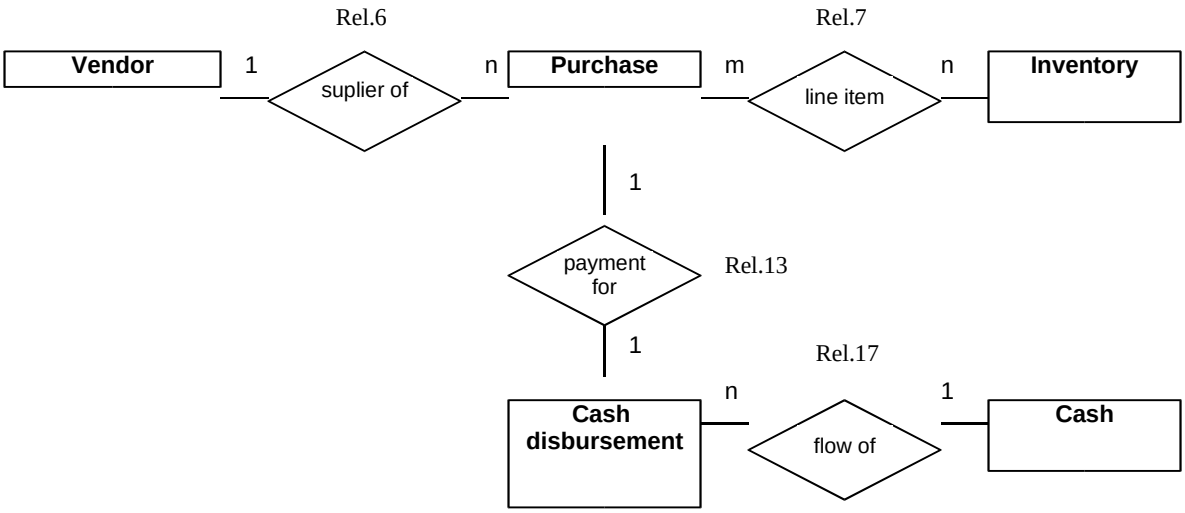


Figure 2.2.3.2 (a) *Expenditure cycle – Inventory from the Retail shop E/R model.*

Relationship sets		Entity sets		
Name	Type	Name	Type	Role
Rel.6	Control	Customer	Agent	issues Order
		Order	Event	is received from Customer
Rel.7	Stock-flow	Order	Event	specifies item in Inventory
		Inventory	Resource	is received by Order
Rel.13	Duality	Order	Event	is part of Sale
		Sale	Event	has some Orders
Rel.17	Stock-flow	Sale	Event	is made to Customer
		Customer	Agent	is the destination for Sale

Figure 2.2.3.2 (b) *Explanations to the Expenditure cycle diagram.*

3.2.3.3 Expenditure cycle – Equipment

Fig.2.2.3.3 reflects the activities at the Expenditure cycle – Equipment. Equipment acquisition is initiated by the retail shop and is placed to a vendor. When the equipment is received from a vendor, this acquisition is updated, and a cash disbursement from the shop to a vendor occurs. The amount of cash available at the shop in decreased.

The cost of equipment is allocated periodically for each equipment asset. Each cost allocation transaction is identified by the time and includes the dates for the period of cost allocation. The relationship set 'payment for' between General administrative service and Cash Disbursement is shown with dashed line in order to stress that its nature is different. Periodical cost allocation of the equipment does influence only indirectly on cash resource. But it's impossible to reflect explicitly within REA framework.

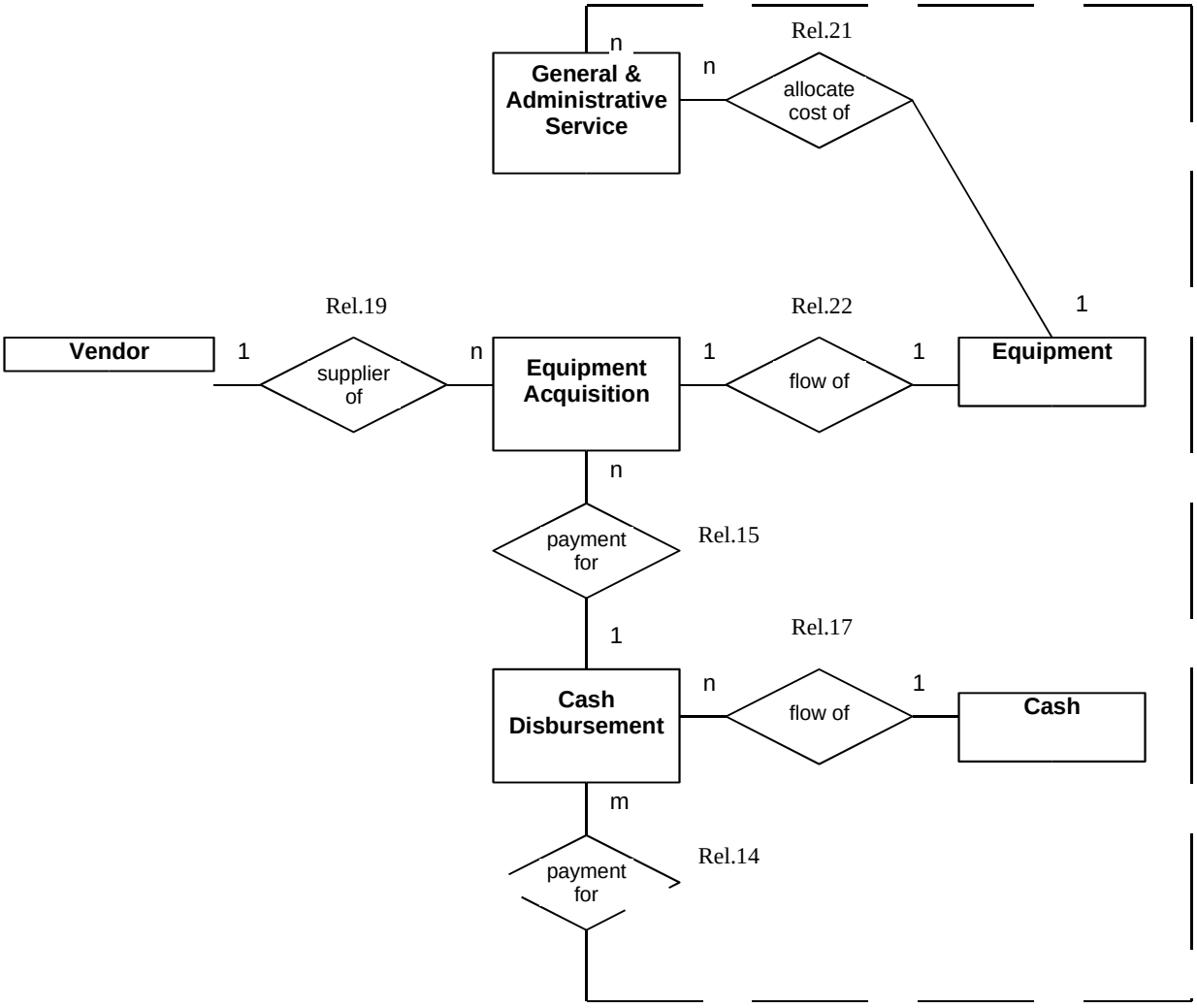


Figure 2.2.3.2 (a) *Expenditure cycle – Equipment from the Retail shop E/R model.*

Relationship sets		**Entity sets**		
Name	**Type**	**Name**	**Type**	**Role**
Rel.14	Duality	Cash Disbursement	Event	is payment for General & Administrative service
		General & Administrative service	Event	is paid by Cash Disbursement
Rel.15	Duality	Cash Disbursement	Event	is payment for Equipment Acquisition
		Equipment Acquisition	Event	is paid by Cash Disbursement
Rel.17	Stock-flow	Cash Disbursement	Event	is outflow of Cash
		Cash	Resource	is decremented by Cash Disbursement
Rel.19	Control	Equipment Acquisition	Event	is done by Vendor
		Vendor	Agent	is an agent of Equipment Acquisition
Rel.21	Stock-flow	General & Administrative service	Event	is allocation cost of Equipment
		Equipment	Resource	cost is allocated by General & Administrative service
Rel.22	Stock-flow	Equipment Acquisition	Event	is inflow of Equipment
		Equipment	Event	is bought by Equipment Acquisition

Figure 2.2.3.3 (b) *Explanations to the Expenditure cycle - Equipment diagram.*

3.2.3.4 Expenditure cycle – Services

Fig. 2.2.3.4 represents the Expenditure cycle – Services activities of the retail shop. If the retail shop enterprise requires some outsource services as the shop rent, advertising, utilities, est., it initiates cash disbursement from the shop to a vendor of the services. The amount of cash available at the shop in decreased.

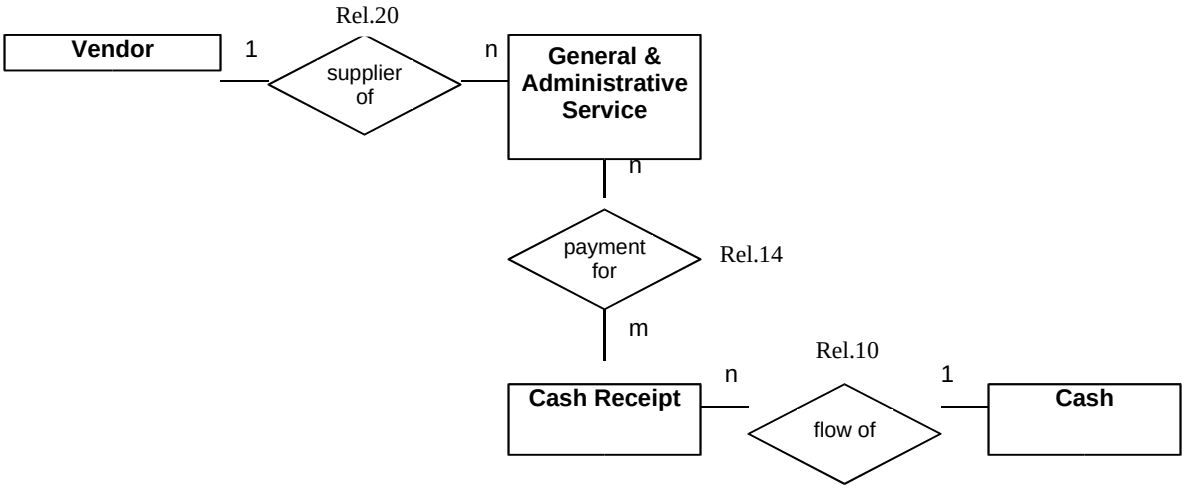


Figure 2.2.3.4 (a) *Expenditure cycle – Services from the Retail shop E/R model.*

Relationship sets		**Entity sets**		
Name	**Type**	**Name**	**Type**	**Role**
Rel.10	Stock-flow	Cash receipt	Event	is inflow of Cash
		Cash	Resource	Is incremented by Cash Receipt
Rel.14	Duality	Cash Disbursement	Event	is payment for General & Administrative service
		General & Administrative service	Event	is paid by Cash Disbursement
Rel.20	Control	Vendor	Agent	is an agent of General & Administrative service
		General & Administrative service	Event	is done by Vendor

B.3.6

Figure 2.2.3.4 (b) *Explanations to the Expenditure cycle - Services diagram.*

B.3.7 Payroll cycle

The Payroll cycle is represented in Fig. 2.2.3.5. The retail shop employees are paid salary for the service they provide to the shop. The salaries payment is based on job agreements between employees and the shop and initiates cash disbursement from the shop to a vendor of the services. The amount of cash available at the shop in decreased.

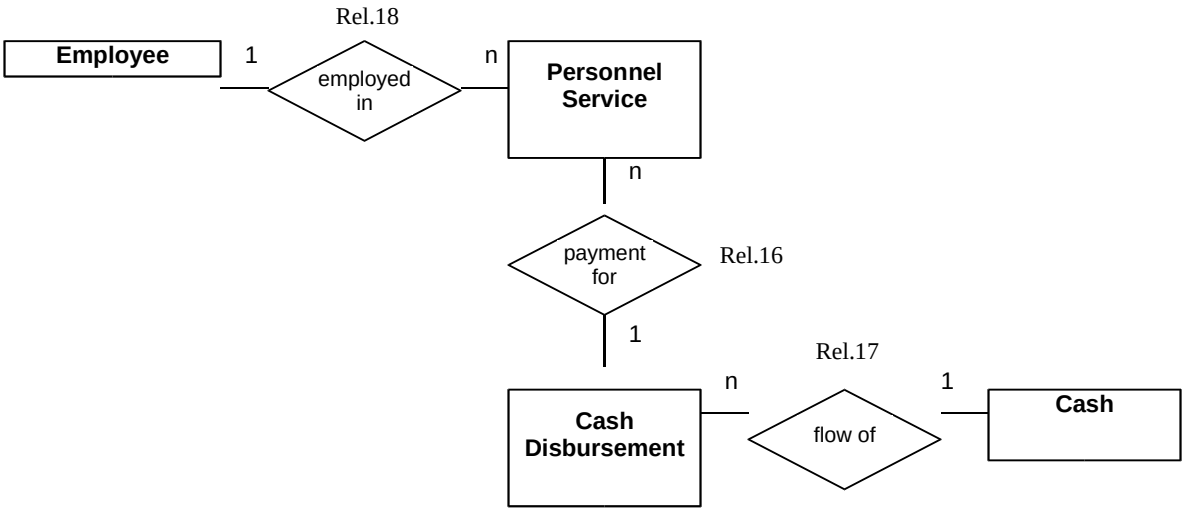


Figure 2.2.3.5 (a) *Payroll cycle from the Retail shop E/R model.*

Relationship sets		**Entity sets**		
Name	**Type**	**Name**	**Type**	**Role**
Rel.16	Duality	Personnel Service	Event	is paid by Cash Disbursement
		Cash Disbursement	Event	is payment for Personnel Service
Rel.17	Stock-flow	Cash Disbursement	Event	is outflow of Cash
		Cash	Resource	is decremented by Cash Disbursement
Rel.18	Control	Employee	Agent	is an Agent
		Personnel Service	Event	is done by Employee

Figure 2.2.3.5 (b) *Explanations to the Payroll cycle diagram.*

3.2.3.5 Finance cycle

The Finance cycle activities are shown in Fig. 2.2.3.6. A stockholder performs financial activities and initiates a capital transaction like investing funds to the enterprise (Cash Receipt) or drawing funds from the enterprise (Cash Disbursement).

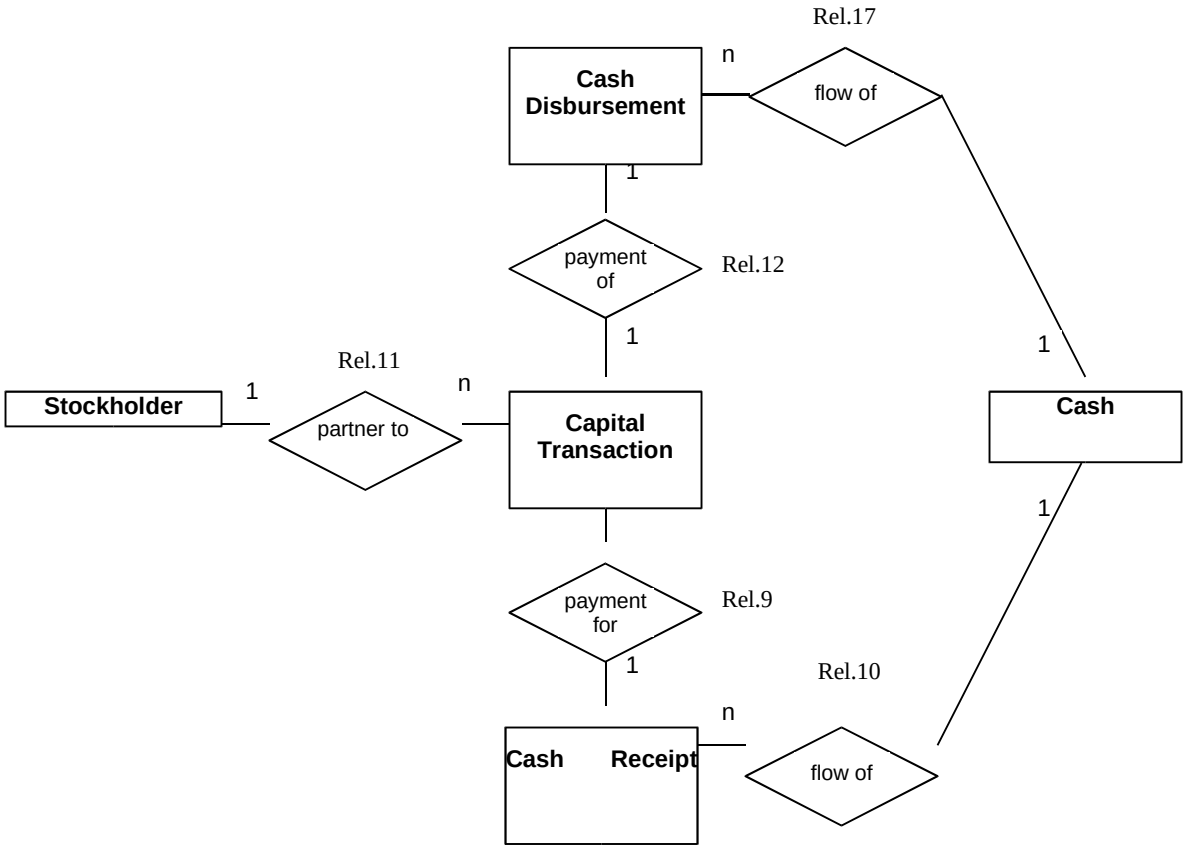


Figure 2.2.3.6 (a) *Finance cycle from the Retail shop E/R model.*

Relationship sets		**Entity sets**		
Name	**Type**	**Name**	**Type**	**Role**
Rel.9	Duality	Cash Receipt	Event	is payment for Capital Transaction
		Capital Transaction	Event	pays to Cash Receipt
Rel.10	Stock-flow	Cash receipt	Event	is inflow of Cash
		Cash	Resource	Is incremented by Cash Receipt
Rel.11	Control	Stockholder	Agent	is agent for Capital Transaction
		Capital Transaction	Event	is done by Stockholder
Rel.12	Duality	Cash Disbursement	Event	is payment of Capital Transaction
		Capital Transaction	Event	is paid by Cash Disbursement
Rel.17	Stock-flow	Sale	Event	is made to Customer
		Customer	Agent	is the destination for Sale

Figure 2.2.3.6 (b) *Explanations to the Finance cycle diagram.*

3.3 REA model in UML notation

This section examines a possibility to represent a REA model in the UML notations. The REA model is considered at the conceptual level and the information level. The UML potential for enterprise conceptual modeling is also discussed in chapter 6.1.

As mentioned in section 2.1.3, the REA model is a special use of the E/R model. The E/R model is represented in the UML language: entity sets correspond to UML classes and relationship sets correspond to associations.

The UML notations include about 200 concepts. But in order to represent the REA model a limited set of UML concepts is only required. The section shows some parallels and disagreements between UML and REA concepts. Several ways to represent the REA model in UML are suggested.

3.3.1 Entity sets vs. classes

There are similarities between an entity in REA model and an instance in UML.

The basic idea of object-oriented programming (OOP) is to encapsulate the properties of a thing (or a concept) and behavior of the thing into an instance. So an instance could be identified both by its properties and behavior. A class in UML has attributes describing a data structure of instances of the class. A class also has operations to describe the behavior of the instances belonging to the class.

Presenting the REA model in UML notations it is possible to model an entity set as a class. But there is no complete correspondence between a class and an entity set. The reasons are named below in this section.

Declarative features

Both an entity in REA and an instance in UML represent a thing from a world or a concept describing a world. Attribute sets' values connected to an entity conform to attribute values connected to an instance.

Analogously an entity set corresponds to a class. Attribute sets associated with an entity sets conform the attributes of a class.

Behavioral features

The main difference between E/R and UML models concerns with the behavioral aspect. The E/R model is a static representation of data structures. The model does not define operations on data. The behavior of entities is not specified in E/R. But an instance in UML can own both properties and behavior.

Representing the REA model in UML, the real challenge is to model operations of a class. The first task in this process is to explicitly find out the behavior associated with an entity set. Since the REA model does not specify the operations associated with entity sets, the behavioral information should be obtained outside the REA model.

But some hints for modeling operations can be found in the REA diagrams.

It was mentioned in the section 2.2.1 that the REA model was initially intended for database design. The REA model served as an abstract representation of data model. And in order to come from an abstraction to a real database model, there is a need to define

operations on data. This problem is handled by the *conclusion materialization process* introduced by W.E. McCatrhy in [1], p.567. The process defines on what information should be produced and how to produce this information. Resuming the ideas of conclusion materialization, the following conclusion can be done:

- The entity sets of the interest of accounting are resources. This means that resources are subjects for accounting activity. Accounting is concerned with values of properties associated with resources. So the attributes of classes representing resources are subjects for accounting operations. Attribute values can be changed or unchanged as the result of an operation.
- All information which is in the interest of accounting can be derived from events. This fact applies that events are objects for accounting activity. Events affect values of properties associated with resources. Events have time points when they occur. So a class representing an event will have an attribute for the time point and operations to affect classes representing resources. An event class also can have a state attribute to show the state of the event accounting activity.
- There are four kinds of procedures to produce the accounting information: triggered, adjustment, view, derivation procedures. In order to model the operations to realize the procedures, state charts, activity and sequence diagrams can be used.

  Triggered and adjustment procedures change values of resources attributes. The procedures will change the state of events. So they can be modeled as state charts. The conditions of changing the states of events can be shown by OCL.

  View and derivation procedures can not change values of resources attributes but use them to present accounting information. But in these cases a user will be in focus of the procedures. So to model operations realizing the procedures the following UML diagrams can be used: use cases, activity diagrams, sequence diagrams. The navigation paths can be shown by OCL.

Modeling the procedures in UML should be considered in regards with both entity sets and relationships. The relationships are discussed in section 2.3.2.

There is no final solution on how to realize procedural features associated with REA model in UML. REA model is not sufficient to supply the necessary information. For example, accounting activities like periodical allocation of resources, taxation, treating claims need additional algorithmic description. The description can depend on a domain or even from a particular case.

The possible answer is to model the REA domain-oriented patterns from REA accounting practice and database design. The patterns should include literate description of a case, REA models on three abstraction levels (value chain level, business processes middle level, task level), associated design pattern (conclusion materialization) and algorithm descriptions. The REA patterns should be realized as patterns in UML. UML patterns should include use cases, class diagrams, behavior diagrams, OCL constraints.

3.3.2 Relationships

Referring to the section 2.2.1, there are two kinds of relationships in the REA model: associations and generalizations. All the relationships in REA (stock-flow, duality, control) have a dynamic nature. The only static relationship is responsibility, specifying

accountability type of an employee within the organization. The REA model only names the dynamic concepts; it does not specify their behavior.

Associations

The associations between entity classes in REA can be shown as associations in UML. The roles of entity sets in associations will correspond to the roles of classes in UML. This is enough to provide correspondence between a model and users' intuition how their business activities are organized in general. But there should be interpretation of the inverse directions of associations in a complete description of the domain.

Another possibility is to model REA associations as association classes in UML. An association class has a meaning as a description of a link between a specific member of one class and a specific member of another class connected with an association. The purpose of an association class is to provide proper distribution of attributes among classes.

Association classes can be applied to model triggers procedures to place triggers operations and attributes affected by the operations. An association class serves as a location for attributes and operations that are properly belong to an association between classes. So the association will be at the level of the classes' instances.

Generalizations

Generalization in REA corresponds to a generalization relationship in UML.

An example could be an entity set 'Employee' which is a superset for subsets 'Salesperson', 'Accountant', 'Purchasing Clerk'. This hierarchy can be directly represented by generalization in UML: a superclass 'Employee' and subclasses 'Salesperson', 'Accountant', 'Purchasing Clerk'.

But in order to stress the semantics of the top-level generalizations we have in REA, like a resource, an agent, an event, they can be realized as stereotype classes. The stereotype classes can be used as building blocks in patterns. Stereotype classes will capture particular sets of attributes and operations realizing particular behavior patterns. In chapter 6.1 UML stereotypes are discussed in details.

Partitive relationships

The REA model does not provide part-whole relationships. But the events can be divided into subparts. For example, an order includes a number of order lines, and every order line can be treated independently.

UML provides two types of part-whole relationships: aggregation and composition. They are binary associations.

The events and their subparts can be modeled using a week binary association – composition. Thus an order is a composition of order lines. An order class will include operations to create order line objects and accumulate summary information about the order lines.

Partitive relationships can be used to get rid from many-to-many relationships. Many-to-many relationships can be substituted with a week association through an additional entity.

3.3.3 Use cases

The REA model is a static domain model of an enterprise. The step to describe performance of the enterprise model is use cases. Section 6.1.2.1 explains how use case diagram can be used for enterprise conceptual modeling.

Agents in REA diagrams can be actors in use cases, relationships between agents and other entity sets can serve to define activities of the agents.

The idea of the use cases modeling in connection with the REA is to make use cases for every event entity and combine them into a use case diagram corresponding to a cycle in a REA model. As it was proposed by Geerts, McCarty and Alumni [10], a use case will explain the event on a more detailed level of abstraction - task level. Geerts, McCarty and Alumni defined tasks in REA analysis as "compromises to full specification (that is, they are economic events where an analyst doesn't try to specify full patterns)", [10], p.9.

The use case diagram modeling can be done in iteration process. The first iteration will include the most general use cases. Later in design process, every use case may be specified downward into sub-cases for the tasks. But the purpose of the use case modeling is to find domain-specific behavioral patterns which business transactions can follow.

The scope of further detailed elaboration of the use cases is a question of a specific domain. If the domain understanding requires to trace high diversity of possible behavioral ways, the use case diagram tends to be large and messy.

Use cases also can be made as domain specific patterns. These patterns will have generalized images for actors and may be extended for a particular case.

3.3.4 Activity diagram

A possible solution to avoid massiveness of use cases is an activity diagram. The diagram shows activities spread in time and between actors. The topic is also treated in section 6.1.2.5 in respect with possible modifications for enterprise conceptual modeling.

Especially helpful would be an activity diagram with swim lanes. Agents can serve as actors. The swim lanes are attached to actors to show the activity area of the actors.

It is recommended to make a separate activity diagram for a single business process.

The diagram includes actions. Control flows show relationships between the actions. Actions will illustrate the how the events and resources behave in a business process: what happens to resources and how the states of events are changed.

3.3.5 Class diagram

It is impossible to translate directly the REA diagram into a class diagram. There are several problems to be solved:

- Agents in REA can be actors, but an actor in UML is never to be a class. The actors can be shown as classes for the purpose of domain analysis.
- All entities – agents, events and resources - are shown in the same manner, so it's difficult to identify events and things explicitly from the REA diagram.

- Many-to-many relations should be transferred into many-to-one, one-to-one relations. A solution is part-whole relationships and additional entities.
- For domain analysis it is possible to have Avoid non-binary relations as control (event-agent-unit). But for the design stage non-binary relations should be avoided. A solution could be replacing a multi-way relationship by a connecting entity set and binary relationships.

REA axioms can be used for validation purposes of a class diagram.

The topic is further discussed in section 6.1.2.2 regarding enterprise conceptual modeling in general.

3.3.6 Modeling process to convert REA diagram into UML

The following steps can be recommended to convert a REA diagram into UML notations:

1. Identify agents, event, resources in REA diagram.
2. Identify cycles in REA diagram.
3. Make use cases – describe the system from the user view point at the task level. Agents are actors.
4. Make activity diagrams for every cycle.
5. Make class diagrams to show classes and relationships between them.
6. Make class diagrams with attributes and operations.
7. Model the behavior of classes with state charts and sequence diagrams.
8. Specify constraints using OCL.

3.4 UML model of retail shop enterprise

3.4.1 Modeling approach and assumptions

This section gives explanations on how the REA model of a retails shop enterprise can be represented in UML. The initial REA diagram is presented in Appendix D; the refinement of the diagram has been introduced in chapter 2.2. The purpose of the modeling is to provide understanding of the domain of a retail shop enterprise.

The process of representing the REA model of a retail shop enterprise in UML should follow the recommendations suggested in chapter 2.2. But the REA model was not supported by any additional description of the domain. There is no information on results of the conclusion materialization process. The REA diagrams alone are not sufficient to model the functioning of a retail shop enterprise. So there is a need to introduce some assumptions for the modeling.

The traditional way of UML modeling starts from the use case analysis. Here the text models are suggested as an initial stage of this process. Text models are similar to the literate description of use cases.

The text models aims to facilitate the REA model of retail enterprise at the information level and modeling behavior features of enterprise functioning. As far as the E/R diagram of a retail shop enterprise from Appendix D represents only conceptual level, we can directly represent it as a class diagram in UML. This class diagram is a conceptual description of retail shop enterprise functioning. But in order to place attributes to the classes there is a need of a REA model on information level. To model the behavior of REA entities we need additional behavioral description. The text models are made to solve the both problems – supplement the informational REA model and behavioral description.

The text modeling was done in an iteration manner. Each following model is an extension of a preceding one. A successor model uses an ancestor model as a pattern and expands it with new events and operational details. The retail enterprise functioning is gradually specifies in the text models.

The text models are not intended to describe all the details of a retail functioning, but consider the basic economic activities of a retail shop.

The Text model 1 is the first iteration to describe how a retail enterprise functions. This is a most general representation of a retail enterprise. The model simplifies an enterprise value chain and describes two basic economic events Sale and Buy, which are the essence of a retail enterprise. The model establishes a pattern for of the following models.

The Text model 2 introduces Buy Inventory, Buy equipment, Buy Service, Payroll events. The events are similarly to the Buy event from the Text model 1. They are described using the Buy event as a pattern.

These Text models 1 and 2 have to be used for making class diagrams in UML without attributes and for model the behavior with activity diagrams. The Text model 2 will be a basis for a UML class diagram for a retail enterprise.

The Text model 3 extends the Text model 1 with operational details. It presents a simplified paper flow associated with the events.

The Text model 4 extends the Text model 2 with operational details and uses the Text model 3 as a pattern for a simplified paper flow.

These Text models 3 and 4 have to be used for making class diagrams in UML with attributes. The Text model 4 will be a basis for a UML class diagram for a retail enterprise with attributes and operations.

The introduced textual models represent a possible straightforward scenario. Analysis of parallelism between events is not shown in the text models. Later the models can be extended with several scenarios and specified with parallel activities.

Every text model can be used as a pattern in business modeling for the domain of a retail shop enterprise.

The behavior of REA entities also can be presented as state charts and sequence diagrams, but activity diagrams with swim lanes are proposed as more suitable.

Validation of UML diagrams supposes conformity of the UML model of retail shop enterprise to the REA axioms. The axioms are presented in section 2.2.2.

Revenue cycle is treated as an example illustrating how to transform REA model into UML notations. Revenue cycle for a retail shop enterprise has been described in REA notation in section 2.2.3.1. The UML class diagrams and activity diagrams for the revenue cycle of a retail enterprise are explained in section 2.4.3.

All other cycles introduced in section 2.2.3 can be treated the same way.

3.4.2 Text models

3.4.2.1 Text model 1

A retail shop enterprise has two events: Sale and Buy. Sale event aims to sell products from the enterprise to the customer. Buy event fulfils the shop's resources to make it able to perform Sale: the shop buys some goods from a supplier.

The Text model 1 includes Enterprise, who owns Inventory and Cash resources. Enterprise sells products to Customer and buys products from Supplier.

Event: **Sale**

Inside Agent: Enterprise

Outside Agent: Customer

Resources: Inventory, Cash

Activities:

1. Customer orders some products from Enterprise.
2. Enterprise checks Inventory if it is able to satisfy the order.
3. If no, Sale is impossible.
4. If yes, Enterprise decrements Inventory by the ordered products.
5. Enterprise delivers the products to Customer.
6. Enterprise calculates total amount of the delivery.
7. Customer receives the delivery of the products from Enterprise.
8. Customer pays for the delivery to Enterprise.
9. Enterprise checks the payment if it satisfies the total amount of the delivery.
10. If yes, Enterprise increments Cash by the total amount of the delivery.
11. If no, Enterprise processes the payment due.

Event: **Buy**

Inside Agent: Enterprise

Outside Agent: Supplier

Resources: Inventory, Cash

Activities:

1. Enterprise checks if the Inventory level is lower then a certain level.
2. If no, there is no Buy.
3. If yes, Enterprise orders some products from Supplier.
4. Enterprise receives the delivery of the products from Supplier.
5. Enterprise increments Inventory by the delivered products.
6. Enterprise checks if Cash is enough to pay the total amount of the delivery.
7. If yes, Enterprise pays for the delivery to Supplier.
8. Enterprise decrements Cash by the total amount of the delivery.
9. If no, Enterprise processes the payment due.

D.3

3.4.2.2 Text model 2

The Text model 2 specifies Buy with similar events:

- Buy Inventory - the enterprise buys products from a vendor to fulfill the store of the enterprise in order to provide Sale;
- Buy Equipment - the enterprise buys some equipment from a vendor to support activities of the enterprise;
- Buy Service - the enterprise buys some service from a vendor to support activities of the enterprise;
- Payroll - the enterprise pays salary to its employees.

As far as the source REA diagram of the Retail Enterprise does not contain order events connected to Purchase, Equipment Acquisition, General & Administrative Service, Personal Service events, the ordering activity is omitted for Buy Inventory, Buy Equipment, Buy Service, Payroll events in the Text model 2.

The model contains Equipment resource.

Supplier is extended by Vendor who provides products, equipment and services to the enterprise.

Enterprise is extended by employees of the enterprise:

- Salesperson – sells products to customer,
- Shipping Clerk – delivers products to customer,
- Supply Agent – buys products and equipment for the enterprise,
- Supervisor – controls how Vendor supplies service to the enterprise,
- Accountant – assists all monetary operations,
- Employee – any employee of the enterprise.

The model extends the Text model 1 with Finance Transaction event. The model includes a Stockholder agent, who owns capital of the enterprise. Stockholder can invest capital to the enterprise or take capital from the enterprise.

Event: **Sale**

Inside Agents: Salesperson, Shipment Clerk, Accountant

Outside Agents: Customer

Resources: Inventory, Cash

Activities:

1. Customer orders some products from Salesperson.
2. Salesperson checks Inventory if it is able to satisfy the order.
3. If no, Sale is impossible.
4. If yes, Salesperson decrements Inventory by the ordered products.
5. Shipment Clerk decrements delivers the products to Customer.
6. Accountant calculates total amount of the delivery.
7. Customer receives the delivery from Shipment Clerk.
8. Customer pays for the delivery to Accountant.
9. Accountant checks the payment if it satisfies the total amount of the delivery.
10. If yes, Accountant increments Cash by the total amount of the delivery.
11. If no, Accountant processes the payment due.

Event: **Buy Inventory**

Inside Agents: Supply Agent, Accountant

Outside Agents: Vendor

Resources: Inventory, Cash

Activities:

1. Supply Agent receives the delivery of the products from Vendor.
2. Supply Agent increments Inventory by the delivery.
3. Accountant checks if Cash is enough to pay the total amount of the delivery.
4. If yes, Accountant pays for the delivery to Vendor.
5. Accountant decrements Cash by the total amount of the delivery.
6. If no, Accountant processes the payment due.

Event: **Buy Equipment**

Inside Agents: Supply Agent, Accountant

Outside Agents: Vendor

Resources: Equipment, Cash

Activities:

1. Supply Agent receives the delivery of the equipment from Vendor.
2. Supply Agent increments Equipment by the delivered equipment.
3. Accountant checks if Cash is enough to pay the cost of the delivered equipment.
4. If yes, Accountant pays for the delivery to Vendor.
5. Accountant decrements Cash by the total amount of the delivery.
6. If no, Accountant processes the payment due.
7. Once a period Accountant allocates cost of Equipment.

Event: **Buy Service**

Inside Agents: Supervisor, Accountant

Outside Agents: Vendor

Resources: Cash

Activities:

1. Supervisor receives the service from Vendor.
2. Accountant checks if Cash is enough to pay for the service.
3. If yes, Accountant pays for the service to Vendor.
4. Accountant decrements Cash by the payment.
5. If no, Accountant processes the payment due.

Event: **Payroll**

Inside Agents: Employee, Accountant

Resources: Cash

Activities:

1. Accountant checks if Cash is enough to pay salary to Employee.
2. If yes, Accountant pays salary to Employee.
3. Accountant decrements Cash by the salary.
4. If no, Accountant processes the payment due.

Event: **Finance Transaction**

Inside Agent: Accountant

Outside Agents: Stockholder

Resources: Cash

Activities:

1. If Stockholder transfers capital to the enterprise, Accountant receives the transfer amount.
2. Accountant increments Cash by the transfer amount.
3. If Accountant transfers capital to Stockholder, Stockholder receives the transfer amount.
4. Accountant decrements Cash by the transfer amount.

3.4.2.3 Text model 3

The model extends the Text model 1 with operation details. The model includes a simplified paper flow and accounts.

Event: **Sale**

Inside Agent: Enterprise

Outside Agent: Customer

Resources: Inventory, Cash

Activities:

1. Customer orders some products from Enterprise.
2. Enterprise registers the **order**. The order contains information:
    - order number,
    - order date,
    - Enterprise name,
    - Enterprise address,
    - Customer name,
    - Customer address,
    - Customer delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the order.
3. Enterprise checks Inventory if it contains the product name and product quantity recorded in the order.
4. If no, Sale is impossible.
5. If yes, Enterprise decrements Inventory of the ordered product name by the product quantity.

6. Enterprise delivers the products to Customer to the delivery address recorded in the order.
7. Enterprise calculates cost of the delivery.
8. Enterprise produces an **invoice**. Invoice contains information:
    - invoice number,
    - invoice date,
    - order number,
    - Enterprise name,
    - Enterprise address,
    - Customer name,
    - Customer address,
    - Customer delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the invoice.
9. Customer receives the delivery of the products and the invoice from Enterprise.
10. Customer pays for the delivery to Enterprise.
11. Enterprise issues a receipt to Customer. The **receipt** contains information:
    - payment date,
    - Enterprise name,
    - payment amount.
12. Enterprise checks the payment if it satisfies the total amount of the invoice.
13. If yes, Enterprise increments Cash by the payment.
14. Accountant enters the payment amount to the debit side of Cash account and to the credit side of Sales account.
15. If no, Enterprise processes the payment due.

Event: **Buy**

Inside Agent: Enterprise

Outside Agents: Supplier

Resources: Inventory, Cash

Activities:

1. Enterprise checks Inventory product names, if their product quantity is lower then a certain level.
2. If no, there is no Buy.
3. If yes, Enterprise orders the products which product quantity is lower then a certain level. The **order** contains information:
    - order number,
    - order date,
    - Supplier name,
    - Supplier address,
    - Enterprise name,
    - Enterprise address,
    - Enterprise delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the order.
4. Enterprise sends the order to Supplier.
5. Enterprise receives the delivery of the products from Supplier.
6. Enterprise receives an **invoice** from Supplier. Invoice contains information:
    - invoice number,
    - invoice date,
    - order number,
    - Supplier name,
    - Supplier address,
    - Enterprise name,
    - Enterprise address,
    - Enterprise delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the invoice.
7. Enterprise increments Inventory by the delivered product name and product quantity.

8. Enterprise checks if Cash is enough to pay the total amount of the invoice.
9. If yes, Enterprise pays for the total amount of the invoice to Supplier.
10. Enterprise decrements Cash by the payment.
11. Accountant enters the payment to the credit side of Cash account and to the debit side of Purchase account.
12. If no, Enterprise processes the payment due.

3.4.2.4 Text model 4

The model extends the Text model 2 with operation details. The model includes a simplified paper flow and accounts from the Text model 3.

Event: **Sale**

Inside Agents: Salesperson, Shipment Clerk, Accountant

Outside Agents: Customer

Resources: Inventory, Cash

Activities:

1. Customer orders some products from Salesperson.
2. Salesperson registers the order. The order contains information:
    - order number,
    - order date,
    - Enterprise name,
    - Enterprise address,
    - Customer name,
    - Customer address,
    - Customer delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the order.
3. Salesperson checks Inventory if it contains the product name and product quantity recorded in the order.
4. If no, Sale is impossible.
5. If yes, Salesperson decrements Inventory of the ordered product name by the product quantity.

6. Shipment Clerk delivers the products to Customer to the delivery address recorded in the order.
7. Accountant calculates cost of the delivered products.
8. Accountant produces an invoice. **Invoice** contains information:
    - invoice number,
    - invoice date,
    - order number,
    - Enterprise name,
    - Enterprise address,
    - Customer name,
    - Customer address,
    - Customer delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the invoice.
9. Customer receives the delivery of the products from Shipment Clerk and the invoice.
10. Customer pays product cost recorded in the invoice to Accountant.
11. Accountant issues a receipt to Customer. The **receipt** contains information:
    - payment date,
    - Enterprise name,
    - product name,
    - product cost,
    - payment amount.
12. Accountant checks the payment if it is equal to the total amount of the invoice.
13. If yes, Accountant increments Cash by the payment amount.
14. Accountant enters the payment amount to the debit side of Cash account and to the credit side of Sales account.
15. If no, Accountant processes the payment due.

Event: **Buy Inventory**

Inside Agents: Supply Agent, Accountant

Outside Agents: Vendor

Resources: Inventory, Cash

Activities:

1. Supply Agent receives the delivery of the products from Vendor.
2. Supply Agent receives an invoice from Vendor. **Invoice** contains information:
    - invoice number,
    - invoice date,
    - order number,
    - Vendor name,
    - Vendor address,
    - Enterprise name,
    - Enterprise address,
    - Enterprise delivery address,
    - product name,
    - product quantity,
    - product price,
    - total amount of the invoice.
3. Supply Agent increments Inventory by the delivered product name and product quantity.
4. Accountant checks if Cash is enough to pay the total amount of the invoice.
5. If yes, Accountant pays for the total amount of the invoice to Vendor.
6. Accountant decrements Cash by the payment.
7. Accountant enters the payment amount to the debit side of Purchase account and to the credit side of Cash account.
8. If no, Accountant processes the payment due.

Event: **Buy Equipment**

Inside Agents: Supply Agent, Accountant

Outside Agents: Vendor

Resources: Equipment, Cash

Activities:

1. Supply Agent receives the equipment from Vendor.
2. Supply Agent receives an invoice from Vendor. **Invoice** contains information:
    - invoice number,
    - invoice date,

- order number,
- Vendor name,
- Vendor address,
- Enterprise name,
- Enterprise address,
- Enterprise delivery address,
- equipment name,
- equipment quantity,
- equipment price,
- total amount of the invoice.

3. Supply Agent increments Equipment by the delivered equipment.
4. Accountant checks if Cash is enough to pay the total amount of the invoice.
5. If yes, Accountant pays the total amount of the invoice to Vendor.
6. Accountant decrements Cash by the payment.
7. Accountant enters the payment amount to the credit of cash account and enters acquisition cost to the debit side of Equipment Acquisition account.
8. If no, Accountant processes the payment due.
9. Once a period Accountant allocates cost of Equipment.

Event: **Buy Service**

Inside Agents: Supervisor, Accountant

Outside Agents: Vendor

Resources: Cash

Activities:

1. Supervisor receives the service from Vendor.
2. Supervisor receives an invoice from vendor. **Invoice** contains information:
    - invoice number,
    - invoice date,
    - order number,
    - Vendor name,
    - Vendor address,
    - Enterprise name,
    - Enterprise address,

- service description,
- service price,
- total amount of the invoice.

3. Accountant checks if Cash is enough to pay the total amount of the invoice.
4. If yes, Accountant pays the total amount of the invoice to Vendor.
5. Accountant decrements Cash by the total amount of the invoice.
6. Accountant enters the payment amount to the debit side of Service account and to the credit side of Cash account.
7. If no, Accountant processes the payment due.

Event: **Payroll**

Inside Agents: Employee, Accountant

Resources: Cash

Activities:

1. Accountant issues payroll for Employee. **Payroll** contains information:
   - Employee name,
   - Employee job title,
   - Payment date,
   - Salary amount.
2. Accountant checks if Cash is enough to pay salary to Employee.
3. If yes, Accountant pays salary to Employee.
4. Accountant decrements Cash by the salary.
5. Accountant enters the payment amount to the debit side of Personal Service account and to the credit side of Cash account.
6. If no, Accountant processes the payment due.

Event: **Finance Transaction**

Inside Agent: Accountant

Outside Agents: Stockholder

Resources: Cash

Activities:

1. If Stockholder transfers capital to the enterprise, Accountant receives the transfer amount.
2. Accountant increments Cash by the transfer amount.
3. Accountant enters the transfer amount to the debit side of Cash account and to the credit side of Capital Transaction account.

4. If Accountant transfers capital to Stockholder, Stockholder receives the transfer amount.
5. Accountant decrements Cash by the transfer amount.
6. Accountant enters the transfer amount to the debit side of Capital Transaction account and to the credit side of Cash account.

3.4.3 UML diagrams

UML diagrams represented in the Appendix E.1 explain the Revenue cycle of a retail shop enterprise. Revenue cycle has been described in REA notation in section 2.2.3.1. This cycle corresponds to a Sale event introduced in the text models in section 2.4.2.

The diagrams presented have an objective to illustrate some possible variants of transforming REA model into UML notation. The complete modeling of a case was not intended.

The following class diagrams and activity diagrams have been constructed:

- Appendix E.1.1 includes a class diagram for Revenue cycle. This diagram is a 'direct' translation of the REA diagram represented in Fig.2.2.3.1. The REA entity sets are shown as stereotype classes, relationship sets are modeled as stereotype associations. Roles of classes in the associations correspond to entity sets' roles.

  Stereotypes are used simply to mark the classes and associations. They explain the meaning of these of elements in a short way. But stereotypes should be carefully modeled and supported by invariants. Especially it's difficult to model stereotyped associations. So this approach is under consideration. The problematical aspects and benefits of using stereotypes in business modeling are treated in chapter 6.1.1.

  This representation is easy to read and understand but it is not informative to explain the retail shop enterprise functioning. Only external agent – Customer – is shown at the diagram. This diagram can be used as a part of high-level conceptual model.

- Appendix E.1.2 contains a refinement of the pervious class diagram. The source is the Text model 3, section 2.4.2.3. The class diagram illustrates a Sale event.

  Entity sets are modeled as classes with stereotypes. Internal and external agents are represented. The diagram helps to understand spheres of business activity of the agents.

  Resources are modeled as compositions of resources' lines. The same approach is applied for events. It helps to model the essence of economical events more carefully. Thus a *stock-flow* relation is modeled as an association between a resource line and an event line. A *duality* relation is shown as an association between one event line and another event line.

  This class diagram can be refined further and extended with attributes and operations. The source can be the Text model 4.

- Appendix E.1.2 represents an activity diagram for the Sale event. The basis for the diagram is the Text model 1. The diagram illustrates activities of the external agent Customer and the internal agent Enterprise in the Sale event. The Sale event has three states: ‘Sale is registered’, ‘Sale is delivered’, ‘Sale is paid’.
- Appendix E.1.2 contains a refinement of the pervious activity diagram. The source for the diagram is the Text model 3. The Enterprise is specified to Salesperson, Shipment Clerk and Accountant internal agents.

  These activity diagrams can be used for refinement class diagram represented in Appendices E.1.1, E.1.2 for setting up operations for the classes.

  These activity diagrams can specify a paper flow supported business processes. A state ‘Sale is registered’ can correspond to issuing an invoice.

  These activity diagrams can be extended with parallelism between activities. For example, delivery and payment of the sale may be done concurrently. The analysis of parallelism requires more information on the domain.

4 Business enterprise and enterprise solution

Today's business environment creates a large demand for software applications to perform management solutions for business enterprises – enterprise solutions.

Examining recent enterprise strategies and enterprise solutions indicates the new impact they make to software engineering process. **The process tends to shift from software engineering to modeling engineering.**

The chapter has the following objectives:

- analyze the present enterprise solutions and their strategies,
- signify the tendencies in software engineering related to present enterprise solutions development.

Chapter 3.1 establishes a base for modeling an enterprise and enterprise solution. The chapter starts with definition of a business enterprise and explains general aspects of enterprise functioning:

- Section 3.1.1 introduces Porter's value chain for distinguishing business activities of an enterprise according to their impact into value creation process. Business processes and actors are explained in connection with enterprise's business activities.
- Section 3.1.2 describes enterprise resources which are grouped as tangible, intangible and human.
- Section 3.1.3 suggests level-oriented approach for description of enterprise functioning.
- Section 3.1.4 names possible enterprise stakeholders.
- Section 3.1.5 introduces the organization of an enterprise as a social structure. The organization contains policy, culture and structure.

Chapter 3.2 explains the artifact of enterprise solution and defines its role in enterprise functioning. The chapter shortly covers enterprise solution architecture and recommends the standard IEEE Std. 1471-2000 as a conceptual framework for enterprise solution architecture. The chapter includes the following parts:

- Section 3.2.1 puts requirements for enterprise solutions. The requirements are structured as functional, non-functional and domain requirements.
- Section 3.2.2 considers enterprise solution a software-intensive system and briefly describes possible variations of enterprise solution architecture using tiring approach.
- Section 3.3.3 discusses the standard IEEE Std. 1471-2000 for software-intensive systems. The standard is proposed to be used for enterprise solution architecture. Possible advantages and disadvantages of the standard are named.

4.1 Business enterprise

Business enterprise is a social organization performing production and trade of products or services based on a private ownership of wealth. The term *enterprise* is used further in the text instead of business enterprise.

Products and services producing and trading by an enterprise are *goods*. The goods have some monetary significance or *value*.

4.1.1 Business processes

The monetary value created by an enterprise is a result of activities and processes inside and outside an enterprise. The Michel E. Porter's *value chain framework* defines the value creation process. It is the most widely known model serving as a tool to identify ways to increase customer value.

The value chain framework is illustrated at the Fig.3.1.1. The value chain can be split into a sequential chain of activities relevant for en enterprise. The Porter's value chain framework identifies nine activities. They are distinguished as primary activities and support activities.

Primary activities directly participate in the enterprise value creation and trade. Primary activities come up as a sequence of:

- *inbound logistics* is supplying an enterprise with materials needed for making products,
- *production operations* is making products,
- *outbound logistics* is delivering products to customers,
- *marketing and sales* is finding customers and trading products to customers,
- *customer service* is facilitating and supporting of the marketing and sales.

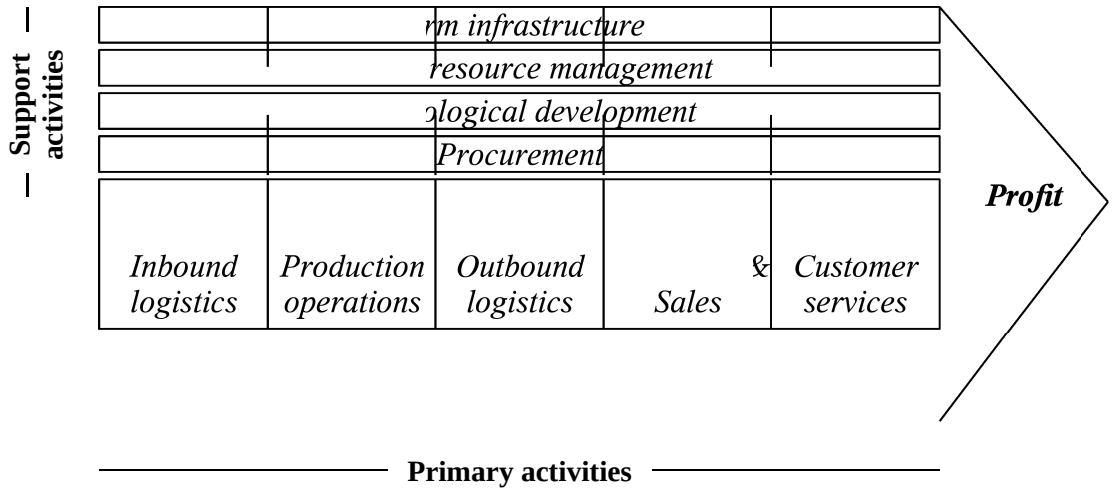


Figure 3.1.1. *Porter's value chain framework*

Support activities participate indirectly in the value creation and trade. They provide infrastructure for primary activities and support all activities of the value chain:

- *firm infrastructure* includes activities that are required for all primary and support activities:
    - *management* is running an enterprise,
    - *planning* is setting up arrangements for enterprise's activities,
    - *finance* is providing funds and capital for investment in an enterprise,
    - *accounting* is keeping information about all payments to and from an enterprise,
    - *legal affairs* is assisting an enterprise in contacts with legal authorities and providing legal basis for enterprise functioning ,
    - *government affairs* is assisting an enterprise in contacts with government authorities and political units,
- *human resource management* is supplying an enterprise with people needed to perform in the enterprise's activities,
- *technological development* is developing goods and improving activities,
- *procurement* is supplying an enterprise with materials and technical devices for its functioning.

Both primary and support activities participate in the value creation. As a result of the functioning of all the activities in the value chain framework, an enterprise gains profit. *Profit* is a difference between the value received by an enterprise as a result of value chain activities and the value expended by an enterprise for value chain activities.

Enterprise's activities imply a number of actions. An *action* is a unit of work to be done by an enterprise. Enterprise's activities may be described by business processes. *Business process* is one or more actions and sub-processes performed by an enterprise. Business processes can be hierarchically divided according to levels of detail into sub-processes.

A business processes involve actors. An *actor* is an individual or a group of individuals who participate in the processes. An *internal actor* is the actor located in an enterprise. An *external actor* is the actor located in an enterprise's environment.

An actor plays different roles in the business processes. A *role* reflects the functions an actor has in a business process.

The task of an enterprise is to examine efficiency of each activity and to find the ways to improve it. The most effective way to do it is to distinguish and examine the core business processes. C*ore business processes* are business processes bringing the most contribution into the value creation.

4.1.2 Enterprise resources

In order to create value, an enterprise needs *resources.* Resources reflect capacity of an enterprise to undertake a particular activity. Resources can be grouped into tangible, intangible and human:

- *Tangible resources* include financial and physical assets of an enterprise:

- cash,
- raw materials, semi-products, products,
- machinery and equipment,
- buildings,
- natural resources property as plants and land.

These resources can be quantified and monetary estimated;

- *Intangible resources* include:
  - good will containing the enterprise 's reputation, brand recognition, and relationships with enterprise environment and customers,
  - intellectual property including trade secrets, patents, know-how, managerial innovativeness, information on an enterprise and enterprise environment,
  - copyrights containing exclusive rights to a literary, dramatic, musical, or artistic work or to the use of a commercial print or label.

  These resources are difficult to quantify and value;
- *Human resources* are people needed by an enterprise:
  - employees,
  - external consultants,
  - 'business partners'.

  They are actors in business processes. These processes are to create value with tangible and intangible resources.

Today the value of intangible resources may exceed the value of tangible resources. Intangible as well as human resources play a key role in performance of an enterprise.

4.1.3 Levels of enterprise functioning

Concerning enterprise functioning, the *level-oriented approach* has been chosen. An enterprise may be viewed in a perspective of strategic, tactical and operational levels of enterprise functioning:

- *Enterprise strategy* is planning in advance how to gain success in enterprise activities. It is a set of enterprise goals. Enterprise strategy is a plan for future activities answering the question "What to do?" in order to achieve success in enterprise functioning. *Enterprise strategic level* concerns with enterprise prospects and related activities of enterprise functioning.
- *Enterprise tactic* is a plan or method how to gain a desired result. It is a set of methods to accomplish the enterprise strategy. Tactic answers the question "How to do?" so as to reach the goals. *Enterprise tactical level* deals with practical concrete directions and related activities realizing the enterprise strategy.
- *Enterprise operational level* relates to day-to-day routines of enterprise functioning. Operational routines follow the directions of enterprise tactic.

The levels of enterprise functioning are related to *time span*. The levels' hierarchy can be set up in time dimension. Strategic level refers to functioning of an enterprise in a long time period, tactical level refers to a medium period, and an operational level refers to a short period.

Each level of enterprise functioning contains plans and methods.

Plan is an arrangement to carry out some activity. Plan has some steps and ordering relations between steps. A *step* may be an action and the resources needed by the action. A *schedule* is a plan with time points for the start and optionally for the end of steps.

Method is a description of connected actions necessary to perform a business task.

The levels of enterprise functioning have different influence on enterprise efficiency. The influence can be distinguished as a direct influence and a feedback influence:

- The *direct influence* shows the contribution of the strategic, tactical and operational levels to enterprise performance. An upper level makes influence on a level below. The strategic level controls the tactical one, which controls the operational level;
- The *feedback influence* shows response of an enterprise - the output of enterprise performance which affects the levels concerned. The feedback influence comes from a lower level to an upper one. Enterprise performance result at a lower level is a source for adjustments at an upper level.

Every enterprise has a hierarchical set of aims: missions, goals and objectives. These aims determine the strategic and tactical directions of enterprise activity:

- *Mission* is a set of possibilities for an enterprise. It is an overall dream that provides the direction to an enterprise for many years. Mission is guided by a vision on what an enterprise is. It focuses on a limited number of goals. Mission gives a sense of main purpose, direction and opportunity of an enterprise and stresses the major value that the enterprise wants to honor;
- *Goal* describes realistic directions that are specific with respect to magnitude and time. Goal has some quantified parameters for mission. Goal indicates what an enterprise wants to achieve. It refers to a strategic level of enterprise functioning;
- *Objective* is a concrete task for a defined period. Objective is quantitative and precise. It refers to a tactical level of enterprise functioning.

The purpose of an enterprise is to make a profit by the creation of value that customers are willing to pay. This purpose supposes a collection of missions, goals and objectives of the enterprise. Missions, goals and objectives sets can have a hierarchical structure according to their importance for enterprise functioning.

Primary activities of an enterprise functioning described in the section 3.1.1, may be identified from enterprise goals. Support activities more relate to objectives.

4.1.4 Stakeholders

En enterprise has *stakeholders*: enterprise property owners, customers, employees, suppliers, distributors, retailers and other individuals and enterprises with whom the enterprise has mutually profitable relationships.

4.1.5 Organization

An enterprise has an organization – a group of people with a purpose of an enterprise's business. Organization is related to the management activity in the Porter's value chain. Organization supposes three elements:

- organizational policy,
- organizational culture,
- organizational structure.

4.1.5.1 Organizational policy

Organizational policy defines a general scope and directions on how an enterprise deal with stakeholders, with environment, on the range of the enterprise's products, industries in which the enterprise operates, geographical scope of enterprise activities and other main business topics. The aim of the organizational policy is to allow an enterprise's employees act consistently on important issues.

4.1.5.2 Organizational culture

Running business, an enterprise accumulates some experience on internal operations and cooperation with outside world and related people's behavior. The experience forms *organizational culture* – a collection of shared stories, beliefs, experiences and norms. It is a set of non-formal behavioral rules and values which increase the efficiency of the enterprise functioning.

4.1.5.3 Organizational structure

People employed in an enterprise are committed to participate in business processes and perform some functions. These functions determine job roles and compose a set of job responsibilities for every job role. The distribution of human resources according to their functions is mapped to o*rganizational structure.*

Organizational structure is a composition of job roles within an enterprise. Several job roles may be associated in groups (departments or divisions) and sub-groups in order to gather people performing the same task. Organizational structure maps the relations between divisions and job roles.

The divisions compose a hierarchical structure, which reflects the levels of enterprise functioning. A division has managers who lead the division at a corresponding level. Top-level managers work at the strategic level, medium-level managers perform at the tactical level, low-level managers deal with the operational level.

4.1.6 Workflow

Workflow is a set of procedures which reflects enterprise functioning at the operational level.

Workflow determines:

- the order of operations,
- links between business processes,
- links with actors of business processes who belong to the organizational structure and environment,
- shows information flow within the organization and between the organization and environment.

4.1.7 Environment

Environment is a reality external to an enterprise. An enterprise is a self-dependent unit producing and trading value, but it is relevant only in cooperation with environment. Environment creates a demand for value producing by an enterprise and consumes this value. The environment puts constrains on enterprise activities.

Organizations and people in outside environment are of great importance for an enterprise. They participate directly or indirectly in an enterprise's business. The role-oriented approach can be used to group the people and organizations according to their meanings for an enterprise. They can be grouped into categories called *parties:*

- *Customers* are individual people, enterprises and non-business organizations willing to pay for product produced by an enterprise. Customers create demand for the product and buy it. An enterprise acts as a *supplier* of the product. An enterprise tries to reach the demand and sell the product to customers;
- *Business partners* are individuals or organizations cooperating with an enterprise. They help each other to create value in order to receive a mutual benefit.

  There are two main groups of business partners: suppliers and distributors.

  Suppliers provide an enterprise with goods or services in order to supply with resources needed for the enterprise functioning.

  Distributors deliver the products or services produced by an enterprise to its customers.

  With development of Internet technology and electronic commerce, a new paradigm appeared: Business-to-Business (B2B) collaboration. It concerns customers also as partners (trade partners);
- *Competitors* are individuals or organizations creating the same value as an enterprise. Competitors attract enterprise's customers and by that reduce the ability to sell products of the enterprise. Enterprise tries to achieve a competitive advantage – to make significant contribution to perceived customer benefit, which has a big potential and is difficult to imitate;
- *Regulation authorities* are government and government organizations (taxation, legislation, custom, administration and other authorities) and non- government

organizations which establish rules and guidelines for business and other activities within the social life. The regulation authorities affect enterprise business processes as well as customers' demand.

4.1.8 Management decisions

Management choice is a process of selecting from a number of alternative actions. The result of management choice is a *management decision.* The goal of the choice is to increase the value created by an enterprise.

The choice is made on different levels of enterprise functioning:

- the strategic choice affects enterprise strategy, missions and goals;
- the tactical choice affects enterprise tactics and objectives;
- the operational choice makes influence on enterprise operations.

The input for management choice is information and knowledge about the aspects of enterprise functioning:

1. customer demand,
2. environment,
3. business processes of an enterprise.

The choice is guided by some rules of business management.

Management decision contains instructions regarding:

- value being created by an enterprise,
- resource allocation,
- business processes optimization,
-

- organizational structure,
- relations with environment.

Management decision initiates an iteration process from the three stages:

- *planning* is a management activity of setting goals and creating a plan on how to achieve that goals at a corresponding level of enterprise functioning.
  - The planning activity implies *scheduling* sub-activity, which specifies plans with time constraints,
- *implementation* is execution of the plan,
- *control* is comparing the execution results with goals set in the plan.

If the goal hasn't been achieved, the plan is to be adjusted. Management activity includes decision making, planning, scheduling and control.

Management solution is a combination of management decisions on organizing and running an enterprise.

4.2 Enterprise solution

The intangible resource *information* is crucial to the effective management of an enterprise. Information reflects changes inside an enterprise and in environment, helps to form knowledge, predict future situations and take management decisions. Without quality information, wrong management decision may be made and knowledge acquisition may not take place. As a result, there may be loss of valuable resources as money, profit, time, business relationship and so on.

In view of great importance of qualitative and timely information for operation an enterprise, there is a need for a powerful software application allowing an enterprise's employees, customers and partners to access the information and empower them to take informed decisions in their business activities.

The software application should provide complete process automation. A major emphasis is placed on control the processes. The application must provide analytics, control and planning (which are future control). But this kind of application is not only a tool to facilitate human activities. In some cases it should replace human actions. The application should 'double an enterprise', that is to deliver business capabilities. A term *virtual organization* is most appropriate for this application. In other words, the application must perform a management solution for an enterprise. Such applications are called enterprise solutions.

Enterprise solution can be viewed as a software system solving a problem for a single user type. Enterprise solution is a tool for:

- making management solutions,
- process optimization (process reengineering),
- manage enterprise's information,
- performing business activities,
- representing better quality and speed of the work.

Enterprise solution functions on the operational level of an enterprise.

General requirements for an enterprise solution may be formulated as follows:

- capturing all business activities of an enterprise, including management decisions,
- providing collaboration with business partners,
- being accessible at any place, any time and from any device.

4.2.1 Enterprise solution requirements

An enterprise solution has *stakeholders*. They are any individuals and organizations being interested in or affected by the enterprise solution. The stakeholders participate in the *requirements elicitation* for an enterprise solution. This is a process through which customers and developers of an enterprise solution determine the requirements for an enterprise solution.

Enterprise solution requirements define services expected from an enterprise solution and constraints for the enterprise solution. A s*ervice* is a function a software system has to serve. A service gives some benefit to the enterprise functioning. A *constraint* is a restriction which limits a service.

Enterprise solution requirements can be grouped as functional, non-functional and domain requirements.

4.2.1.1 Functional requirements

Functional requirements name the services or functionality that an enterprise solution should provide. They describe the value that an enterprise solution brings in the value created by an enterprise.

Functional requirements specify processes which have to be implemented by enterprise solution, input information accepted by an enterprise solution, output information expected from an enterprise solution and data held by an enterprise solution.

Functional requirements for an enterprise solution will include:

- description of business processes which an enterprise solution has to carry out,
- description of information inputs into an enterprise solution
  - from documents participated in an enterprise's workflow,
  - from interaction between actors of the business processes. The actors' interaction is described in the enterprise's workflow and organizational structure. The actors can belong to an enterprise's organizational structure and to the environment,
  - from other software systems interacting with an enterprise solution,
- description of information outputs expected from an enterprise solution in the form of
  - printed documents, reports and displays satisfying an enterprise's workflow and interaction between actors,

  - transfers to other software systems interacting with an enterprise solution,
- description of data that that should be held by an enterprise solution.

4.2.1.2 Non-functional requirements

Non-functional requirements also called *operational requirements*, describe features of an enterprise solution concerning the quality of functional requirements provided by the enterprise solution. These features are constraints to an enterprise solution.

The following operational requirements are especially important for an enterprise solution:

- *performance* defines how fast the enterprise solution must execute the tasks and how efficient hardware utilization is,
- *security* confirms that an enterprise solution follows a security policy of an enterprise,
- *scalability* is the ability of an enterprise solution to perform under increased number of users, the amount of data and number of transactions being managed,
- *interoperability* how the enterprise solution will interact with systems in other organizations,
- *availability* defines how long a user is waiting for a response from an enterprise solution to the user's request,
- *reliability* settles up the acceptable failure rate,
- *manageability* claims that an enterprise solution must have tools to manage the solution,
- *maintainability* is the ability of an enterprise solution to be repaired and maintained easily and fast.

4.2.1.3 Domain requirements

Domain requirements come from an application domain. The domain requirements for an enterprise solution should reflect fundamentals of a business enterprise domain. The domain requirements combine both functional and non-functional requirements for an enterprise solution.

4.2.2 Enterprise solution architecture

An enterprise solution is a software system. A *software system* includes programs, configuration files for setting up these programs, system documentation describing a structure of the system, user documentation explaining how to use the system.

En enterprise solution consists of sub-systems and components.

Sub-system is independent from services provided by other sub-systems. It is composed from modules and has interfaces for cooperation with other sub-systems. A *module*

supplies other modules with one or more *services* and depends on services provided by other modules. A software model may be composed from other modules.

A *component* is an executable software module with an interface for communication with other components.

Enterprise solution architecture describes the organization of an enterprise solution, its sub-systems and components and relationships between them. It is based on the enterprise solution requirements.

Enterprise solution architecture is of great importance for an enterprise solution. It defines directions for design and whole evolution of an enterprise solution. The cost of development of software and growing complexity of enterprise solutions requires decisions about an enterprise solution:

- overall design structure,
- goals,
- requirements,
- development stages.

These decisions have to be made at the early stage of enterprise solution development – *architectural design.* This is a process to define a collection of hardware and software components, their interfaces and establish the structure for the development of an enterprise solution.

Enterprise solutions architecture has been changed in connection with development of technology and business needs. From the technology point of view these changes can be classified according to the number of tires in the architecture.

In the **centralized architecture based on mainframes** all the devices in the system have a central administration. All transactions, data checking, computations, navigation and minimal user interface are located on a mainframe.

In the **client-server architecture** a database level is separated from a client. Business logic is distributed between the levels. Transactions, data checking, computations are executed at the database level. The client level performs transactions, computations checking, navigation and user interface functions. This architecture gives more elaborated user interface, more interactivity and possibilities to set up the system according to users needs.

Multi-tired Web applications have three levels:

- data level (data services),
- middle level (business logic services),
- client level (presentation services).

Usually a client level is realized as applications for a Web browser or GUI-applications with some functions for data checking and computations, navigation and UI elements.

A middle level is realized as a Web application server. It keeps transactions, checking, computations and navigation logic. There are possibilities for dynamic generating of client elements and setting up the system.

A data level is realized as a DBMS server. It performs data storage and has minimal resources for transaction management and data checking.

Generally modern enterprise solutions have multi-tired architecture based on XML in the middle level. These systems are similar to the multi-tired Web applications architecture. The difference is the middle level. It consists of the two parts:

- XML Web services level (transaction server),
- UI services level (interaction server).

Transaction server executes several tasks: generating a client part, B2B processes automation, transactions, data checking, computations and settings logic of a client part for concrete clients.

The main advantage of the system is that a client part can be located at the different devices connected to the Web: PC and various smart devices (mobile phones, pocket PC, PDA, GPS etc.). The client level can execute data mapping logic, computations, data checking, and navigation.

The main weakness of this architecture is a need to support different communication highways. The solution is to use a concrete method for every type of devices. This approach can be realized as **a platform for mobile solutions**. It has three basic parts:

- clients applications,
- interactive application platform,
- service platform.

Every mobile device has a browser as a client application. A service platform contains applications, Web servers and services. An interactive application platform is a middle level between clients and a service platform. It supports standards of the clients and helps to perform solutions on different devices.

An example of this solution is the Microsoft .Net platform presented in the next chapter (section 4.2).

4.2.3 Architectural standard IEEE Std 1471-2000

Recently some efforts have been made to define a standard for architectural description of software systems. One of the attempts is the standard IEEE Std1471-2000, IEEE's Recommended Practice for Architectural Description of Software-Intensive Systems. The standard was developed by the Institute of Electrical and Electronics Engineering, Inc. (IEEE). This is a non-profit technical professional association. It leads in many technical areas including computer engineering.

IEEE Std1471-2000 is a formal standard defining architectural description and a conceptual framework for it for software intensive systems. The standard is not meant for obligatory use and has to be updated according to the changes in computer technology.

The standard defines a *software- intensive system* as any system where software makes a considerable impact into the system life cycle.

An enterprise solution can be considered as a software-intensive system. Therefore IEEE Std1471-2000 may be applied as a technique for the enterprise solution architectural description. The figure 3.2.3 is taken from the standard IEEE Std1471-2000 [29]. It represents a conceptual model of architectural description.

4.2.3.1 Concepts

The standard presents a *conceptual framework* for architectural description which is a collection of terms and concepts for architectural design. The considered concepts are:

- *System* can be any system. A system is a purposeful aggregation of interrelated components working together in order to achieve some objectives. Examples are a software system, a part of a software system, an enterprise.
- *Context* is an *environment* of a system. It includes any details of reality which is outside to a system. An environment may contain another system interacting with the system.
- *Stakeholder* is a person or organization having an interest or interests in a system.

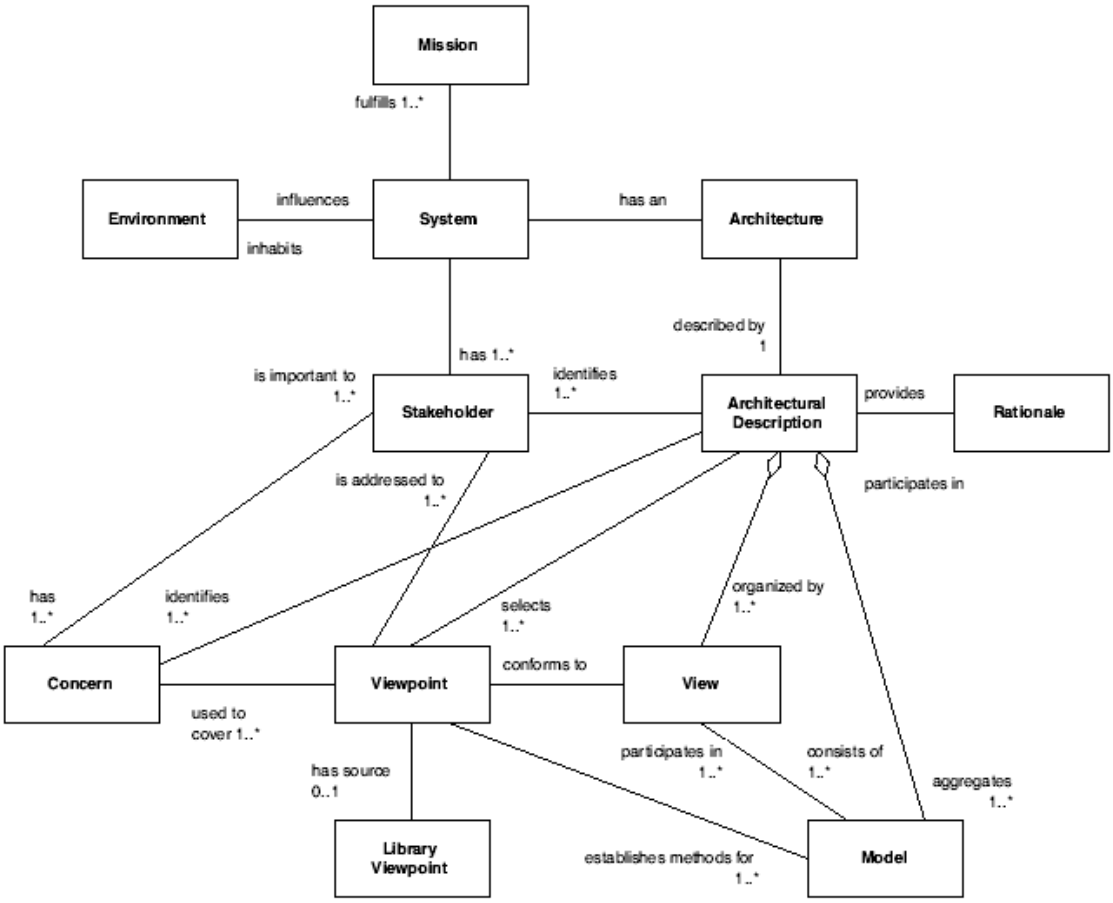
Stakeholders have different roles regarding their use of architectural descriptions.

The roles are:

  - a client who is an acquire of a system,
  - a user who use a system,
  - an architect who create an architectural description of a system,
  - a developer who develops a system,
  - an evaluator who estimates a system.

- *Concern* is an interest of a stakeholder regarding a system.

- *Mission* is a main goal of a system. Stakeholders expect a system to accomplish its mission in order to meet a set of the system's objectives.



5.B.3.1.1

Figure 3.2.3. *Conceptual model for architectural description*

- *Architecture* is an abstraction of a system. It reflects organization of a system:
    - components aggregated in a system,
    - relationships between this components,
    - relationships between a system and its environment,
    - rules governing system life cycle.

  Architecture is a structured set of concepts.

- *Architectural description* is a set of products and artifacts for documenting an architecture.

Architectural description may be used for:

- description of a system and its life cycle,
- communication among stakeholders,
- evaluation of architectures,
- system development management and execution,
- system change management,
- verification of a system's implementation with its architectural description.

- *View* is an **actual representation of a whole system** attended to a set of particular concerns, which cover a single perspective.
- *Viewpoint* is a **representation framework for a view**. This is a set of conventions for constructing, interpreting and analyzing a view:
  - languages – notations, model, product types,
  - analysis techniques,
  - modeling methods.
- *Library viewpoint* is a source of a viewpoint, if the viewpoint is not originated in the architectural description. A library viewpoint may include heuristics and patterns. These are assisting guidance on how to construct a view.

  Pattern, also called template, represents one set of concerns relative to an architecture.
- *Architectural model* is a part of architectural description. A model provides a content of an architecture. An architectural model is associated with a viewpoint. A model is developed by using conventions of the associated viewpoint.
- *Rationale* is an explanation of use of an architecture description. A rational explains the selection of viewpoints with regard to stakeholders and their concerns.

Stakeholders, concerns and viewpoints are reusable elements. They may be used in architectural descriptions in various systems.

4.2.3.2 Relationships

The relationships and constraints for the relationships between the concepts in the conceptual framework can be expressed by the clauses:

1. A system has one or more stakeholders.
2. A system fulfills one or more missions.
3. A system inhabits from an environment.
4. A system has an architecture.
5. An environment influences a system.
6. An architecture is described by an architectural description.
7. An architectural description provides a rationale.
8. An architectural description identifies one or more stakeholders.
9. An architectural description identifies one or more concerns.
10. An architectural description selects one or more viewpoints for use.

    Constraints:
    The selection is based on the decision of stakeholders to whom the architectural description is devoted and concerns of these stakeholders.

11. An architectural description is organized by one or more views.
12. An architectural description aggregates one or more models.
13. A stakeholder has one or more concerns.
14. A concern is important to one or more stakeholders.
15. A viewpoint is used to cover one or more concerns.
16. A viewpoint is addressed to one or more stakeholder.
17. A viewpoint has source in one or zero library viewpoint.

    Constraints:
    If a viewpoint is not originated in the architectural description, it has a source in a library viewpoint. Otherwise a viewpoint has no a source.

18. A viewpoint establishes methods for one or more models.
19. A view conforms to a viewpoint.

    Constraints:
    A view is well-formed with respect to its viewpoint:
    - Each view in an architectural description corresponds to exactly one viewpoint.
    - Each view in an architectural description conforms to the specification of the corresponding viewpoint.
    - Each view addresses one or more concerns of stakeholders.

20. A view consists of one or more models.

21. A model participates in a view.

22. A model participates in an architectural description.

    Constraints:
    A model uses methods of the associated viewpoint.

Views may be used for completeness and consistency checking.

An architectural description is *complete* if it addresses all stakeholders' concerns. It means that a set of selected views must meet all the identified concerns of all identified stakeholders.

An architectural description is *consistent* if the selected views are consistent with each other.

4.2.3.3 Documentation

Every architectural description will include the following elements:

- identification of an architectural description, version, and overview information containing date of issue and status, issuing organization, change history, summary, scope, context, glossary, references;
- identification of the system stakeholders: users, acquires, developers and maintainers;
- identification of the system stakeholders concerns related to the architecture: missions of the system, feasibility of creating the system, suitability of using the system to achieve its missions, the risk of the system development and functioning for the system stakeholders, how the system should be maintain and deployed;
- specification of the selected viewpoints: a viewpoint name, stakeholders being addressed by a viewpoint, concerns being addressed by a viewpoint, language, modeling techniques, analytical methods being used to construct a corresponding view, source for a library viewpoint;
- rationales for the selected viewpoints about stakeholders and their concern covered by a viewpoint;
- architectural views: a view identifier, representation of the system constructed with the languages, modeling techniques, analytical methods of an associated viewpoint, configuration information;
- records about inconsistencies among the architectural description of the system: inconsistency among views, analysis of consistency for all views;

- a rationale for selected architecture: a rationale for selected architectural concepts, a rationale for alternative concepts.

4.2.3.4 Activities

To produce an architectural description, the following activities should be taken:

- identification of stakeholders,
- identification of stakeholders' concerns,
- selection of viewpoints and formulation of a rationale for the selection. This activity supposes a choice of languages, techniques and models to describe a system behavior,
- representation of a resulting architecture through views. A view must conform to a previously defined viewpoint,
- listing inconsistencies among views.

4.2.3.5 Problematical aspects

The standard is on its development stage. There are a number of open problems to be solved in the future. Some of the problems are named below.

The intention of reusable viewpoints is to ease a process of creating architectural descriptions. Ideally the process must be reduced down to the identifying stakeholders, their concerns and choosing appropriate representational framework – reusable viewpoints from a library viewpoint. But now only little knowledge is captured in the patterns. There is a need for unified repository for reusable viewpoints.

Another topic to be developed is view checking. There is no clear guidance in the standard on how to check and analyze views. The unified techniques are needed for this purpose.

During a process of inter-view consistency checking, is a multiple-view integration problem must arise. It is hardly to require that views should not overlap. The views have different notations but describe overlapping issues. For example, to describe an architecture of a software system, an architect may chose the viewpoints: conceptual, logical, physical. But a set of concepts for a conceptual view should be a basis for a logical and physical views, a set of concepts for a logical view should be a basis for a physical view.

The standard tries to capture the system, its structure and its environment in one conceptual framework. But in the development process this framework is of not much use. The main problem is that the framework do not consider system life cycle. It is mentioned in the standard: "It has long been recognized that "architecture" has a strong influence over the life cycle of a system." But the standard treats that problem only as possible scenarios. There is no explicit recommendation on how exactly to use the

framework. The standard should provide a connection with the system life cycle at the conceptual level.

Another problem is how to bind the meta-model represented by the standard to the development of a concrete software system. There should be recommendations on what are the possible specializations of the concepts and how and when to use them in their interdependences. So there is a need for guidelines what concrete parts of the model to use depending on the context.

The problem of the standard conceptual framework interpretation is treated in [26].

The standard is not obligatory to use, informal and opened for suggestions and improvements. Formalization and unification of the standard can help to solve the mentioned problems.

5

6 Microsoft enterprise strategy

The *enterprise strategy* is based on vision and determines main principles, approaches and directions in development of enterprise solutions. The success of enterprise solution highly depends on its vision and strategy. Vision designates the enterprise solution artifact and explains its role in enterprise and enterprise environment.

Microsoft enterprise strategy aims to support business solutions for large-scale as well as medium size and small enterprises. Microsoft enterprise strategy is system-oriented.

The chapter covers the present Microsoft enterprise strategy approach implemented by the family of Microsoft .Net products.

Enterprise solution architecture plays a key role for enterprise strategy. It determines directions for design and evolution of an enterprise solution. In chapter 4.1 the main concepts of the present Microsoft enterprise solution architecture are described. The Microsoft enterprise solution architecture is briefly compared with the Standard IEEE Std.1471-2000.

Chapter 4.2 gives a short introduction to the Microsoft .Net platform. Microsoft .Net operates at the operational level of enterprise functioning.

6.1 Microsoft enterprise solution architecture concepts

Enterprise solution deals both with business enterprise concepts, described in the section 3.1, and information technology supporting an enterprise functioning. The information is reflected in the enterprise solution architecture. *Enterprise solution architecture* is a tool to describe organization of an enterprise solution at the conceptual level.

The chapter represents the most important concepts of the Microsoft enterprise solution architecture.

The source for the chapter is [28] found in the online MSDN library.

6.1.1 Enterprise solution architecture scope

The scope of the Microsoft enterprise solution strategy captures information on an enterprise and its functioning. The scope of the strategy must cover all the levels of enterprise functioning – strategic, tactical and operational. The levels have been described in the section 3.1.3.

The topics of information relevant for the enterprise strategy are called *concerns*.

A concern expresses an interest of one or more *enterprise solution's stakeholders* – persons who have any influence on an enterprise solution: end-users, engineers developing an enterprise solution and related systems, managers, enterprise domain experts and anyone in the environment being affected by an enterprise solution. A stakeholder has one or more concerns.

The following concerns are in the scope of the Microsoft enterprise strategy:

Microsoft enterprise solution architecture concepts

- *goals and objectives* include guide information for enterprise functioning,
- *business processes and organization* include information on business processes, involved resources and actors, workflow, organizational structure scheme and job role descriptions, principles and rules on relations with customers and business partners, competitors and regulation authorities,
- *systems and data* include information on software systems and database management systems (DBMS), used by an enterprise,
- *technology* contains information on hardware used by an enterprise.

These concerns have different levels of detailed elaboration covering strategic, tactical and operational levels of enterprise functioning. The concerns are related with each other and should be considered as parts aggregated into a whole complex system. The meaning of these parts for the enterprise strategy is in their interdependency and entire structure.

The information on an enterprise may be considered from different *perspectives.* A perspective reflects a stockholder's viewpoint. It is a stockholder's unique vision on how to achieve a business goal by the enterprise solution. The Microsoft enterprise strategy groups the possible viewpoints into four perspectives:

- *the business perspective* explains an enterprise's business,
- *the application perspective* describes software applications supporting the enterprise functioning,
- *the information perspective* defines the data needed for the enterprise functioning,
- *the technology perspective* refers to hardware and software supporting the enterprise functioning.

The concerns are used to cover one or more stakeholders' perspectives for an enterprise solution (Fig.3.2.1). The concerns are described below.

6.1.1.1 Goals and objectives

The Goals and objectives concern describes the directions in enterprise functioning on the strategic and tactical levels and place a guideline for development of an enterprise solution:

- enterprise's goals and objectives for business performed by an enterprise;
- data management policies explain principles and directions of storing and retrieving data necessary for enterprise functioning;
- prospects and plans for developing new or improving present software systems to support goals and objectives in the enterprise's business.

The described concern is addressed to a business, application and information perspectives.

6.1.1.2 Business processes and organization

Information on main and support activities of an enterprise and its organization defines functionality for enterprise solution. The Business process and organization concern is used to cover the business and information perspectives. The concern will focus on:

- all business processes and sub-processes owned by an enterprise or a partial collection of the processes related to functioning of an essential part of the enterprise, resources involved into the business processes, internal and external actors participating in the business processes,
- organizational structure with job roles descriptions, organization culture and organizational policy,
- workflow connected business processes and actors, organizational structure, environment and information flow,
- aspects of data management policies in respect with the workflow,
- patterns of information production and consumption on the operational level of enterprise functioning – data connected to document flow: documents, presentations, spreadsheets covering business processes.

6.1.1.3 Systems and data

Business processes and organization functionality is supported by software systems in order to achieve an enterprise's objectives. The following points are important:

- description of collection of software applications used by an enterprise - application portfolio,
- descriptions of services that support and automate business processes, linking users, information and functionality of business processes,
- description of the interaction and interdependences of the software applications – applications' interfaces,
- network connectivity software components assisting network communication between the server, desktop and mobile devices,
- operating systems providing the resource management which means running of different components of an enterprise solution, and providing users with a virtual machine that is more convenient to use then the actual hardware,
- data models presenting standard structure to store the data produced and consumed during enterprise functioning,
- structured data stores as DBMS. DBMS supports different views of data for different users, data retrieval, data security, data recovery, data access for different applications.

The Systems and data concern is addressed to the application, information and technology perspective.

6.1.1.4 Technology

The technology supports applications. The software systems and data used by an enterprise require hardware to run at it. This may include:

- plans for developing new and revising old hardware technology platforms,
- server hardware,

Microsoft enterprise solution architecture concepts

- desktop hardware,
- mobile devices: personal digital assistant, mobile phone, pager etc.,
- network communication devices,
- printers

and other hardware.

The Technology concern is used to meet the application and technology perspectives.

6.1.2 Architectural levels

The Microsoft enterprise solution architecture may be described from different perspectives. It can have several levels: business, application, information or technology. An enterprise solution has only one architecture. But the value of enterprise architecture is in the interrelations between individual perspectives.

The Microsoft enterprise solution architecture defines correspondence between architectural perspectives and architectural levels:

- *Business architecture* aims to collect functional requirements for the enterprise solution and to reflect the structure of an application domain: enterprise processes and organization;
- *Application architecture* transforms functional requirements to the enterprise solution architecture;
- *Information architecture* shows data structures and information flow;
- *Technology architecture* must translate operational requirements to the enterprise solution architecture.

The perspectives are sources for the requirements elicitation for an enterprise solution. Enterprise solution requirements have been introduced in the section 3.2.1.

Functional requirements contain the main value an enterprise solution delivers to an enterprise. Operational requirements define how to support the functional requirements technically. That’s why application and technologies architectures are considered as the most valuable and deciding for an enterprise solution.

Application and technology architectures should be regarded in mutual cooperation and interdependence. They must take advantages of each other.

6.1.2.1 Application architecture

Application architecture is the structure of an enterprise solution from the application viewpoint. Enterprise solution is considered as a set of automated services that implement functional requirements for an enterprise solution.

Application architecture covers:

- a structure of an enterprise solution. Enterprise solution is viewed as automated services and interfaces to other services and applications,

- how the structure implements functional requirements. Application architecture supports business architecture.

Well-designed application architecture will match the following aspects:

- correspond to functional requirements,
- use the latest technology. Application architecture has to exploit reusable software components to implement functional requirements i.e. services. *Software component* is an executable software module with an interface for communication with other components. Software component provides one or more services,
- map to the technology architecture.

Application architecture is a base for enterprise application development and further for deployment packages and policies.

6.1.2.2 Technology architecture

Technology architecture is the structure of an enterprise from the technology viewpoint. Technology architecture describes technologies used in the enterprise solution architecture and interrelations between them.

Technology architecture explains:

- a structure of enterprise solution as a set of hardware and software interfaces to that hardware,
- how the structure implements operational requirements:
  - support application architecture,
  - support technology architecture,

Well-designed technology architecture should satisfy the points:

- servers should be configured in a way to match application components used in application architecture,
- network hardware settings have to support information flow shown in information architecture.

Technology architecture is used for setup and operation of data center and later for installation of network of devices.

6.1.3 Views

The Microsoft enterprise strategy considers viewpoints as a representation framework or scope.

The idea of multiple views is broadly used in the Microsoft enterprise architecture. An architectural description can contain several models which participate in several views.

There is a correspondence between every architectural perspective and an enterprise architecture level. Enterprise architecture level can be business, application, information or technology architecture.

Microsoft enterprise solution architecture concepts
Microsoft enterprise solution architecture contains several views for every architectural perspective. The views can be conceptual, logical, physical and implementation.

The purpose of an architect is to create enterprise solution architecture as set of models. The models can participate in different views which conform to viewpoints. A viewpoint provides a framework language to describe a model. This language includes a collection of concepts, relationships among them, description element related to the concepts and guidance how to use these concepts.

The representational framework or scope can be collected into *patterns*. Patterns are based on real successful experience in enterprise solutions and provide ready solutions for architectural models. The architectural models or its components may be composed from patterns.

All the possible views for the Microsoft enterprise solution architecture are represented below.

6.1.3.1 Conceptual views

It is the most abstract view. The conceptual view is used to cover the concerns of the stakeholders: a business user of an enterprise solution and an enterprise solution architect.

The purposes of conceptual view are:

- define functional requirements of an enterprise solution,
- create a business model for an enterprise solution,
- discuss a business model with a business user.

The conceptual view should be described in the terms easily understandable for non-technical people.

Application architecture conceptual view helps to find out functional requirements and a business user vision of an enterprise solution. The output of design the conceptual view is a business model.

A *business model* is a description of business processes in a way to represent enterprise functioning at the operational level.

The modeling elements can include:

- UML diagrams
    - Use case diagram,
    - Activity diagram,
- Business entity modeling elements
    - REA diagrams,
    - ORM diagrams

and other.

Microsoft enterprise solution strategy stresses importance of the application architecture conceptual view. The architectural decisions implemented at the conceptual view affect all further stages of development an enterprise solution.

Application architecture conceptual view is described in details in section 4.1.4.

Technology architecture conceptual view is a description of technology areas used in enterprise solution. The view defines and names these areas and a structure in which the areas are composed. The purpose of the view is to provide a basis for discussion the technical aspects of the application architecture between a client and a supplier of an enterprise solution.

The technology architecture conceptual view must guarantee that the technology areas:

- meet all operational requirements (which should be implemented as Web services) for an enterprise solution,
- are available to the client.

The high level diagram of the view will include common elements required for any Web service or software system:

- *service platform* provides software supporting Web services:
  - operations system,
  - data storage,
  - networking

  and hardware to run at;
- *service framework* is the Enterprise Application Server, which is a host for Web services of enterprise solution. It supports functions required specifically for Web services: process, logic and state management and other functions;
- *service delivery* supports presentation of result of Web services' activity to a user of enterprise solution for any devices: client services, user portals;
- *service creation and deployment* provides the tools and describes methodologies, processes and technology patterns to realize Web services and create an enterprise solution;
- *service integration* concerns with Enterprise Application Integration - combining operating systems and Web services.

6.1.3.2 Logical views

The logical view is the most abstract view. The logical view is used to cover concerns of the stakeholders: an enterprise solution architect and an enterprise solution developer.

The intentions of logical view are:

- define the main components that implement functional requirements of an enterprise solution,
- define the relationships of the main components,
- create application models for an enterprise solution,
- discuss application models with a developers.

The conceptual view should not show technical details of the components and their interrelationships.

Microsoft enterprise solution architecture concepts
Application architecture logical view uses as input functional requirements implemented within the business model. The design output of the logical view will be application models.

Applications models are logical views of the business model of an enterprise solution. Application models illustrate an overall structure of an enterprise solution and define how to meet functional requirements:

- Data management,
- Processes steps,
- Interfaces between parts of the model in terms of logical messages and sequences,
- Data and states.

The modelling elements can include UML diagrams:

- Class diagram,
- Sequence diagram, collaboration diagrams.

Technology architecture logical view describes enterprise solution architecture in terms of technology packages from enterprise software vendors. These packages are usually servers providing some operational functionality.

Elements of the technology architecture logical view are databases, mail systems, transaction support, messaging.

6.1.3.3 Physical views

The physical view is described in technical terms. It is the most concrete view. The physical view is addressed for concerns of the stakeholders: an enterprise solution architect and an enterprise solution developer.

The objectives of physical view will be:

- show the particular implementation components of an enterprise solution. Enterprise solution is viewed as a software and hardware system,
- illustrate the relationships of the implementation components,
- physical view is used in the design and implementation processes of enterprise solution life cycle.

Application architecture physical view has applications models as an input and implementation models as output.

Implementation models defines how to map application models to the technology. The models should specify data control structures, message handling. The purpose of the models is to give a complete specification

The modeling elements will contain:

- Detailed business logic written as code,
- UML notations.

Technology architecture physical view describes in details the names of technology elements, connectivity between them and their actual physical location at the very detailed level. The view contains operating systems, servers, networks and other elements.

6.1.3.4 Implementation views

The implementation view covers concerns of the following stakeholders: a client enterprise who orders enterprise solution and …

The purposes of implementation view are:

- show how to setup the implementation components of an enterprise solution in the client organization,
- describe how and what technical facilities (software and hardware) should be used in a client organization to run an enterprise solution,
- implementation view is used at the setup process and operational process of enterprise solution life cycle.

Implementation view is considered by the Microsoft enterprise strategy as to be owned by a client of an enterprise solution.

6.1.4 Application architecture, conceptual view

Application architecture plays the key role in Microsoft enterprise solution architecture. Application architecture contains three views: conceptual, logical and physical. The views have been briefly described in chapter 4.1.3.

The conceptual view is the most influential to the enterprise solution though is defines the basis for enterprise solution development. All the development activities will focus on implementation the enterprise solution architecture at the conceptual level. The proper architectural description will increase productivity of developers and quality of an enterprise solution and reduce time to sell the enterprise solution to a client.

The elements used by the view are in connection with terms of a business enterprise introduced in chapter 3.1.

The elements of the application architecture conceptual view are the following:

- *Services* are code units implementing the parts of enterprise solution logic. They have interfaces to provide communication with other services and systems. The interfaces are based on messages. The services can operate not only at one computer, but across the whole network

  Services must support business processes and associated data. The states of business processes are determined by *business rules*: business process algorithms. Business rules are implemented as *enterprise solution logic*.

  Services have *policy*: a general management rule to govern services.

  There are two types of services:

  - *Business services* are services implementing business process functionality;

Microsoft enterprise solution architecture concepts

  - *Process services* are services assisting business services;

- *Messaging* is infrastructure of messages using by services to communicate with each other across the platform. Messages participate in business processes and have routes in the messaging infrastructure;
- *Contracts* are sets of explicit rules for communication between services. When two services communicate together through interfaces, they will produce and accept messages according to a contract. A contract specifies messages, format of the messages, and sequence to send the messages;
- *Policies* are sets of run-time rules for defining behavior of services. Policies provide services with management and security. Policies identify the scope of services behavior with respect to the stakeholders. For every stakeholder a service can be viewed differently in a way of service functionality available for a stakeholder;
- *Sate* is a status of a business process at a time point. It is a source of current information about a process. A state can be stored in a database of in a file system. Services manage states of business processes; this is the main rationale of services. The important constraint is that services must ensure states consistence;
- *Processes* are business processes. A business process is sequence of actions. They are explained in section 3.1.1. Processes should be carried out by process services. They provide progress in functioning of an enterprise solution: moving from one state to another. Execution of an action is realized as a call of a business operation.
- *Applications* are compositions of process services and interfaces. Infrastructure parts of application (user interfaces) in is separated from business parts.

  Interfaces may be of two kinds: business interfaces and user interfaces. The interfaces are realized and process services to govern communication. Business interfaces provide communication between process services. User interfaces provide interaction between an enterprise solution and a user.

  Services implementing business processes may consist of layers:

  - *a service façade* defines the functionality visible for enterprise solution environment. It assists implementation of contracts. A service façade can hide interfaces internally used in the solution;
  - *a business process layer* assists a business process within a service;
  - *a business entity layer* implements business processes and includes business entities. Business entities are objects implementing business rule. Business entities do not store data, but contain references to the data stores;
  - *data representation layer* stores the data used by the business entity layer. The layer does is not domain-oriented. It holds generic functionality to store data.

  This model is appropriate for communication between a user and services and for business-to business scenario, when business process services communicate with each other.

6.1.5 Comparison with the standard IEEE Std 1471-2000

IEEE Std1471-2000 standard for the enterprise solution architecture description is introduced in the previous chapter (section 3.2.3).

The motivations to compare the Microsoft enterprise solution architecture with the standard are:

- the standard gives a reasonably good conceptual framework for system architecture,
- Microsoft enterprise solution strategy should enforce the formalization of enterprise solution architecture.

A comparison could give some ideas for further improvement of the standard and constructing a meta-model for Microsoft enterprise solution architecture.

There are two main aspects for consideration:

1. Semantic relation between parts of architecture,
2. Life-cycle development support.

The standard has proper semantics for the meta-model, but needs stronger life-cycle development support. The system life-cycle should be included to the standard meta-model.

The Microsoft enterprise solution architecture leads in the second aspect, but lacks the first one.

There is no complete direct correspondence between concepts of the standard and the Microsoft enterprise solution architecture. Some parallels could be found.

The basic concepts of the Microsoft enterprise solution architecture are listed in comparison with the standard:

- *Enterprise solution* corresponds to a *system* in the standard.
- *Stakeholders* of Microsoft enterprise solution corresponds to *stakeholders* in the standard.
- *Enterprise solution architecture* corresponds to an *architecture* term in the standard. But Microsoft enterprise solution strategy does not distinguish architecture and architectural description.
- *Perspectives* may correspond to *viewpoints* suggested by IEEE Std 1471. A viewpoint is a way to consider and solve a problem. A viewpoint is possessed by a stakeholder. A viewpoint is external to an enterprise solution.

  Topics of information relevant for the enterprise strategy may correspond to the *concerns* IEEE Std1471-2000.
- *Model* may corresponds to *model* concept in the IEEE Std 1471-2000. It is a part of architectural level in Microsoft enterprise solution architecture;
- *Patterns* may correspond to the ‘Library Viewpoint’ in the IEEE Std 1471-2000. The architectural models or its components may be composed from patterns.

Microsoft enterprise solution architecture concepts

- *Views* partly correspond to the *views* in the IEEE Std 1471-2000. Views can be multiple in the Microsoft enterprise architecture.

  But a view in Microsoft enterprise solution architecture looks like a combination of terms from the standard: stakeholder, concern, viewpoint, view. It is an attempt to view the enterprise solution architecture from stakeholders' viewpoints at different stages of the enterprise solution life-cycle. Every stage has a unique set of stakeholders with their roles unique for the stage and with their sets of concerns unique for the stage.

- An *architectural level* has no correspondence with the standard. An architectural level is associated with several views. It can contain several models which participate in several views. The rationale of an architectural level is to represent architecture in dynamics, how it will change during the life-cycle stages. Enterprise architecture level can be business, application, information or technology architecture.

Microsoft enterprise solution architecture has a semantic gap between parts of architecture representing different stages of life-cycle: architecture levels. Also there is no clear semantic between stakeholders and views. As a result, patterns for constructing enterprise solutions are selected manually. There is no explicit prescription on when and how to use the patterns.

These topics may be clarified by introducing a meta-model. That model should combine all the main architectural concepts in a structure and specify them with invariants.

There are some particular aspects to be determined by the architectural meta-model:

- Determine all the possible and necessary combinations between views, architectural levels and models; specify them with invariants;
- Define the structures of those combinations;
- Identify possible sets of stakeholders, their roles and concerns for a concrete life-cycle stage;
- In particular, specify how to bind a business architecture level with the enterprise solution; determine the acceptable techniques and methodology for modeling the business architecture level.

6.2

6.3 Microsoft .NET

6.3.1 Product overview

The Microsoft .Net technology is a platform to build applications. The platform gives possibilities to develop Web client, server and service applications, to cooperate them together and with other solution components. Microsoft .Net provides integration of different solutions, including solutions for other platforms, through XML Web services.

In general, Microsoft .Net is a set of services. A *service* is a code unit which handles a limited set of tasks. These services are realized as an intermediate code independent from a platform. The services are executed by a virtual machine - Common Language Runtime (CLR). CLR performs resource management and applications control.

The Microsoft .Net is a product family with a number of software components. The main products are:

- .Net Framework is a platform for development, implementation and execution of Web services and applications.
- Visual Studio .Net is an integrated development environment for XML Web services and Web applications.

  .Net Framework together with Visual Studio .Net gives a complete solution for developers. The products realize a programming model based on common class libraries for different programming languages.
- .Net Enterprise servers – SQL Server 2000, Exchange 2000, Commerce Server 2002 and others. They provide deploying, management and coordination of Web services.

  For example, Commerce Server 2002 is a corporate server, provides fast development of e-commerce Web sites. Commerce Server 2002 uses integration with Visual Studio .Net development environment and .Net Framework components. The server has a built-in support of .Net My Services, .Net Passport, .Net Alerts components.

  SQL Server 2000 is a compact DBMS. It can be used for fast development of enterprise applications for mobile devices.

Microsoft .Net contains products for smart clients:

- .Net extensions for smart devices (mobile phones, pocket PC etc.) - allow to download and locally execute applications with easy-to-use UI, to use resources of local networks and Internet. A user may apply to any data from any place. Some application areas are on-line deals, team calendar planning, completion questionnaires.

  An example is Smart Device Extensions (SDE) for Visual Studio .Net. It helps to make applications for mobile phones and personal digital assistants (PDA).

Microsoft .Net includes a set of XML Web services. They can be combined with other XML Web services and clients applications. There are some examples of the products:

- .Net My Services SDK and .Net Alerts SDK - are Web services. The products simplify applications and services development. The Web services are oriented to the end-user and use SOAP net protocol for data exchange. These service

applications can be used in Web applications for smart devices, in other Web services and client applications realized on different platforms. The .Net Alerts service allows to apply for wide range of .Net Alerts clients including Windows XP.

- SQL XML for SQL 2000 - is a set of components for extension XML functionality for SQL Server 2000. It helps developers to simplify data integration, increase applications interoperability, supplies XML functionality for any level of multi-tier architecture applications – data level, medium level and client applications.
- Microsoft XML Core Services (MSXML) - is a set of XML services. It provides applications with .Net XML functionality. MSXML supports the newest versions of XML standard and recommendations of W3C XML Schema.
- Office XP Web Services Toolkit - allows to use Web services in MS Office XP applications and to integrate MS Office XP solutions using VBA package (Visual Basic for Applications).

6.3.2 Microsoft .Net Framework

6.3.2.1 Architecture

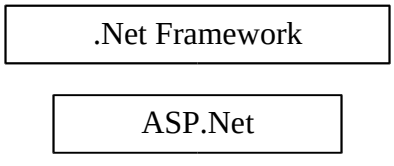
Microsoft .Net Framework is a key component of Microsoft .Net. It is a high-performance multi-language platform for development, implementation and execution of Web services and applications.

The main components of .Net Framework are presented in the Fig.4.2.2.1. There are Common Language Runtime (abbreviated CLR), Class Library and ASP .Net.

CLR is an environment for execution of .Net applications, memory management, security control, integration with programming languages. Developers do not work directly with CLR, but with unified Class Library located above CLR. The Class Library contains classes for interaction with operating system services (OS services), working with data and XML documents and solving other programming tasks. The part of the Class Library is a programming model for development Web application ASP .Net.

OS Windows services support COM+ technology, provide transaction management, message queues and other OS functions. Microsoft .Net Framework interacts with OS and provides standardization of system calls.

The main .Net Framework components are briefly explained in this chapter.



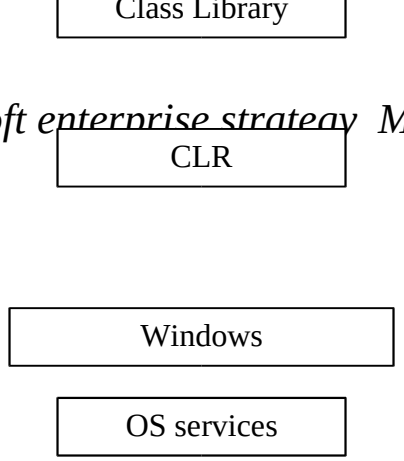


Figure 4.2.2.1. *Microsoft .Net Framework main components.*

6.3.2.2 Data exchange

There are three possibilities for data exchange in .Net:

- Windows Forms – is a set of classes to make GUI for Windows applications. Win Forms also carry functionality provided by .Net platform. For example, multithread data processing and inheritance, ActiveX control elements.
- Web Forms – is a set of classes to make GUI for Web applications. A Web Form is a dynamic Web page with control elements and data processing model. It contains HTML code and a scrip code fragments. A client browser sends a request for a page to a server, the script code fragments are executed at the server side. The resulting page has HTML-like format and is sent to the client browser.
- Web Services - is a set of classes for data exchange between server and client components. Web services use XML as a format understandable for different platforms and use Internet as a common platform to integrate applications.

6.3.2.3 CLR

Common Language Runtime (CLR) is an environment for execution of .Net applications. CLR performs functions for resource management and applications control: exception handling, security control, version control, provides debugging facilities; controls object lifetime using reference counters and garbage collection.

This functionality is available for any programming language supporting a special Microsoft standard – Common Language Specification (CLS). At present there are several languages using CLR: Microsoft Visual Basic .Net, Microsoft Visual C# .Net, Microsoft Visual C++ .Net and others.

CLR integrates code components written on different languages. The code integration process is accomplished during a design stage, not an execution stage.

First a code unit is written in a programming language from .Net language family. Then this code is compiled for CLR. At this step the code is called **managed code**. Managed code has **metadata** created during the compilation. Metadata describes a code component itself and other components used for making this component. Metadata contain information on types, members and references used in the code. CLR uses metadata for class detection, loading, code generation for a concrete platform, security control. The environment checks if all the necessary resources named in metadata are available.

The managed code is written in **Microsoft Intermediate Language** (MSIL). This code does not depend on a concrete platform. When the program written on MSIL is called for execution, the intermediate language code is transformed into a concrete platform code. This is a task of **Just-In-Time Compiler** (JIT). The compiler is a .Net component depending on a platform. Every method is compiled after calling. Compiled code is stored in memory. All the following calls use the compiled code.

An **assembly** is a minimum unit for embedding, version control, design reuse and security control in .Net. It is a collection of one or several files and all associated binary data. Every assembly includes a manifest. This is metadata including information on classes, types, references to other assemblies. Using metadata gives several advantages. There is no need to store information about objects in the object register, when a component is transferred to another computer. Component deletion requires only deleting the assembly containing this component.

Though .Net components do not require registration, if an assembly is used by more than one application, it should be stored in a special storage – **Global Assembly Cash** (GAC) or **Download Cash**.

Another component of CLR is **Common Type System** (CTS). It determines types supported by CLR. The types are divided into two groups: value types and reference types. The groups have subtypes. Value types describe values presented by a byte sequence. Reference types express values presented by a byte sequence location. The top of class hierarchy in the .Net Framework is System. Object. All CTS types inherit from it.

6.3.2.4 Class Library

The Microsoft .Net Framework Class Library contains classes which can be used from any programming language supported .Net. The Class Library provides the following functions:

- general and user-defined classes support,
- exception handling support,
- input/output operations and streams,
- OS functions reference,
- data access,
- possibility to make Windows applications,
- possibility to make client and server Web applications,
- possibility to make Web services.

All the classes in the Class Library are organized as namespaces. Every namespace contains classes and other types referring to a specific task or a group of tasks.

The System namespace is a root in the Microsoft .Net Framework Class Library. It contains fundamental data types in .Net Framework. This namespace includes System. Object. It is a parent class for all classes in .Net, primitive and expanded classes, and more than 100 additional classes for exception handling, domain application management, garbage collection and for other functions.

System. Console class helps to create console applications for .Net.

System. Type class is a point to enter the Reflection namespace. The class Type contains methods for getting information about class members. Using classes from the Reflection namespace, it is possible to know metadata associated to these classes during a program execution.

System. Web namespace located in Microsoft .Net Framework Class Library is a base for development ASP.Net applications.

There are more namespaces in the Class Library, but the description is skipped.

6.3.2.5 ASP .Net

ASP .Net is a programming model for development Web applications independent from a client's platform. In general a client could be any device connected to Internet. ASP .Net applications are capable to integrate with server components and solve a wide range of business tasks.

ASP .Net is a basis for development Web services and Web Forms. ASP .Net is based on **Active Server Page** technology. The technology helps to create dynamic Web pages. These are Web pages combining HTML code with a script language code. A dynamic Web page is a file located at the Web server. The script code fragments are interpreted on a server side. The execution result has HTML-like format and supplements the script code fragments in the initial page version. The resulting page is sent to a client browser.

The main advantage of ASP .Net technology comparing with ASP is that code of a Web page is not interpreted, but compiled and cached at the server side. This increases applications productivity. When a client browser send a request to a server for a Web page, the system checks, if there is a compiled version of the page. The compiled page is written in the intermediate language MSIL. It's executed by .Net environment, and the result is sent to the browser.

ASP .Net applications use data stored in DBMS in order to generate dynamic fragments. ASP .Net applications access the data through **ActiveX Data Objects** (ADO). ADO .Net is a class library of control elements. A client receives ADO objects in a content of Web pages and executes them. ADO technology helps to save data in XML and load data from XML format. This technology helps to exchange data between Web services in distributed applications.

6.3.3 Visual Studio .Net

Visual Studio .Net is an integrated development environment for XML Web services and Web applications. The functionality of Visual Studio .Net offers the following advantages to application developers:

- an intuitively understandable, unified multi-language development environment;
- .Net languages, which can interact with each other and are familiar to developers,
- high-productivity tools for life-cycle development of enterprise solutions – from requirement analysis and planning to product maintenance.

In general, programming languages and functionality of Visual Studio .Net have to be used within the .Net Framework.

6.3.3.1 Programming languages

The programming languages used in .Net are the following:

- *Visual Basic .Net* is the most easy and productive tool for Windows and Web applications development. The language totally supports OO programming model, visual inheritance of forms, simplified access to Windows functions, multithread data processing, development of Windows NT services and consol applications;
- *Visual C# .Net* is a modern component-oriented language, specially designed for .Net platform. It combines the best features from C, C++, Java and Delphi languages. The language provides:

  - automatic garbage collection,

  - reference manipulations and direct memory access,

  - property and event objects support,

  - attributes support,

  - built-in support of common types,

  - multiple inheritance is available only from interfaces,

  - C API (Application Programming Interface), Windows API and COM+ at the language level,

  - assembly support,

  - type checking,

  - automatic variable initialization;

- *Visual C++ .Net* is a low-level application control tool. The main distinction of the language is support of both the code model for .Net (managed code model) and the code model for Windows (unmanaged code model). The language contains ATL Server – an extension of the Active Template library. It helps to crate compact and high-performance Web services and applications;
- *Visual J# .Net* allows Java-programmers to work within .Net platform.

6.3.3.2 Web service and Web application development

Visual Studio .Net provides reach functionality for fast and simple development of XML Web services, Web applications based on Web Forms and Windows applications based on Windows forms. Microsoft .Net supports Web Forms and Windows Forms, but Visual Studio .Net gives facilities to simplify use of the technology. With the help of Web Forms it's possible to create interfaces for Web applications in the same way as for Windows applications. These Web applications are transparent for all browsers and smart devices.

Mobile Internet Toolkit is an additional package supporting code generation for smart devices: WAP-, HTML-telephones, Pocket PC, Palm and RIM Blackberry pagers. It is possible to use Web Forms for usual browsers and for smart devices within the same project, combining business logic and data access for both platforms.

Windows Forms are used to develop client applications. It is possible to integrate Windows applications with Web services. ADO .Net technology provides data access.

The Server Explorer helps to access server components. It provides working with databases, supports event queues, operating system services and other server components.

The Component Designer allows visual design of server components. Developers can drag-and-drop resources into the designer, set the attributes and write code. Resources are considered as usual objects.

6.3.3.3 Life-cycle development support

There are two products from the Visual Studio .Net family for development enterprise solutions:

1. Visual Studio .Net Enterprise Architect – helps to design large-scale enterprise solutions,
2. Visual Studio .Net Enterprise Developer – supports groups of developers to make XML Web services and enterprise solutions for any devices.

The products contain recommendation for effective architectural design of enterprise applications using enterprise schemes and templates, facilities for Web services and applications design, development and testing for groups of developers. These aids support:

- database modeling facilities based on Visio technology,
- program modeling facilities based on Visio technology,
- enterprise templates,
- testing facilities - Application Center Test,
- version control facilities - Visual SourceSafe.

7.C.3.3.1 Object Role Modeling

Visual Studio .Net modeling tools support the Object Role Modeling (ORM) methodology mentioned in the section 2.1.3.

ORM is used information analysis at the conceptual level and database modeling.

ORM has advantages for working with application domains. It is possible to create conceptual models which are understandable for customers, and also to analyze information and manipulate with data types.

7.C.3.3.2 Enterprise templates

Enterprise templates have two basic components: the initial project structure (project template) and policy.

Enterprise templates facilitate a general project scheme. Thus a distributed application can contain projects for every three architectural layers – UI, business logic and database. Besides the initial project structure, it's possible to include standard components and other repeatedly used program elements in the templates. The result is a high-level application structure which is a started point for a project development.

Enterprise templates are available in Visual Basic .Net and Visual C# .Net.

Policy is a XML document describing project in Template Description Language (TDL). It helps an enterprise solution architect to manage the Visual Studio .Net environment including task list, instrument panel, Solution Explorer, designers, editors etc. An architect can determine technologies available for developers, set initial value range, and include design documents.

7.C.3.3.3

7.C.3.3.4

7.C.3.3.5 Testing

The Application Center Test allows developers to collect performance information make functional testing of Web services and applications Visual Studio .Net. The Application Center Test supports a script language and can simulate the workload.

7.C.3.3.6 Implementing Enterprise Solutions

Visual Studio .Net contains facilities for implementing enterprise solutions:

- implementation as an integrated part of any project;
- enterprise solutions may be implemented to Web servers, non Web servers, traditional medium as CD-ROM or theirs combination;
- total class support for implementing program development;
- total support of Zero Administration for Windows including side-by-side installation.

8 Navision Jamaica model

At present there is a need for a software tool to construct enterprise solutions satisfying the rapidly changing business environment. This software tool must help to form enterprise solutions very quickly, because customer requirements vary pretty much in time. Such a tool should be able to:

- consider existing and forecast the future diversity of applications and devices a user works with,
- predict growing number and complexity of business processes,
- capture great amount of information a user should percept and control, and events which influence on business,
- provide modeling facilities for a user with a comfortable user interface and helping to make business solutions in the most efficient way.

These rationales require the tool to be universally accessible regarding place, time and device, provide cooperation with business partners, integrate with other business applications and Web services and rely on user experience in work with information systems and applications.

The resent reply for this demand is a Navision Jamaica multi-tier platform for implementing and operating enterprise solutions.

Section 5.1 gives an outline of Jamaica based on the papers [14], [16].

Section 5.2 explains Jamaica modeling concepts. The conceptual model for Jamaica modeling concepts is suggested and represented in UML notation in Appendix E.2. The conceptual model can be used to improve understanding of Jamaica modeling concepts.

Section 5.3 proposes possible development directions for Jamaica model.

8.1 Jamaica overview

The main objective of Jamaica development is a focus on the customers and the market needs rather then on technology in order to give the fast response to changing customer requirements and fast solution implementation. This is a key intent influencing all Jamaica features described below.

Traditional software engineering includes the following stakeholders:

1. **a customer** – acts as a client and an expert in his business area,
2. **a consultant** – interviews the customer and makes a system design,
3. **a programmer** – implement the design into the running system.

A customer can test and evaluate the system when it has been implemented, and ask the consultant for improvement if necessary. The consultant makes changes in the design and submits it to the programmer for changes in the program. This iteration process may

cause high turn-around time and errors while information comes from one party to another.

Jamaica approach to software engineering supposes only two participants:

1. **a customer** – acts as a client and an expert in his business area,
2. **a business architect** - interviews the customer and makes a solution in cooperation with the customer.

A programmer is not needed. The solution is created by **modeling** and may be run immediately. A customer self can make changes to the system with the help of an architect and see the feed-back immediately.

The purposes of the Jamaica approach are:

- to shrink the distance between a customer request and a running implementation
- to minimize the amount of errors and turn-around time.

Jamaica also provides a possibility to use custom code within the same development environment and programming model, it means that a solution or its parts may be also programmed.

The Jamaica platform possesses **high adaptability** to changing requirements. It provides access to the same modeled solution from different devices without writing any code. The platform also can be extended to new devices. A user can access the solution with a user interface intelligently adapted to a particular device.

The architecture of the platform allows to choose what functionality put to the client side and to a server side within the solution. The customer decides the distribution of functionality between a client and a server.

The **standard** for Jamaica platform is .NET framework. The platform uses a standard database and Microsoft SQL Server to perform requests to the database. The Jamaica designer modeling tool is hosted within Visual Studio.NET.

The Jamaica platform can **integrate** with other systems and applications:

- The platform provides abilities to consume **Web services** for an existing solution without any special programming to adapt it. From the other side it's possible to expose a solution as a Web service to make it available for external consumers.
- The platform supports **document exchange** between applications and by that ensures Business-to-Business and Partner-to-Partner integration. The platform uses XML templates for mapping reports to the form understandable for customers. Customers can modify their reports themselves.
- It is possible to provide users with **self-service** by granting on-line access rights to a solution.

The Jamaica platform meets **user requirements** for operating in changing business environment. It supplies implementation of Business-to-Business scenarios, Web shops and self-services for business partners, provides access to a solution from any device connected to the Internet.

The intention to use **user experience** is to give users a feeling that they use Jamaica just as they usually use Internet. Thus each user gets a portal arranged as a number of activity centers. The portal is personalized according to a specific role a user plays in business

process. An activity center accumulates several logically connected tasks, which correspond to real business activities as ‘sale’, ‘purchase’, ‘delivery’ and others.

A user interface is automatically generated upon the actual model and provides primary **user assistance,** like UI widgets, page titles and instructions to a user, as well as secondary user assistance, which contains more topical description. The user assistance subsystem is generated from the model itself and implements inductive navigation principle, when a user is offered to do the next possible actions within the modeled solution.

The platform supports **multi-language** and allows a user to switch a preferable language any time.

For now the Jamaica platform implemented the architecture with the thinnest client - a Web browser and all the clients hosted by an ASP provider. It’s possible to develop a Windows SmartClient to make a client application with reach functionality.

The user experience concepts realized in Jamaica are Web navigation, Portal and Activity Centers, User tasks, integrated user assistance, digital feedback loops, embedding Web Services. The tasks for future implementation are category browsing, unified search, subscriptions, personalization, collaboration support, PIM integration, productivity integration.

8.2 The Jamaica modeling concepts

The traditional modeling approach for enterprise solutions supposes the next modeling concepts:

- Database – contains database tables,
- Business Logic – contains queries to the Database,
- Forms, Reports – present information from the Database to a user,
- Code Units – reusable code units or components customized to the solution to provide functionality of the system,
- Code – programming code unit to provide functionality the system,
- Help files and Documentation – provide user assistance.

The general idea of the Jamaica modeling approach is that an enterprise solution is formed by modeling. The requirements for Jamaica modeling concepts are:

- modeling concepts must be understandable for all parties involved in forming the enterprise solution,
- modeling concepts must have enough expressive power to represent real business processes.

Jamaica has the following basic modeling concepts:

- Business Model,
- Business Object, Business Object Base,
- Element, Element base, Element Type,

- User Task, Task Step, Task Page,
- View, Sub-View, Document, Report,
- Portal, Activity Center.

Any business process involves a number of business entities, like ‘a salesperson’, ‘a customer’, ‘an order’, ‘a stock’ and so on. The purpose of the Jamaica run-time designer is to map the real business world to the **Business Model**, i.e. business concepts to the modeling concepts. Business model concerns with business logic, analytics, documents, help/user assistance, user interface and a database. Business model describes the enterprise solution.

Any business entity from a real world is a **Business Object** in Jamaica. Business Objects may be classified in a smaller set of more abstract concepts according to similarities they have. This higher level of abstraction is formed by **Business Object Bases**. For instance, ‘an employee’ and ‘a customer’ Business Objects have the same Business Object Base named ‘party’ which entails common things from both an employee and a customer.

Business Objects have to be modeled in Jamaica as **Business Object Types**. A Business Object is an instance of a corresponding Business Object Type. Business Object Types relate in a certain way with other Business Object Types, they can have some crosscutting functionality and can entail features from other Business Object Types. The main purpose of modeling a solution in Jamaica is to create Business Object Types built upon Business Object Bases.

The ‘building blocks’ of Business Objects are **Elements**. Elements represent something useful in a context of different Business Objects and capture most of the functionality of Business Objects. Elements are independent and exist only in a frame of Business Objects. An example is ‘Delivery Address’. Elements contain data and functionality to process the data. Elements may depend on other elements and exchange data between each other.

Analogously to abstraction levels for Business Objects, Elements are based upon **Element Bases**, but any Element has only one Element Base. An element is an instance of an **Element Type.** Element Types are designed upon the Element Bases. For instance, Delivery Address Element Type may be built using the Address Element Base. There are some examples of Element Bases:

- Address – for postal addresses,
- Classification – for classifying Business Objects into groups and categories,
- Containment – for specifying a relationship between a Business Object which consists of a number of Business Objects,
- Milestone – for expiration date. When the expiration date occurs, the Milestone Element triggers some event,
- Registration (equivalently to Aggregation) – for storing a set of elements for legal and reporting purposes,
- Reconciliation – for keeping track of reconciliation of resources between Business Objects,
- Relation – for setting up relations between Business Objects.

Summing up, mapping business concepts to a Jamaica solution includes the next main activities:

- Choose a Business Object Base and create a Business Object Type upon it,
- Choose Element Base and Create Element Type upon it. This process is called typification,
- Specify data and functionality of the Business Object Type by putting the Element Types into the Business Object. This process is called configuration.

The functionality of Elements and Business Objects is invoked by **methods**. Methods belong to Elements and Business Objects and are called by **events** through 'event-wiring'. An event is something happened which has an influence on the solution. An event may be caused by a system ('system event') according to the modeled Business Logic, or by a user through an **action** ('user event') according to a user's choice in the user interface. Actions may contain custom code.

The only way of interaction between a user and Business Objects is through **User Tasks**. They supply a user with access to data and functionality of Business Objects and provide a basis for automatically-generated user interface. There is a basic set of possible User Tasks:

1. Create a Business Object,
2. View a Business Object,
3. Modify a Business Object,
4. Delete a Business Object,
5. Print a Business Object,
6. Run a specific method,
7. Transform a Business Object (create a new Business Object based on existing one).

User Tasks can be chained and contain hierarchical structure of **Task Steps**. Task Steps can contain a number of **Task Step Pages**.

The structure that gets a set of data of Business Objects is a **View**. A View is a number of Elements on a Business Object. Elements can contain **Sub-Views** and so on. But the View term does not correspond to SQL terminology. A View is an interface through which User Tasks can access the database and Business Logic. An instance of a View is a **Document**. It has a format understandable outside the Jamaica solution. A **Report** is a Document containing combined information from data of several Business Objects.

Users in Jamaica model are Business Objects with a Login Element. Users may be assigned **roles** according to activities they do in a business process. Roles are used to give user permission for one specific **portal** with particular **activity centers.** An activity center is also a portal to a specific set of Tasks which represents related business activities.

The first Task Step in a User Task, also called a 'Hot Task', is shown to a user in a web part in an activity center. It can be a Select Task or a Report.

The main architectural concept in Jamaica is the **N-tired framework**, where each tier communicates with above and below tiers through Views. The tiers are:

- Web browser client,
- UI subsystem – exchanges device-specific pages with the client,
- User Tasks – keeps track of Tasks,
- Business Logic – creates Business Objects and Elements instances, executes methods, offers interfaces for Web services and system level communication,
- Storage – provides communication with database, offers interface for analytics on the raw data.

One of the key intension of Jamaica modeling is to exploit generic and reusable modeling concepts to combine a solution. The Jamaica modeling concepts may be combined in **Design Patterns** for business solution. A proposed frame for modeling business processes is REA accounting model introduced in the Chapter 2. The REA model may be converted into UML using the Jamaica modeling concepts presented above in the section.

The summary of Jamaica modeling principles:

- create the high-level abstraction specification on a specific domain, understandable for a domain user – Business Model through Business Types, Element Types, Tasks, Events and Actions, Views and Documents.
- the code is generated directly from the model using high-level metadata and Design Patterns,
- the modeled code may be customized and coexists with non-modeled.

Future work in modeling is to create and widely exploit reusable Design Patterns. There are three possible directions for further development:

- to model a universal economic system,
- to model a specific application domain,
- to model a system which is extendable by new concepts without changing the system in general.

The conceptual model explaining the Jamaica modeling concepts is represented in UML notation in Appendix E.2:

- Appendix E.2.1 represents a class diagram for Business Objects and elements,
- Appendix E.2.2 contains a class diagram for describing basic modeling Jamaica elements.

8.3 Jamaica development possibilities

This chapter gives some suggestions for development possibilities of the Navision Jamaica system. The suggestions are given in respect with modeling concepts.

Comparing with Microsoft enterprise solution strategy, Jamaica should be used for small and medium enterprises with limited set of stakeholders' roles and simple business processes. The reason is that it could be difficult to keep consistency of the solution while modeling. The potential problem is keeping Jamaica model consistency while trying to satisfy all stakeholders' viewpoints.

The next problem or a source for previous problem is that it's difficult to keep the whole solution in mind at a time. So the solution should be leveled in order to be user-manageable.

If to make parallels with the concepts from the Microsoft enterprise solution architecture, Jamaica modeling is a process of creation a business model within the application architecture conceptual view. Microsoft enterprise strategy has universal scope regarding size of an enterprise: supports large-scale as well as medium size and small enterprises. Microsoft enterprise strategy is more system-oriented. Jamaica strategy can be called as more business-oriented.

Microsoft enterprise solution architecture (application conceptual view) is not suitable enough for business modeling. It's complicated for a user and involves many technical terms and implementation details. Jamaica enterprise solution strategy states that a user should not be concerned with technical details. The process of construction of an enterprise solution has only one stage – modeling engineering. But the 'payment' for this optimization is limited scope of the target market.

In order to enforce consistency of Jamaica solution, the possible suggestion is to involve some ideas from the standard IEEE std. 1471-2000 described in section 3.2.3. The standard gives an appropriate architectural conceptual framework for Jamaica, even it has some incompleteness. The main disadvantage of the standard is absence of a system's life cycle at the structure of conceptual framework. The standard supplies some recommendations at the examples of scenarios on how to use the frame work at different stages of a system life cycle.

In the case of Jamaica modeling this critical point on the standard is not important, because the development of Jamaica solution has only one stage. It accumulates all the traditional development stages like requirements analysis and elicitation, development, implementation, testing, validation in one stage called modeling. Feasibility study is out of scope of Jamaica enterprise solution.

The concepts of architectural description framework recommended by **the standard IEEE std. 1471-2000 can be specified for Jamaica** as following:

- Stakeholders are users in the Jamaica model,

- Concerns are stakeholders' rationales for Jamaica solution,
- Viewpoints determine a representation language and concepts they provide for modeling. A particular structure of business objects and elements created by a user and an architect may be put into a viewpoint library,
- Viewpoint library is a set of Jamaica modeling concepts and their relationships. Domain specific patterns should be specified,
- Views determine users permission for specifying portals, activity centers in Jamaica,
- Models can be of two types: a business model and a navigation/presentation model connected to the business model.

The standard framework can be also used to validate the model.

Another suggestion is to **make distinguishes in modeling process into sub-stages**:

- Requirement analysis and
- Development.

This will help to observe and model a solution as a whole at a high abstract level.

The modeling notations can be UML. The use of UML for conceptual modeling is discussed in chapter 6.1. Modeling enterprise value chain in general can be done by UML class diagram notation. Modeling business rules is suggested to do by state charts and activity diagrams. Use cases can be helpful for modeling a representation part of a solution. REA notations can be used as supplementary for conceptual modeling.

The next point concerns with Business Objects and Elements. The combination of BO and elements is not clearly understandable for a user. Intuitively one can say that Business Objects belong to conceptual level, but Elements belong to information or database level. Elements have a meaning of attributes for Business Objects.

As it was mentioned above, Jamaica model has similar meaning as the application architectural conceptual view in Microsoft enterprise strategy (section 4.1.4). Microsoft application concept use layers. The idea of **layering** can be used in Jamaica in order to separate conceptual and information levels. Business entity layer does not contain data. This is a layer for Business Objects. Implementation details as Elements should be hidden from a user as much as possible. Elements should be considered at the business process layer.

But from the other hand Elements affect the conceptual model in Jamaica. Another suggestion is **do not distinguish Elements as Business Objects**.

Continuing the discussion on Business Objects and Elements, there is another point difficult to understand for a user. Instances of Business Objects and Elements are treated the same way as Business Objects' and Elements' types. The rationale for that is optimization of design stage for building solutions. But this cause some inconsistency in conceptual terminology: Business Objects vs. Business Object Types, Elements vs. Element Types.

Another development direction is to consider solution to be controlled at present, past and future. This can be achieved by using business **ontology**. This topic is discussed in chapter 6.3.

Enterprise ontology can be used for validation a business model and for induction navigation in user interface. The possible way is to present ontology as a lattice. The ontological structure can be modeled in UML.

An effective way to use the lattice model in UI is to try to guess what the user is searching for by looking at examples of what he has been choosing. The lattice model is very suitable to this kind of choice.

Finding the least upper bound of the chosen concepts (the generalization concept which is common for the chosen concepts and situated in the hierarchical structure nearest to the chosen concepts), we can induce that the user is interested in actions/topics related to other specializations of the least upper bound. These specializations will show the possible navigation paths for a user interface. Constraints may explicitly specify:

- if a user can go to any of the induced paths or not, and
- how the choice have been made if several paths are allowed.

The mechanism of inducing queries can be used also for validation of the business model. The ontology must be carefully defined and supported by constrains realizing axioms.

9 Future enterprise solution

This chapter refers to business enterprise and enterprise solution subjects introduced in chapter 3, Microsoft enterprise strategy described in chapter 4 and Navision Jamaica model discussed in chapter 5.

The **goal** of future enterprise solution can be expressed as following: use the enterprise IT resources to increase the value produces by an enterprise.

Methods to achieve this goal will be:

- *business planning* is based on analysis of enterprise functioning and considers recourse allocation and scheduling the activities to construct enterprise solution;
- *software engineering*. As it was mentioned in chapter 3, software engineering is transforming into modeling engineering. Ideally constructing future enterprise solution should involve only one stage – business modeling;
- collecting *empirical experience* in business modeling, creating and using enterprise solutions. The experience should be collected in a form of reusable patterns and prescriptions how to use the patterns.

The need for reconsidering enterprise solution is cased by modern tendencies in technology and business life. **Technology factors** affecting efficiency of enterprise solution include fast running computers, Internet development and smart devices. Now algorithms efficiency is not a critical point. The considerable impact into effectiveness of enterprise solution will be made by semantics of data, knowledge representation, use of business ontologies and integration ontologies into enterprise solutions.

These factors in information technology development cause **changes in business**: development of new business spheres (e-commerce), new business infrastructure involving large data diversity, business processes are organized in a way to use new technical advantages. The complexity of business processes tends to grow.

In this respect functional requirement for enterprise solutions become more complicated. **Business modeling** plays a key role in enterprise solutions. It defines how well functional requirements will be implemented.

Future enterprise solution should be more **business-oriented**. The reason is that business user can understand business processes and participate in constructing enterprise solutions.

Referring to the Visual Studio. Net, it's important to notice that there are no development tools for business modeling ([30], *Identifying Business Requirements*):

'The tools used in business model planning are primarily business tools – rather that development tools – including accounting, financial planning and modeling, project management, and other management information tools. Visual studio, enterprise Edition provides no specific business planning tools.'

The modeling tools can be easily understandable for domain experts, be enough expressive and gives solid basis for further enterprise solution development. UML can be relevant for these tasks and acceptable both by enterprise solution developer and enterprise solution user.

This chapter concentrates on business modeling and discusses its future development possibilities.

Section 6.1 examines UML notations as a tool for business modeling.

Section 6.2 considers conceptual modeling to be used as input to the design stage of enterprise solution. Conceptual modeling is described in regards to business enterprise and enterprise solutions modeling and software applications modeling in general.

Section 6.3 discusses the use of ontologies for business modeling. The work done in the section relates to terminology represented in Appendices B and C.

9.1 UML notation

UML was constructed for the purpose of software specification, design, construction and maintenance. The idea to extend the use of UML for modeling real world has come later with development of software engineering and changes in business needs. UML notation can be used for description of application domains at the early stages of software system development to support requirements analysis. The application domain description is an input to the design phase. Now there is no commonly accepted guidance how to model a real world system in UML.

Unified Modeling Language (UML) is a graphical object-oriented modeling notation language.

UML was created by James Rumbaugh, Grady Booch and Ivar Jacobson. The motivation was the growing demand to describe business applications in a unified formal way. By the beginning of 90s a number of tools for creating business application models were made. Those tools had no correspondence between each other, which caused a problem in CASE tools development. Computer Aided Software Engineering (CASE) is software developing with the help of computer tools. CASE tools were initially intended to support DBMS: CASE tools provided automatic database generation according to its graphical structure made at the screen by a user. Now CASE tools assist software engineering process in requirements analysis, system modeling, code generation, debugging and testing.

Originally UML development was carried out by Rational Rose company specializing on CASE tolls. From 1995 OMG consortium has actively participated in development and standardization of UML notations.

Now UML is a language for visualization, specification, construction and documentation of software intensive systems. A software-intensive system was introduced in chapter 3.2.3 and then mentioned in chapter 4.1 in connection with enterprise solution.

UML 1.5 is the last version standardized by OMG; the coming version is UML 2.0.

The core UML features are:

- UML is object-oriented. It is well-structured for object-oriented languages;
- UML is a language of visual modeling. It provides graphical representation of a model as one or more diagrams;
- UML is a descriptive tool. UML is not a programming language; it does not contain algorithms;

- UML is platform independent notation. It is abstract regarding a concrete language specification or a development tool.

Object Constraint Language (OCL) was constructed to extend the expressiveness of UML. OCL is designed to denote invariants, pre- and post- conditions for UML constructs as Boolean expressions. OCL is included in UML.

Possibilities for UML extension

Development directions of UML notation have been influenced by technology and business evolution. Enterprise solution is one of the topics for UML progress. People from computer societies put some extensions to the UML notations. The future UML extensions should:

- adjust the notation to the new tendencies in enterprise business processes and software engineering / modeling engineering,
- improve the notation to make it more friendly and expressive.

One of the danger aspects of UML development is appearance of new dialects coming from different computer societies. The result can be inconsistency in the language and loss of the basic UML idea – to be a unified modeling language.

An idea to avoid that danger is to use the existent frame of UML notations and the *extension mechanisms* proposed by the UML designers:

- *Tagged value* is a pair of strings (a tag and a value) written in a form:

  tag = value

  A tagged value is used to represent arbitrary information; it can be also a keyword.

  There are several predefined tags in UML: **location**, **persistence**, **semantics**, **transient**. But they are not fairly relevant for enterprise solution modeling, because they relate to a classifier. Classifier is a specialization of a generalizable element. Classifier is not usually used by the analyst, but rather for understanding the UML notation.

  The only predefined tag really useful for enterprise solution modeling is **documentation**. It is a comment explaining something about a document;

- *Stereotypes* is a facility to make new building blocks in UML. Stereotypes are defined by existing building blocks. Stereotypes are specialized versions of UML elements.

  Stereotypes can be shown by a string of stereotype keywords in guillemets <<**stereotype name**>> or optionally by a specific icon.

  There is a number of predefined stereotypes in UML including stereotypes for business modeling. They are discussed in section 6.4.1;

- *Constraints* are invariants for UML structures used to specify the structures precisely. Invariant must be always held by a structure. Constraints can be written in OCL.

  There are predefined stereotypes for constraints:

<<invariant>> indicates that the constraint always hold for a structure,

<<postcondition>> specifies that the constraint must always hold after the completion of the attached operation,

<<precondition>> shows that the constraint must hold before the attached operation is called.

The listed UML extension mechanisms can be used in future enterprise solution. Stereotypes seem to be the most powerful mechanism for introducing new elements within the UML notation for enterprise models. Stereotypes can be used together with OCL constraints to explicitly specify the structure.

The rest of this chapter has a goal to discuss possibilities of using UML/OCL notations for business enterprise and enterprise solution modeling.

Section 6.1.1 describes UML stereotypes as a technique with a grate potential for enterprise modeling. Some advantages and dangers of using stereotypes are named. Stereotypes are also mentioned in respect with UML extension for business modeling.

In section 6.1.2 the use of UML diagrams are discussed regarding business modeling. The section considers also some modifications which could be done to adjust UML diagrams for business needs.

Section 6.1.3 proposes three new kinds of UML diagrams for business modeling: hybrid, agile and agent-based. Hybrid and agile diagrams might be used in the early stage of analysis of an enterprise solution. Agent-based diagram may be helpful in modeling business logic.

9.1.1 UML stereotypes

Stereotype is a specialized UML element. Stereotype is based on an existing element and used to construct a new UML element.

The possible use of stereotypes in business modeling has been already discussed in sections 2.3 and 2.4.1 in respect with representing REA notation in UML. Class diagrams in section 2.4.3 illustrate the discussion results. Stereotypes were used to model classes representing REA entities: resources, events, agents.

9.1.1.1 Advantages of using stereotypes

Using stereotypes in enterprise solution modeling is of a great potential. The main advantage is that stereotype notation allows to introduce new modeling elements without breaking the UML notation meta-model.

Another strong advantage is visual simplification of UML diagrams. Being introduced once in the model, stereotypes can be used many times. They allow to hide the details of a modeling element without sacrificing its semantics. Stereotypes can be used to construct a conceptual model for enterprise solution to present business objects in concise and expressive way. A conceptual model with stereotypes will be easily readable and understandable for a domain expert. At the same time the model will be enough expressive and useful for an enterprise solution developer.

OCL can be used to define invariants for stereotypes to describe them precisely.

Stereotypes can be used in enterprise solution modeling for the following tasks:

- marking element of a model,
- setting constraints for structures in a user-friendly way,
- validating a model,
- representing business objects as icons.

9.1.1.2 Dangers using stereotypes

In spite of all advantages of stereotypes, using stereotypes is a point at issue.

Usually stereotypes are used without proper declaration or without declaration at all in order to speed modeling process. But this approach can cause misuse of stereotypes, because they were not defined in details. The right way to use stereotypes is first to declare and then to apply stereotypes in a model.

Stereotypes can be declared without tag or constraints content. In this case stereotypes are used only for marking model elements. On the one hand, stereotypes represent a structure in a user-friendly manner. On the other hand, they do not bring much constructive impact to the model. The same or better effect may be achieved by using other UML elements.

Summing up all the named potential problems, it's important to note that stereotypes should be used very carefully.

9.1.1.3 UML stereotypes for business modeling

Facing the demand for modeling business activities in UML, the notation was extended. The UML Extension for Business Modeling, v1.1 was published in 1997. The later proposals for UML extension for business modeling suggested by Rational® software development company are based on the ideas of the named version.

The Extension used *stereotypes* to introduce new graphical modeling elements:

- *Use case model* is a model for enterprise activities description in interaction with external parties participated in the activities. Enterprise activities are modeled as use cases. Parties are modeled as actors. The relationships between activities and parties are shown;
- *Use case package* is a package consisting from use cases, actors and relationships between them;
- *Use case system* is a package consisting from use case packages, use cases, actors and relationships between them;
- *Object model* is a model for describing things from enterprise domain;
- *Object system* is a subsystem of an object model. Object system includes organization units, workers, work units, entities and relationships between them;

- *Organization unit* is a subsystem to represent a unit of an enterprise organization structure. It contains organization units, work units, workers, entities and relationships between them;
- *Work unit* is a subsystem to represent one or more entities participated in a business task;
- *Worker* is a class representing an actor of a business process. Workers can communicate with each other and manipulate by entities;
- *Case worker* is a worker who is an external actor of a business process. For example, a customer;
- *Internal worker* is a worker who is an internal actor of a business process. For example, a salesperson;
- *Entity* is a class to describe any thing a worker manipulate with. For example, a sale, an invoice;
- *Communicates* is an association between classes. It represents interaction between classes and can be used for navigation. The association can be one- or two-way. For example, a customer communicates with a sale;
- *Subscribes* is an association between classes. It has a target and a source. The association defines a set of events. For example, a sale is a target or a subscriber. It involves a set of events or sources: a payment, a delivery and an invoice. The subscriber is notified if events occur in the target. A sale if complete when a delivery, a payment, an invoice occur.

The introduced stereotypes are represented by keywords in guillemets. Class stereotypes worker, case worker, internal worker, entity have specific icons in UML notation.

The modeling elements proposed by the Extension can be successfully used for future enterprise solution modeling to construct conceptual models.

9.1.2 UML diagrams

A business model for an enterprise solution can be represented in UML notation. Requirements for a business model are as follows:

- reflect enterprise business processes and organizational structure,
- be easy readable and understandable for a user of enterprise solution,
- be precise and consistent within the model,
- as sources for modeling, use user interviewing, UML diagrams and documentation available at the enterprise which maps enterprise functioning; keep consistency with those sources,
- provide basis further development of an enterprise solution.

A business model can be constructed by a limited set of UML diagrams specially adopted for business modeling. The notation presented in the UML Extensions for business can be used in the diagrams.

The section illustrates a problem of choice between user-friendly representation and scarifying semantics of UML diagrams when modeling business.

The possible choice of UML diagrams is suggested below. The UML diagrams recommended are treated in connection with their traditional usage in business modeling, modification in diagrams notation and possible information sources for constructing the diagrams. The suggested modifications are inspired by ideas from [25]. The purpose of adjustment is to make UML diagrams more popular for non-technical users.

The considered sources for constructing the UML diagrams for business modeling are UML diagrams and other documents explaining enterprise functioning. Those documents are widely used in enterprise solution requirements elicitation or in business reengineering.

The description is supported by small examples referring a Retail Enterprise model developed in chapter 2.

9.1.2.1 Use case diagram

Use case diagrams traditionally serve the following tasks:

- definition of usage requirements for a future enterprise solution,
- analysis of usage requirements for an existing enterprise solution.

Use case diagram is not expressive for modeling business processes. When trying to model behavioral features of enterprise functioning, some problems occur. The topic has been discussed in section 2.3.

Use case diagrams do not show inputs and outputs of the business processes. Use case description represents steps in processes. Trying to model the processes in details, a use case diagram can be too messy and confusing. Activity diagrams and class diagrams with stereotypes are preferable for this purpose.

Constructing a business use case, a question can arise regarding the business actors participated in a use case. How to distribute roles for actors: what actors are internal and what are external? An answer is to make a boundary frame. The boundary can be a whole enterprise, an organization unit, a subunit or a team which is not in the organizational structure but is organized by some business objective. If an actor within the boundary, it is an internal actor.

An actor can play both internal and external roles for different use cases. It is possible to draw the boundaries lines so that each actor within a team will be an internal and outside a team – an external actor.

In order to mark the mapping with organizational structure, it is possible to mention an owner of a use case as an owner of a business process. This can be done as a comment to a use case.

Traditional use case diagram can be modified into two kinds of diagrams: business use cases and business process modeling use cases. As it was mentioned above, a use case diagram is not suitable for detailed behavior modeling. But it can be used as a sketch to name business processes and actors involved.

Business use cases

A business use case hides the actors that function inside a business process. Details of cooperation between internal actors are not available. This diagram gives an overview of enterprise solution usage requirements. It shows main goals of an enterprise and the context.

Text models 1 and 3 presented in chapter 2.4.2 can be used for modeling business use cases. The models introduce external actors Customer and Supplier and one internal actor Enterprise.

Another source is [17] Jacob Nielsen, *The Ace User Experience Vision* representing scenarios for communication between a user and a software system. A user can be modeled as an external actor, and system is the only internal actor.

Business process modeling use cases

Business process modeling use case represents cooperation of internal agents as well as external agents. This approach is suitable for describing business processes.

A business process modeling uses cases show goals and sub-goals of an enterprise and its context. That sub-goal can be shown as entity stereotypes icon described in section 6.4.1. For example, a diagram for the sale process can contain business goals:

- 'Make sale' goal including
- 'Make delivery' and 'Make payment' sub-goals.

To distinguish internal and external actors, case worker and internal worker icons also can be used. Continuing the sale example, the actors will be:

- Customer is a case worker communicating with 'Make payment',
- Salesperson is an internal worker communication with 'Make sale',
- Shipments Clerk is an internal worker communication with 'Make delivery',
- Accountant is an internal worker communication with 'Make payment', 'Make sale'.

Text models 2 and 4 (section 2.4.2) contains several internal actors: salesperson, supply agent, accountant, shipment clerk. These models can be used for business process modeling use case diagrams.

But if to model a use case for the cycle 'Payroll' represented in section 2.2.2.5, then an employee will be an external actor. Even an employee belongs to the enterprise organization structure; he/she acts externally to the payroll process. It is an issue of a boundary question described above.

Information sources for modeling use case diagrams can be UML diagrams and other documents: activity diagrams, organization charts and essential use cases.

9.1.2.2 Class diagram

A class diagram is a static representation of a system. A class diagram is traditionally used for conceptual and domain modeling, but from the enterprise solution developer's viewpoint. A class diagram aims to communicate internal structure of an enterprise solution to others. It is a key diagram in a UML model because it defines basic things of a system and relationships between them.

Usually a class diagram is avoided in domain model documentation for users. Enterprise functioning is modeled more often by other notations, not UML. The argumentation against UML class diagrams is that they are complicated and massive for a user and not visually expressive in a user-friendly way.

But the meaning of class diagrams for business modeling is underestimated. It can describe main entities of enterprise functioning and their interrelations. A class diagram helps to obtain an overview of enterprise functioning.

An example refers to a class diagram based on the REA model of a retail enterprise introduced in section 2.4.3. It is easy for a user to percept information from the diagram, because a user can recognize familiar things represented by stereotypes: resources, events and agents. Attributes of classes demonstrate properties of those things, operations show responsibility areas of the things, association ends names explain roles the things play in business processes, relationships between things illustrate structure of enterprise functioning. A class diagram gives a general abstract representation of things corresponding to real world.

But stereotypes are not the only way to make class diagrams user-friendly. A business model can have several class diagrams hierarchically structured into packages. A high-level diagram will contain packages; lower level diagrams will explain the packages. This approach will help a user to understand and discuss the model gradually starting from the easiest representation and coming to more detailed diagrams if necessary.

To construct the high-level class diagram it's convenient to use predefined business stereotypes described in section 6.4.1. Stereotypes must be shown as icons. If the predefined stereotypes are not enough to express enterprise functioning, new stereotypes with icon representation can be defined. This diagram may be called business class diagram.

Enforcing visual expressiveness for a user, a business class diagram can be constructed as a combination of use case and class diagrams notation using stereotypes. The diagram can contain a class representing a business process, a class showing input into that process (Cash), output (Receipt), and actors participating in a process. To show the relationships between classes, one-directional associations and classes' role names can be used. Not all the relationships between classes may be drawn at the diagram, but only between a business process and its input, its output and its actors. This will lead to loss of some semantics, but will be easier for a user to understand.

For example, a Payment class with <<business process>> stereotype may be associated with Cash <<resource>> playing a role 'is input for'.

The Payment <<business process>> with a role 'has output' can be associated with <<document>> Receipt.

A Customer <<actor>> can play a role 'makes', am Accountant <<actor>> plays a role 'accept' in associations with the Payment <<business process>>.

The most relevant sources for construction a business class diagram modeling can be activity diagrams, REA model, literate description of business processes, usage scenarios and user stories. Other sources can be collaboration diagrams, sequence diagrams, state chart diagrams.

9.1.2.3 Sequence diagram

Sequence diagrams are not very popular in business process modeling. A diagram tends to be quite big, when a sequence of actions is long. But the diagram notation is easy to understand and read.

Usually sequence diagrams are used for modeling the logic of a usage scenario or a path through one or more business cases. A mistake would be to represent the entire usage scenario in details at one diagram.

Business dynamic sequence diagram can be used to view dynamic behavior of a business process. To make the diagram more illustrative for a user, objects may be represented as icons. Messages should be as much as possible concise and short, trying to keep natural language phrases.

Sources for modeling business dynamic sequence diagrams are class diagrams, use cases, usage scenarios, user stories, literate business process descriptions.

9.1.2.4 Collaboration diagram

In business modeling practice, collaboration diagrams are used for representing behavior of complex object interaction. But the diagram is not expressive enough and takes some efforts from a user to catch its semantics.

A business dynamic collaboration diagram should use icons to represent classes. A separate diagram can be modeled for every business use case. The diagram is suggested as a supplementary notation. It’s preferable to use activity diagram instead of.

Sources for business dynamic collaboration diagrams are class diagrams, use cases, usage scenarios, user interface flow diagrams, user stories, literate description of business processes and other documents.

9.1.2.5 Activity diagram

Activity diagram is widely used for business modeling. It seems to be the most successful graphical notation to represent a dynamic behavior of enterprise functioning. Activity diagrams are used for analysis and design of business processes and business rules governing the processes. The diagrams are rather suitable for design the logic flow of any complex operation.

Business activity diagrams may be represented in two levels: business dynamic high level activity diagram and business dynamic detail activity diagram. The rationale is the same as to present class diagrams at different levels: help a user to understand a business model.

A business dynamic high level activity diagram does not show any actors involved in a business process. The diagram is very simple. It looks like a state machine including a small number of states of a business process:

- an initial state,
- an intermediate state represented by an icon. It names the business process,
- a final state symbol.

A business dynamic detail activity diagram provides an overview of the logic for a business process. It explains a high level activity diagram. The diagram can use swim lanes to mark spheres of activity for actors participating in a business process.

An activity diagram for a sale business process represented in section 2.4.3 is a variant of a collaboration diagram.

Documentation sources for modeling business dynamic activity diagrams are class diagrams, essential use cases, literate description of business processes, usage scenarios, user stories.

9.1.2.6 State chart diagram

State chart diagram is not widely used for business modeling. The main reason is that state chart diagram does not consider participants of a business process. It only shows states of a business process and the direction of activity flow.

Traditionally the state chart diagram is used for analyzing and modeling behavior of a business process. But usually the business process involves a number of sub-processes interacting with each other, so to illustrate the interaction is quite problematical.

But state chart diagram can be used more efficiently in combination with other diagram notations for modeling business logic. State chart diagram can be rather helpful in being part of an activity diagram. The topic is discussed in section 6.1.3.3 Agent-based UML. State chart is used in combination with activity diagram to model business processes as automata.

The main advantage of state chart diagram is the possibility to define complex states as macros. This mechanism helps to model behavioral patterns.

The sources for state chart diagrams are literate description of business rules, class diagram, source code, system use case, usage scenario.

9.1.3 New UML diagrams

This section gives a suggestion for adjusting UML notation for business modeling.

The hybrid and agile diagrams introduced below are intended to make UML notation more popular among non-technical users, for example, domain experts participating in the enterprise modeling process.

The agent-based UML is proposed to model business processes in a more detailed level stressing communication between participants of business processes.

9.1.3.1 Hybrid diagram

Hybrid diagrams have no analogous in UML. The diagrams combine UML diagrams notation in order to represent both dynamical and static aspects of enterprise functioning at one picture. Agile diagrams should use icons to represent classes.

A hybrid diagram can be constructed as a mixture of use case goals and activity diagrams with swim lanes introduced earlier in the section. This kind of diagram will contain elements describing:

- Actors,
- A business processes,
- Input to the business processes,
- Output of the business processes.

Individual icons may be assigned to represent different kind of classes for input/output for business processes. Use cases and use case models also can be input for a business process.

The central concepts of the diagram are business processes. All other icons are supplementary classes associated only with business processes (associations between those classes are not shown in order to keep the diagram simple). The supplementary icons serve to show resources, documents and actors involved in business processes. Cooperation between business processes is not represented as associations.

Disadvantage of the hybrid diagram is that it's too general. Not all the relevant associations are shown. It can be used as one of the views in a business model.

9.1.3.2 Agile diagram

Agile diagram is also a hybrid diagram. But it's supposed to spend less time for its construction.

Agile diagram aims to represent a business process. The steps in agile diagram modeling are as follows:

1. make a business use case diagram and use it to
2. show business use case goals opposite to the enterprise activities.

The agile diagram will include use case goals (not classes representing process as in a business class diagram), actors, inputs and outputs to the business process.

The agile diagram is very simple but still it captures the essence of a business process. It is akin to a REA model showing events, agents and resources

Possible sources for hybrid and agile diagrams can be UML diagrams (business use case diagrams, business activity diagrams and business collaboration diagrams), organization flow charts, organization structure documents and others. Agile diagrams can be constructed after or simultaneously with the named UML diagrams.

All the previously named diagrams in section 6.1.3 and suggested modifications for diagrams in section 6.1.2 are only attempts to find out techniques for adjusting UML notation for business modeling and make it more popular for non-technical users. The diagrams using icons should be considered as a first step in business model representation. They must imply refinement diagrams. But still the described diagrams can serve as a guide for a business model. Users can recognize themselves as participants of business processes and can see where to go further to obtain more detail-elaborated description. The supposed diagrams can be an orientation guide both for users and developers of an enterprise solution.

The diagrams notation should be improved further. Ideally the diagram notation must be supported by more elaborated patterns describing a particular business needs and bound to a particular domain.

9.1.3.3 Agent-based UML

Enterprise solution can be viewed as a multi-agent application. *Agents* are active objects running on behalf of users. Agents act concurrently, using browsers and cooperating with the enterprise solution business logic.

As it was described in the section 3.1, business processes involve actors who operate with resources. Actors interact with each other at different steps or actions composing a business process. If a business process engages several actors and consists of many actions, interaction between agents becomes complicated.

Agents will represent any actors of a business process. If consider an enterprise solution involved in enterprise functioning, enterprise solution and its parts also can be actors.

Agents interact with each other by messages. Agents must understand each other and know which messages to expect in a particular situation. Agents can be separated into groups according to the activities. Each group of agents will have:

- a set of incoming and outgoing messages,
- a set of rules defining the sequence of actions.

A set of messages can be defined by a role an agent plays in a business process. One agent can play several roles. A set of rules determines an agent's behavior. These are business rules. An *interaction protocol* regulates interaction between agents. It reflects business rules and determines the flow of control between roles and within a role of agents.

UML allows to model interaction between agents. There is an extension of the language called Agent UML (AUML). It suggests a hybrid diagram namely *protocol diagram* for multi-agent communication. The diagram combines elements from the interaction diagram and the state chart notation.

But existing UML notation also allows to model interaction between agents.

To analyze agents' behavior, collaboration diagram can be used. The diagram represents agents participating in a use case and the sequence of messages between them.

In section … the activity diagram with swim lanes represents interaction between actors participated in the sale business process. The difficulty is that activity diagrams can be too big and difficult to read if a process is complicated and modeled in details.

The solution can be to use using stereotypes to improve visual expressiveness of activity diagrams with swim lanes:

- <<role>> describes a role of an agent or group of agents (like Customer, Enterprise). A role names a space of control flow for bounded by swim lanes. It is an area of an agent activity;
- <<channel>> describes the linkage between roles via messages. It names a space of control flow bounded by swim lanes. It is an area of inter-agents communication;

- <<synchronization point>> describes the point in time where agents exchange messages. It belongs to a channel. It is a semantic extension of UML;
-

- <<send>> describes a transition of an incoming message. An incoming message comes from an agent (sender) to another agent (receiver) through a synchronization point;
- <<receive>> describes a transition of an outgoing message. An outgoing message comes from an agent (receiver) to another agent (sender) through a synchronization point when the incoming message has been received;
- <<timeout>> is connected to a synchronization point. When the timeout is reached and a receiver did not deliver a message, control flow of the receiver goes to the state pointed by the timeout transition.

UML is powerful notation to represent activity states of a business process as automata. Complex states can be defined as macros. A macro state can accumulate several states. A macro can be named and used in an activity diagram. This mechanism helps to model hierarchical structured protocols and use them as patterns.

B

9.2 Conceptual modeling

Conceptual modeling is a process of describing in formal notations some aspects of the real world. The result of the process is a conceptual model. A *conceptual model* is simplified formal description of a domain.

A conceptual model may be used in software engineering as input to the design stage. This topic is treated in section 6.2.1 in respect with business enterprise and enterprise solution and in section 6.2.2 regarding modeling software applications in general.

Conceptual modeling requires retrieving and combining concepts within a domain, which is formal terminology work, tools and techniques for describing concepts.

The term conceptual modeling can be explained in respect with conceptual structure and models.

Conceptual structure is an abstraction being a composition of concepts and relations between them. An example of conceptual structure is terminology. It combines terms which are concepts, and generic, partitive, associative relations between concepts. Another example is E/R model introduced in chapter 2.1. The E/R model contains entities and relationships between them.

The main purpose of conceptual modeling is to give a description of a domain. *Description* relates a subject to a content. A subject is a thing in a domain. A content is an object of a description and can be expressed as statement(s) in some language.

Description can be specialized into designation and definition found as main concepts in terminology work. *Designation* is a word or a word phrase naming a thing in a domain. *Definition* includes both a subject and an object of description. The subject and the object are described in the same language.

A *model* is a specialization of description. It is a description of something in a domain. A model can be considered as an abstract structure partially describing things in a domain. It is impossible to give a complete description of a domain. The abstraction depends on a viewpoint to the domain. It defines the scope of the model: what concepts should be included in the model.

This explanation of a model has similarities with a Conceptual framework for architectural description given in the standard IEEE Std 1471-2000. The standard was introduced in section 3.2.3. A model being part of an architectural description of a software-intensive system depends on viewpoints.

The conceptual model content is a composition of model elements. A *model element* describes object types and their properties in a domain. An object type is similar to a class in UML or a general concept in terminology. The conceptual model can also include individual concepts as examples of generic concepts.

The conceptual model represents single object types, relations between object types and structures of object types. Relations can be specialization/generalization, part/whole and association.

Modeling includes the following three aspects:

- *Modeling notation* is a representation instrument for the model. The notation can be based on graphical elements or languages;
- *Modeling semantics* considers the meaning of a model. The meaning can be divided into the meaning of modeling notations and the meaning of the model specified by its content. The modeling notation may have a formal syntax (grammar) with non-terminal and terminal symbols. Terminal symbols are chosen by a constructor of the model. Non-terminal symbols are assigned to sentences in the grammar in a compositional way. Semantics is concerned with this assignment. These non-terminal symbols can be mapped to the real world by an *interpretation*;
- *Model engineering* is a mental process having a part of the real world as a subject and resulting a model of the domain. The process requires a domain expert and a modeling expert. A domain expert has knowledge on the domain. A modeling expert has skills in modeling and knowledge on modeling language.

9.2.1 Enterprise conceptual model

Understanding and describing enterprise domain is the first step in constructing enterprise solution. The conceptual model of enterprise can serve for this purpose.

As it was described in chapter 4.1.3, conceptual modeling has an important meaning for Microsoft enterprise strategy. The strategy considers a conceptual view as bringing the most valuable impact into application architecture.

In Navision Jamaica model described in chapter 5, the constructing of a business model is the main and the only stage of creating an enterprise solution.

This section defines the goal of enterprise conceptual modeling, stakeholders and stages of the modeling process, most relevant representation viewpoints and possible modeling notations. REA models presented in the report are referred as examples of enterprise conceptual models. Some problematical aspects and possible directions for enterprise conceptual modeling are mentioned.

The goal of constructing an enterprise conceptual model is to understand enterprise functioning. The resulting model is called a *business model* in both the Navision Jamaica (section 5.2) and the Microsoft enterprise strategy (section 4.1.3.1).

Enterprise conceptual model should represent:

- business processes,
- resources involved in the processes,
- agents participating in the processes.

Stakeholders of the process of constructing an enterprise conceptual model are:

- *an architect* having a role: use the knowledge of a domain expert to model an enterprise solution;
- *a domain expert* having a role: explain enterprise functioning to an architect. A domain expert can an enterprise stakeholder namely a manager, a functional managers, a senior stuff member.

An enterprise solution is considered as software-intensive system. It is reasonable to use the ideas of the standard IEEE Std 1471-2000 introduced in section 3.2.3. The enterprise conceptual model should meet different viewpoints to satisfy different stakeholders of enterprise solution. Enterprise conceptual model can have multiple views structure.

Enterprise conceptual model should meet at least three basic viewpoints:

- Business – to make an enterprise model mapping enterprise goals hierarchy, business processes, organizational structure. The goal of this viewpoint is to model business;
- Functional - to make an enterprise solution model reflecting the enterprise solution rationale, environment, stakeholders, and services. The goal of a functional viewpoint is to implement functional requirements;
- Technology – to make a model of enterprise solution as technology product involving software and hardware components, information and data needed. The goal of a technology viewpoint is to implement non-functional requirements.

The listed viewpoints can be grouped in another way. Microsoft enterprise strategy names four viewpoints: the business perspective, the information perspective, the application perspective and the technology perspective. Functional viewpoint is divided into two sub-viewpoints: information and application perspectives. The perspectives are introduced in section 4.1.1.

All the level of enterprise functioning can be mapped into enterprise conceptual model.

Stages in the process of constructing an enterprise conceptual model will be:

- capturing concrete experience of modeling enterprise functioning into patterns;
- observation and reflection of the experience and finding concepts;
- formalization of abstract concepts and generalization; setting up part/whole and association relationships;
- testing concepts in situations of different relevant viewpoints;
- structuring concepts into a model reflecting a concrete viewpoint(s);
- evaluating experience in using the model.

The notations for an enterprise conceptual model could be:

- **REA**. In general, REA notation could serve in conceptual, information and data levels. The REA diagrams represented in the report are at the conceptual level. The REA model of a Retail Enterprise shown in Appendix D is an example of enterprise conceptual model. Other examples are REA diagrams done for representing cycles of the Retail shop enterprise. The diagrams are introduced in section 2.2.3.

  Disadvantages of REA notation are that it's impossible to model the behavior and difficult to model high level or low level of details of business processes (for example, actions composing a business process or business processes as individual concepts composing a value chain);

- **UML/OCL** notations are very suitable for enterprise conceptual modeling. The notation has been discussed in chapter 6.1.The UML notation permits to model behavior and to produce arbitrary level of detail elaboration. UML notation has a built-up mechanism for extension, so it gives some reasonable freedom to adjust the notation according the modeling needs. OCL language serves for setting up invariant for models.

The named notations may be used together in conceptual modeling but to represent different viewpoints. UML/OCL notations are more flexible.

There is a lack of theoretical foundation for enterprise conceptual modeling. The possible solution could be using business ontology for constructing and validating enterprise conceptual models. It might be also created an ontology for enterprise conceptual modeling itself. This ontology could consult an architect on what notations to use, what viewpoint to consider, what models to construct, what steps to do in the constructing process. These two ontologies must be combined together.

Another problematical aspect is a lack of software tools for enterprise conceptual modeling. Requirements for the tools:

- Tools should have a language, an interpreter of the language, a database. This topic is closely corresponded to the theoretical foundation aspect named above;
- Graphical representation of a model and modeling process;
- Integration with enterprise solution tools.

This theme is discussed in the following.

9.2.2 Application model

As it was mentioned in chapter 3 Business Enterprise and Enterprise Solution, chapter 4 Microsoft Enterprise Strategy and chapter 5 Navision Jamaica Model, the impact of first stages in software engineering brings considerable value to the resulting software products. The software engineering process becomes more model-oriented. In the future with capturing business and programming experience, the software engineering will be replaced by modeling engineering.

Modeling engineering will be focused on making of application models which can be used for automatic code generation. Application modeling is a stage in the application development process using Model Driven Architecture technology. The technology is described as following.

Model Driven Architecture (MDA) is a software development technology. MDA provides methodology for development of software applications capable to work on different platforms.

The MDA concept has been developed by the Object Management Group (OMG) consortium. The main purpose of OMG is standard and specification development to regulate the practical use of new information technologies for various hardware and software platforms. One of the technologies is UML. OMG takes an active part in further UML notation development.

The basic idea of MDA is to distinguish business logic modeling as a separate and significant stage in software engineering process. The business logic concept was introduced in section 4 Application Architecture, Conceptual View. Business logic implements business rules which are combined in business process algorithms. Microsoft enterprise strategy calls business logic *enterprise solution logic*.

Following the idea of NDA, Impact of human efforts in application development are shifted from programming to modeling.

According to MDA, the first step in development of a software application is creating an application model. An *application model* defines:

- component parts of an application,
- structure of an application,
- behavior of an application.

This model is called *Platform Independent Model* (PIM). PIM should be written in UML. Creating an application model, a modeler has no concern with implementation details. These details are different for concrete software and hardware platforms. PIM model of an application is to be created once, but may be used for different platforms.

At the second step, a modeler should choose the concrete platforms for running the application and create *Platform Specific Models* (PSM). These models adapt PIM with one or more integrated development environments. The models also include interfaces to support interaction between the application and other applications.

The last step is automatic code and database generation. The code generated is based on PIM and PSM models. If there are several PSM models, the code has to be generated for every PSM. The result of the code generation is a software application capable to run at different platforms.

MDA gives some advantages to application developers:

- localization of all business logic at one place – application model,
- it allows creating an abstract model for the whole application independent from a platform,
- automatic code and database generation.

MDA is not a competitor to Microsoft .Net introduced in chapter 4 or to other modern platforms for application development as CORBA (Common Object Request Broker Architecture), J2EE (Java2 Enterprise Development Edition), Sun One, Borland Enterprise Studio. MDA stands at a higher level of application development. It can be considered as a meta-technology for application development.

OMG consortium supposes the MDA technology can be applied for any of the present platforms. The main task to adjust MDA for some technology is to create a PSM model.

Due to the strong advantages of MDA, OMG predicts consortium optimistic future for MDA technology. MDA is expected to be the leading technology for application development. MDA will not be affected by new platforms or present platforms' development.

Perspectives for MDA development include automatic GUI generation.

Future application development scenario will be a cycle from the following stages:

1. Creation/changes of a model,
2. Code generation.

If necessary, a developer can change the model, but he/she does not need to change the code.

Realization of the process enquires a tool for interpreting a model and code generating.

Development of MDA will cause a shift in specialization of developers of software applications. The knowledge and skills in application domain then on programming will bring more value for developers.

A modeling language can be UML.

Referring MDA to the Microsoft enterprise strategy discussed in chapter 4, MDA might be a possible direction for development Microsoft enterprise strategy.

Navision Jamaica modeling described in chapter 5 can be considered even as more advanced than MDA in regards with application development process. The Jamaica application development process includes only one stage – modeling. The application model is directly executed. But the application model in Jamaica has some limits discussed in section 5.4.

Further development of MDA technology should focus on the following topics:

- Application models should be concise, precise, expressive, consistent, easy understandable and readable;
- Interpretation tools of application models should be able to transform an application model directly to a solution (as it is done in Navision Jamaica).

These development directions are closely connected with development of conceptual modeling, applying ontologies and using UML notations. UML notation has been discussed in section 6.1. Business enterprise and enterprise solution ontologies are considered in section 6.3.

9.3

9.4 Ontology and enterprises

The notion of ontology came from philosophy to computer science. In philosophy the term *ontology* is a synonym for metaphysics. It is a branch in philosophy concerning with the nature of existence in the real world.

In computer science *ontology* means a set of concepts and their interrelations. This is a complete description on what exists in a subject area. A subject area could be a model, a software system or an application domain.

Enterprises can be an area for constructing ontology. Enterprise ontology may be applied for referring and teaching the subject of enterprises and for business and software engineering. Enterprise ontology will pursue the following goals:

- to provide informational base for sharing knowledge on enterprises. Enterprise ontology can be used as a formal standard for enterprise stakeholders;
- to supply reuse of knowledge on enterprises;
- to explicitly specify assumption for the enterprise domain,
- to assure that a high level specification for enterprise solution is well-defined and evaluated with respect to the real world;
- to assist navigation in user interfaces and business logic of enterprise solutions,
- to analyze enterprise domain. Regarding enterprise solutions, enterprise ontology can be used for consistency checking of models of enterprise solutions.

The subject of enterprise must be considered together with related areas. The domains relevant for enterprise ontology are:

- social domain,
- enterprise domain,
- information technologies domain,
- enterprise solution domain.

Figure 6.3 illustrates the interrelations of the domains using Venn diagrams. Domains are considered as sets of concepts and can be visualized as areas drawn on the plane. Those sets can intersect, join or include each other. The interrelations between the considered domains can be viewed as a chain of inclusions (Fig.6.3.a):

social domain > enterprise domain > enterprise solution domain > information technologies domain.

From the other hand, enterprise solution domain can be represented as intersection of enterprise and information technologies domains (Fig.6.3.b).

The subject of ontology has been treated by modern philosophers. John D. Sowa and Nicola Guarino have brought the most considerable impact to modelling ontologies. Their work helps to bind ontology discipline and information technologies. This chapter refers to some of their ideas.

A process of constructing ontology is similar to setting up a terminology system. Terminology provides a basis for ontology construction.

The work done in the chapter is directly linked to the previous chapters 2-5. These chapters give foundation to establish terminology for the areas of enterprise and enterprise solution. The terminology is represented as term dictionaries in Appendix B. Appendix C is done in cooperation with Tom Østerby. The appendix gives a refinement for the term dictionaries and provides a base for constructing enterprise ontology.

The chapter describes the subject of ontology in general and enterprise ontology. An attempt to construct enterprise ontology is made.

The chapter consists of the following parts:

- Section 6.3.1 contains ontology overview. The section gives a definition for ontology and introduces types of ontologies. The section also describes ontological entities which can be simple concepts, relations and mediations, and ontological relations. Ontological entities and relations are explained in

Fig. 6.3. *Venn diagrams for social, enterprise, enterprise solution, information technology domains.*

- Section 6.3.3 gives a description of Sowa's strategy for a high-level ontology called Sowa's diamond. Sowa's ontology can be considered as a framework for business ontology.
- Section 6.3.4 describes the need for business ontology within different kind of models discussed in chapters 2 – 5: REA model, business enterprise, enterprise solution, Microsoft and Navision Jamaica strategy. The process of constructing ontologies is illustrated by examples of REA and business enterprise and enterprise solution domains.

  The central source for setting up business ontologies is Appendices B and C. Business ontology' concepts are viewed as categories of Sowa's diamond.

9.4.1 Ontology overview

In general ontology can be identified as a system of categories of a real world. This system of categories does not depend on a representation language but appears as a consequence of a particular viewpoint to the world.

Ontology structure includes simple concepts, relations and mediations according to Sowa. The relations can be generalization, part/whole or thematic associations.

The syntax and semantics of ontology can be verified via axioms.

9.4.1.1 Definition

There are several definitions for ontology corresponding to different viewpoints. The section gives a brief description of ontology definitions introduced by N.Guarino and P.Giaretta in [23].

Ontology is a philosophical discipline studying the subject of existence.

Besides that ontology can be viewed as a conceptual system. This conceptual system can be supposed as a basis for knowledge base. The conceptual system will provide a set of concepts and their semantics.

Another point of view also considers ontology at the semantic level, but as a formal semantic description. Ontology expresses the meaning of things in formal terms and structures.

Ontology can be understood as a specification of a "conceptualization". This definition is popular within artificial intelligence society. According to this viewpoint, ontology provides a specification for knowledge representation.

There is a set of viewpoints which view ontology as a syntactical system providing the order and methods for describing knowledge. This understanding of ontology comes to the following three definitions.

Ontology is a representation of a conceptual system through a logical system. The logical system can be characterized by specific formal properties. Here the features of the concepts are emphasized as a key aspect for a concept representation within ontology. Another standpoint proposes the logical system to be characterized only by its specific purposes. Context of a concept is a governing factor for the concept representation within ontology.

Another understanding of ontology as a syntactical system says that ontology is the vocabulary used by a logical theory.

The last definition given by N.Guarino and P.Giaretta also explains ontology as a syntactical object. Ontology can be viewed as a meta-level specification of a logical theory.

Summing up, ontology can be described as taxonomy of terms, terms' definitions and inference rules. Ontology depends on a context, is valid within a group and based on common meaning of terms, their relations and structures. Ontology is independent from a representation language.

According to John D. Sowa, ontologies can be classified as terminological, axiomatic, and prototype-base. Terminological and axiomatic ontologies are described in the following being relevant for business modeling.

9.4.1.2 Terminological ontology

Terminological ontology contains concepts and relations which are not specified by axioms and definitions. The conditions for use of the concepts are not determined.

The concepts may be specified by relations as generalization/specialization or part/whole. These relations set up a position of a concept relatively to other concepts in the terminological ontology. But the relations between the concepts do the sufficiently define the concepts.

Terminological ontology can be established as a result of terminological work for a domain. Concepts of a domain are represented by designations.

9.4.1.3 Axiomatic ontology

Axiomatic ontology is a terminological ontology where concepts and relations are specified by axioms and definitions. The complexity of the logic is not restricted. A goal of axiomatic ontology is to construct a theory for a domain.

Axioms and definitions specifying axiomatic ontology are stated in logic or a computer-oriented language that can be translated to logic. Such languages are Prolog, SML, Lisp, SQL, OCL and others.

Comparing axiomatic and terminological ontologies, axiomatic ontology is smaller than terminological, but supports more complex inferences. The difference between the ontologies is in complexity of logic which specifies concepts and relations. Axiomatic ontology has notations for forming the ontology and for producing deductions.

9.4.1.4 Ontological entities

Description of ontological structure assumes a common agreement that ontologies are distinguished as:

- *universals* are comparable to general concepts in terminology and classes in UML. The ontology of universals is *primary ontology*;
- *particulars* are analogous to individual concepts in terminology and instances in UML. Every particular is connected to exactly one universal. Particulars help to understand universals.

9.4.1.5 Ontological relationships

There are two types of relationships for ordering concepts in ontologies: generalization and part/whole.

Universals in ontology are ordered by *taxonomic relationships* which is *isa*. The relationship *isa* is similar to generalization in object-oriented domain. Ordering in terminology is called generic relation.

There are some dangers using generalization relationships as confusion of senses about concepts, reduction of senses caused by different views to the concepts, overgeneralization, restriction by role type for a concept, confusion of taxonomic roles for concepts.

The ordering in ontology can be represented graphically as a lattice or a tree. Both representations have advantages and discourages. A *tree structure* is simple, it provides simple inheritance: one child has only one parent. But it’s impossible to represents all the concepts using the structure.

A *lattice structure* gives more realistic representation of domain concepts. A lattice supports multi-inheritance: one child can have many parents. This approach reflects multiple-view representation of a concept. Moreover this structure allows making induced queries, which is important for navigation. The topic was mentioned in regards to Navision Jamaica system in section 5.6.

In section 6.3.4 Sowa’s diamond is introduced. It is an example of the lattice structure for ontology.

Another type of ontological relationships is *part/whole*. It corresponds to aggregation and composition in UML and partitive relations in terminology.

There is non-hierarchical kind of ontological relationships called *thematic associations*. Two concepts are linked together via roles. It is the analogue of UML associations and association ends explaining the roles which classes play in the associations. In terminology, there are associative relations.

9.4.2 Ontological engineering

Ontological engineering is a process to construct ontology. The process is similar to making terminology work. It focuses on identifying and collecting terms in a domain. The result of ontological engineering is a model of ontology.

It's important to note, that there is no single correct solution as a result of constructing ontology. The ontology can be evaluated only qualitatively.

There is a set of requirements for ontologies proposed by Gruber [24]:

- *clarity* implies that ontology effectively reflects the meaning of terms;
- *coherence* means that all the definitions should be logically consistent and all derived statements should follow the ontology's axioms;
- *extendibility* stands that term dictionaries shared within ontology might be extended without reconsidering existing terms;
- *minimal encoding bias* denotes that the ontology model should be specified at the level of representation, not by symbol coding;
- Minimal ontological commitment requires that ontology should include only most relevant assumption on a domain. Some space should be left for further extensions and specifications.

Ontological engineering is a new branch in computer science. There are no common approved and verified techniques for constructing ontology. Ideally this kind of technique should specify the procedures for constructing ontology and at what stages of the process to do these procedures.

The process of constructing ontology involves the following stages:

- *specification* defines purposes for constructing ontology, use and users of ontology;
- *conceptualization* provides a conceptual model for a domain;
- *formalization* transforms conceptual model into formal model;
- *realization* implies coding of the formal model in a programming language.

The named stages involve a number of sub stages for intermediate knowledge representation. These stages are accomplished not consecutively but as knowledge are accumulated. There is a recommendation to start from a glossary of terms, then construct concept classification trees and binary relations diagrams and finally the intermediate knowledge representation.

A simple ontology can be constructed by a sequence of steps:

1. Define the domain and a scope of ontology, its purpose and use;
2. Check existing ontologies if they can be reused as a part of the ontology being constructed;
3. Determine terms relevant for ontology;
4. Set up classes and class hierarchy;
5. Define slots which are properties of classes;
6. Define facets which are cardinality of slots, slot-value type, range of slots;
7. Make instances of classes.

9.4.3 Sowa's division - diamond

Sowa's diamond is graphical representation of Sowa's high –level axiomatic ontology. is a lattice, where the top represented everything in the world. The bottom represents the absurd type. Sowa's diamond consists of categories which reflect three dimensions of Sowa's axiomatic ontology; abbreviation for categories is shown in parenthesis:

1. Sowa's dimension represents Peirce's categories Firstness (1), Secondness (2) and Thirdness (3):

– *Thing* is a simple concept;

– *Relation* is an association between one or more simple concepts;

– *Mediation* is a structure. It combines one or more things and relations between them,

2. Physical/Abstract dimension:
   – *Physical* (P) is an entity having a location in space and time,
   – *Abstract* (A) is an entity having no a location in space and time,
3. Temporal dimension:
   – *Continuant* (C) is an entity being a stable object over some time interval,
   – *Occurrent* (O) is an entity being a process which is not stable during some time interval.

These three dimensions give 12 possible combinations which represent high-level categories in Sowa's ontology. There are also 6 intermediate categories resulting combinations of Peirce's and Physical/Abstract dimensions: Actuality, Form, Prehension, Proposition, Nexus, Intention.

The 12 Sowa's basic categories are briefly introduced below. The abbreviated types of dimensions are shown in parenthesis:

1. Object (P1C) is a Physical Thing considered as a Continuant. It can be recognised by properties stable during the lifetime of the Object;
2. Process (P1O) is a Physical Thing considered as an Occurrent.

   A Physical Thing can be viewed as stable or dynamic according to the level of detail and time scale;

3. Schema (A1C) is an Abstract Thing considered as a Continuant. The time or time-like relationships are not particular. According to Sowa, a Schema is an uninstantiated Thing;
4. Script (A1O) is an Abstract Thing considered as an Occurrent. Script represents time or time-like sequences;
5. Condition (P2C) is a Physical Relation considered as a Continuant. Object is a prehending entity for the Condition. Object has a stable relationship to prehended entity during some time period;
6. Behavior (P2O) is a Physical Relation considered as an Occurrent. Object or Process is a prehending entity for the Behavior. Object has a changing relationship to prehended entity during some time period;
7. Description (A2C) is an Abstract Relation considered as a Continuant. Sowa explained a Description as a Proposition about a Continuant. A Schema is related to the structure of some Continuant. Description can be viewed as an application of a Schema to describe a Physical or Abstract entity;
8. History (A2O) is an Abstract Relation considered as an Occurrent. According to Sowa, a History is a Proposition about an Occurrent. A Script is related to the structure of some Occurrent. A History does not need to be true;
9. Situation (P3C) is a Physical Mediation considered as a Continuant. Within a Situation, Sowa supposes an animate agent (not necessary a human being) who differentiates a region of time and space and has some Intention;
10. Execution (P3O) is a Physical Mediation considered as an Occurrent. An agent directs the Execution following some Purpose. This Purpose explains the behaviour of some Object related to some Process;
11. Reason (A3C) is an Abstract Mediation considered as a Continuant. Sowa explains a Reason as an Intention of some agent concerning some Continuant. Reason gives explanation, why a situation is certain;
12. Purpose (A3O) is an Abstract Mediation considered as an Occurrent. According to Sowa's explanation, some agent has a Purpose and directs the Execution. The agent has an Intension as a goal motivation for its activity.

Sowa's diamond is a topic for discussion. It can be extended with social and mental dimensions. Social will show what is connected to social life. Mental will consider what is going on in a human being's mind.

The most general concepts found in the term dictionaries may be connected to categories in Sowa's diamond. It can serve as a high-level ontology for business ontology.

UML can be used for representation of Sowa's diamond.

9.4.4 Business ontology

The need for business ontology has a number of reasons:

- Ontology supplies with reusable conceptualization on business. This conceptualization is based on shared meaning for concepts among the enterprise stakeholders.

- Ontology can advise what concepts and relations have to present in a business model.
- Ontology can give directions on what kind of decisions should be taken during constructing a business model.
- Ontology can be used to describe a business model for a particular enterprise domain through specialization and instantiation of concepts.
- Ontology can represent a set of business issues.
- A business model based on business ontology for a particular domain can serve as a precise set of functional requirements for enterprise solution.

Chapters 2 - 5 describe the subjects of REA model and a retail enterprise, business enterprise, enterprise solution, Microsoft enterprise strategy, Navision Jamaica model. The description provides a basis for terminology for the domains resulted in Appendix B.

Appendix C provides a refinement for the term dictionaries represented in Appendix B. Appendix C defines a syntactical structure to represent terminology and contains terms and definitions satisfying this structure. It is the more formal representation of the terminology.

Appendices B, C is a base for enterprise ontology. The terminology can be extended with new terms relevant for enterprise domain. These terms may be derived from the text of the report or from external sources.

This chapter represents an attempt for constructing enterprise ontology. Appendix C is already a terminological ontology. The goal is to refine Appendix C that the formal ontology will be a result using the Sowa's diamond and some formal notation either a logic language or UML.

To formalize the terminological ontology, the concepts can be viewed as things, relations and structures. Most of those concepts will relate to a *social* area which is proposed as an extensional area for Sowa's Abstract/Physical dimension. So the concepts for the terminology will be:

- social thing,
- social relations,
- social structures

recognized in a Time Point (Occurent) or a Time Period (Concurrent).

Things can be described by their properties or attributes. An attribute is characterized by name, category and value type:

(1) attribute (AttributeName, Category, ValueType),

where *Category* is one of the 12 basic categories or specialized categories of the12 basic categories in Sowa's diamond.

Relations are determined by definitions. A definition name of a relation has a number of things involved into a relation and roles of the things in the relation:

(2) relation (RelationName, list (Thing x Role)).

Constraints for the definition of relation:

(3) every Thing in relation has a Role,

(4) number of Things in relation is not less than 2.

Structures are formed by a set of things and relations. Structures can be recursive, so any structure is identified by a name and sets of things, relations and structures:

structure (StructureName, set (Thing), set (Relation), set (Structures)).

Constraints for the definition of structure:

(5) every member in set of Things which forms a structure is found in the list of Things defining a Relation in set of Relations which forms the structure

and conversely

(6) every member in the list of Things defining a Relation in set of Relations which forms a structure is found in the set of Things which forms the structure,

(7) every member in the set of Relations which forms a structure holds constraints *(3), (4)* for relations,

(8) every member in the set of Relations forming a structure is found in the set of Relations forming a member of the set of Structures which forms the structure

and conversely

(9) every member in the set of Relations forming a member of the set of Structures which forms a structure is found in the set of Relations forming the structure,

(10) every member in the set of Structures which forms a structure holds *(5), (6), (7).*

This chapter represents a selection of some general concepts from terminology for domains of REA model, business enterprise, enterprise solution, Microsoft and Jamaica. The selection is done more elaborated for REA and business enterprise. Other domains could be treated in the same way to construct ontologies.

REA concept system represented in section 6.3.4.1 shows a special viewpoint where everything can be divided into things and relations between things. Retail enterprise is considered as a REA model.

Business enterprise and enterprise solution concept systems (sections 6.3.4.2, 6.3.4.3) give a base for specialization for Microsoft enterprise strategy and Navision Jamaica conceptual systems.

The formalization of some of the concepts is done for REA terminology illustrating the methodology for ontology constructing.

9.4.4.1 REA

The most general concepts for REA model are introduced in the Fig.6.3.4.1. The REA model itself is a structure or mediation according to Sowa's categories being a combination of REA components.

The first line in conceptual system includes a concept 'REA component' which is not in the terminology. The intention of the line is to introduce a concept for specialization of REA parts. REA model is considered as aggregation of REA components.

REA components
    Business resource

- Cash
- Equipment
- Inventory

Business event
- Capital transaction
- Cash disbursement
- Cash receipt
- Equipment acquisition
- General & Administrative service
- Order
- Personnel service
- Purchase
- Sale

Business agent
- Customer
- Stockholder
- Vendor
- Employee
  - Salesperson
  - Supply agent
  - Shipment clerk
  - Accountant
- Unit

Relationship sets
- Control
- Duality
- Responsibility
- Stock-flow

REA model

REA model of a Retail Enterprise

Fig. 6.3.4.1. *Generalization structure of concepts for REA domain terminology.*

Here is formalization for some concepts for REA terminology. Fig. 6.3.4.2 illustrates formalization for things (a), relations (b) and structures (c) in REA. Comparing to Sowa's categories, resource is a physical static thing, event is physical occurring thing.

Thing	**Attribute name**	**Attribute category**	**Attribute value type**
Business resource	Name	Description	string
	Quantity	Description	number
	Time period	Continuant	(time, time)

Business event	Name	Description	string
	Time/date	Occurent	time
	Location	Description	string
	Quantity	Description	number
Business agent	Name	Description	string
	Address	Description	string
	Accountability	Description	set of strings

(a)

Relation name	**Thing**	**Role**
Control	Business resource	Exchange transaction
	Business agent	Inside
	Business agent	Outside
Duality	Business event	Increment
	Business event	Decrement
Responsibility	Unit	Superior
	Unit	Subordinate
Stock-flow	Business resource	Stock
	Business event	Flow

(b)

Structure name	**REA model**
Things	Business resource, Business event, Business agent
Relations	Control, Duality, Responsibility, Stock-flow
Structures	REA model of a Retail Enterprise
Structure name	**REA model of a Retail Enterprise**
Things	Cash, Equipment, Inventory, Business event, Capital transaction, Cash disbursement, Cash receipt, Equipment acquisition, General & Administrative service, Order, Personnel service, Purchase, Sale, Customer, Stockholder, Vendor , Employee, Salesperson, Supply agent, Shipment clerk, Accountant
Relations	Rel.1 – Rel.22 (introduced in section 2.2.3)
Structures	

(c)

Fig. 6.3.4.2. *Formalization for things (a), relations (b) and structures (c) in REA.*

The section represents some general concepts from business enterprise domain. The main source is Appendix C representing terminology for business enterprise. Other sources are found in Appendices B and C containing terminology and in chapter 3.1 explaining business enterprise domain.

The hierarchical structure represented below illustrates the attempt to find most general concepts and establish a hierarchy for business enterprise domain. The work is not considered completed. The structure represents different viewpoints.

The terminology work must be continued, viewpoints structure should be more elaborated. The first sketch of business enterprise ontology is represented as following:

Enterprise
    Retail Shop
Organization
Organizational Culture
Organizational policy
Organizational structure

Desired Situation

- Goals
  - Mission
  - Objective
- Enterprise Strategy

Relation
- Action Flow

Good
- Physical Thing
  - Raw Material
    - Product
    - Semi-Product
- Service

Tool
- Machinery

Business Action

Business Process

Assets

Enterprise Resource
- Stock
- Tangible Resource
  - Inventory
  - Financial assets
    - Cash
      - Money
  - Physical assets
    - Equipment
    - Machinery
    - Product
    - Raw Material
    - Semi-Product
- Intangible Resource
  - Intellectual Property
  - Goodwill
  - Copyright
  - Business information
    - Step
    - Plan
      - Schedule
    - Business method
    - Enterprise Tactic
    - Account
      - Cash Account
      - Bank Account
      - Capital Account
      - Wage Account
      - Sales Account
      - Purchase Account
    - Ledger
    - Account item

- Contract
- Human Resource
  - Employee
  - Consultant
  - Business Partner

Profit
Value
Event

- Purchase
- Personnel service
- Equipment Acquisition
- Buy
- Activity
  - Economic Activity
    - Financial Activity
    - Cash Transaction
      - Cash Receipt
      - Cash Disbursement
      - Capital Transaction

Enterprise environment
Competitor
Vendor
Customer
Business partner

Role

- Buyer
- Consultant
- Business Agent
  - Customer
  - Vendor
  - Employee
  - Stakeholder
    - Employee
    - Stockholder
    - Business partner
      - Customer
      - Supplier
      - Vendor
      - Distributor
- Actor
  - Agent
  - Accountant
    - Actor

Location

- Address

Party

- Competitor
- Agent
- Business Partner
  - Supplier

- Vendor
- Distributor

Person
- Employee

Activity
- Trade
- Main Business Activity
- Economic activity
  - Financial activity
- Primary activity
  - Production operation
  - Customer service
  - Marketing
  - Inbound Logistics
  - Outbound logistics
  - Sale
  - Customer Service
- Support activity
  - Accounting
  - Finance
  - Firm infrastructure
  - Procurement
  - Technical development
  - Government affairs
  - Legal affairs
    - Claim
    - Commitment
    - Contract
  - Management Activity
    - Human Resource Management
    - Decision making
      - Management choice
    - Planning
      - Scheduling
    - Control

Management Decision

Management Solution

Level of enterprise functioning
- Enterprise operational level
- Enterprise strategic level
- Enterprise tactical level

Workflow

Document
- Order
- Invoice
- Receipt

9.4.4.3 Enterprise solution

This section represents the first sketch of business enterprise solution ontology. The sources are Appendices B and C containing terminology and from chapters 3.2, 4.1 and 5 explaining business enterprise solution domain. The terminology and ontology work should proceed.

System
- Software system
  - Enterprise solution

System environment
- Enterprise

Architecture
Architectural description
Rationale
Stakeholder
View
Model
- Business model
  - REA model
  - Value chain framework
- Viewpoint
- Concern

Software engineering
Modeling engineering
Mission
User
User role

10

11 Conclusion and future work

The work done in the report examines some of the existing business modeling techniques in respect with enterprise solutions. Possible development directions for business modeling are suggested.

The REA diagram notation examined in the thesis is considered as suitable for modeling the domains of accounting and business enterprises. The main advantage of REA model is providing a structure for modeling accounting domain supported by a small set of simple axioms setting invariants for the structure. The domain of accounting is built into the REA model. The notation is easy to read and understand by non-technical people. But unfortunately the notation is limited to express different levels of enterprise functioning. The REA model is appropriate at the analysis stage of development of enterprise solutions.

In contrast to REA, UML notation is more universal and flexible to model any kind of business activity considered in any scope and any level of details. The scope could be enterprise –wide or business process/sub-process -wide and so on going further down specifying the details of enterprise functioning. The UML notation also suits well to model activities in scale of corporation, industry and macro-economy level. These cases are not considered in the thesis, but could be meant in stressing enterprise functioning as interaction with its environment. UML has a powerful mechanism for extending the notation staying in the frame of UML language. It helps to serve nearly all categories of enterprise solution stakeholders.

The thesis covers resent enterprise solutions and considers different approaches by examples of Microsoft enterprise strategy and Navision Jamaica model. Both variants have reasonable rationales but do not completely satisfy present business needs. Navision Jamaica model permits to make the solution by modeling but lacks the multi-view representation of the solution. Navision Jamaica model is oriented to small enterprises. Microsoft enterprise strategy has a more systematic approach and allows constructing solutions for any enterprise. But the process of constructing enterprise solutions is complicated and requires considerable amount of human resources.

The comparison of two strategies illustrates a dilemma of choosing between a scale of a target market and efficiency of the process of constructing enterprise solutions. The reason could be found in insufficient connection between information technology and business. There is a lack of theoretical foundation and tools for business modeling.

The possible solution is constructing ontologies for business enterprise and enterprise solutions. Ontologies can be used for construction and validation of conceptual models, navigation in user interfaces and business logic of enterprise solutions, construction enterprise solutions.

The report represents a ground work to make a business enterprise and enterprise solution ontology. The ontology construction is not completed and requires more work.

The efforts should be done in further elaboration of terminology. The concepts found should be revised; the relations between concepts must be refined. More concepts can be collected from the domains descriptions represented in the report and outside sources.

The terminological ontology represented in the thesis can be used as a basis for axiomatic ontology. The terminological ontology has to be presented in formal notations. Axioms should specify invariants for ontology and derivation rules. Formal notation can be BNF-grammar. E/R notation and UML notation can be expressed in translated in BNF-grammar. Another recommended notation is Pierce algebra. It's close to computer languages notations and may be interpreted for execution as well as BNF-grammar. UML/OCL notation is also suitable for ontology formalization. The next step to construct axiomatic ontology is construction and programming a database for the ontology.

Ontology work done in the thesis illustrates some problematical aspects.

One of the experienced difficulties is to keep **consistency of terms and their interrelations within an ontology**. In the represented ontology there is some overlapping in generalization/specialization and in part-whole relations.

The main reason is different unstructured viewpoints applied to one model. Viewpoints establish roles for concepts and even specify the fact participation of concepts in and certainly 'specify' structures.

A lattice structure is a compromise to represents different viewpoints in the same structure. But still, if try to represent all the possible viewpoints, the ontology will have no value due to increasing massiveness of a model and will require a big number of constraints to realize the intentions of every viewpoint.

Considering the named difficulties, viewpoints modeling seems to be a key factor for ontology efficiency. Referring to the standard IEEE Std.1471-2000, the work to establish viewpoints may be done in following steps:

4. Determine stakeholders for ontology,
5. Define their perspectives,
6. Group the perspectives into viewpoints, set up the relations between viewpoints and stakeholders,
7. Establish modeling techniques for every particular viewpoint.

The result should be a **viewpoint structure** for ontology.

The viewpoints for enterprise ontology are recommended to group according to the levels:

- *Macro-economical level* may be not so relevant, but helps to 'navigate' at the stage of a national economy, world-wide economic tendencies,
- *Micro-economical level* helps to understand the economical activities of the enterprise,
- *Enterprise level* deals with daily enterprise running.

There can be a number of different sub-viewpoints at every level.

To form the valuable viewpoint structure a number of experts should be engaged in the process. The related areas are:

- philosophy discipline,
- economy, finance and corporate budgeting, accounting, marketing, logistics, organization theory and management and supporting subjects in mathematics and statistics,

- software engineering, modeling engineering, particular technology platforms, logic theory, knowledge representation, artificial intelligence.

Another aspect for further development is **consistency and integrity between interrelated ontologies**.

Choosing enterprise solution as a main purpose, the following interrelated domains are considered: REA model, business enterprise and its specialization for a Retail Shop, enterprise solutions and its specializations - Microsoft enterprise strategy and Navision Jamaica model. The future work should concentrate on REA and business enterprise, enterprise solutions and Microsoft strategy domain. The business enterprise domain can be treated independently from REA or can be combined with REA domain. Enterprise solution ontology can be used later for specialization for different technology platforms.

Sowa's diamond is recommended as meta-ontology. It can be extended with Social dimension. The most general concepts of the ontologies should be bound to Sowa's categories.

The borders and interrelations between ontologies can be considered of two kinds regarding to changes of the concepts:

1. Shared concepts. For example, Enterprise term belong to enterprise ontology and enterprise solution ontology;
2. 'Influencing' concepts. For example changes Information Technology influences changes in Business Infrastructure and vice versa.

Most often the concepts in interrelated ontologies belong to both kinds at the same time.

These two kinds of concepts should be treated individually in respect with one of the basic requirements for ontology: expansibility. The relations between concepts in ontology are supposed to be weak in order to be easy extendable in future. But the second type of concepts illustrates a strong dependency between concepts and causes some imbalances in ontologies integrity. The solution is in finding suitable most general concepts in connection with Sowa's categories extended with social domain.

Summing up, the important and interesting directions for future work are refinement business enterprise and enterprise solution terminological ontology and constructing axiomatic ontology. Theoretical aspects as ontology viewpoint structure and ontologies integrity needs further elaboration.

13

14

15

16

17

18

19

20

21

22

23 Appendices

A Bibliography

[1] William E. McCarthy, *The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment*, The Accounting Review (July 1982) pp. 554-78

[2] Cheryl L. Dunn and William E. McCarthy, *The REA Accounting Model: Intellectual Heritage and Prospects for Progress*, The Journal of Information Systems, Spring 1997, pp. 31-51

[3] Guido L. Geerts and William E. McCarthy, *Augmented Intensional Reasoning in Knowledge-Based Accounting Systems*, The Journal of Information Systems (an early version of this paper was presented at the Twelfth International Workshop on Expert Systems and Their Applications, June, 1992, Avignon, France)

[4] Guido L. Geerts and William E. McCarthy, *An Accounting Object Infrastructure for Knowledge-Based Enterprise Models*, IEEE Intelligent Systems Their Applications, July-August 1999, p. 89-94

[5] Guido L. Geerts and William E. McCarthy, *The Ontological Foundations of REA Enterprise Information Systems*, November 1999, March 2000, August 2000

[6] Robert Haugen and William E. McCarthy, *REA: A Semantic Model for Internet Supply Chain Collaboration*, Business Object Component Workshop VI: Enterprise Application Integration

[7] Julie Smith David, Gregory J. Gerard and William E. McCarthy, *Design Science: Building the Future of AIS*

[8] William E. McCarthy, *An Entity-Relationship View of accounting Models*, The Accounting Review, October 1979, p. 667-86

[9] William E. McCarthy, *Semantic Modeling in accounting Education, Practice and Research: Some Progress Hierarchies with Resource-Event-Agent Object Templates*, Conceptual Modeling: Current Issues and future Directions, Editors: P.P. Chen, J. Akoka, H. Kangassalo and B. Thalheim. Springer Verlag, Berlin and Heidelberg, 1999, pp. 144-53

[10] Guido L. Geerts and William E. McCarthy, *Modeling Business Enterprises as Value-Added Process Hierarchies with Resource-Event-Agent Object Templates*, Business Object Design and Implementation J. Sutherland and D. Patel (eds.), 1997, Springer Verlag, pp. 94-113

[11] Guido L. Geerts, William E. McCarthy and Stephen R. Rockwell, *Automated Integration Of Enterprise Accounting Models Throughout The Systems Development Life Cycle*, Intelligent Systems in Accounting, Finance and Management, Volume 5 (1996), pp. 113-28

[12] *User Task Design*, Hawaii User Experience

[13] Ventura Vehicles

[14] Lars Hammer, Pavel Hruby, *The Software We Built*, DTU.ppt, Microsoft Power Point Show, 2002

[15] *Pattern: Model-View-Presenter*, Microsoft, 2002

[16] *Navision Jamaica Executive Overview*, Navision

[17] Jacob Nielsen, *The Ace User Experience Vision*, SAO-PRESS-HAWAII.TNG User Experience Vision (short).ppt, Microsoft Power Point Show, 2001

[18] Tom Østerby, *Information Modeling*, Compendium for the course Object-Oriented Domain Modeling, 2003

[19] Simon Bennett, Steve McRobb, Ray Farmer, *Object-Oriented Analysis and Design using UML*, McGraw-Hill Publishing Company 1999

[20] Hector Garcia-Molina, Jeffrey D. Ullman, Jenifer Widom, *Database Systems: The Complete Book*, Prentice Hall 2002

[21] Martin Fowler, *Analysis Patterns: Reusable Object Models*, Addison-Wesley 1997

[22] *Longman Dictionary of Contemporary English, New Edition, Longman 1987*

[23] Nicola guarino, Pierdaniele Giaretta, *Ontologies and Knowledge Bases* in *Towards Very Large Knowledge Bases* ed. N.J.I. Mars, IOS Press 1195, Amsterdam

[24] Gruber T. R., *A translation approach to portable ontologies*. Knowledge Acquisition. No.5 (2) 1993

[25] *Business process modelling*, Discussion on the Rational Unified Process forum, April 2002

[26] http://ic2.epfl.ch/publications/documents/IC_TECH_REPORT_200259.pdf

[27] http://www.agilemodeling.com/essays/modelingTechniques.htm

[28] http://msdn.microsoft.com/architecture/enterprise/default.aspx?pull=/library/en-us/dnea/html/eaarchover.asp

[29] http://www.idi.ntnu.no/~letizia/swarchi/IEEE1471.pdf

[30] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsentpro/html/veconIdentifyingBusinessRequirements.asp:

B Term dictionaries

B.1 REA model

Retail shop is a shop to sell goods to customers. The goods are for the customers' own use, not for sale for anyone else.
Agent is a participant in the accounting process.
Capital Transaction is an event of cash transaction between a stockholder and the enterprise as a result of financial activity.
Cash Disbursement is an event of cash transaction from the enterprise to a vendor or employee as a payment for goods or services or to a stockholder as a result of financial activity.
Cash is a resource having monetary value available for the enterprise.
Cash Receipt is an event of cash transaction from a customer to the enterprise for goods sold or from a stockholder to the enterprise as a result of financial activity.
Claim is temporary imbalance between sale and cash receipt.
Control relationship is a three-way relationship connecting a Resource, an Agent belonging to an enterprise and having 'inside' role, and an Agent not belonging to an enterprise and having 'outside' role.
Customer is a person or enterprise buying products from the retail enterprise.
Duality relationship is a binary relationship connecting two interplaying Events. The Events mutually depend on each other. One Event increments the related resource set and has a role 'increment'. Another mirror event decrements the related resource set and has a role 'decrement'.
Employee is a person hired and paid by the retail enterprise and doing personal service.
Equipment Acquisition is an event where the enterprise buys equipment from a vendor.
Equipment is a set of things needed for the enterprise to perform its main business to sell goods to customers but not for sale in the frame of the main business.
Event is a REA representation of any economic activity.
General & Administrative Service is a management activity event required for running the enterprise.
Give is a basic enterprise economic activity representing resource outflow. It is opposite to Take.
Inventory is products to be sold by the retail enterprise to customers and amount of these products presented at the stock of the enterprise.
Order is an event where a customer makes a request to the enterprise for goods he wants to buy.
Party is a generalization for Agents and Units.
Personnel Service is an event where an employee provides service to the enterprise.
Purchase is an event where the enterprise purchases goods from a vendor.
Resource is a REA representation of any economic object that is under control of an enterprise.
Responsibility is a binary relationship connecting a 'superior' Unit and a 'subordinate' Unit. It describes a hierarchical structure of the Units.
Sale is an event where the enterprise sells goods to a customer.

Stock-flow relationship is a binary relationship connecting a Resource with role ‘stock’ and an Event with role ‘flow’. The Event causes a change in the Resource.
Stockholder is a person or an enterprise buying products from the retail shop owns the resources of the retail enterprise.
Take is a basic enterprise economic activity representing resource inflow. It is opposite to Give.
Unit is an Agent representing a set of individual Agents.
Vendor is a person or enterprise selling products or services or equipment for the retail enterprise.

B.2 Double-entry accounting

Account: a section in the ledger. *See* ledger.

If the debit side of an account is greater then the credit side, the account represents either an asset or an expense or loss. If the credit side of an account is greater then the debit side, the account represents capital, liability, income or a gain. *See also* cash account, bank account, capital account, wages account, sales account, purchases account, equipment acquisition account, general & administrative service account.

Accounting equation: the relation which is hold at any point of time in the accounting process:

Capital = Assets – Liabilities

or alternatively

Assets = Capital + Liabilities.

Accounting period: dividing up the life of the enterprise into discrete periods for the purpose of reporting its financial state.

Accounting process: a process of reporting monetary operations within the enterprise.

Acquisition cost:

Assets: all resources possessed by the retail shop enterprise – products, equipment, cash and claims.

Bank account: an account to register cash transactions via bank. Entries are done in the same way, as with cash account. *See* cash account. In the considered accounting model there is no difference between cash and bank account in the simplicity purpose.

Business transaction: *see* transaction.

Buyer: a person or an enterprise who buys goods or services.

Capital account: an account to register cash transactions between a stockholder and the enterprise. When the stockholder invests money into the enterprise, it is entered at the credit side of the capital account and debit side of the cash account.

Capital or **Owner's equity**: proprietor's ownership claims in the business. In the case of the Retail Shop Enterprise, capital is a stockholder's claims in the enterprise. This may appear as an equation:

stockholder's claims in the enterprise = net resources possessed by the enterprise

Cash account: an account to register cash transactions.

When cash is received by the enterprise, it is entered on the debit side of the cash account and credited to the account related to the cash transaction. When cash is paid out by the enterprise, it is entered on the credit side of the cash account and debited to the account related to the cash transaction.

Cash transaction: a transaction where goods or services are paid for in cash or by cheque, when they are received or delivered.

Cheque: a form of payment in business via bank. A cheque contains the next information:

- cheque date,
- cheque number,
- bank account number of the payee,
- payee name,
- bank account number of the drawer,
- drawer name,
- payment description,
- amount of money.

Claims or **future assets**: intangible resources arise from imbalances between sales and payments for the sales and between purchase and payments for purchase.

Cost allocation of equipment: *see* depreciation.

Credit: a right side at the ledger. *See* ledger.

Credit note: a document issued by a seller to inform a buyer on the amount, which the seller has overcharged the buyer on the invoice, or if the buyer returned goods to the seller. The credit note is entered at the ledger of the Retail Shop Enterprise, and the seller pays to the buyer the amount, recorded in the credit note. A credit note contains the next information:

- invoice number,
- invoice date,
- order number,
- credit note number,
- credit note date,
- seller name,
- seller address,
- buyer name,
- buyer address,
- buyer delivery address,
- value overcharged,

- payment due date.

Credit transaction: a transaction where goods or services are paid for some days after delivery.

Customer account: an account to register credit transactions.

When the enterprise issues an invoice to the customer, it is entered on the debit side of the customer account and credited to the sales account.

Debit: a left side at the ledger. *See* ledger.

Debit note: a document issued by a seller to inform a buyer on the amount, which the seller has undercharged the buyer on the invoice. The debit note is entered at the ledger of the Retail Shop Enterprise, and the buyer pays to the seller the amount, recorded in the debit note. A debit note contains the next information:

- invoice number,
- invoice date,
- order number,
- debit note number,
- debit note date,
- seller name,
- seller address,
- buyer name,
- buyer address,
- buyer delivery address,
- value undercharged,
- payment due date.

Depreciation or **Cost allocation of equipment**: allocation of the cost of equipment over the accounting period that reflects its useful economic life to the enterprise according to the amount which is 'consumed' during the period.

Discount: a percentage reduction of a retail price. It is deducted from the retail price amount the buyer is charged for goods.

Drawings account: an account to register cash transactions between a stockholder and the enterprise.

When the stockholder withdraws money from the enterprise, it is entered at the debit side of the drawings account and credit side of the cash account.

Double entry bookkeeping: a systematic method or recording business transactions in a ledger. *See* ledger.

Equipment acquisition account: an account to register cash transactions from the enterprise to a vendor, when equipment acquisition is paid by the enterprise.

The transaction is entered to the debit side of the equipment acquisition account and to the credit side of cash account.

Employee account: an account to register credit transactions.

When the enterprise issues payroll, it is entered on the debit side of the employee account and credited to the payroll account.

Enterprise: *see* Retail shop enterprise.

Enterprise account: an account to register credit transactions between, where the enterprise is a party involved in the transaction.

Entry: a record of a business transaction at the ledger.

Expenditure: charges of the enterprise to provide its business activities.

General & Administrative Service account: an account to register cash transactions from the enterprise to a vendor, when service is paid by the enterprise.

The transaction is entered to the debit side of the general & administrative service account and to the credit side of cash account.

Invoice: a document issued by a seller to inform a buyer how much he's owed for the goods supplied. The buyer checks the invoice against the order and the delivery. If the invoice is correct, the invoice is entered at the ledger of the Retail Shop Enterprise, and the buyer pays to the seller the total amount, recorded in the invoice. An invoice contains the next information:

- invoice number,
- invoice date,
- order number,
- seller name,
- seller address,
- buyer name,
- buyer address,
- buyer delivery address,
- product name,
- product quantity,
- product price,
- total value of the invoice before VAT,
- VAT payable,
- total value of the invoice including VAT,
- payment due date.

Ledger: a book for recording business transactions according to the double entry bookkeeping. The purpose of the ledger is to provide the total amount of income and expenditure for each type of entry, the value of the assets owned by the enterprise.

The pages of the ledger are split into the left side called debit and right side called credit. The ledger is divided into section called accounts. The money value of each transaction is

entered once on each side of the ledger. For example, the sale transaction must be entered to the debit side of Cash account and to the credit side of Sales account. Each side contains the next information:

- entry date,
- entry details,
- entry amount.

Liabilities: deductions which must be done from assets. These are taxes, claims by outside parties, provisions.

Net resources: resources possessed by the retail shop enterprise after all deductions.

Order: a document issued by a buyer to request goods he/she wants to buy from a seller.

Owner's equity: *see* capital.

Price: *see* retail price.

Provisions: capital amount to allow for liabilities which are anticipated but not yet quantified precisely, or for reduction in asset values.

Profit: the maximum amount of money that can be withdrawn in a period from the enterprise while leaving capital intact.

Purchases account: an account to register cash transactions from the enterprise to a vendor, when purchase is paid by the enterprise.

The transaction is entered to the debit side of purchases account and to the credit side of cash account.

Receipt: a document issued by the enterprise to a customer for goods which have been paid for in cash. It is not legally required. A receipt contains following information:

- payment date,
- Enterprise name,
- product name,
- product cost,
- payment amount.

Retail price or **Price**: money amount which a buyer is charged for goods.

Retail shop enterprise or **Enterprise**: an enterprise performing a retail trade business in order to make profit by selling goods to customers. The retail shop enterprise acts as a seller, when it sells goods to customers, or as a buyer, when it buys goods or services from outside to provide sales.

Revenue: income of the enterprise getting from sales.

Sales account: an account to register cash transactions from a customer to the enterprise, when sale is paid by the customer.

The transaction is entered to the credit side of sales account and to the debit side of cash account.

Seller: a person or an enterprise who sells goods or services.

Transaction or **Business transaction**: any business activity which involves money or capital transaction.

Vendor account: an account to register credit transactions.

When the enterprise receives an invoice from vendor, it is entered on the credit side of the vendor account and credited to the purchases account.

Wages account: an account to register cash transactions from the enterprise to an employee, when salary is paid by the enterprise.

The transaction is entered to the debit side of wages account and to the credit side of cash account.

B.3 Business enterprise

Accounting is keeping *business information* about all payments to and from am *enterprise*.

Action is the unit of work to be done in an *enterprise*. Actions are building blocks in processes and they are ordered by *action flows*. An action is performed by one or more *actors* playing *roles* in the action.

Activity is something witch an *enterprise* might do. Activity may be described by *business processes.*

Actor is a person doing a *business process*. The actor may play a specific role designating the function in the process. An actor is normally located in the *enterprise*. An actor can sometimes be located in the *enterprise environment*.

Business enterprise see *Enterprise*.

Business information is information being an *intangible resource* for an *enterprise*.

Business partner is a *party* cooperation with the *enterprise*. Business partners are part of the *environment* for the enterprise.

Business process is one or more *actions* and sub-processes performed by one or more *actors* in an *enterprise*.

Cash is money in form of bills or coins. Cash belonging to an *enterprise* is a *tangible resource*.

Competitor for an *enterprise* is a *party* offering the same *goods*. Competitors are part of the *environment* of the *enterprise.*

Control is a *management activity* where a *schedule* is compared with the present state.

Copyright is an *intangible resource* as an exclusive right to a literary, dramatic, musical, or artistic work or to the use of a commercial print or label, as granted by law.

Core business processes is the set of *business processes* giving the biggest contribution to the value creation.

Customer is a *party* buying *goods* from an *enterprise*. Customers are part of the *enterprise environment.*

Customer service is a *primary activity* of an *enterprise* assisting actual or potential *customers*.

Distributor is a *business partner* who delivers the *products* or *services* produced by an *enterprise* to its *customers*.

Enterprise is a social organization performing services, production and trade of *goods*. An enterprise is based on private ownership of wealth.

Enterprise environment is the part of the world having interaction with or of interest for the *enterprise*.

Enterprise operation level relates to day-to-day routines of an *enterprise*.

Enterprise resource is the means for an *enterprise* to create value. Resources may be divided into *tangible resources*, *intangible resources* and *human resources*.

Enterprise solution is a kind of software system solving a problem for a single user type. An enterprise solution is a description generated by *modeling engineering* and the resulting models are directly executable.

Enterprise strategy is a set of enterprise *goals*.

Enterprise strategic level concerns with enterprise *goals* and related activities of enterprise functioning.

Enterprise tactic is a set of methods to accomplish the *enterprise strategy*.

Enterprise tactical level deals with practical concrete directions and related activities to realize the *enterprise strategy*.

Equipment is instruments needed for performing *services*. Equipment owned by an *enterprise* is a part of its *tangible resources*.

Finance is an activity for providing funds and capital for investment in *tangible resources* and *intangible resources*. Finance is an activity in *firm infrastructure*.

Firm infrastructure is *support activities* being required for *primary activities* and other support activities. Firm infrastructure includes *management, planning, finance, accounting, legal affairs* and *government affairs*.

Goal is a state to be reached by doing something.

Good is a *product* or *service* sold by an *enterprise* to *customers*.

Goodwill is an *intangible resource* generated when an *enterprise* does something positive that has value. Goodwill can include the company's reputation, brand recognition, and relationships with *enterprise environment* and *customers*.

Intellectual property is an *intangible resource* as trade secrets, patent, *know-how*, managerial innovativeness, information on an *enterprise* and *enterprise environment*.

Government affairs are activities of interest of an *enterprise* concerning government authority or political units. Government affairs are an activity in *firm infrastructure*.

Human resource management is *support activities* for supplying an *enterprise* with *employees* having *know-how* to perform the activities in the enterprise.

Inbound logistics is *primary activities* for supplying an *enterprise* with *raw materials*.

Know-how is the technical knowledge and skill required to do something.

Legal affairs are activities of interest of an *enterprise* concerning legal authorities. Legal affairs is an activity in *firm infrastructure.*

Machinery is mechanical or electrical tools necessary for making *products*. Machinery owned by an *enterprise* is a part of its *tangible resources.*

Management is *activities* concerned with running an *enterprise*. Management includes *decision making*, *planning*, *scheduling* and *control*. Management is an activity in *firm infrastructure*.

Management choice is a process of selecting from a number of alternative *actions.* The result of management choice is a *management decision.* The goal of the choice is to increase the value created by an enterprise.

Management decision is a result of *management choice*. Management decision contains instructions regarding value, *resources, business processes, organizational structure* and relations with *environment.*

Management solution is a combination of *management decisions* on organizing and running an *enterprise*.

Marketing is *primary activities* funding potential *customers*.

Main business activity is activities responsible for creating monetary value.

Mission is a set of possibilities for future development of an *enterprise*. These possibilities may be outside of present *resources*.

Modeling engineering is a process describing something (the subject) by a formal notation. The notation may be UML, and the subject may be an *enterprise solution.*

Objective is a *goal* which is considered attainable.

Organization is a part of an *enterprise* which includes the *organizational structure*, the *organizational policy* and the *organizational culture*.

Organizational culture is a part of *organization*. Organizational culture is a collection of beliefs, experiences and behavioral norms shared by people within an organization.

Organizational policy is a part of *organization*. Organizational policy defines a course on how an *enterprise* deals with *stakeholders*, with *environment*, the range of *goods* and the scope of *activities*.

Organizational structure is a part of *organization*. Organizational structure defines on how *human resources* are distributed in accordance with their *roles* in *business processes*.

Outbound logistics is *primary activities* delivering *products* to *customers*.

Party is a group of individuals, non-business organizations and *enterprises*.

Plan is the result of *planning*. A plan has some steps and ordering relations between steps.

Planning 1) is a *management activity* resulting a *plan;* 2) is an activity in *firm infrastructure*.

Primary activity is a sequence of activities in the *value chain framework*. Primary activities are directly contributing to value creation. Primary activity elements are *inbound logistics, production operations, outbound logistics, marketing and sale* and *customer service*.

Procurement is a *support activity* for supplying an *enterprise* with material and technical devices for its functioning.

Product is a *good* sold to *customers* by an *enterprise*. A product is an outcome of the production process. Products belonging to an enterprise are a part of its *tangible resources*.

Production operations are *primary activities* making *products*.

Profit for an *enterprise* is the difference between the value received and the value spent.

Raw materials are used and consumed in a production process. Raw materials are owned by an *enterprise* is a *tangible resource.*

Role is something played by an *actor*.

Sale 1) is an exchange of *goods* for money; 2) is *primary activities* for an *enterprise* selling *goods* to *customers*.

Schedule is a *plan* extended with time points for the start of the *steps*.

Semi-product is a *product* being part of another product. Semi-product belonging to an *enterprise* is a part of its *tangible resources*.

Service is a *good* being sold to *customers* by an *enterprise*. A service is a non-tangible good proving time, skills and experience.

Stakeholders for an *enterprise* are the set of persons or organizations having an interest in the enterprise. Stakeholders may be owners, employees, business partners, customers, suppliers, distributors and others.

Step is an element in a *plan*. A step may be an *action* and the *resources* needed by the action.

Support activity is activities in the *value chain framework* necessary for the *primary activities*. Support activities are *firm infrastructure, human resource managements, technical development* and *procurement.*

Supplier is a *business partner* providing an *enterprise* with *goods* in order to supply with *resources* needed for the enterprise functioning.

Tangible resource includes financial and physical assets of an *enterprise*. Examples are *cash, raw materials, semi-products, products, machinery, equipment* etc.

Trade is buying and selling *goods.* A retail enterprise is doing trade.

Technical development is *support activities* for developing goods *and improving activities*.

Value chain framework is a model describing the value creation processes for an *enterprise*. The framework is divided into *primary activities* and *support activities*. The model is constructed by Michel E. porter.

Workflow is a set of procedures which reflects enterprise functioning at the operational level.

B.4 Enterprise solution strategy

Application architecture is a part of an *enterprise architecture* describing

ANSI, "the American National Standard Institute is a private, non-profit organization that administers and coordinates the U.S. voluntary standardization and conformity assessment system", *see* http://www.ansi.org/about_ansi/overview/overview.aspx?menuid=1

Application domain is a part of real world which is in a scope of interest of application software.

Application perspective is a *perspective* in the *Microsoft enterprise solution strategy*. The perspective describes software applications supporting the functioning of an enterprise's business.

Application *see application software*

Application software is *software* solving problems of an *application domain*

Architectural design is a process of defining *hardware* and *software* components and their *interfaces* to make a framework for sub-system control and communication and the development of a *software system*. The output of this process is a description of the *software architecture*.

Availability is a *non-functional* requirement. Availability defines a time span of a user waits for a response to his/her request to an *enterprise solution*.

Business perspective is a *perspective* in the *Microsoft enterprise solution strategy*. The perspective concentrates on functioning of an enterprise's business.

Business process and organization is a *concern* in the *Microsoft enterprise solution strategy*. The concern expresses main and support activities of an enterprise and its organization. The concern is used to cover the *business perspective* and *information perspective*.

Coals and objectives is a *concern* in the *Microsoft enterprise solution strategy*. The concern describes the directions for enterprise functioning on the strategic and tactical levels and for development of an *enterprise solution*. The concern is addressed to the *business perspective* and *application perspective*.

Concern is a peace of information relevant for the *enterprise solution strategy* and an *enterprise solution's stakeholders*. A stakeholder has one or more concerns. Each concern addresses one or more *perspectives*.

Constraint is a restriction which limits a service. *See enterprise solution requirements*.

Domain requirements is

Domain requirements reflect fundamentals of an application domain. Domain requirements can be *functional requirements* and *non-functional requirements*.

Domain *see Application domain*

Enterprise architecture is a conceptual tool for understanding a structure and functioning of an enterprise.

Enterprise solution requirements see *software requirements*.

Enterprise solution strategy is approaches, principles, directions for development of enterprise solutions.

Enterprise solution's stakeholder is a person having direct or indirect influence on the *enterprise solution requirements*.

Functional requirements is a set of statements of services the software system should provide.

Hardware is electronic devices and their parts which allow to store, process and delivers data and run the *software*.

IEEE Std 1471-2000, IEEE's Recommended Practice for architectural Description of Software-Intensive Systems, is a formal standard defining *architectural description*.

IEEE, the Institute of Electrical and Electronics Engineering, Inc. is a non-profit technical professional association. IEEE leads in many technical areas including computer engineering. *See http://www.ieee.org/*

Information perspective is a *perspective* in the *Microsoft enterprise solution strategy*. The perspective defines the data needed for the enterprise functioning.
Interface is a boundary part between two interacting sub-systems through which they communicate with each other.
Interoperability is a *non-functional* requirement. Interoperability defines how an *enterprise solution* will interact with systems in other organizations.
Maintainability is a *non-functional* requirement. Maintainability defines the ability of an *enterprise solution* to be repaired and maintained easily and fast.
Manageability is a *non-functional* requirement. Manageability requires an *enterprise solution* to have tools to manage it.
Non-functional requirements is a set of constraints for services offered by a software system.
Operational requirements *see non-functional requirements*
Performance is a *non-functional* requirement. Performance requirement defines the efficiency of hardware utilization.
Reliability is a *non-functional* requirement. Reliability set up the rate of failure acceptable for an *enterprise solution.*
Requirements elicitation is a process through which customers and developers of an enterprise solution determine the *enterprise solution requirements*.
Scalability is a *non-functional* requirement. Scalability requirement defines the ability of an *enterprise solution* to perform under increased number of users, the amount of data and number of transactions being managed.
Security is a *non-functional* requirement. Security requirement demands an *enterprise solution* follows a security policy of an *enterprise*.
Service is a function which an enterprise solution should provide. *See enterprise solution requirements*.
Software architecture describes the sub-systems of a *software system*, relationships between the sub-systems and the environment. *See architectural design.*
Software engineering is 1) a process of transforming a user's *requirements* into *software*; 2) an engineering discipline for studying and developing theories, methods and tools for the software engineering process.
Software is computer programs, procedures and associated documentation and configuration data which is needed for correct operation of these programs and procedures.
Software requirements is a description of services a *software system* should provide and constraints for the software system. *See functional requirements, non-functional requirements, domain requirements.*
Software system is an automated system which receives input information and delivers output information. A software system consists from a number of related software components, description of the system, user documentation explaining how to use the system.
Systems and data is a *concern* in the *Microsoft enterprise solution strategy*. The concern focuses on *software systems* used by an enterprise. The concern is used to meet the *application perspective, information perspective* and *technology perspective*.
Technology is a *concern* in the *Microsoft enterprise solution strategy*. The concern relates to hardware used by an enterprise. The concern is addressed to the *application perspective* and *technology perspective*.
Technology perspective is a *perspective* in the *Microsoft enterprise solution strategy*. The perspective covers hardware and software supporting the enterprise functioning.

B.5 Microsoft .Net

API (Application Programming Interface) is a set of functions available for using in software applications.
ActiveX is a group of Microsoft technologies for programming component object applications based on COM.
Business logic is a set of functions of the middle level in the multi-tired based applications architecture.
COM (Component Object Model) is a Microsoft standard mechanism including interfaces. With the help of this mechanism, objects can offer their services for other objects. COM is a basis for many object technologies including OLE and ActiveX.
COM+ is an extension for COM. “COM+ builds on COM's integrated services and features, making it easier for developers to create and use software components in any language, using any tool.” - http://www.microsoft.com/com/tech/COMPlus.asp
OLE (Object Linking and Embedding) – until 1996 it was a common name for OO Microsoft technologies based on COM. After including the term ActiveX 1996, OLE term is used for based on COM technologies for creating complex documents by linking and embedding document items.
Smart device is a mobile device connected to Internet: Tablet PC, PDA, mobile phones, devices based on GPS, etc.
Service is a discrete unit of code which handles a limited set of tasks.
Web service is a discrete unit of code with a limited functionality. It based on XML and allows Web applications to share data.

B.6 Navision Jamaica modeling concepts

Action a combinations of a method and an event, can be linked with another action, is trigged by a user through a user task, can contain custom code.
Activity center is a portal providing a group of related activities. Examples: Sales, Setup.
Address is an element base to define a postal address.
Business object (BO) is a modeling block to represent a business entity. Examples: customer, invoice, spare part.
Business object base (BOB) is a modeling abstract concept to classify BO for similar business entities. Example: an abstract for customer and employee.
Business object type (BOT) is a type for a set of particular BO. Examples: customer, sales invoice. Accumulates crosscutting functionality of BO. Relations between BOT may be used for high-level consistency checking.
Classification is an element base to define groups and categories of BO.
Containment is an element base "for specifying a relationship between a "header"-type Business Objects and a number of "line"-type Business Objects" ([16], p.6).
Dependency is a relation between elements used to transfer value of one element to another trigged by a system event, depending on the state of the elements.
Document is an instance of a specific view.
Element Base is a modeling abstract concept to classify similar elements. Examples: address, classification, milestone.
Element Type is defined on the element base, assigns a type for a set of particular elements.

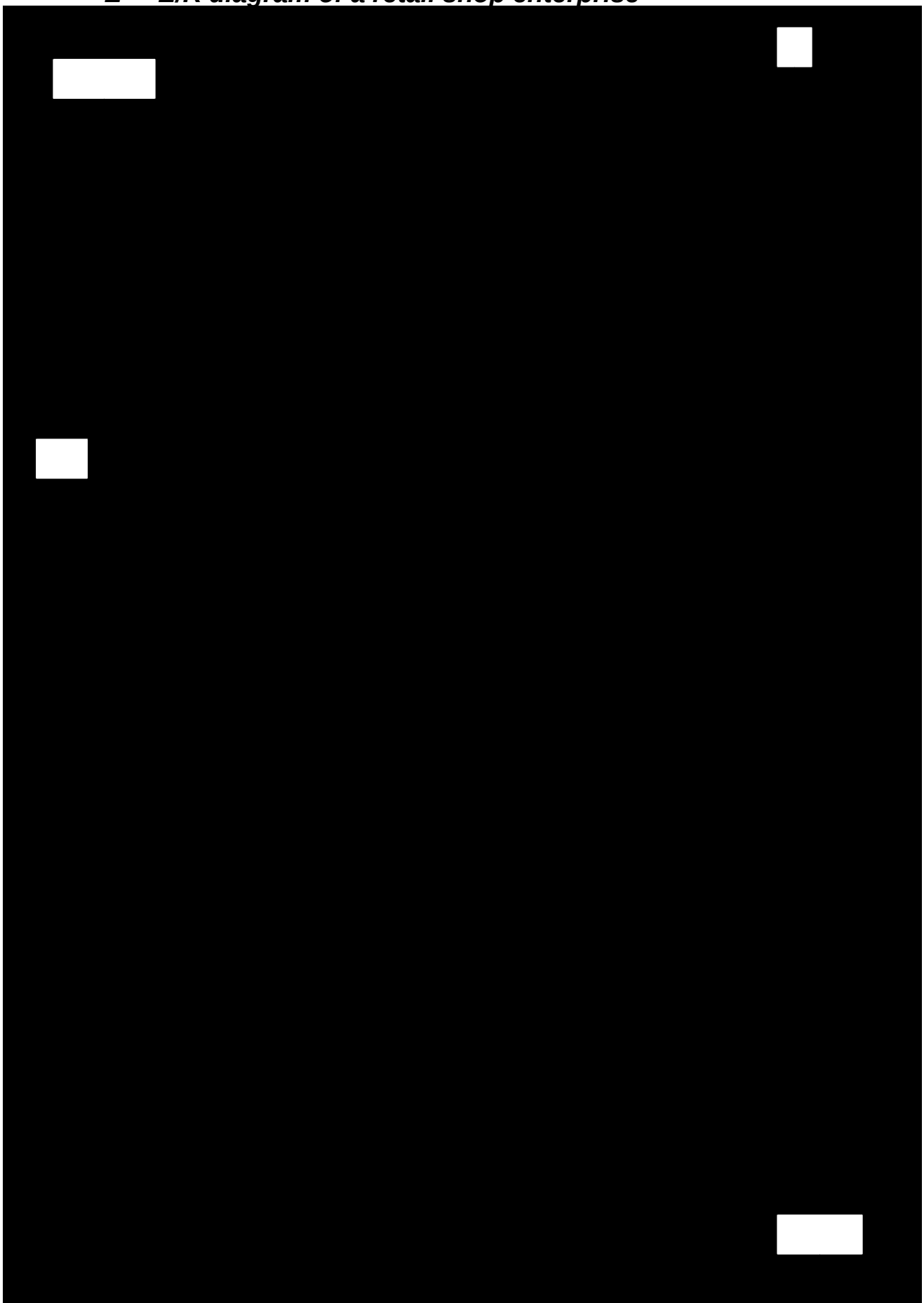
Element is a modeling block, which carries both data and functionality to process that data, exists in the context of a BO. Elements capture the knowledge via design patterns.
Event is an occurrence that is relevant for the information system, can be trigged by the system or a user through an action.
Event-wiring is an invocation of a method by an event.
Hot task is the first step in a user task shown to a user in a web part in an activity center. It can be a select or a report.

Inductive navigation is a system functionality, which provides user navigation according to the context of preliminary user steps, business model current data and constraints.
Methods provide theirs functionality and belong to elements and BOs. Methods are called by events through "event-wiring".
Milestone is an element base to define an expiration date, "the Element fire an event when the expiration date occurs" ([16], p.6).
Portal is a role specific entry to the user task for a particular user. In Jamaica project requirements there is one portal for one user.
Reconciliation is an element base to keep track of reconciliation of resources.
Registration (Aggregation) is a "set of elements ... permanently stored for legal and reporting purposes" ([16], p.6).
Relation is an element base to define a relation between BOs.
Role is a modeling abstract to define the permission, portal, activity center for the user.
Sub-View is a view which an element contains. Sub-view also contains elements.
Task step page is a part of a task step.
Task step is a part of a user task in a multiple task sequence.
Template is a mapping document (XML file) from a BO view to the paper to generate a report.
Tier is a basic modeling concept. Examples: web browser client, UI subsystem, user tasks, business logic, storage. "Each communicates with the tier above it and the tier below it in the loosest possible way, mostly through Views" ([16], p.9).
Typification is a modeling principle that one element type is defined based on one element base.
User task is the only way to provide user interaction with a BO, access to data and functionality, bases for automatically-generated UI. User task may consist of a sequence of sub tasks. There are seven types of user task: Create, view, Modify, Delete, Print, DoAction, Transform.
User is a user of a business information system, is modeled as a BO with login element.
View is a number (some or all) of elements on a BO, is used to update data on the BO.

C

D Terms and definitions

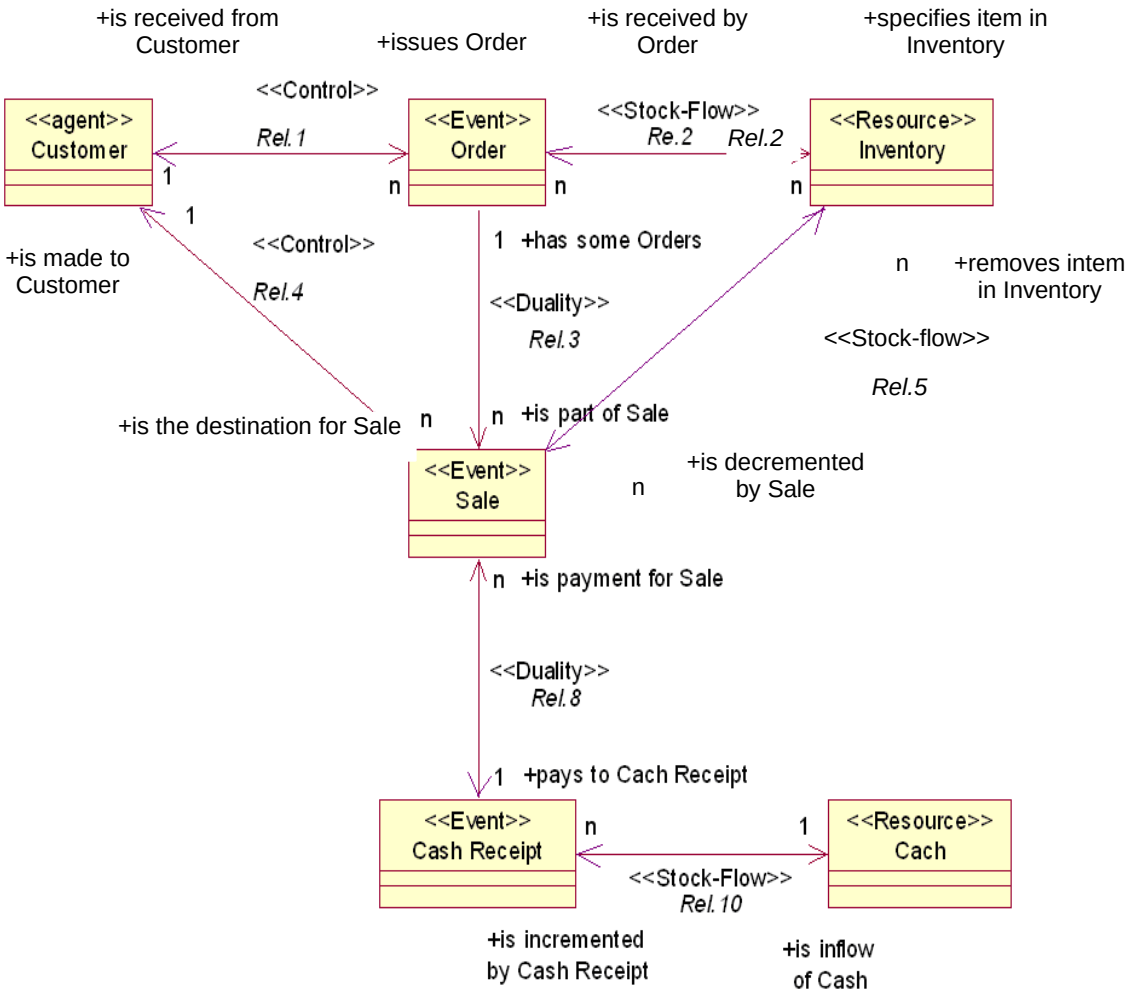
E E/R diagram of a retail shop enterprise



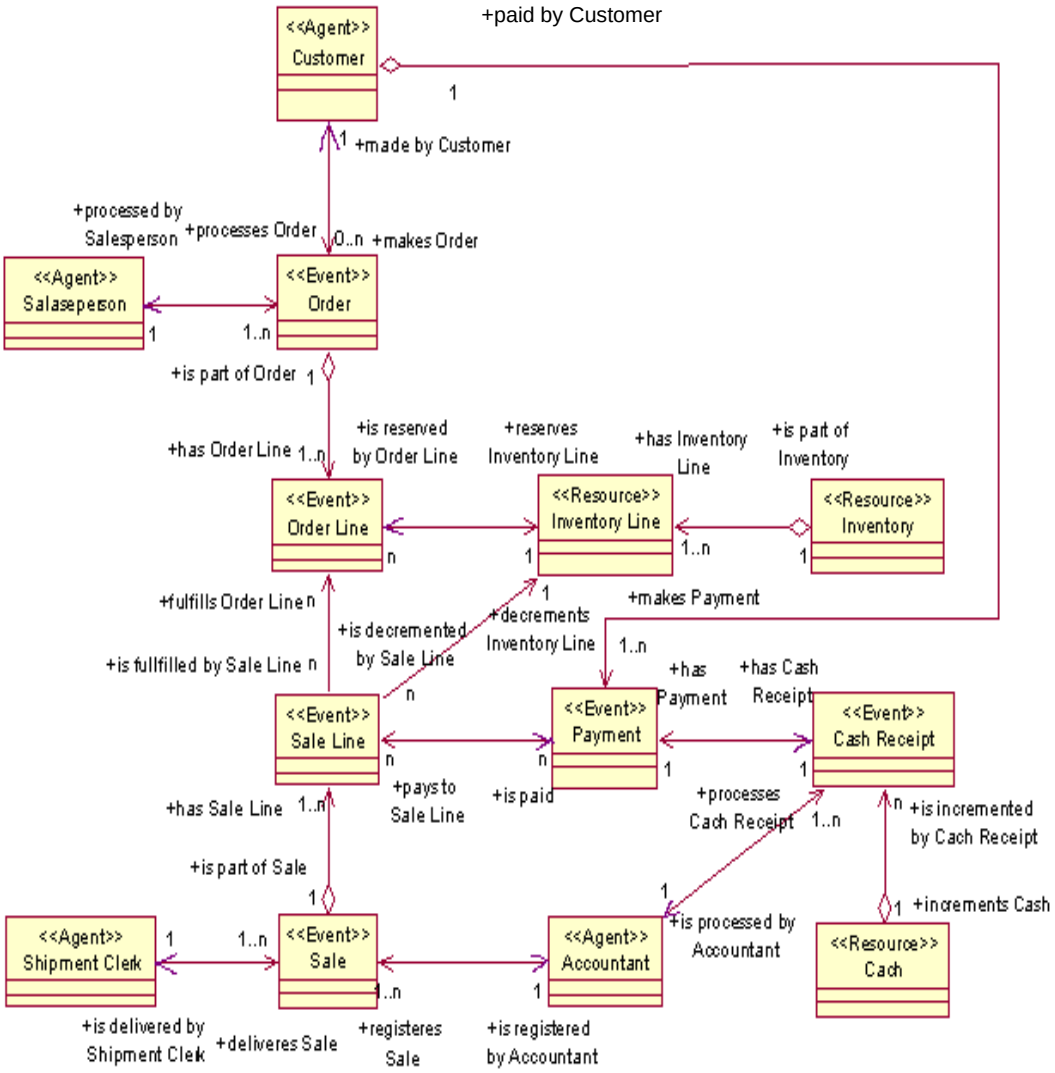
F UML diagrams

F.1 UML diagrams for a retail shop enterprise

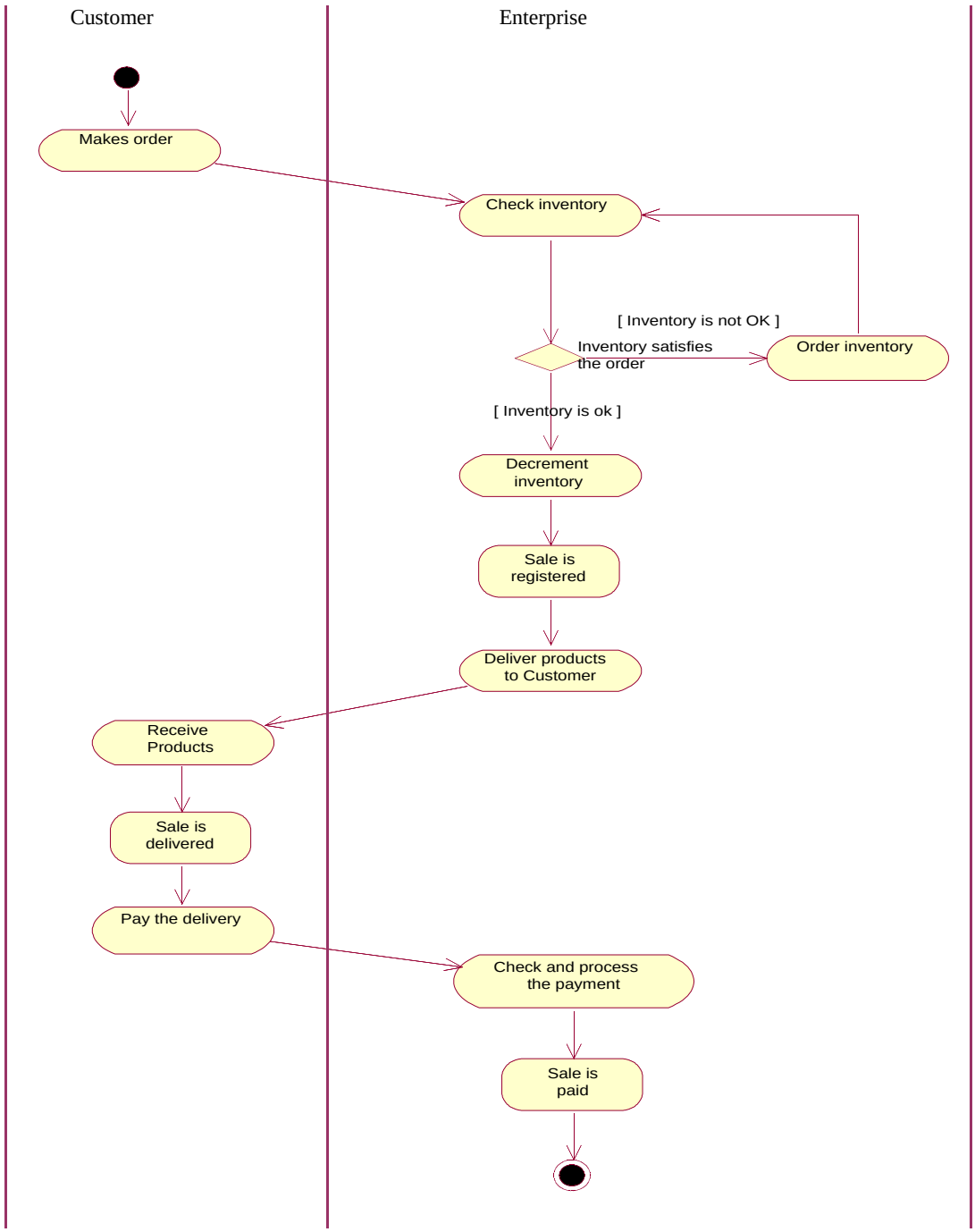
F.1.1 Class diagram for the Revenue cycle



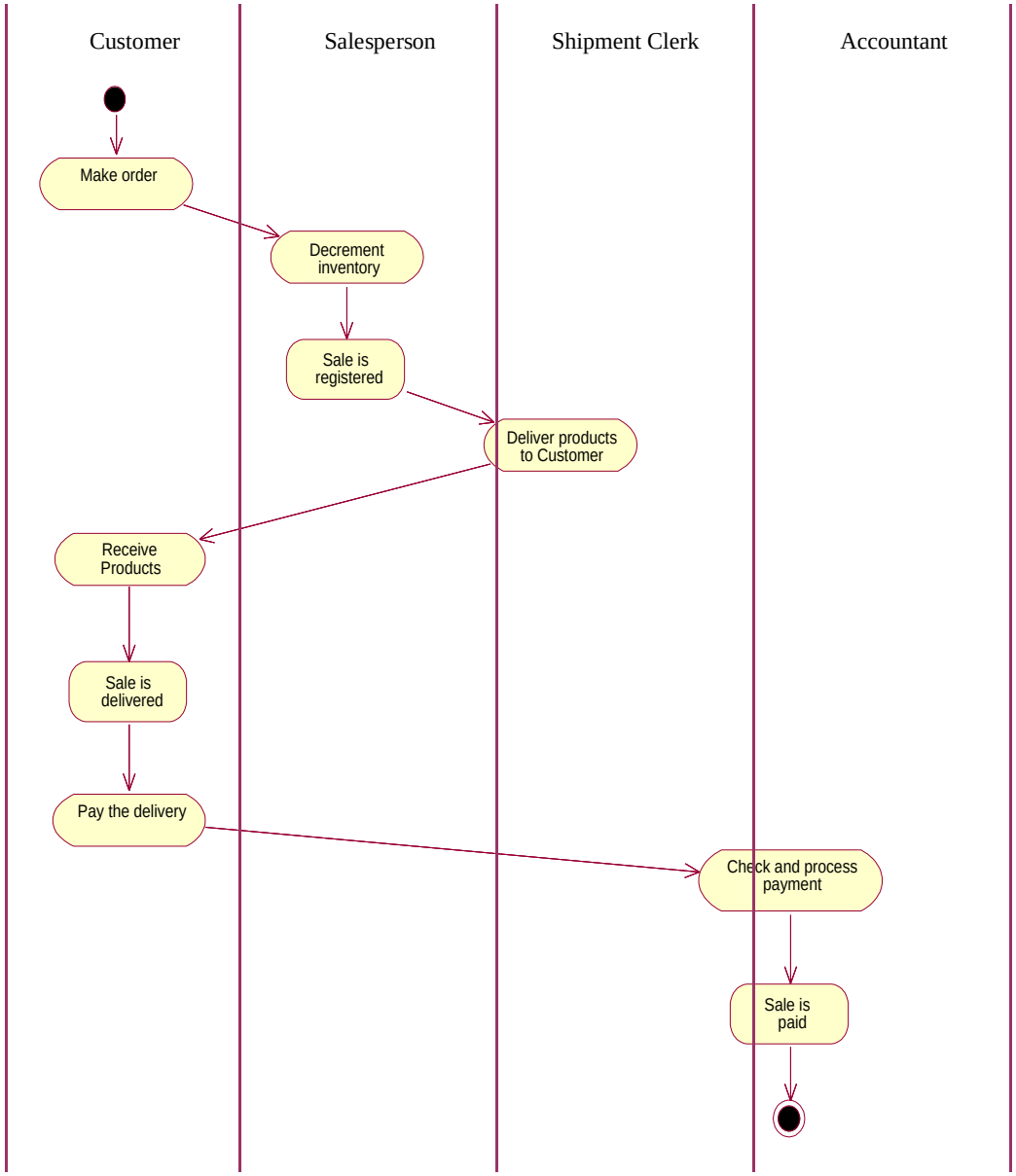
F.1.2 Class diagram for the Sale event



F.1.3 Activity diagram for the Sale event

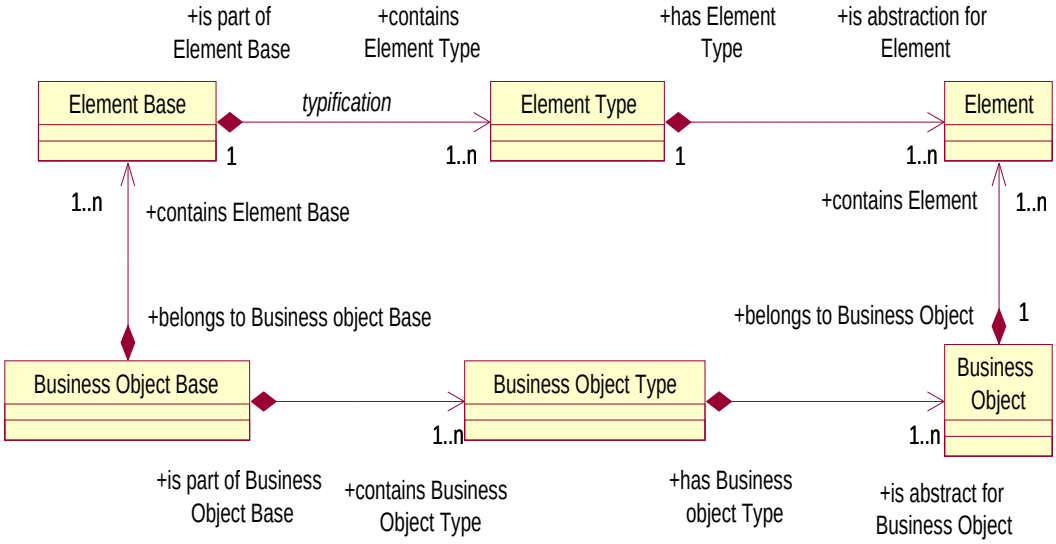


F.1.4 Activity diagram for the Sale event, refined

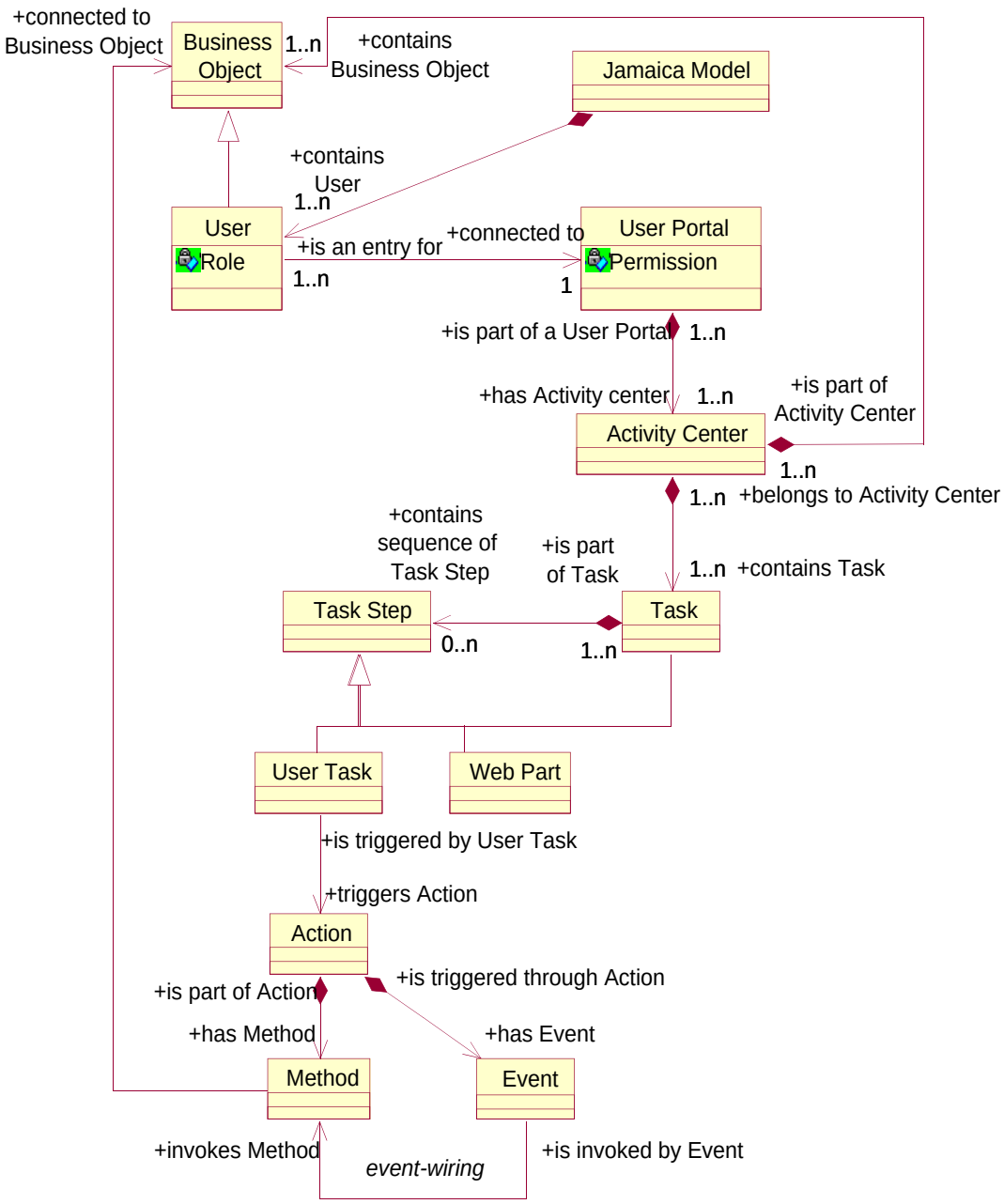


F.2 UML diagrams for the Jamaica conceptual model

F.2.1 Class diagram for Business Objects and Elements



F.2.2 Class diagram for Jamaica conceptual model


+connected to
Business Object
Business Object
1..n
+contains
Business Object
Jamaica Model
+contains
User
1..n
User
Role
+is an entry for
1..n
+connected to
1
User Portal
Permission
+is part of a User Portal
1..n
+has Activity center
1..n
+is part of
Activity Center
Activity Center
1..n
1..n
+belongs to Activity Center
+contains
sequence of
Task Step
+is part
of Task
1..n
+contains Task
Task Step
0..n
1..n
Task
User Task
Web Part
+is triggered by User Task
+triggers Action
Action
+is part of Action
+is triggered through Action
+has Method
+has Event
Method
Event
+invokes Method
event-wiring
+is invoked by Event

G Sowa's diamond