Chapter 6

CASE STUDY AND EXAMPLES

The **case study** research strategy contributes (in general) to capturing some practical perspectives of the investigated problems. It is helpful for considering the knowledge of the practitioners in exploring the research area. According to Yin [85], a *case study* is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident. The *case study* as a research strategy comprises an all-encompassing method – with the logic of design incorporating specific approaches to data collection and data analysis. Although in the past *case study* had been considered only as an exploratory tool [54], they have proved to be more than just an exploratory strategy. Some of the best and most famous *case studies* have been both descriptive and explanatory [85].

Usually, *case studies* are the preferred strategy when HOW or WHY questions are being posed, when the investigator has little control over events and when the focus is on a contemporary phenomenon within some real-life context [85].

In order to realize successfully this strategy, it is essential to design properly the particular *case study*, to collect precisely consistent evidence and to analyze it.

Particularly in the current research, the *case study* strategy is proposed being applied following a specific goal, namely to **test the applicability of a proposed research approach**.

Hence, in the current chapter, the applicability of *SDBC* will be demonstrated by means of a test case study carried out at a **large insurance company in The Netherlands**. Since it prefers not to be mentioned by name, the company is referred to as '*Icomp*' (a name given by us, standing for 'Insurance company'). The goal of the *case study* is not only to provide practical evidence about the strengths of *SDBC* but also to **validate** some of the essential ideas and concepts suggested within the current book. Following the *Icomp* case, we present small illustrative examples for the sake to briefly illustrate things that are not 'covered' by the *Icomp* case.

This chapter is structured as follows: Section 6.1 presents the case study background, bringing elicitation on the case's focus, problem, and goals as well as on the selection of the target organization (where the case study has been carried out). Section 6.2 outlines the collected information, to be used as an input for the application of *SDBC*. This information has been kindly delivered to us by representatives of the target organization. The particular application of the *SDBC* approach is reflected in Section 6.3. Section 6.4 contains concluding remarks. And in the end, Section 6.5 presents illustrative examples, as above mentioned.

6.1 Background

The preparation of the *case study* has been driven by its goal and also by the consideration of several other relevant aspects. Among them are the selection of a target organization, the *case study* focus, and the problem definition. Those aspects will be briefly discussed below:

Goal of the *case study*
From the perspective of the needs of the current research, the main goal of the *case study* has been defined as follows: validation of the conceptual framework of the *SDBC* approach and also of its application guidelines, by applying them in a real-life situation. Taking into consideration the essential elements presented as part of those guidelines, the *case study* should focus on the following aspects:

- *Business processes* analysis and *modeling*, comprising the consideration of the initial (case) information, the structuring of its elements, the identification of relevant *business process modeling units* and their adequate reflection in *business coMponents* (soundly elicited in terms of *structure*, *dynamics*, *data*, and *communicative issues*);
- Derivation of a *software specification model*, comprising the reflection of a *business coMponent(s)* into a corresponding *software specification model*, to be further decomposed into *software coMponents*.

Selection of an organization to be explored
In order to adequately validate *SDBC*, a suitable organization had to be found, an organization – willing to participate in the *case study*, by providing relevant information. The choice of organization was based on the following criteria:

- Size of the organization. The bigger an organization the greater the complexity of its *business processes*; this in turn concerns the sophistication of the support provided by corresponding *information systems* to those *business processes*. For instance, small organizations working in an ad-hock manner are rarely facilitated by sophisticated *information systems* and technologies. Thus, focusing (in this research) on the support to *business processes*, provided by *ICT applications* which are comparatively (more than average) complex, we have had the requirement for a large organization (consisting of more than 2000 employees).

- The business domain. As it is well-known, organizations belonging to some business domains are more dependable on a proper ICT application support than organizations belonging to other business domains. We have targeted business domains related to the financial sector because financial companies are currently among those which greatly depend on *information systems* and technologies.

Hence, as stated before, the *case study* considered in the current chapter, has been carried out in a *financial* (in particular *insurance*) company, namely the company *Icomp*, situated in The Netherlands. *Icomp* delivers financial products (and financial services) to end customers.

Focus of the *case study*
Considering the company *Icomp*, the *case study* has focused in particular on a part of the company's business, namely: the distribution of financial products. This choice has been made not only because such a focus has a direct relation to the core of the business of *Icomp* (this will be seen from the information provided in the following section) but also because the mentioned part of the business of the company strongly requires appropriate *business process modeling* and is dependent on support by *information systems* and technology. Therefore, relevant improvements in these directions could be much useful for *Icomp*.

Problem definition
Considering the available actual information about *Icomp* (this information is reflected in the following section), we have defined two problems to be addressed in the current *case study*:

- The *environment* of *Icomp* demands a sounder and more flexible way in which the company specifies and modifies its financial-products-related *Business Processes*, grasping adequately all essential aspects.
- A better clarity would be appreciated about the impact of eventual reorganization within the company's financial-products-related business activities. This is driven by the necessity of introducing relevant technology in support of the mentioned activities.

Goal in context
Considering the main goal of this *case study*, which has already been formulated, and in relation to the defined problems (addressed above), we have made the following elaboration of the general case study goals, in the light of particular benefits that the case study could bring to *Icomp*:

- Provide insight into the way in which the financial-products-related *business processes* of the company could be modeled so that there is a possibility for flexible modifiability (facilitated by re-use options, for example), soundness, and completeness (regarding the essential aspects of a business reality).
  - This might include the *modeling* of the essential issues characterizing the company and its *environment*.
- Provide insight into the way in which a *software specification model* could be (soundly) derived on the basis of a *business process model*.

- This might include a proposition concerning the introduction of an *ICT application* and a demonstration how its specification could be realized.

The following section will provide information about *Icomp*.

6.2 Icomp

As stated before, this *case study* has been carried out at the company *Icomp*. The current section will briefly introduce it, considering one particular view on *Icomp*, namely the financial products distribution part of its business (this view has already been mentioned in the previous section). Further on, we will mean by '*Icomp'* just those things concerning the company, which are associated with this particular view.

This perspective on *Icomp* has been taken (as explained) because of its direct relation to the core of the company's business, namely distribution of financial products to end customers through brokers. As the studied information shows, particularly this essential part of the business of *Icomp* would (eventually) need an application support.

Distributing financial products through *brokers* means that there are a number of (insurance) financial companies, a number of brokers, and a number of end customers, concerning this distribution mechanism. *Broker j* distributes products of a number of companies (including *Icomp*, if it has an agreement with *Icomp*) to a number of end customers. *End customer k* might be advised by a number of *brokers* about the products of a number of financial companies. Hence, *Icomp* uses a number of *brokers* through which it distributes its (financial) products to a number of *end customers*. Thus, we could relate *Icomp* basically to two actor-role types, namely 'BROKER' and 'END CUSTOMER', as shown on Figure 6.1. BROKER could be fulfilled by any of the intermediary (brokerage) companies registered with *Icomp*. END CUSTOMER could be fulfilled by any human or organization interested in the financial products distributed by *Icomp*.

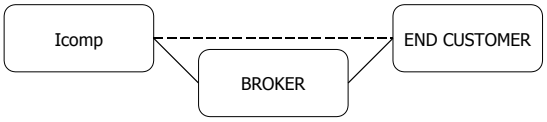

Fig. 6.1. Brokers, facilitating the relations of Icomp with end customers.

Thus, on the figure, the line between *Icomp* and END CUSTOMER is dashed, indicating that the relation to *end customers* is indirect; it comes through *brokers*.

The *brokers* collaborating with *Icomp*, distribute its financial products on the basis of an agreement specifying which products the particular *broker* could sell to *end*

customers and what commission the *broker* would get from *Icomp*. The following information elaborates further on the *Icomp-broker* relation:

- An agreement can be started/changed/ended between *Icomp* and a *broker*.
- A *broker* might receive support from *Icomp*. For example, if a *broker* has been successful in selling products (of *Icomp*) to representatives of a particular customer segment, it might be useful that *Icomp* provides to the *broker* a specialized training concerning this particular segment.
- The commission paid by *Icomp* to a *broker* is as follows:
  - For each new agreement, a *broker* gets 'starting commission'.
  - For each month in which an *end customer* keeps his/her insurance (particularly advised by a *broker*), the *broker* gets 'monthly commission'.
- A *broker* must pay a premium to *Icomp* for an agreement initiated.

With respect to the financial products distribution, *Icomp* has relations not only with intermediary (brokerage) companies but also with *re-insurance companies*, *product development companies*, *investigation companies*, and other (less important) ones.

It is possible that in some cases a *re-insurance company takes* over insurance risk from *Icomp*.

In complicated situations, *Icomp* relies on *investigation companies* for the provision of expert support, for instance: realization of an expertise.

For keeping its product portfolio actual, *Icomp* receives support from *product development companies* delivering new financial (including insurance) products.

As for *Icomp* itself, it is essential to consider its being divided into five departments: Account Management, Acceptance, Claims, Finance, and Marketing.

The *Account Management department* manages the *Icomp-broker* relations. It proposes agreement(s) to a *broker* and once an agreement is signed, the department controls its execution by making sure that the *broker's* results are in accordance to what is in the agreement.

The *Acceptance department* handles requests of end customers for financial product(s), for example, a request for a property insurance.

The *Claims department* deals with claims of end customers and the (eventual) investigation of these claims by experts.

The *Financial department* deals with payments, including the premium payments received by *Icomp* from *end customers*, the payments of *Icomp* to *end customers* for claims, the commission payments that *brokers* receive from *Icomp*, and the payments of *Icomp* to product development companies.

The *Marketing department* is responsible for the product strategy of *Icomp*, dealing with product development, and also with advertising and public relations.

The following section focuses on the application of the *SDBC* approach.

6.3 Applying SDBC

Based on the *case study* background on one hand and on the information about *Icomp*, on the other hand (addressed subsequently in the previous sections), this section is to elaborate on how *SDBC* could be applied within the context of the *Icomp* case:

The section is divided into three sub-sections:

- Sub-section 6.3.1 will focus on the considered (within the *case study*) information and the identification (based on this information) of several relevant *business coMponents*.
- Sub-section 6.3.2 is to consider the specification and elaboration of a particular *business coMponent*.
- Sub-section 6.3.3 will demonstrate the derivation of a *software specification model*, based on the specified *business coMponent* (Sub-section 7.3.2).

Since the steps-to-follow in applying the *SDBC* approach have been introduced, explained, and discussed in the previous chapter, we will now just follow those of them relevant to the tasks within the case, without explaining in much detail those steps.

6.3.1 From the Case Information to Business CoMponents

As mentioned before, in this sub-section, we will show how *business coMponents* could be identified based on the case information (Section 6.2). We provide below a roadmap (fully consistent with the *SDBC* outline (Chapter 5)) which gives in advance information about the modeling activities (steps) to take place within the current sub-section, in order to achieve what we have already defined as a goal [54]:

Step 1 : Building of a generalization hierarchy for the explored domain.
Step 2 : Identification of relevant actor-roles.
Step 3 : Identification of the corresponding inter-role actions (relations).
Step 4 : Elaboration of those relations with semiotic norms.
Step 5 : Decomposition of *Icomp* and a related positioning of the relations.
Step 6 : Construction of a SCI chart.
Step 7 : Derivation of *business coMponents*.
Those seven steps will be addressed within the current sub-section.

Building of a generalization hierarchy for the explored domain
Structuring and positioning semantically the case information is in line with Activity 1 (from the *SDBC* Input/Output Model (Chapter 5)).

As a starting point from the case information (Section 6.2) we select the entities (natural/legal persons) who collaborate with the target company (*Icomp*). They are (in alphabetical order): *intermediary companies*, *investigation companies*, *product development companies* and *re-insurance companies*. *Investigation companies* are actually a sub-type of *consultancy companies* (according to the interviewed specialists from *Icomp*). The rest of the mentioned ones are sub-types of *financial companies*. Being an insurance company itself, *Icomp* is a financial company too.

This information is sufficient for identifying a **`generalization hierarchy`** (organizational business objects model) for the explored domain. The hierarchy is charted in accordance with the guidelines proposed [54], with the aim of bringing order in the original input information. The organizational business object *model* regarding *Icomp* is shown in Figure 6.2-a:

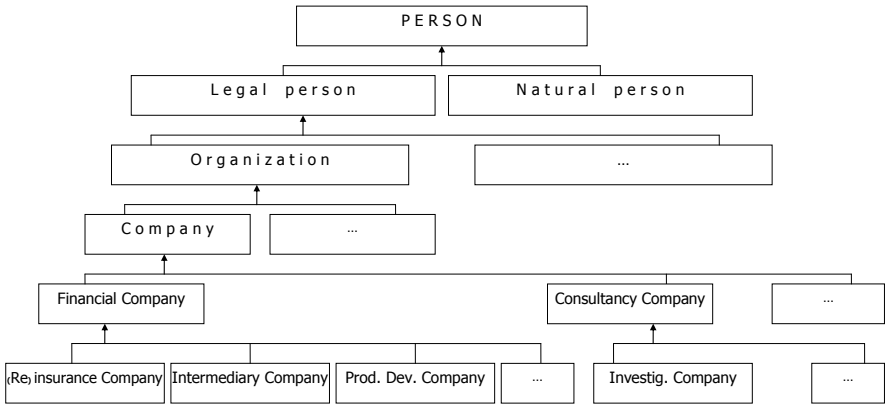

Fig. 6.2-a. The organizational business object model concerning the case study.

Hence, as seen from the figure, *Icomp* collaborates with:

- three types of financial companies, namely – 1) re-insurance companie(s), 2) intermediary companie(s), and 3) product development companie(s);
- investigation companies which are type of consultancy companies.

The position of *Icomp* within this model is also clear – *Icomp* is an insurance company.

Besides these types of companies, *Icomp* collaborates also, of course, with its customers. According to the considered case information, a customer of *Icomp* might be any person, legal or natural. This is illustrated on Figure 6.2-b where 'LP' stands for *legal person* and 'NP' stands for *natural person*.

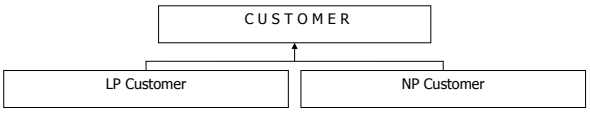

Fig. 6.2-b. The business object model regarding the customers of Icomp.

In the rest of this chapter, any customer, no matter to which of the two customer sub-types (Figure 6.2-b) belongs would be called '*customer*'. The reason is that *Icomp* does not distinguish between its customers in any way. It is to be stated also that besides the term '*customer*', the materials concerning *Icomp*, including the case briefing, contain also the terms '*client*' and '*end customer*'. Actually, those two are synonyms of '*customer*'. Therefore, they will be left out for the rest of this chapter and only the term '*customer*' will be used.

Identification of relevant actor-roles

Following the roadmap, the next step is to produce an **`actor role model`** based on the *business object models*. As studied and motivated already, if the actor-role concept is applied, then it would be easier to model complex *systems* – we have mentioned as well examples of role substitution, which are in support of this view: for instance, if a manager sends a fax, then (s)he plays the role 'Secretary'. Hence, in such a case, if we do not model an actor-role, we should either model the individual natural/legal persons (which is too much complicated in such situations), or oversimplify those issues, which might lead to limitations in the *enterprise model* being created). Further on, the term '*role*' will be used, meaning '*actor-role*'. (see Chapter 2).

We proceed, thus, towards a *role* identification which is in tune with the *SDBC* application guidelines [54]. It starts with an initial consideration of the roles which are typical for each of the identified company types (Fig. 6.2-a). In particular, the starting point is to find (formulate) a suitable word for each of those *roles*. This is done by studying the case information (*customers* should be considered, too). The next step is to find out whether there is information of a type of company which in some situations takes roles different from its typical role (in the *Icomp* case we do not have such examples).

However, formulating a word does not give full information about the meaning of the *role*. Therefore, the word should be extended with some elaboration. By *role* elaboration is meant describing what characterizes a particular role. We do this as follows:

- The typical *role* type for a reinsurance company is formulated as REINSURER: one fulfils REINSURER if taking over a risk from an existing insurance. A re-insurance company is not expected to take other *roles*.
- The typical *role* type for an intermediary company is formulated as BROKER: one fulfils BROKER if matching *customers* to relevant financial companies, in particular – insurance companies (in this case), by 1) giving to *customers* financial consultations about those companies; 2) directing *customers* to particular companie(s) if there is a match between *customer* requirements and company product(s) – this direction is realized by advising for a product of a particular company. An intermediary company is not expected to take other *roles*.
- The typical *role* type for a product development company is formulated as SUPPLIER: one fulfils SUPPLIER if delivering financial products to insurance companies. A product development company is not expected to take other roles.
- The typical *role* type for an investigation company is formulated as EXPERT: one fulfils EXPERT if realizing expertises (analyses and investigations) for insurance companies. An investigation company is not expected to take other *roles*.
- In is necessary also to formulate the *role* type CUSTOMER: one fulfils CUSTOMER if purchasing financial (including insurance) products and providing specialized information upon request.
- *Icomp*, as the target company in this study, fulfils the role type INSURER; one fulfils INSURER if selling financial (including insurance) products.

We stress upon the fact that we have identified *role* types (rather than particular roles). This means that, speaking of REINSURER, for example, we are not interested in any particular instance(s) related to this *role* type. It could be fulfilled by many re-insurance companies.

Those (identified) *role* types are expected to relate somehow to *initiators*/*executors* of particular *transactions*. This, as a part of the modeling output reflected in the current section, would facilitate the identification of *business coMponents*.

Figure 6.3 shows the identified *role* types and also their elaborations. They are depicted in rectangles outlined by double line. Attached to them are rectangles outlined with single line. The role elaborations are depicted in them.

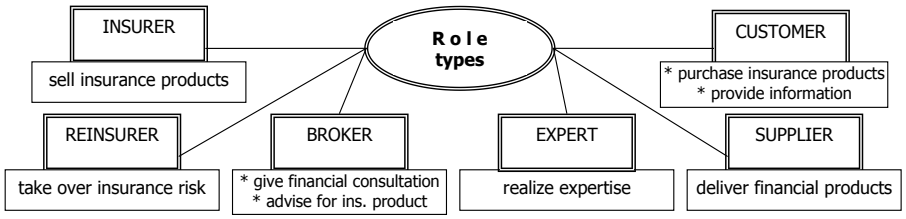

Fig. 6.3. Basic role types within the Icomp case.

As seen from the figure, there are six *role* types: INSURER corresponds to insurance companies (such is the company under study (*Icomp*)), CUSTOMER corresponds to the customers of insurance companies, and the other *roles* are straightforwardly derived from the hierarchy model represented on Figure 6.2 a: REINSURER (Figure 6.3) is the *role* type typical for a Reinsurance company (Figure 6.2 a), BROKER (Figure 6.3) is the *role* type typical for an Intermediary company (Figure 6.2 a), and so on. As for the *role elaborations* which are also depicted on Figure 6.3, they have been formulated based on the case study information and interviews with employees of *Icomp*.

Identification of inter-role relations

Based on the identified major *role* types, the actions (relations) among them are studied. We will call those relations *inter-role relations* from now on, or '*relations*' for short. Studying the *relations* would be useful with regard to a consideration of the structure and dynamics of the explored *enterprise system*. As a first step in identifying the existence of relations, the interviewed *Icomp* employees were asked to answer whether or not a *relation* exists between each two of the *role* types. Table 6.1 contains the collected data. As seen from the table, only the grey rows correspond to an existing relation. For example, from the third row (from top to bottom) it is seen that there exists a relation between INSURER and EXPERT.

INSURER	REINSURER	Yes, a relation exists
INSURER	BROKER	Yes, a relation exists
INSURER	EXPERT	Yes, a relation exists
INSURER	SUPPLIER	Yes, a relation exists
INSURER	CUSTOMER	Yes, a relation exists
REINSURER	BROKER	No, a relation does not exist
REINSURER	EXPERT	No, a relation does not exist
REINSURER	SUPPLIER	No, a relation does not exist
REINSURER	CUSTOMER	No, a relation does not exist
BROKER	EXPERT	No, a relation does not exist
BROKER	SUPPLIER	No, a relation does not exist
BROKER	CUSTOMER	Yes, a relation exists
EXPERT	SUPPLIER	No, a relation does not exist
EXPERT	CUSTOMER	Yes, a relation exists
SUPPLIER	CUSTOMER	No, a relation does not exist

Table 6.1. Identified inter-role relations.

The next step, according to the *SDBC* guidelines [54], is to describe briefly each identified relation. In achieving this, we will firstly consider in more detail the particular *role* types and secondly – address the aspects that concern their relations.

The first sub-task could be realized through binary relationships - a *binary relationship* does concern two entities and is usually described by: noun-verb-noun; the nouns correspond to the entities and the verb describes the relation among them. If we take, for example, the role type COMPOSER, related to the expression: 'writing songs', then we could form a binary relationship. It would be between COMPOSER and song (between the *role* type and something related to its *output*). The verb 'write' describes how those two relate.

Thus, looking at Figure 6.3, we could analogously form binary relationships since, as it could be seen, a *role* type name plus a *role* elaboration could be considered as a *noun-verb-noun* expression. Thus we have:

INSURER - sell - (insurance) products
REINSURER - take over - (insurance) risk
BROKER - give - (financial) consultation
BROKER - advise for - (insurance) products
EXPERT - realize - expertise
SUPPLIER - deliver - (financial) products
CUSTOMER - purchase - (insurance) products
CUSTOMER - provide - (specialized) information

Further, we extend (based on Table 6.1) each of the above eight expressions with one more noun corresponding to a *role* type which relates to the *role* type represented within the particular expression. This is done as follows: For a particular role type, we can see the 'candidate' matches from the table. Thus, we have to choose any of them. The criterion is how it matches the context of the expression. For example, starting from INSURER, we see from Table 6.1 that it relates to REINSURER, BROKER, EXPERT, SUPPLIER, and CUSTOMER. Therefore, we ask the question: To whom does INSURER sell insurance products? The answer (according to the case

information) is: ‘to CUSTOMER’. Therefore, we extend the first expression with CUSTOMER:

INSURER - sell - (ins.) products - CUSTOMER

If we go further, we see from Table 6.1 that REINSURER relates only to INSURER; we ask the question: From whom does Reinsurer take over risk? The answer is: ‘from INSURER’. Therefore, we extend the second expression with INSURER:

REINSURER - take over - (ins.) risk - INSURER

We continue analogously:

BROKER - give - fin. consultation - CUSTOMER
BROKER - advise for - financial products - INSURER
EXPERT - realize - expertise - INSURER
SUPPLIER - deliver - financial products - INSURER
CUSTOMER - purchase - ins. products - INSURER
CUSTOMER - provide - spec. information - EXPERT

We now need to consider the above expressions and check them against redundancy since there is a risk to describe twice one and the same thing, like in the following two expressions:

INSURER - sell - insurance products - CUSTOMER
CUSTOMER - purchase - insurance products - INSURER

Considering the case information, we have concluded that the information in the above two expressions is about one and the same thing, namely the INSURER’s selling of insurance products to CUSTOMER. Therefore, we randomly choose one of the above two expressions and leave out the other one. Let’s select the first one.

Further, we will use the above expressions as an input for building the so called Icomp RR chart, to facilitate the description of relations (‘RR’ stands for ‘Roles & Relations’). In order to build the chart, we need to consider the above expressions, putting the role type names in boxes. The names of those of the role types which relate to the realization of a particular activity (for example, the activity: ‘sell insurance products’) are underlined. Next to that, the name of the role type corresponding to the target (within the case study) organization should disappear. On its place we put the particular name of the organization (*Icomp*). Hence, we should replace the type role (INSURER) with the instance role (‘*Icomp*’). This is because we are not interested in any company which could fulfil INSURER but in this *role* as performed in particular by the company *Icomp*.

Between each two boxes, concerning role types and characterizing a particular relation, we should put together all the text (from the corresponding expressions above) which is between the names of the role types. For example, we take the text: ‘realize expertise’ from the line:

EXPERT - realize - expertise - INSURER

The RR chart is depicted on Figure 6.4. As seen from the figure, each line contains two role type names (the name of the target company is in some places instead a role type name) and in between is the description of the relation. All those are derived straightforwardly from the previously constructed expressions. As it could be seen, we have also given a unique code to each relation (**R1** to **R7**). Onwards, we will refer to each of the modeled relations using those codes.

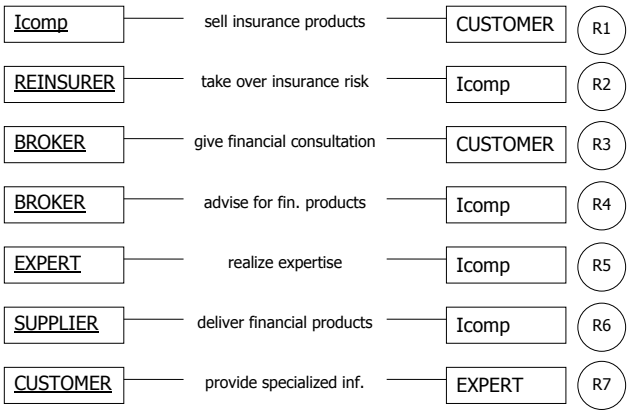

Fig. 6.4. RR model for the Icomp case.

Norm elaboration

According to [54], an important step to take place on that basis is adding further precision to the descriptions of the relations, by applying *organizational semiotics*, and in particular - *norm analysis* (see Chapter 3). The construction of *norms* has been considered in the mentioned chapter. Hence, no explanations will be made here about how the *norms* are constructed on the basis of the information on corresponding relations. A *norm* will be attached to each relation (Figure 6.4), being given a name containing '**N**' (from the word '*norm*') and the code of the relation. The seven constructed *norms* are:

NR1
Whenever BROKER has advised CUSTOMER in favour of a Icomp's product and CUSTOMER fits within Icomp's policy
If CUSTOMER decides to purchase this product
Then Icomp
Is obliged to insure CUSTOMER according to the concrete product details and based on a payment from CUSTOMER, made accordingly.

NR2
Whenever there is a long run relation between Icomp and REINSURER
If an insurance to be realized by Icomp would include a unacceptably high risk for Icomp and the insured objects fit within REINSURER's policy
Then (if asked) REINSURER
Is obliged to take over risk(s) from Icomp regarding the particular insurance.

NR3
Whenever CUSTOMER has a request for consultation to BROKER
If an insurance company having got an agreement with BROKER has an appropriate product with regard to the CUSTOMER's particular request
Then BROKER
Is obliged to consult CUSTOMER about this product.

NR4
Whenever there is an agreement between Icomp and BROKER
If a product of Icomp is a best match with regard to a CUSTOMER's request
Then BROKER
Is obliged to do advice for CUSTOMER in favour of Icomp's product(s).

NR5
Whenever there is a non-standard situation regarding a stated claim
If Icomp asks EXPERT for an expert evaluation (expertise)
Then EXPERT
Is obliged to realize an expertise with regard to the stated claim.

NR6
Whenever there is an agreement between Icomp and SUPPLIER about delivery of insurance products
If CUSTOMER wants to have a product whose production and delivery falls in the mentioned agreement as a responsibility of SUPPLIER, and Icomp has ordered this financial (in particular - insurance) product to be developed
Then SUPPLIER
Is obliged to deliver the financial product.

NR7
Whenever EXPERT is involved in an expert evaluation (expertise)
If EXPERT asks CUSTOMER for specialized information
Then CUSTOMER
Is obliged to cooperate by providing the required information.

Positioning of the relations

We have realized so far an identification and a thorough elaboration of the essential *relations* concerning the *Icomp* case.

Our position towards *Icomp* as the company under study requires adding more precision about the way *Icomp* handles the mentioned relations internally. Said otherwise, it is of interest to know which of the departments (organizational units) within the company are involved in which of the relations. Such information would be of significant importance for specifying an ICT application which, for example, might operate across some (or all) of the mentioned departments.

Therefore the next step should be to position the relations in terms of the *Icomp* organizational units. Those units have been defined based on the information about *Icomp* (Section 6.2).

We consider Figure 6.4 and leave out of consideration the relations **R3** and **R7** because they do not relate directly to *Icomp*. We take then the remaining relations (which all concern *Icomp*) and conduct interviews in order to clarify for each particular relation the corresponding involved *Icomp* department(s). Of course, it appears that often a relation concerns more than one department. For example, the relation between *Icomp* and BROKER comes firstly through the *Account management department* (considering the agreement and also the *Icomp* – BROKER collaboration in general) and secondly, through the *Financial Department* (as long as payments are concerned).

Figure 6.5 contains the results. The names of the *Icomp* departments are put in rectangles. Each relation (in circle) having connection with a department is linked to it

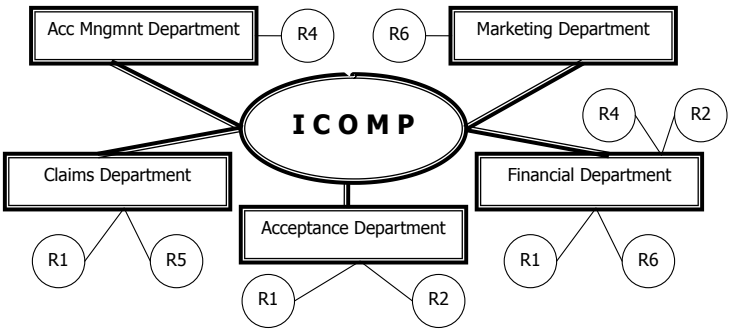

Fig. 6.5. Relations and organizational units concerning the Icomp case.

We have purposefully simplified slightly the way we look at the organizational structure of *Icomp* because this would make our further modeling activities easier to understand. However, the modeling complexity would be sufficient for adequately demonstrating the strengths of the *SDBC* approach.

SCI chart

Based on the modeling results which have been achieved so far, we will (according to [54]) apply the SCI chart in order to summarize the initial case information ('*SCI*' stands for '*Structuring Customers' Information*'). The modeling outputs depicted in Figures 6.3, 6.4, and 6.5 should be a sufficient basis for constructing the chart. The *SCI* chart is presented in Figure 6.6 where the following abbreviations are used:

am	stands for	'Account management department';
md	stands for	'Marketing department';
fd	stands for	'Financial department';
ad	stands for	'Acceptance department';
cd	stands for	'Claims department'.

On the figure, the target organization (*Icomp*) is represented within the rectangle with rounded corners. Inside are depicted the five departments (source: Figure 6.5); within an attached rectangle is an elaboration concerning *Icomp*. In rectangles around the rounded cornered rectangle, are depicted the five considered *role* types plus their elaborations (source: Figure 6.3). On the basis of Figures 6.4 and 6.5, the *role* types are linked (where appropriate) to corresponding departments within *Icomp*. Also, where appropriate, the roles are linked to each other. Each line is given a number.

In this way, through the *SCI* chart, we have achieved a compact, complete, and focused view on the target organization (and relevant information).

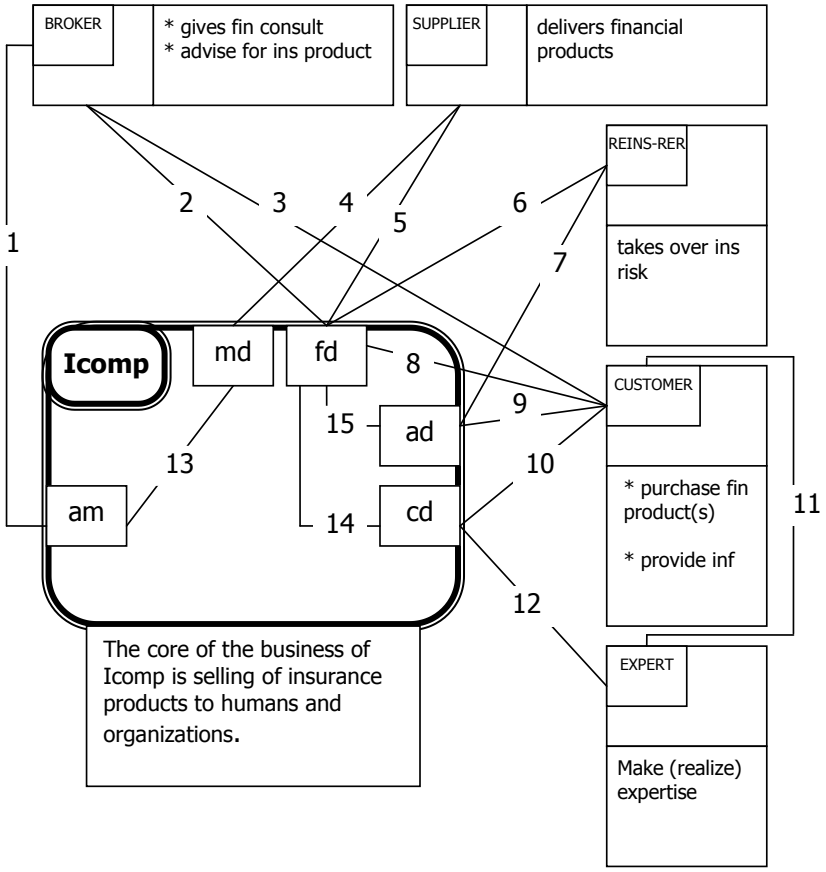

Fig. 6.6. Icomp SCI chart.

Derivation of business coMponents

According to the guidelines for application of *SDBC* [54], a *SCI* chart could facilitate the identification of *business coMponents*, particularly using the notations concerning *transactions* (see Chapter 5).

We consider the lines from the *SCI* chart - each line originates one or more *business process patterns*. In those patterns, we consider the organizational units within *Icomp* as *roles*. Hence, the set of *business process patterns*, derived from the *Icomp SCI* chart is:

1	***am***	***start***	***AGREEMENT***	***BROKER***
1	*am*	*end*	*AGREEMENT*	*BROKER*
1	*am*	*manage*	*AGREEMENT*	

2	*fd*	*pay*	*COMMISSION*	*BROKER*
3	*BROKER*	*advise*	*(for a) PRODUCT*	*CUSTOMER*
4	*SUPPLIER*	*deliver*	*new PRODUCT*	*md*
5	***fd***	***pay***	***PREMIUM***	***SUPPLIER***
6	*fd*	*pay*	*re-insurance PREMIUM*	*REINSURER*
7	*REINSURER*	*start*	*REINSURANCE*	*ad*
7	*REINSURER*	*end*	*REINSURANCE*	*ad*
8	*CUSTOMER*	*pay*	*PREMIUM*	*fd*
8	***fd***	***pay***	***CLAIM***	***CUSTOMER***
9	*ad*	*start*	*CUSTOMER AGREEMENT*	*CUSTOMER*
9	*ad*	*end*	*CUSTOMER AGREEMENT*	*CUSTOMER*
9	*CUSTOMER*	*give*	*HEALTH INFORMATION*	*ad*
10	*CUSTOMER*	*declare*	*DAMAGE*	*cd*
10	*cd*	*state*	*COMPENSATION*	*CUSTOMER*
11	*CUSTOMER*	*give*	*HELATH STATEMENT*	*EXPERT*
12	*EXPERT*	*give*	*EVALUATION*	*cd*
13	*md*	*provide*	*new PRODUCT*	*am*
14	*cd*	*order*	*CLAIM PAYMENT*	*fd*
15	*ad*	*order*	*PREMIUM PAYMENT*	*fd*

This output represents the starting point for the identification of *business coMponents*. Essential for this is the discovery of *transactions*. It is claimed (and motivated) that the above output could facilitate the mentioned discovery. Next to that, this output's being focused adds value to the overall consistency of the set of *transactions* and *business coMponents* (being identified).

We will take, for illustrative purpose, several of the above *business process patterns*. Through the identification of *transactions*, we will reflect those patterns in coordination-structure models (that represent the actor-roles, transactions, and the system boundary), identifying in this way *business coMponents*. We will consider, in particular, the patterns represented in bold on the above list, starting with the following one:

1	*am*	*start*	*AGREEMENT*	*BROKER*

Firstly, we are clear what the system under study is. It is *Icomp* (the *Account management department* is one of its departments) - this is reflected in one of the *roles* in the above expression. Secondly, we are clear which the *roles* under consideration are; in this case they are 'am' and 'BROKER'. Hence, we could model this as presented in Figure 6.7:

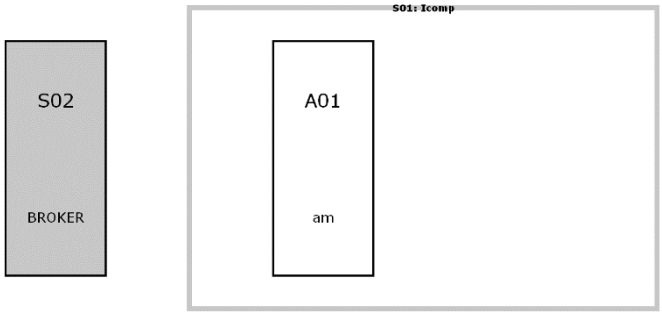

Fig. 6.7. Representation of a pattern.

What we know also from the pattern is the essence of the *inter-role relation*: 'start agreement'. However, reflecting it directly in one *transaction* would not provide a complete view since we need to analyze this information and identify the starting transaction (see *Definition 6*). To achieve this, one would (usually) answer the helpful question: 'what is the cause'. We have done this, discovering that a broker could have an agreement started only based on an application (submitted). Therefore, the *starting transaction* would be:

T2 application F2 application <*A*> has been submitted

We then ask what happens next. It is that **am** receives an application from a **broker** and, before being able to start an agreement with the **broker**, **am** needs an approval by a *controller* within *Icomp* (we have not considered it so far because of it not having a significant importance). Thus, we identify an additional *role*-type, namely CONTROLLER. As for the corresponding *transaction*, it would be:

T3 approval F3 approval concerning application <*A*> has been done

Based on this approval is the starting of an agreement, by **am**:

T1 agreement F1 agreement based on application <*A*> has started.

Hence, taking the information from Figure 6.7 and also the identified *transactions* (see above) plus the new *role*-type (*controller*), we are able to build the relevant **business coMponent**. This is depicted in Figure 6.8 (see the notations presented in Chapter 5):

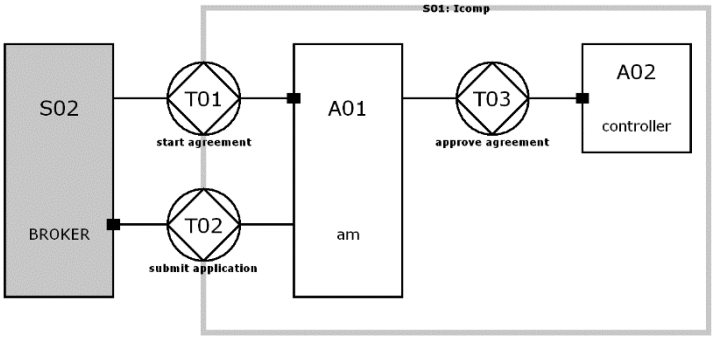

Fig. 6.8. An identified business coMponent – structural view.

Considering the above modeling output and in line with the principles of *LAP* and *enterprise ontology* (see Chapter 3), we construct (see Figure 6.9) a model that elaborates on the communicative aspects concerning the *transactions*.

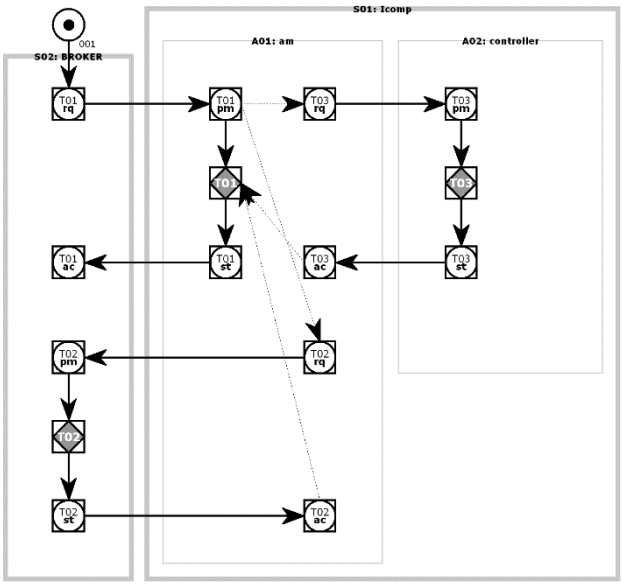

Fig. 6.9. An identified business coMponent – communicative view.

We will be more detailed about the elaboration of identified *business coMponents* in the following sub-section. In this sub-section we just consider the identification of *business coMponents*.

We continue with the rest two *business process patterns* to be considered and reflected in *business coMponents*:

5	*fd*	*pay*	*PREMIUM*	*SUPPLIER*
8	*fd*	*pay*	*CLAIM*	*CUSTOMER*

Proceeding analogously, we will identify *business coMponents* based on the *patterns*.

Since both *patterns* concern payment, we propose using a re-usable *business coMponent*, in particular a *general business coMponent*. It is to be extended afterwards. We are not going to explain those issues since they have been considered in Chapter 5.

A general payment *business coMponent* specified in the same notations is depicted in Figure 6.10.

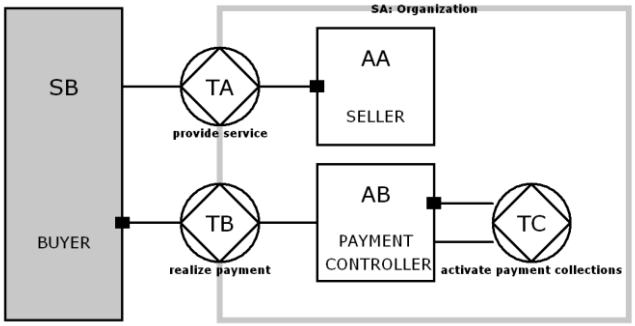

Fig. 6.10. A general payment business coMponent.

As seen from the figure, in the general case, we have an organization providing a service to a customer and claiming therefore payment in return. Usually, the entity delivering the service is not the entity handling the payment: there are two internal *role*-types depicted on the figure, therefore, namely SELLER and PAYMENT CONTROLLER. SELLER delivers a service to the customer ('BUYER') and informs about this PAYMENT CONTROLLER who as a result of self-activation (on a periodic basis) would handle the payment.

Taking the first of the two considered *patterns*, we extend straightforwardly the model shown in Figure 6.10. BUYER in this case would be *Icomp* (its *Marketing department* (**md**) buys a financial product and its *Financial department* (**fd**) has to pay to a corresponding supplier). This is represented on Figure 6.11:

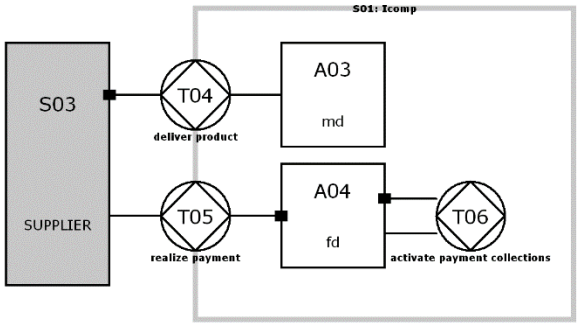

Fig. 6.11. A possible extension of the general payment business coMponent.

As for the second *pattern*, we again reflect straightforwardly the information: this time the payment should be directed to a customer. However, before the payment could be initiated (as studied from the case information) it is necessary that an expert (external to *Icomp*) investigates the case. Considering this accordingly, we derive a *business coMponent*, as represented in Figure 6.12:

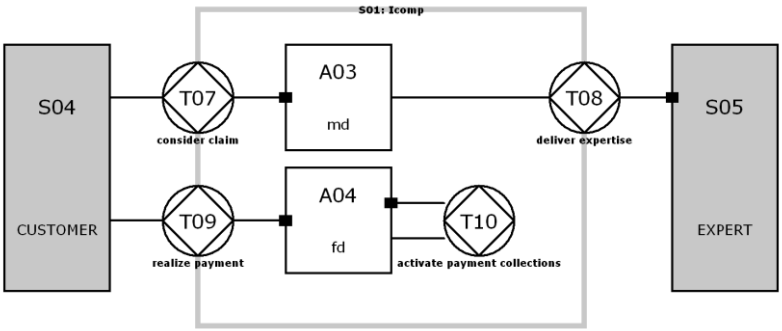

Fig. 6.12. Another possible extension of the general payment business coMponent.

So, we have demonstrated *business coMponents* identification. In the following sub-section, we will consider the elaboration of a particular (identified) *business coMponent*.

6.3.2 Elaborating a Business CoMponent

In the previous sub-section, we have demonstrated the identification of *business coMponents*, using *SDBC*. As mentioned at the beginning of the current section: in this

sub-section, we will demonstrate the specification and elaboration of an identified *business coMponent*; in the following sub-section, we are going to demonstrate the derivation of a *software specification model*, based on the *business coMponent*.

As for the particular *coMponent* to be considered, it will not be one of the coMponents identified on the basis of the *SCI* chart (Sub-section 6.3.1). It will be, instead, a *Business CoMponent* resulting from a business improvement proposal concerning *Icomp* (the conceptual framework of *SDBC* allows for **business re-design**, as a possible design step, whenever this is considered necessary).

Our reason for introducing a business improvement proposal is that such an improvement is expected to create an adequate foundation for realizing a useful software support to *Icomp* while simply automating any currently existing *business processes* within the company would bring less value to it.

Therefore, we will address below:

- the problem concerning the need for improvement;
- a relevant business improvement proposition;
- a resulting *business coMponent*, to be adequately specified and elaborated.

The *business coMponent* will be elaborated in terms of *structure*, *dynamics*, *data* and *communicative issues* (Figure 6.13), as according to the *SDBC* approach.

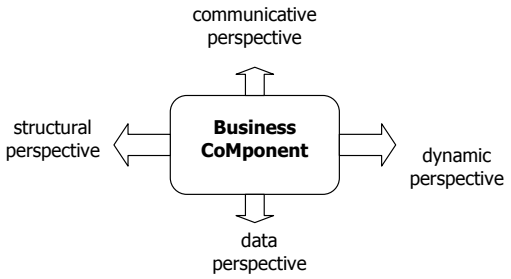

Fig. 6.13. Elaborating a business coMponent.

Problem statement

Regarding some relatively simple cases in which an advice is straightforwardly deliverable (based on relevant information and rules), using human brokers is too expensive. It would be more appropriate if human brokers are used just in cases in which their particular expertise is to be applied.

The Financial Mediator – a proposal

We have made, reflecting the above problem, a business improvement proposal according to which a new business unit is to be introduced, namely a **`Financial Mediator`** (**FM**).

*The **FM** facilitates Dutch insurance companies. In order to use **FM**, a company should subscribe (for free). **FM** brings about the following useful deliverables:*

- *advice (to customers or insurance brokers) on what of the offered (by the registered companies) products best satisfies a particular customer demand;*
- *delivery (to customers) of products of insurance companies.*

*Any customer could request (for free) **FM** to do for him/her either advice or delivery of a product. The customer should firstly specify his/her request (choosing from a list): (s)he should make clear whether the request is about a health insurance, auto insurance, and so on, specify the particular demand (for example: to insure a car against theft with the highest possible coverage (that includes car accessories, tires, and so on)), and so on. Based on this, a request processing unit within **FM** generates a standardized specification regarding the customer's request, which is delivered to a match-making unit within **FM**. This unit is to further realize a match allowing the **FM** to do the advice. The match is driven by the particular criterion chosen by the customer (for example: a preference for the cheaper or the most reliable product available). In order to deliver such a criterion-driven match, the match-making unit uses a data bank of relevant rules and procedures. Besides the output given by the request processing unit, the match-making unit needs as well an output from a data search and processing unit. It searches through the information that concerns registered companies, applying procedures to it. This allows for a precise identification of candidate-matches, relevant to the particular customer's request. Thus, given this output plus the (mentioned) standardized specification of the customer's request, the match-making unit would be able to realize a match, applying the mentioned rules and procedures.*

*As for the subscription of (insurance) companies, any (Dutch) company could subscribe for free. This is facilitated by a subscription processing unit within **FM**. This unit could realize a subscription only after another unit within **FM** (a company profile builder) creates a profile of the particular company, making its data available through a data-bank (to be usable also by the data search and processing unit). Usually, **FM** creates 'standard profiles'; however, several special companies could have 'golden profiles' (with more benefits).*

*Allowance: a customer's using **FM** (either for advice, or contract, or product delivery) is to be limited to no more than five times per month. As for (insurance) companies' allowance, a company is allowed to subscribe to **FM** only if it is licensed according to the Dutch financial laws.*

*As for a product delivery: once a customer has chosen a product, (s)he might request that **FM** facilitates the actual product delivery. The customer requests an offer (**FM** is to be authorized to generate offers, based on information from the particular company, kept in its profile). Once **FM** (its offer generating unit) has produced an offer, it should have it first approved by the respective (insurance) company, before delivering it to the customer. From the moment of the delivery, the particular insurance (or other financial product) is in effect – between the customer and the corresponding (insurance) company.*

*A company should pay a commission to **FM** for each realized (through **FM**) insurance (or other product).*

Financial Mediator (FM) – the Business CoMponent

On the basis of the above proposition, we identify a relevant *business coMponent*, namely the **FM** *business coMponent*.

Since we have already demonstrated the *SDBC business coMponent* identification mechanism, we will not demonstrate the identification itself again (it has been done analogously, as in the previous sub-section). The current sub-section aims instead (as stated already) at demonstrating the specification and elaboration of a *business coMponent*.

Hence, we go directly to the identified *transactions* (the *transactions* listed below). The first six of them relate to the **FM**'s delivering advices. The rest (backgrounded grey) relate to the **FM**'s contracting concerning financial (insurance) products.

T1	*Deliver advice*	F1	advice <A> is delivered
T2	*Perform match-making*	F2	match of advice A is made
T3	*Generate customer's information specification*	F3	customer's information concerning advice A is processed
T4	*Generate candidate-matches*	F4	data search and processing is done concerning advice A
T5	*Realize subscription*	F5	Subscription <S> is realized
T6	*Create profile*	F6	profile is created concerning subscription S
T7	*Offer contract*	F7	contract <C> is offered
T8	*Approve contract*	F8	contract C is approved
T9	*Deliver financial product*	F9	the product specified in contract <C> is delivered
T10	*Submit agreement*	F10	agreement concerning contract C is submitted
T11	*Accept contract*	F11	contract C is accepted
T12	*Activate payment collections*	F12	payment activation <A> is realized
T13	*Realize payment*	F13	Commission for product(s) specified in contract <C> is paid

As seen from the above list, **FM** could deliver an advice. This requires that a match-making is performed, based on a standardized specification of the customer's information and on generated candidate-matches. As for the consideration of (insurance) companies, **FM** could offer them subscription. It is completed only after a particular company profile has been created by **FM**.

It is seen also that once a customer has chosen a particular financial (insurance) product, it could be facilitated by **FM** for the product's delivery. **FM** offers a contract based on which the customer would acquire the product. The contract, however, would need to be approved by the particular company, before being offered to the customer. After it has been offered, the customer should accept it and from this moment on, (s)he has rights and obligations concerning the product. For each product delivery, a payment of commission should take place, from the particular financial (insurance) company to **FM**. A payment controller is activated periodically in collecting all payments due for the particular period.

Further, we will reflect those *transactions* in models, offering elaboration in two aspects, namely structural and communicative. We will then elaborate further those models with semiotic *norms* (see the previous sub-section).

Afterwards we will derive, based on those models, Petri Net and ORM models [54], offering elicitation in terms of dynamics and data, respectively. We will attach to the Petri Net models some further refined norms.

Hence we will address further the structural, communicative, dynamic, and data aspect of the considered business reality. This four-aspect business view (Figure 6.13) is in tune with the *SDBC* foundations (Chapter 5).

Financial Mediator – structural and communicative aspects

For clarity's sake, in modeling the above *transactions*, we will firstly consider those of them which concern the **FM**'s offering advice services, and secondly – those concerning the **FM**'s offering contracting services (backgrounded in grey color).

As for the first of the mentioned *transaction* groups, we have reflected it in the model, represented in Figure 6.14. The model concerns the *structural business coMponent* aspect.

As mentioned before, the identification of such a model has been demonstrated, including the identification of *roles* and *transactions*. Hence, we take them directly in building the model, without explaining how we have identified them.

The functionality of **FM** concerns customers and insurance companies (for short – '**IC**'). Hence, we have two major role-types: CUSTOMER and INSURANCE COMPANY. As seen from the figure, in the model, they are reflected as the roles '*Customer*' (**S02**) and '*Insurance Company*' (**S03**); those roles are external with respect to **FM**. The *transactions* **T01** and **T05** concern the **FM-Customer** and **FM-IC** relations, respectively.

Next to that, a number of actions take place within **FM** where we have identified six roles (internal as concerns **FM**). They are: **A01** (**Advisor**), **A02** (**Match-maker**), **A03** (**Request Processing Unit**), **A04** (**Data Search & Processing Unit**), **A05** (**Subscription Processing Unit**), and **A06** (**Company Profile Builder**). *Transactions* **T01**, **T02**, **T03**, and **T04** as well as roles **A01**, **A02**, **A03**, and **A04** concern directly the advice delivery: they are about the mere (**FM**'s) delivery of an advice to a customer. *Transactions* **T05** and **T06** as well as roles **A05** and **A06** concern indirectly the advice delivery: they are about the collection and use of information (concerning insurance companies) needed for performing an advice.

As seen from Figure 6.14, **A01** is to *deliver advice* (**T01**). However, this could be done only based on a realized *match-making* – a matching between what the particular customer requests and what is offered by the insurance companies (registered with **FM**). **A02** is to realize such a *match-making* (**T02**). What **A02** needs in order to realize the match are two things: 1) a complete specification regarding the request of the (particular) customer, a specification presented in standardized notations (the reason is that if such a specification is not standardized it would be hardly matchable with information concerning insurance companies); 2) list of candidate-matches. **A03** must *generate the mentioned standardized specification* (**T03**) and **A04** should *provide candidate matches* (**T04**). In performing **T04**, **A04** is facilitated by two data-banks, namely **DB01** and **DB02**. These data-banks are claimed to be an essential elaboration concerning the model. Using **DB02**, **A04** gets direction what procedures to apply (and

where to find them) in connection to a particular need expressed by a customer. For example, if a customer needs an auto-insurance, following a procedure helps to adequately direct a further searching through companies (in this particular case – searching in their property insurance departments and/or 'schemes'). Based on such an orientation achieved, **A04** could effectively direct its search for relevant (insurance-companies-related) information, using the bank **DB01**. It contains information about the (insurance) companies registered with **FM**, the (financial) products offered by them, and also other related details. Thus, using these (mentioned) two data-banks, **A04** should be able to provide to **A02** a list of candidate-matches (with regard to the particular (customer) request. Therefore, based on the request specification (delivered by **A03**) and this candidate-matches list, **A02** must realize the match-making. However, this should be done according to a particular criterion (like reliability or quality of service, for instance). It should be specified by the customer. Having received this information, **A02** should apply particular procedures in approaching the 'matching' information (this information would be considered in one way if the cheapest (financial) product is the goal and in another way if the most reliable product is to be selected). With respect to this, **A02** is facilitated by the data-bank **DB03**. It allows **A02** to know what procedure (or a combination of procedures) to apply to the 'matching' information based on a criterion chosen by the customer. The data-banks and related information will be considered in more detail further on within the current sub-section, when we come through the data *business coMponent* aspect.

Also, it is seen from Figure 6.14 that **A05** is to realize the subscription (**T05**) of a (insurance) company wishing to use **FM**. Before a subscription could be handled, a company profile is to be built (**T06**) by **A06**. This includes adding data to the data-bank **DB01** which was mentioned already.

As also seen from the figure, the realization of the **T01** *transaction* includes providing (relevant) information to **A02**, **A03**, and **A04** (the dotted lines between **T01** and these roles indicate this). **A02** is to receive the criterion (chosen by the customer) according to which to perform the match-making. **A03** should receive the (full) information submitted by the customer. **A04** should be provided with the type of the customer need (for instance: 'auto-insurance').

We have done, thus, the basic elaboration on the model and will then add further elicitation using *semiotic norms*. This is a logical continuation of the *norm* derivation characterizing the earlier analysis phases (those phases addressed in the previous section, have presented derivation of more general *semiotic norms* intended to 'govern' the ones to be identified).

Since the role of *norms* and their derivation have already been explained in previous chapters, we will directly go to the content of the identified *norms* attached to the *transactions*. Since they concern the essential level, we will add identification to them as follows: a string consisting of '**E**' (from 'Essential'), '**N**' (from 'Norm') and a number of the particular *transaction*. The derived *norms* are below:

EN1
Whenever S02 has requested advice
If A02 has realized match-making
Then A01
Is obliged to formulate and deliver an advice.

EN2
Whenever S02 has requested advice
If A03 has delivered standardized customer specification AND A04 has delivered candidate matches
Then A02
Is obliged to realize match-making.

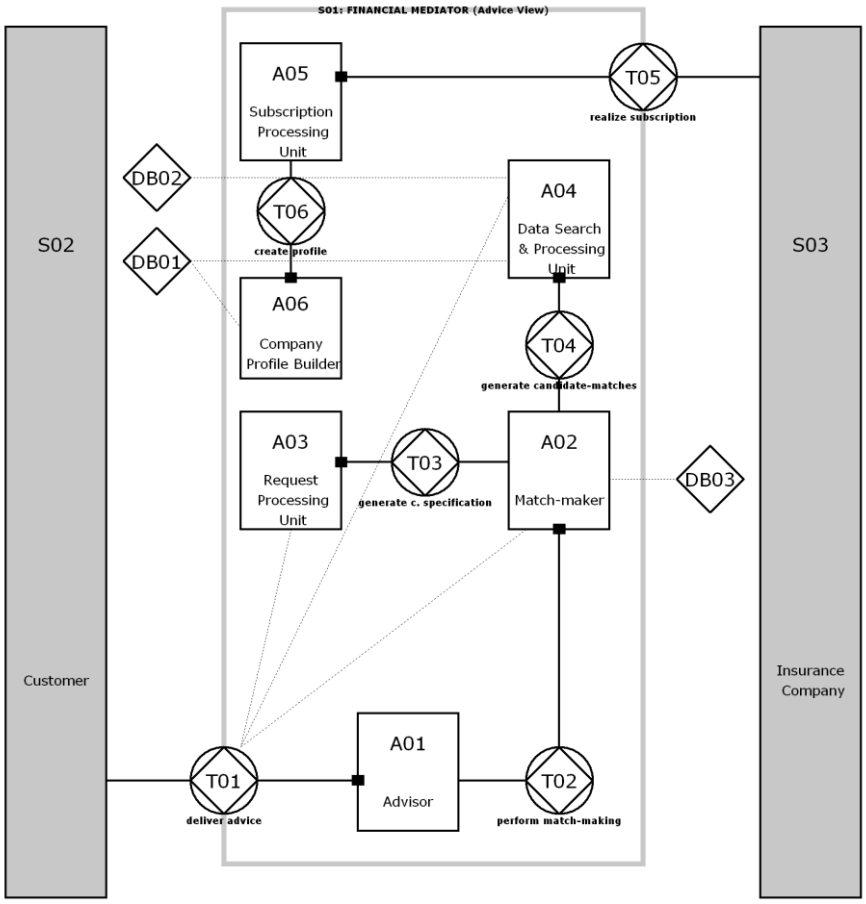

Fig. 6.14. The FM business coMponent (advice view) – structural aspect.

EN3
Whenever S02 has requested advice
If A03 has received submitted customer information
Then A03
Is obliged to delivered standardized customer specification.

EN4
Whenever S02 has requested advice
If A04 has received information about the type of a customer need
Then A04
Is obliged to deliver a candidate-matches list.

EN5
Whenever S03 has requested subscription
If A06 has built a (relevant) company profile
Then A05
Is obliged to realize subscription.

EN6
Whenever S03 has requested subscription
If A06 has received submitted customer information
Then A06
Is obliged to build a company profile.

Based on the model represented in Figure 6.14, we derive a model (Figure 7.15) representing the communicative view on the addressed business reality.

As seen from Figure 6.15, we have added elaboration (concerning the communicative aspect) by applying the *transaction* pattern (see Chapter 3 and Chapter 5) to each of the *transactions* (Figure 6.14).

As seen from Figure 6.15, two sub-processes are to be considered – one of them relates to the FM's delivering advice to a customer, the other one relates to the FM's realizing subscription to an (insurance) company. This is indicated by two starting points (on the figure): starting point **001** and starting point **002**.

As also seen from the figure, the dependence of **T01** on the execution of **T02**, the dependence of **T02** on the executions of **T03** and **T04**, and the dependence of T05 on the execution of **T06** – all these are accordingly reflected in the model.

Therefore, we have considered so far the 'Advice view' over the **FM** *business coMponent* as far as the structural and communicative aspects are concerned. We continue analogously towards the consideration of the 'Contracting view' over the *coMponent*. We proceed analogously and will not offer as detailed explanations as in the previous paragraphs.

The built model corresponding to the contracting view is depicted on Figure 6.16.

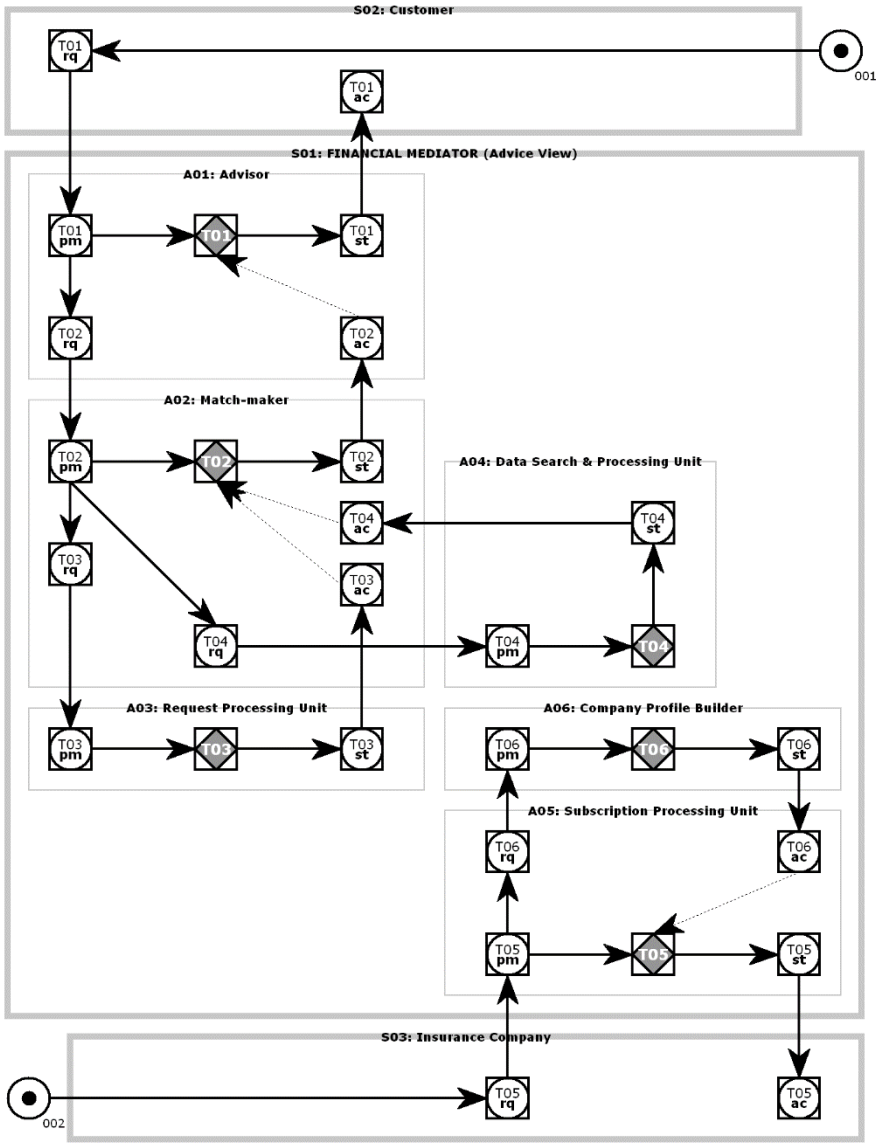

Fig. 6.15. The FM business coMponent (advice ciew) – communicative aspect.

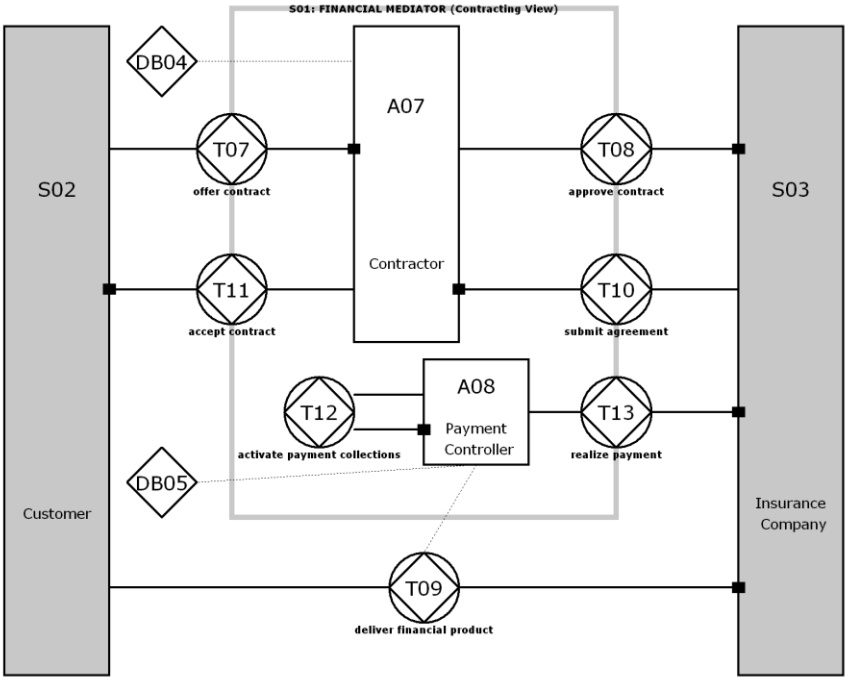

Fig. 6.16. The FM business coMponent (contracting view) – structural aspect.

As seen from the figure, the depicted functionality of **FM** concerns also **customers** and **insurance companies** – **S02** and **S03**. There are internal actors as well: **A07** (**Contractor**) and **A08** (**Payment Controller**).

As seen from the figure, **A07** could offer (**T07**) to **S02** a contract (in doing this, **A07** is facilitated by a data-bank (**DB04**) containing contract templates that concern particular companies). This could be realized only based on an approval (**T08**) of such a contract, from the concerned (insurance company).

It could be seen as well from the figure that **S03** could deliver (**T09**) a (insurance) financial product to **S02**. However, this could be done only based on a submitted (to the company) **customer**-**FM** agreement (**T10**) based on an offer acceptance (**T11**) by **S02**.

It could also be seen from the figure that in some situations, a (insurance) company should realize payment to **FM**. Actually that is about any realized (through **FM**) product delivery. **FM** should be notified about each realized (through it) product delivery. An indication for this is the dotted line between **T09** and **A08**. **A08** has (therefore) the information (it is stored in the data-bank **DB05**) about what each (registered) company

owes to **FM**. **A08** is activated by itself periodically. Then it is to handle the payments accordingly.

We go further (as we already did in the above paragraphs) for norm elaboration. We will not do this for transactions **T12** and **T13** because they are straightforwardly understandable. The derived *norms* are below:

EN7
Whenever S02 has requested contract
If S03 has approved a contract proposed by A07
Then A07
Is obliged to deliver the contract.

EN8
Whenever S02 has requested contract
If A07 has offered a contract not contradicting with the policy of S03
Then S03
Is obliged to approve the contract.

EN9
Whenever S02 has requested a financial product
If A07 has submitted an agreement (about the product) concerning S02
Then S03
Is obliged to deliver the financial product.

EN10
Whenever S02 has requested a financial product
If S02 has accepted a corresponding contract
Then A07
Is obliged to submit to S03 the appropriate agreement.

EN11
Whenever S02 has requested a financial product
If A07 has offered a contract which does not contradict with S02's interests
Then S02
Is obliged to accept the contract.

Based on the model represented in Figure 6.16, we derive a model (Figure 6.17) representing the communicative view on the addressed business reality.

As seen from the figure, three sub-processes are to be considered – one of them relates to the **FM**'s offering a contract to a customer, the second one relates to an insurance company's delivering a financial product, and the third one relates to the **FM**'s payments handling. This is indicated by three starting points (on the figure): **003**, **004**, and **005**.

As also seen from the figure, the dependence of **T07** on the execution of **T08**, the dependence of **T09** on the executions of **T10**, and the dependence of **T10** on the execution of **T11** – all these are reflected accordingly in the model.

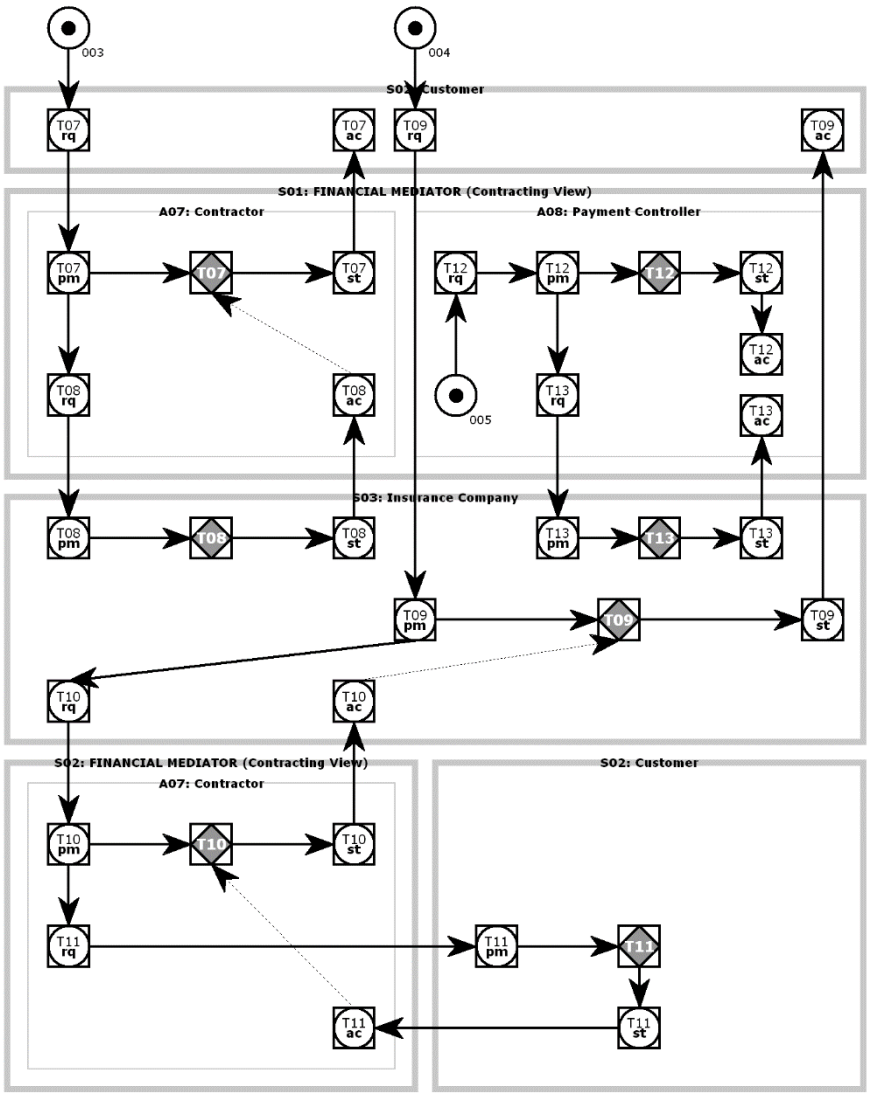

Fig. 6.17. The FM business coMponent (contracting view) – communicative aspect.

Therefore, we have considered so far both 'Advice' and 'Contracting' views over the **FM** *business coMponent* as far as the structural and communicative aspects are concerned.

We continue with consideration of the *dynamic* and *data* aspects.

Financial Mediator – dynamic and data aspects

As for the *dynamic* aspect, it is considered by reflecting the built (so far) models in appropriate dynamic (*work-flow*) ones. We will use *Petri Net* (*PN*) notations (plus *norm* elaboration that concerns the *PN* model).

An introduction to *PN* can be found in Shishkov [54], and also on how to derive a *PN* model in *SDBC*. Thus, we will not explain in detail this derivation.

We will first build a model corresponding to the 'Advice view'.

A basic source for building this dynamic model (which is represented on Figure 6.18) is the (already) constructed communicative one (Figure 6.15).

As seen from Figure 6.18, the two sub-processes, considered within the communicative model, are reflected in the dynamic one ('**Start 1**'/'**Start 2**' from Figure 6.18 correspond to starting points **001**/**002**, respectively, from Figure 6.15). This is logical because such fundamental issues should not change depending on the particular aspect view.

As it is also seen from the figure, the *transactions* (Figure 6.15) are reflected in corresponding *activities*. Those are,

regarding the first sub-process:

- 'FM: Generate standardized (stn.) specification'
- 'FM: Generate candidate-matches'
- 'FM: Perform match-making'
- 'FM: Deliver advice'

and regarding the second sub-process:

- 'FM: Build company profile'
- 'FM: Realize subscription'

The activities '*FM: Generate standardized (stn.) specification*' and '*FM: Generate candidate-matches*' are modeled through the useful '*parallel process*' *PN* mechanism, reflecting the requirement (Figure 6.15) that they both are completed before the activity '*FM: Perform match-making*' could be realized.

We have reflected also (as depicted in Figure 6.18) some particularly important (from the perspective of the *work-flow* of events) communicative acts:

- '*Customer: Request advice*' is a reflection of the 'request' part of the *transaction* **T01**. This is necessary to be considered as an activity within the *PN* model because what actually needs to take place in triggering the flow of events is that a customer requests to receive advice from **FM**.
- '*FM: Process information*' concerns also **T01**; handling an advice delivery should include consideration and processing of the customer information (to be accordingly distributed within **FM**). This has not been modeled as a separate *transaction* because it concerns the 'information' level, not the 'essential' one. However, from the perspective of the flow of events it should be considered.
- '*FM: Realize data search*' actually concerns the execution part of the *transaction* **T04**. Again, because of its concerning the 'information' level, it is not considered as a separate transaction although it has to be considered within the modeled flow of events.

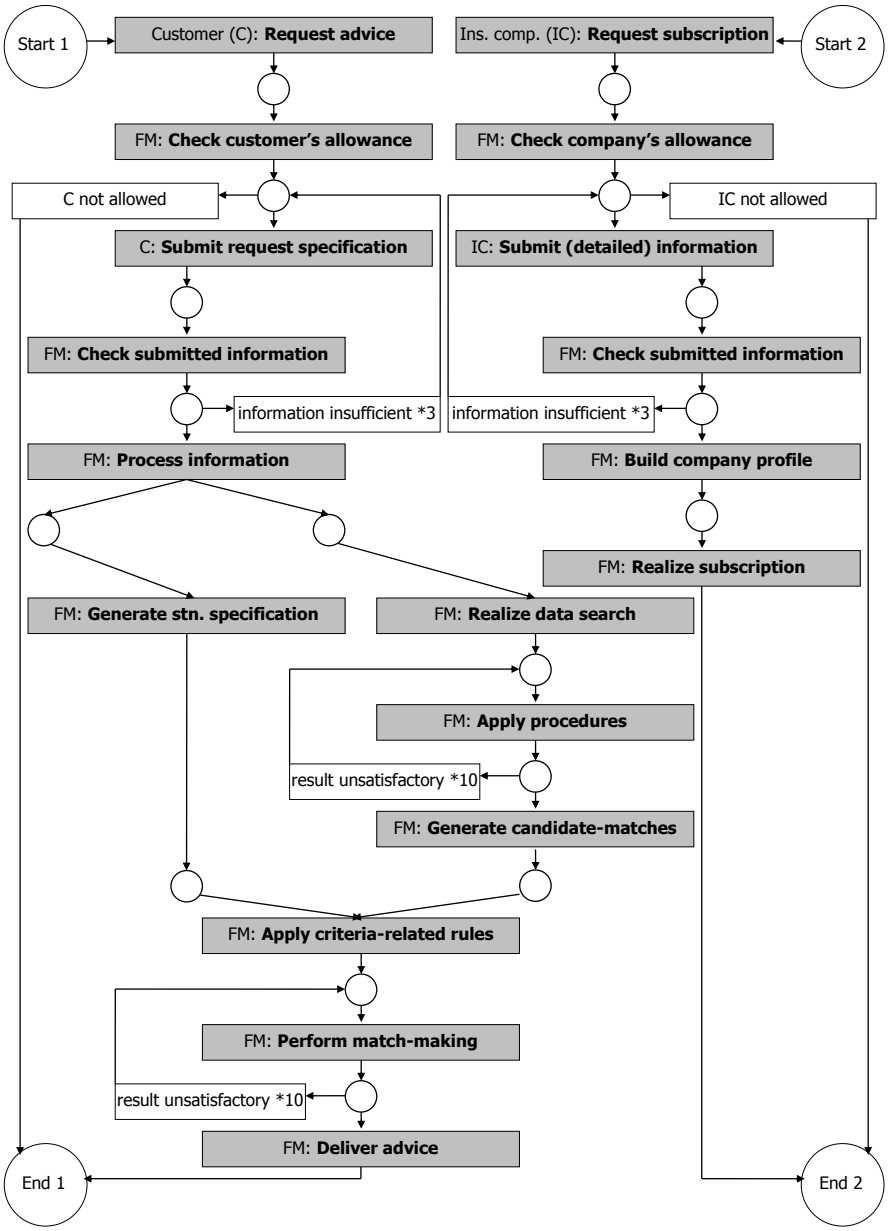

Fig. 6.18. The FM business coMponent (advice view) – dynamic aspect.

- The same applies to '*FM: Apply procedures*'.

- '*FM: Apply criteria-related rules*' concerns in the same way the execution of the *transaction* **T02**.

- '*Ins. comp.: Request subscription*' is a reflection of the 'request' part of the *transaction* **T05**. This is necessary to be considered as an activity within the *PN* model because what actually needs to take place in triggering the flow of events is that a company requests to be subscribed to **FM**.

As for the activities '*FM: check customer's allowance*' and '*FM: Check company's allowance*', (Figure 6.18), they reflect a requirement from the business proposition, according to which: 'A customer's using **FM** (either for advice, or contract, or product delivery) is to be limited to no more than five times per month. As for (insurance) companies' allowance, a company is allowed to subscribe to **FM** only if it is licensed according to the Dutch financial laws'. Those are actually informational (not essential issues) since they concern information checking. For this reason, they are not reflected in the models depicted in Figure 6.14 and 6.15. Since they affect the flow of events, they are to be reflected in the dynamic model: the allowance of customer/company should be checked and if a customer/company happens to be not responding to the mentioned requirements then the customer/company should not be allowed to use the services of **FM**, hence a direct move to the '**end**' point should take place.

As for the activities '*C/IC: Submit request specification / (detailed) information*', and '*FM: Check submitted information*', they concern informationally *transactions* **T01** and **T05**, respectively. Information aspects concerning those *transactions* are not to be reflected at the essential level but have to be considered within the *work-flow* of events. This is because the information providing (by a customer/company) is a key activity from a *work-flow* perspective. This applies also for the check whether the information provided is sufficient (if not, the particular customer/company is to be asked to re-submit the information; the '***3**' means that after **3** unsuccessful entries the user is kicked off – analogous indications are used also in '*result unsatisfactory*' in the same figure).

As for the 'Contracting view', we have derived a model (Figure 6.19) in an analogous way.

As for the *norm* elaboration which is suggested, we have derived several 'information' *norms* (attached to the *PN* models) consistent with the 'essential' *norms* (identified in the previous paragraphs). In their identification we include '**I**' (from 'Informational'), '**N**' (from '*Norm*') and a number. The *norm* (below) is an example of such a *norm*, concerning the activity '*FM: Check submitted information*' (we have assigned this activity a number, namely: number **4**):

IN4
Whenever a customer has requested advice
If (s)he has submitted information to FM
Then FM
Is obliged to check the submitted information.

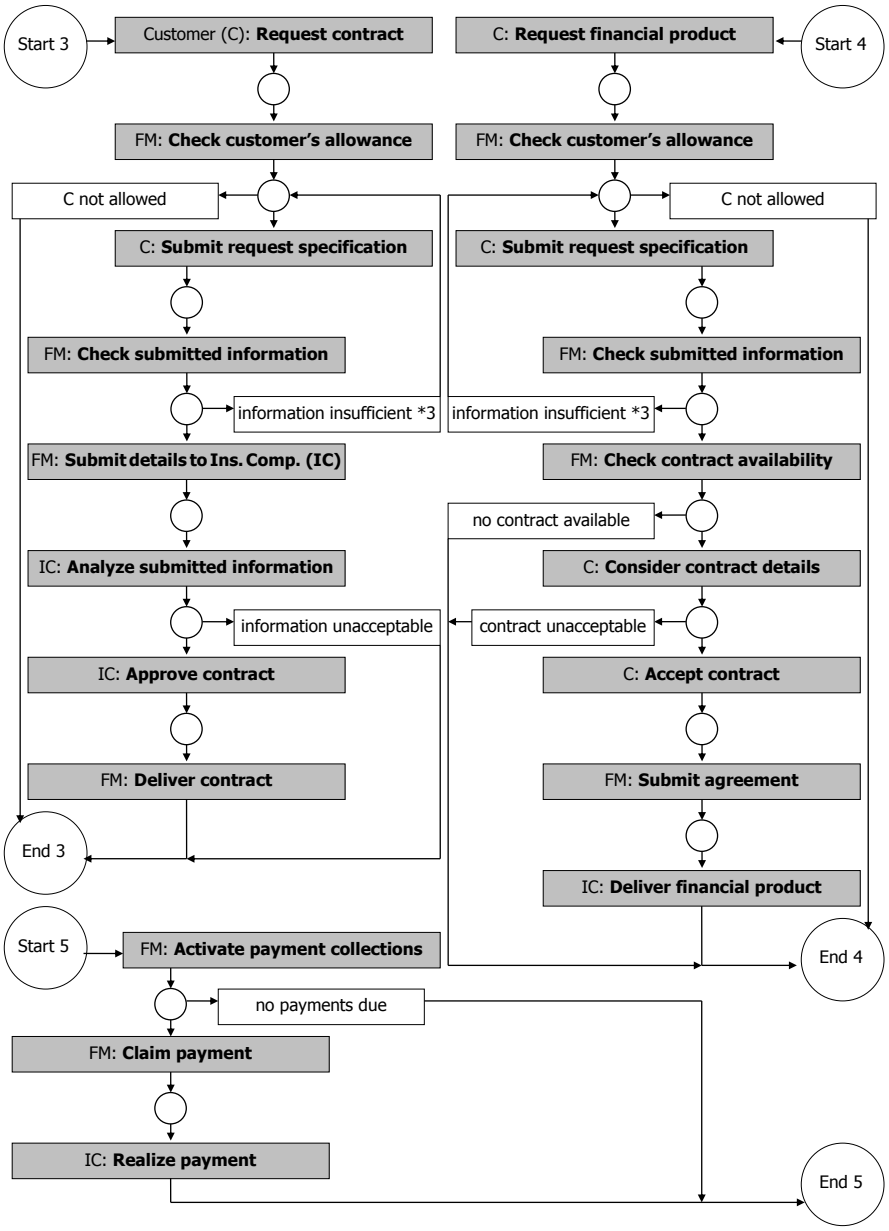

Fig. 6.19. The FM business coMponent (contracting view) – dynamic aspect.

And finally, regarding the *validation* of the derived dynamic models, we could apply *discrete event simulation*, as studied in [54].

As for the data aspect, it is considered by reflecting the models built so far in appropriate *data* ones. We will use the *ORM notations* (see Chapter 5) for that purpose.

We will build a model corresponding to the 'Advice view'. We will not elaborate factually the models which concern the 'Contracting' view mainly because the *data-banks* related to them (**DB05** and **DB05** – Figure 6.16) are to be considered at a later stage, when **FM** would have started to function. For example, information about delivered (through **FM**) financial products (concerning **DB05**), based on which (eventual) payments would be activated, would have appeared after **FM** has started its operation.

Regarding the 'Advise view', we turn to the fundamental link between the models (built so far) and the *data* aspect – those are the *data-banks* **DB01**, **DB02**, and **DB03** (Figure 6.14). We make reflection towards data models, by further modeling those *data-banks*.

Hence, our 'Advise view' (*ORM*-driven) *data* model of the **FM** *business coMponent* should include elaborations of the *data-banks* **DB01**, **DB02**, and **DB03**.

Before proceeding to such an elaboration, we need to add some *data* input to the business proposal information, which is as follows:

- We have selected for consideration the following seven (insurance) financial companies situated in The Netherlands: *Icomp* (situated in a Dutch city, offering products as follows: **1011001** (those codes will be explained further on)); *OHRA* (situated in Arnhem, offering products as follows: **0001010**); *AEGON* (situated in Den Haag, offering products as follows: **1110111**); *Nationale-Nederlanden* (situated in Rotterdam, offering products as follows: **1001110**); *Euro Lloyd Verzekeringen* (situated in Amsterdam, offering products as follows: **0100100**); *Unive Verzekeringen* (situated in Zwolle, offering products as follows: **1111110**); *AXA* (situated in Utrecht, offering products as follows: **1101001**). Details about those companies have been summarized at:
http://www.sdbc.tk/icomp/detailsicomp.htm
http://www.sdbc.tk/ohra/detailsohra.htm
http://www.sdbc.tk/aegon/detailsaegon.htm
http://www.sdbc.tk/nn/detailsnn.htm
http://www.sdbc.tk/ev/detailsev.htm
http://www.sdbc.tk/uv/detailsuv.htm
http://www.sdbc.tk/axa/detailsaxa.htm

As for the possible *customer needs* (to be addressed by **FM**), they might be: '**auto-insurance**', '**health-insurance**', '**life insurance**', and so on. *Procedures* (to be considered concerning them) and their URLs are as follows:

auto-insurance	Procedure 1	http://www.sdbc.tk/pr/pr1.htm
health-insurance	Procedure 2	http://www.sdbc.tk/pr/pr2.htm
life-insurance	Procedure 3	http://www.sdbc.tk/pr/pr3.htm

...

As for the *criteria* consideration (facilitated by *procedures*), which has already been mentioned, the following *procedures* are to be used, corresponding to the four criteria considered (**pay-back**, **reliability**, **quality of service**, **insurance costs**):

Pay-back	Procedure PB
Reliability	Procedure RB
Quality of Service	Procedure QS
Insurance Costs	Procedure IC.

Regarding the *product codes* (used above already), we would like to make the following elaboration: We have considered seven types of (insurance) financial products, namely **Life-insurance-related products**, **Property-insurance-related products**, **Mortgage-related products**, **Pension-related products**, **Travel-insurance-related products**, **Personal-damage-insurance-related products**, and **Lawyer-assistance-insurance-related products**. We have given the following numbers to those types of products:

Life ins.:	*1*
Pr. Ins.:	*2*
Mortg. :	*3*
Pens. :	*4*
Trvl. :	*5*
PersDmg. :	*6*
LwrAsstnc. :	*7*

Then we introduce a string of seven binary digits. Each position there corresponds to the number of a particular type of product.

Thus, the code **0000100**, for instance, should let us know that the particular company (to which this code is attached) offers only *travel insurances* and related (financial) products.

We have presented all this information in Figure 6.20 concerning the *data* aspect of the **FM** 'Advice view'.

The top model on the figure concerns the *data-bank* **DB01** (Figure 6.14); the bottom model concerns **DB03**. The model between them concerns **DB02**.

As seen from the figure, we have consistently conducted *data* elaboration on the model represented in Figure 6.14, considering adequately the factual case information.

Therefore, we have considered so far both 'Advice' and 'Contracting' views. Regarding the 'Advice model', we have elaborated it in structural, communicative, dynamic, and data aspects. Regarding the 'Contracting model', we have elaborated it in structural, communicative, and dynamic aspects.

So, we have demonstrated *business coMponents*' elaboration. In the following sub-section, we will address the reflection of a *business coMponent* in the *specification* of software.

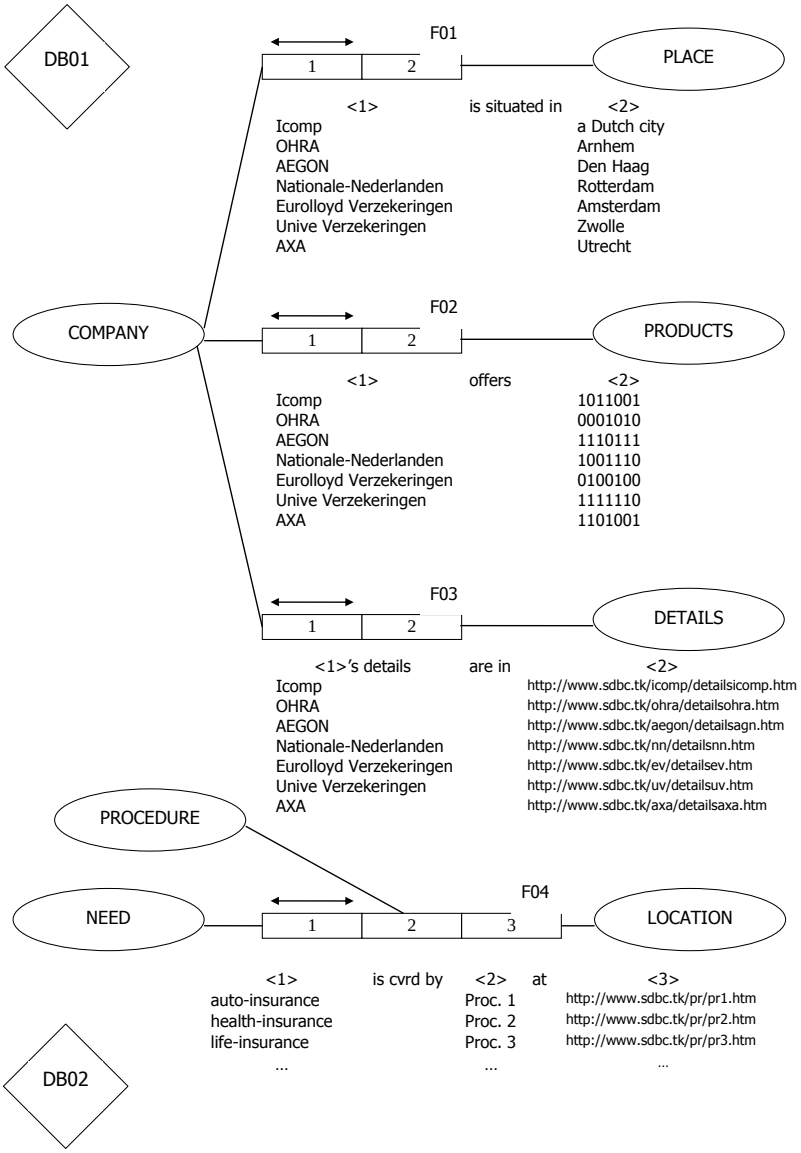

Fig. 6.20. The FM business coMponent (advice view) – data aspect.

6.3.3 Towards Software Specification

In the previous sub-section, we have demonstrated the *SDBC*-driven *elaboration* of a *business coMponent*. As mentioned at the beginning of the current section: in this sub-section, we will demonstrate how a UML-driven [74] *software specification model* could be derived on the basis of a *business coMponent* (in this particular case, on the basis of the *coMponent* considered in the previous sub-section). As already mentioned, this model should reflect the *business coMponent*. However, it is necessary also that the model considers the *user-defined requirements* towards the software system-to be. Said otherwise, this model must have two inputs.

1. A *business process modeling* input coming through a *business coMponent(s)*;
2. A *requirements* input coming through the specification of what the (future) users of the software system-to-be require as an automation.

We need, therefore, to add a *requirements specification* to the business proposition done in the previous sub-section:

According to the user requirements: The **FM** must be automated completely, representing an **ICT application** which must be accessible via the *Internet*. The application should have mechanisms for checking the data accuracy, before performing match-making. And also, the application should be facilitated by a database (containing all the information from data-banks **DB01**, **DB02**, **DB03**, **DB04**, and **DB05**), located on a server in The Netherlands.

Therefore, in going through the further (*software specification*) steps, we will consider both the input *business coMponent* (Sub-section 6.3.2) and the user-defined requirements (the above paragraph).

Use case derivation

Use cases are modeling constructs that serve to link the *application domain* (the business world) to the *software domain*, regarding any *UML*-driven *software specification* [56]. Hence, the first step in reflecting the **FM** *business coMponent* into a (*UML*-driven) *software specification model* must be a *use case derivation*.

According to the SDBC application guidelines [54], a use case derivation is to go through three phases, namely:

- derivation of **essential use cases**;
- derivation of **informational use cases**;
- derivation of **UDR use cases** (‘*UDR*’ stands for ‘*User-Defined Requirements*’).

ESSENTIAL USE CASES are pieces of functionality, reflecting actions from a considered *enterprise system*, which are ‘essential’, as according to the *enterprise ontology terminology* [19].

INFORMATIONAL USE CASES are pieces of functionality, reflecting actions from a considered *enterprise system*, which are *informational*.

UDR USE CASES are pieces of functionality added on the basis of a consideration of the *user-defined requirements* towards the software *system*-to-be.

The *SDBC use case derivation* concerning those three types of *use cases*, is depicted on Figure 6.21 and will be followed further on.

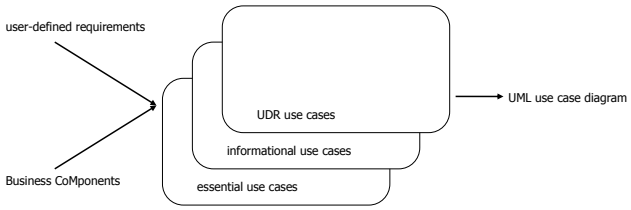

Fig. 6.21. The SDBC use case derivation procedure.

Deriving Essential use cases
As according to [54], we derive the *essential use cases* (Figure 6.22), by mapping them straightforwardly from corresponding *transactions* (Figures 6.14 and 6.16). As for the *UML use case diagram*, the *actors* there reflect straightforwardly the external role-types (Figure 6.14, Figure 6.16). The reason is that we are to automate **FM** completely. Therefore the **FM** perspective is to coincide with the perspective of the software *system*-to-be.

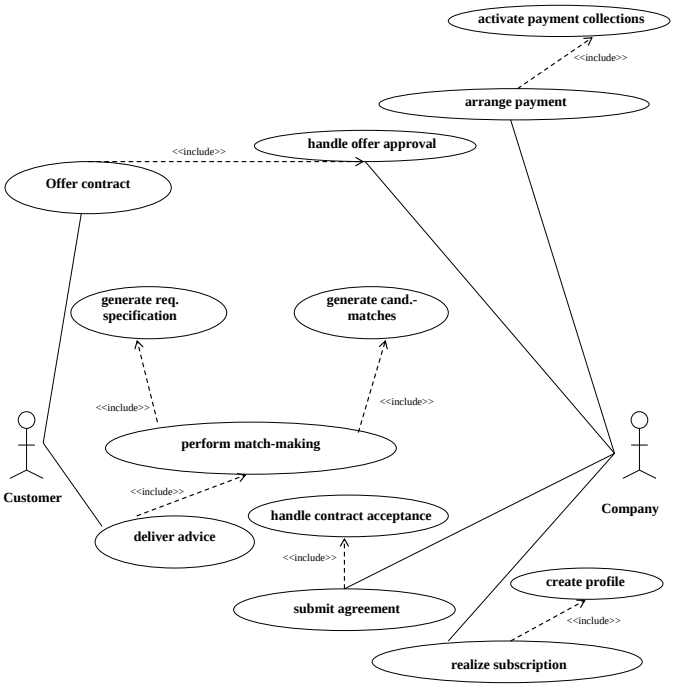

Fig. 6.22. FM: Use case model (identification of the Essential use cases).

Deriving Informational and UDR use cases

Based on the identification of the *essential use cases* and having as a source the dynamic (*PN*) models (Sub-section 6.3.2), where we have reflected the *informational* issues related to the **FM** *business coMponent*, we identify the following *use cases*:

- *'check allowance'*
- *'check submitted information'*
- *'process information'*
- *'apply search'*
- *'apply procedures'*
- *'apply rules'*
- *'submit information'*
- *'check contract availability'*,

All of which reflecting straightforwardly corresponding *PN processes*/*transitions* (Figures 6.18 and 6.19). Next to that, we have added the use case:

- *'request additional information'*,

as an *extension* to the *use case* '*check submitted information*', since in some situations (when the submitted information is insufficient), it might be necessary that additional information is submitted.

We have also identified the following two *UDR use cases* reflecting the (above specified) *user-defined requirements*:

- *'check data accuracy'*
- *'add data in database'*.

Thus, the complete *UML use case model* is depicted in Figure 6.23 where, as seen, the *Informational use cases* are backgrounded in *grey* and the *UDR use cases* are backgrounded in *black*.

Regarding the *use case diagram*: There are two *actors*: **Customer** and (Insurance) **Company**. Concerning Customer (Company) he(it) takes the decision, has the responsibility, has the goal to have an advice/contract/product(fin.) delivered (its information correctly added to the **FM** database and have a subscription facilitating in this way the distribution of its (financial) products). The diagram contains *23 use cases*: '**deliver advice**', '**add data in database**', and so on. There are 15 **<<include>>** relationships (one of them concerns the use cases '**deliver advice**' and '**perform match-making**', indicating that the **FM**'s delivering an advice to **Customer** requires performing a match-making (based on which the advice would be specified)) as well as one **<<extends>>** relationship (in some cases, as mentioned above, if submitted information is insufficient, before continuing its operation further, **FM** would need the submission of some additional information, so the basic use case is '**check submitted information**', and it is extended with '**request additional information**').

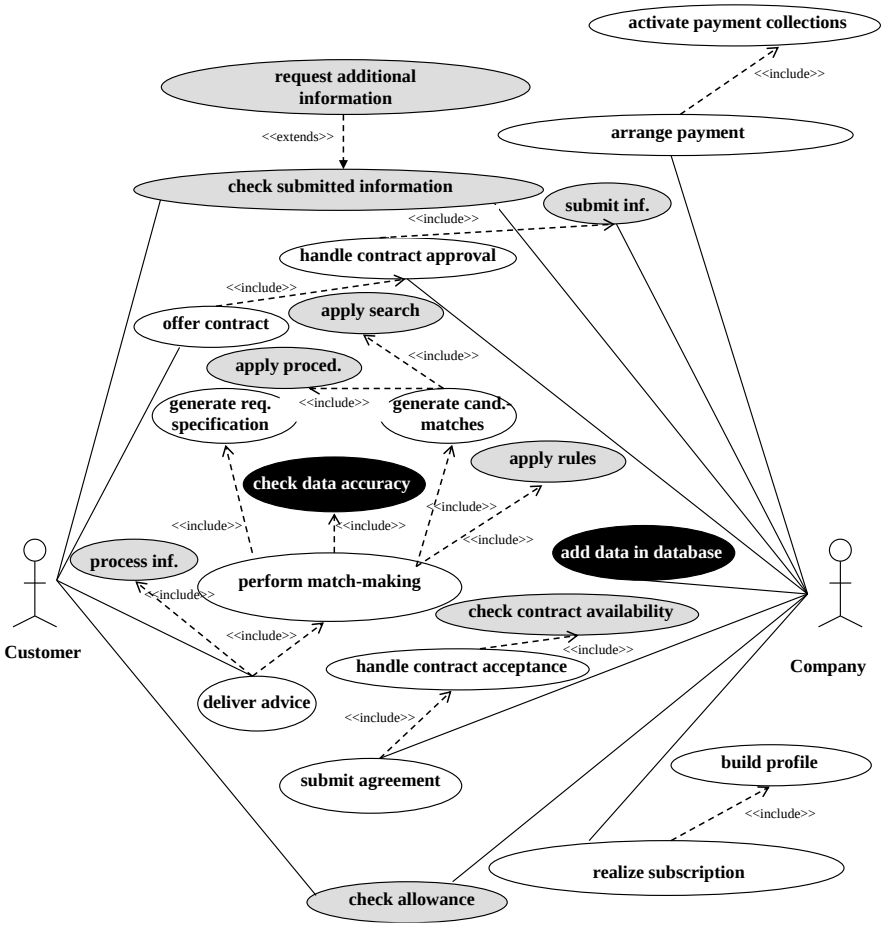

Fig. 6.23. FM: Thorough use case model.

Elaboration

Based on the built *UML use case model*, it is possible to make any further elaboration concerning either particular *use cases* (specifying them in more detail) or the *model* as a whole.

We will proceed with demonstrating how any particular *use case* of interest could be adequately specified. We follow a *use case* specification mechanism inspired by [14, 58]. Below we will just demonstrate the specification of a *use case* from the *model* already built (Figure 6.23).

We have selected, for illustrative purpose, the *use case* 'add data in database' and the mentioned investigation is applied to it – Figure 6.24 (only those extensions related to activity six are depicted).

The *use case* is written at 'system' scope (as opposed to 'enterprise' scope) since it describes an interaction with a computer system. The indicated 'summary' level means that the *use case* is long running (executed over months or years), showing the context in which the user goals operate.

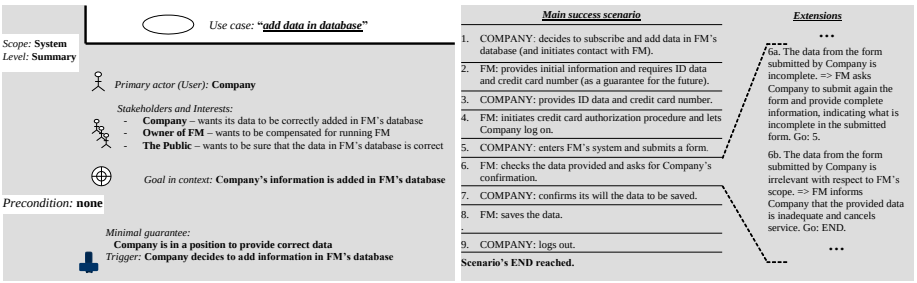

Fig. 6.24. Specification of the use case: 'add data in database'.

For <u>further (dynamic) elaboration</u> (and visualization) of the considered *use case* ('add data in database'), a *UML activity diagram* [74] could be straightforwardly derived based on the *main success scenario* + *extensions* (Figure 6.24). As seen from this figure, there are nine core activities (complemented with extensions), in the mentioned *use case*. Some of them are shown on Figure 6.25, as an overall *UML activity diagram*.

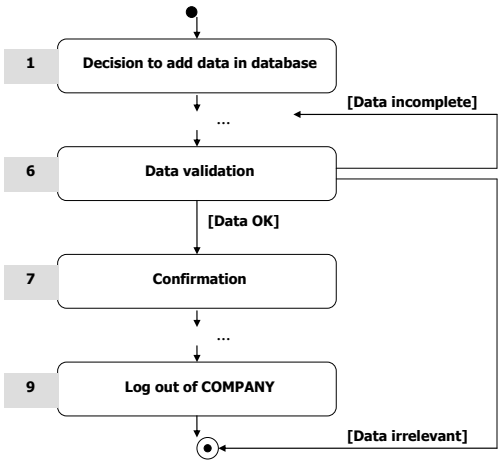

Fig. 6.25. UML activity diagram for the use case: 'add data in database'.

As studied in [54], based on such a dynamic model, it is straightforward to proceed to computer simulation. We will not demonstrate this in the current chapter.

As mentioned above, one might need to elaborate either particular *use cases*, specifying them in more detail, as demonstrated above, or the *model* (Figure 6.23) as a whole. In elaborating the *model* as a whole, one could take either *structural* or *dynamic* perspective.

The PN *business process models* (see Figure 6.18 and Figure 6.19) can be used as a basis for deriving a *dynamic elaboration* of the overall *use case model* (Figure 6.23). However, in realizing this, one should add accordingly information connected to the *user-defined requirements* because this information is certainly missing in the *business process models*.

As for the *structural elaboration* of the overall *use case model*, it could be conducted by reflecting both the *structural business process models* (Figures 6.14 and 6.16) and the overall *use case model* (Figure 6.23) into *UML class diagram(s)*[74]. We will show below only a partial *UML class diagram* concerning just the *use cases* 'realize subscription' and 'build profile'; we reflect also the two profile types, as from the initial case information). The *UML class diagram* is depicted in Figure 6.26:

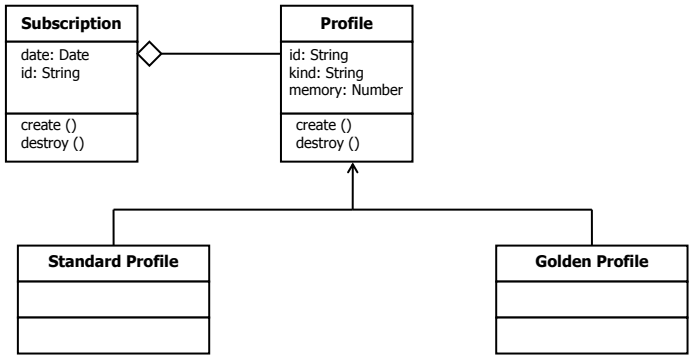

Fig. 6.26. FM: Partial UML class diagram.

In summary, so far in the current chapter, we have demonstrated, by means of the Icomp case, how starting from a case briefing and coming through enterprise models, software specification could be derived.

6.4 Enabling Service-Orientation

The *service-oriented architecture* and its strengths have been discussed in Chapter 4. In the current section, we will demonstrate how the *SDBC modeling* output can be used as a basis for deriving *service-oriented* specifications, such that the resulting software is capable of being delivered to users by means of technology-enabled services. In order to accommodate *service-orientation*, we would need partial re-factoring of some of the

models, presented in the previous sections of the current chapter. Further, we are offering only a partial illustration of the above because the goal is to just demonstrate how *SDBC models* could accommodate *service-orientation*.

We firstly take a partial view and do a slight simplification with regard to the *model* presented in Figure 6.14: We represent the *Customer*, *Advisor, Match-maker*, *Request Processing Unit* (we call it 'Request Handler', for short) and *Data Search and Processing Unit* (we call it 'Data Searcher', for short), as just entities and put them in named boxes, as follows:

- Customer (**C**);
- Advisor (**A**);
- Match-maker (**MM**);
- Request handler (**R**);
- Data searcher (**D**).

Further, we consider *transactions* as just interactions that we represent as connections between entities while the small *grey boxes*, one at the end of each connection, indicate the *executor role* (as according to *LAP* and *enterprise ontology* – see Chapter 3) of the connected entities, similarly to the model represented in Figure 6.14. The connections indicate the need for *interactions* between *entities*, in order to achieve the business objective of financial mediation; with each connection, we associate a single *interaction* (**i**):

- C-A (**i1**);
- A-MM (**i2**);
- MM-R (**i3**);
- MM-D (**i4**).

Further, **C** is positioned in the *environment* of the financial mediation system – **FM**, and **A**, **MM**, **R** and **D** together form the **FM** system. Through **i1**, **FM** is related to its *environment* (represented by **C**). Thus, from the perspective of **C**, there is no difference between **FM** and **A**.

This all is depicted as a *business entity model* in Figure 6.27-a.

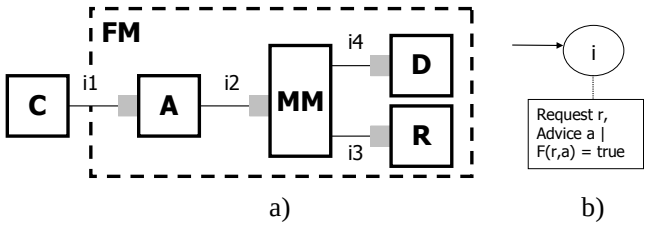

Fig. 6.27. a) FM: business entity model; b) FM service behavior represented by a single action.

What we have illustrated in Figure 6.27-b is the external behavior of **FM**, at a high level of abstraction, and then we move to the abstraction level which concerns the internal behavior of **FM**. With respect to the *external behavior model*, as already mentioned, it should envision the *interaction* between the customer (**C**) and the system (**FM**), and is represented by a single action (expressed by an *oval*) in Figure 3-b. The depicted action has also attributes (put in a *box*) elaborating the result of the *action*.

This single action **i** corresponds to the business objective of the **FM** system: to serve the request (**r**) of a customer, by giving advice (**a**) that satisfies certain criteria (**F**(**r**,**a**) = **true**).

Regarding the *internal behavior model*, it should reflect the *interactions* between the *entities* of the *system*, as exhibited in Figure 6.28. This *model* shows how the *interaction* **i1** between the Customer **C** and the Advisor **A** is made dependent on other interactions (**i2**, **i3** and **i4**) in the system. Each interaction between two entities (e.g., **C** and **A**) represents a **request** (e.g., from **C** to **A**, of type **RequestC-A**) and advice (e.g., from **A** to **C**, of type **AdviceA-C**), where the **advice** satisfies certain criteria (e.g., as expressed by the truth value of function **FA**).

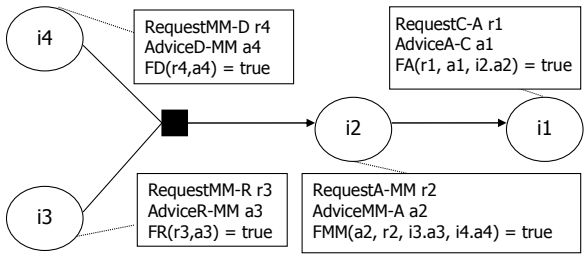

Fig. 6.28. Interactions in decomposed FM system, implementing the FM service behavior.

Assuming that the *models* of Figure 6.27-b and Figure 6.28 represent the same *request* from the *customer* (**r** = **r1**) and the same advice to the customer (**a** = **a1**), it follows that **F**(**r**,**a**) = **true iff** (**FA**(**r1**, **a1**, **i2**.**a2**) = **true** and **FMM** (**a2**, **r2**, **i3**.**a3**, **i4**.**a4**) = **true** and **FR** (**r3**,**a3**) = **true** and **FD** (**r4**,**a4**) = **true**).

We now need to further elaborate this *model*, in order to achieve a better link to relevant real-life enterprise aspects, and we do this, by considering the *transaction* concept – as discussed already in previous chapters, this would allow the *modeling* of failure-scenarios (not only success-scenarios). Further, we acknowledge the essential role of real-life communication and coordination in an *enterprise system*. Hence, we apply the *transaction pattern*, expressing it using a notation well-suited for *SOA*, namely – *ISDL* [64].

Figure 6.29 exhibits the *generic process of an interaction* reflected through the **transaction pattern** and *modeled* at **two different abstraction levels**. At the highest level, the *interaction* is represented by a *single action* which models the *production fact* that is established. Characteristics of the *production fact* are modeled

using the *information attribute*. At a lower abstraction level, the *interaction*'s *communication aspects* are modeled conforming to the *transaction pattern* (see Figure 5.4). *Separate actions* are used to model the interaction's *request*, *promise*, *state*, *accept* and *decline*, and the *production act*. It should be noted that actions $\mathbf{Id_{Ex}}$ and $\mathbf{Id_{In}}$ correspond to the decline of an interaction followed by a unsuccessful negotiation; and actions $\mathbf{Ip_{Ex}}$ and $\mathbf{Ia_{In}}$ represent the promise and acceptance, respectively, which are followed by a successful negotiation.

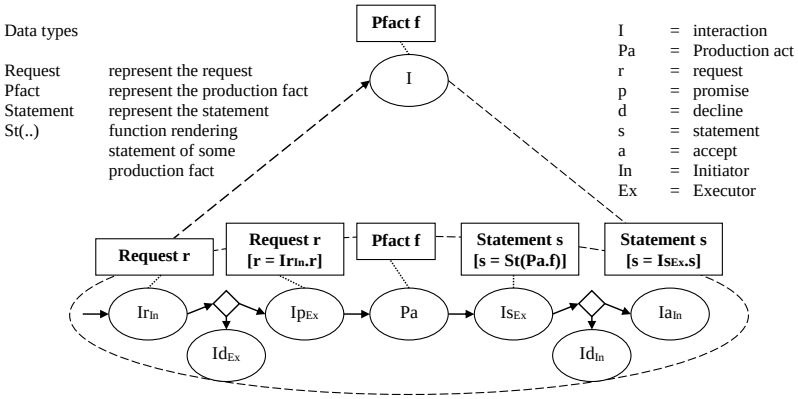

Fig. 6.29. IDSL interpretation of the transaction pattern.

Hence, if we would need to go to a still lower abstraction level, compared to the one in the behavior model (Figure 6.28), we may go for 'zooming in' with regard to each of the four interactions, represented in the model, such that we arrive at a detailed behavior aspect model of the FM, as shown in Figure 6.30:

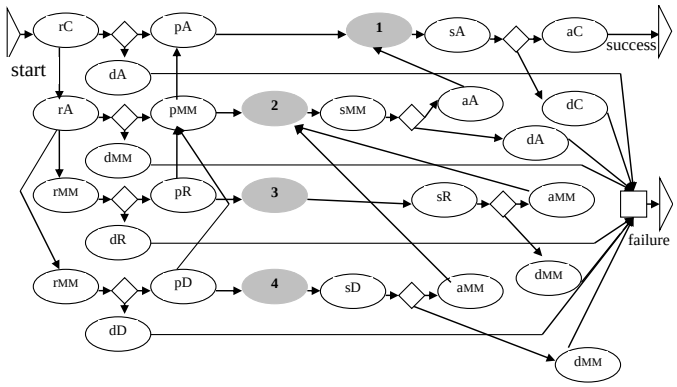

Fig. 6.30. Detailed behavior aspect model of the FM.

It should be noted that the number labels of *production acts* (grey ovals in the figure) correspond to the interactions **i1** – **i4** (Figure 6.28). Further, following one instance of the behavior, we have *two possible outcomes*, namely successful and failure outcomes.

Further, based on the detailed *behavior model* and through simplification, we arrive at a **service-oriented model** (Figure 6.31): we *group* together *coordination acts* based on their relations to *production acts*. Furthermore, we straightforwardly reflect (from the detailed *behavior model*) the information on how those groups relate to each other; we use an alternative way to model the *decline acts*: a *decline-after-request act* and a *decline-after-state act* are represented by a special value of an *information attribute* (e.g., *Result* **r** | **r** = '**decline**') of the *promise* and *accept* acts, respectively. Information attributes of the act and constraints on the values of these attributes are not represented in the figure. The *model*, presented in this way, defines services rooted in the transaction pattern, consistently with the achieved *modeling* output.

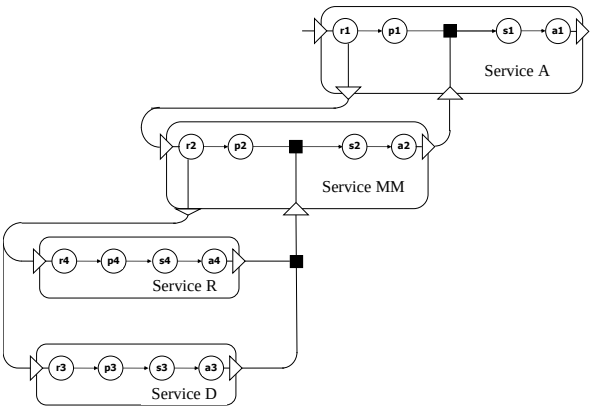

Fig. 6.31. Refined interactions in decomposed FM, implementing the FM service behavior.

Thus, the *business entities* represented in Figure 6.27-a point to the (*application*) *components* underlying the *services* represented in Figure 6.31. This assumes the easiest decision: to do a **one-to-one** mapping between the *business entities* (Figure 6.27-a) and the *application components* (it was implicit that the *business entitiy model* was straightforwardly mapped to an *application components* model where the *application components* correspond to the *business entities*, and this is how we have reached in the end the *service model* represented in Figure 6.31). Nevertheless, such a one-to-one mapping between the two has the disadvantage that identified *services* are tightly coupled. This means that there is a dependency of the *service* provided by one *component* on *services* provided by other *components* (as seen from Figure 6.31). We argue that a solution would be to introduce an additional *application component*, called **orchestrator**, that has the task of coordination – inspired by *service orchestration* (Figure 4.4).

The *orchestrator* is an application-specific *component*, as the coordination is application-specific. The (subordinate) *services*, however, which are coordinated by the *orchestrator*, may be useful for many different types of applications. Their description

may therefore be published through a public or corporate registry, such that they can be discovered, and selected for invokation by an *orchestration component*. Related to its coordination tasks, the *orchestrator* could sometimes supply to one service the result of another service, if this is necessary for the service to perform its task.

Figure 6.32-a depicts the *orchestrator's* (**O**) desired role. It concerns the inter-activities between the 'original' *components* (reflecting corresponding *business entities*) as well as *coordination*. The *orchestrator* mediates not only the interaction between the *customer* (**C**) and the *system* but also all interactions between *components* inside the *system*.

For this reason, in order to enable *orchestration*, we need to firstly *refine* the *business entity model* (Figure 6.27-a), by reflecting there the *orchestration entity* (colored grey in Figure 6.32-b) that mediates the interactions among *entities*.

Then in a similar way (see above) we can reflect this in a *behavior model* and in the end – in a *service model*.

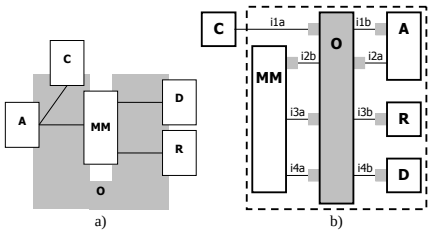

Fig. 6.32. a) Desired role of the orchestrator; b) The refined entity model.

Thus, we have conducted *business re-design* in order to facilitate the accommodation of *service-orientation* in the context of *SDBC modeling*.

6.5 Other Examples

Through the *Icomp* case, we have demonstrated how *enterprise engineering* and *software engineering* could be brought together, inspired by *SDBC*, such that *enterprise modeling* and *application modeling* are adequately carried out (and alignment between the two is supported) as well as the (possible) move towards *service-orientation*. In this, we have considered *requirements* but not so explicitly and we have assumed a *top-down* approach – starting from high-level business information and moving to lower level software specifications. Hence, further elaboration is needed with regard to the above and we provide it by considering two illustrative examples, namely: (i) The eVoting example where we explicitly consider *requirements*; (ii) The Border Security example where we take a *middle-out* (rather than *top-down*) approach. Those two examples will be briefly considered in the following sub-sections.

6.5.1 The eVoting Example [60]

We consider identifying actor-roles (**AR**) and corresponding relations (**R**), in the context of a typical Voting scenario,

FIRSTLY – **AR**s:

AR1 – **CAMPAIGNER**: the one(s) campaigning in favor of a particular policy / party / vision and influencing the people in that way;

AR2 – **VOTER**: the one(s) voting for parliament / president / … and thus executing basic rights in the country;

AR3 – **PRIMARY COUNTER**: the one(s) counting the votes in a particular voting station;

AR4 – **SECONDARY COUNTER**: the one(s) aggregating the final result, by putting together the voting results from the voting stations;

AR5 – **ORGANIZER**: the one(s) organizing the voting process and supporting all above-mentioned accordingly;

AR6 – **CONTROLLER**: the one(s) controlling all above-mentioned;

7 – **SYSTEM**: even though this is not an actor-role, we have to somehow model abstractly the "place holder" where all voting "goes".

SECONDLY - **R**s:

AR1-AR2 suggesting that the CAMPAIGNER is promoting political messages that are supposed to influence the VOTER;

AR2-SYSTEM suggesting that the VOTER provides essential input to the SYSTEM, namely the vote;

SYSTEM-AR3 suggesting that the SYSTEM has impact with regard to each voting station (said otherwise, each voting station has its "own" SYSTEM), by providing the information needed by the PRIMARY COUNTER for calculating the station results;

AR3-AR4 suggesting that the SECONDARY COUNTER needs the PRIMARY CONTER's feedback from each voting station, in order to aggregate the overall voting results;

AR5-ALL suggesting the ORGANIZER of the elections has relationship with all above-mentioned ARs and the SYSTEM as follows: creating conditions for the CAMPAIGNER to do promotion adequately; establishing that the rights of the VOTER are guaranteed; establishing rules and mechanisms according to which the PRIMARY COUNTER and the SECONDARY COUNTER should fulfil their corresponding tasks; establishing and running the voting SYSTEM;

A6-ALL suggesting that the CONTROLLER should execute effective control concerning all above-mentioned ARs and the SYSTEM, as guarantee that the voting is fair.

This is the basis for our conceptual requirements-driven model; further, we abstract from several issues, such that we do not consider an AR pointing to the one(s) (outside the CAMPAIGNER) who may be somehow influencing the decision of the VOTER – this could have been modeled as an AR by itself but we have not done this because of the lack of technical relevance.

We present our conceptual model on Figure 6.33 and we use simple and intuitive graphical notations: the labels of the ARs are put inside boxes and the SYSTEM is presented as oval, while the Rs are represented as lines (the arrows indicate who is

ADDRESSED in the relationship – for example: if the CAMPAIGNER is influencing the VOTER, then the arrow should be at the VOTER end because the VOTER is addressed by this).

As seen on the figure, we have not only drawn arrows at each line (lines representing Rs) but we have also added labels there: the CAMPAIGNER would influence the VOTER, the ORGANIZER would enable the SYSTEM, and so on.

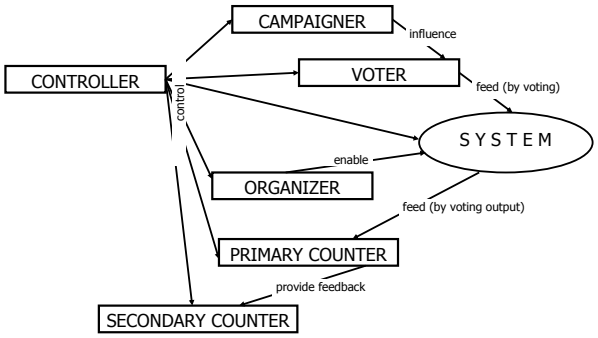

Fig. 6.33. The Voting conceptual model.

Further, we refer to particular **public demands** with regard to a possible introduction of eVoting, and in this case, the demands are:

- *secrecy of vote*, possibly achieved through anonymous credentials, such that not even the system "knows" how a person has voted;
- *cost adequacy*, possibly achieved through smart decisions rather than posh hardware that would generate future "dependencies";
- guarantee against *violations* with regard to the way the system works;
- guarantee against *manipulations* of the final voting results;
- support of *secure communication* between the computers and the servers that is to be possibly cryptography-enriched.
- *controllability* - any third parties should be able to "verify" that the system is working properly;
- guarantee that each vote has been counted and that the person who had voted would *not be allowed to vote again*;
- *fault-reaction* is to be established as guarantee that even if the system (partially) crashes, it would recover and this would not affect its storage and processing functions;
- *ease of use* even by persons who are not of high computer literacy;

no need for extra *qualification* of the election authorities.

We then **elaborate those public demands**:

With regard to the SECRECY OF VOTE demand, there are two things: (i) it is to be guaranteed that **nobody can know how a person has voted**; (ii) it is to be ensured that the person has been marked as "voted", such that (s)he **would not go to vote again**.

With regard to the COST ADEQUACY demand, the only way of avoiding the "big expensive black box" is to conceptualize the eVoting process such that **it is known what technology is needed for what**.

A way to guarantee against VIOLATIONS with regard to the way the System is working, is to present the user with **a simple and exhaustive list of options**, with no possibilities to do anything outside the presented options.

A way to guarantee agains MANIPULATIONS OF THE FINAL RESULTS is to keep things at two levels, such that the **Primary Counters generate the "raw" results** based on which the **Secondary Counters generate the final results** and this all stays stored with **possibility to check in the future**.

The COMPUTER-SERVER communication is to be such that there is guarantee that **a "packet" sent by a computer is received by the server and by noone else**; this is a matter of organization and also a matter of networking protocols.

CONTROLLABILITY can be partially achieved if **all intermediary results get transparent** and then the only remaining challenge is how are the "raw" results generated.

FAULT REACTION is a matter of **recoverability** and this is a non-fuctional concern that has to be addressed from a functional perspective nevertheless.

EASE OF USE is a matter of design.

The issue on QUALIFICATIONS needed for being involved in eVoting is a matter of legislation; as it was mentioned before, sufficient IT literacy among the population is assumed.

We then derive (straightforwardly) *semiotic norms* corresponding to the elaborated demands. We take just for the sake of illustrating this, one eVoting public demand and we reflect it in a specified requirement expressed as a *norm*. We take the SECRECY OF VOTE elaborated demand and we derive the *norm* accordingly:

OS Norm 1:

`Whenever`	John has voting rights
`if`	John is executing eVoting
`then`	the eVoting system
`is`	(i) obliged `to` mark John as "voted"
`is`	(ii) prohibited `from` recording the way John has voted.

Based on *OS Norm 1*, we derive a *workflow pattern* expressed with the notations of *UML activity diagram* [74] – see Figure 6.34:

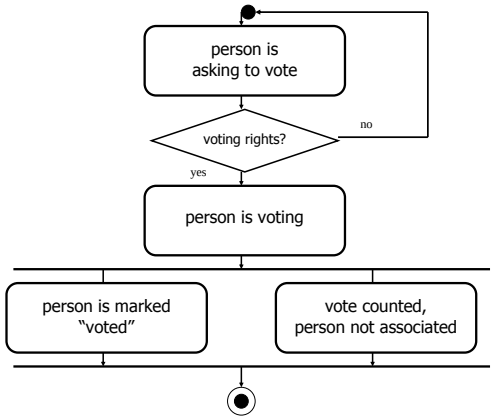

Fig. 6.34. Workflow pattern corresponding to OS Norm 1.

That is how we would methodologically derive *user-defined requirements*, in support of *SDBC modeling*.

6.5.2 The Border Security Example [59]

The Border Security domain is characterized by MANY possible-to-occur situations about the monitoring of illegal migration, combatting related crime, and so on, and there is a need for *context-awareness* and better *interoperability* with regard to the existing (national) border security platforms and *systems*. At the same time, we realize that it is not straightforward applying *context-aware solutions* in the Border Security domain. Hence, research is needed on *Context-Aware Border Security* (*CABS*) control since it would be difficult for a country to supply persons and equipment at every potentially risky border point. A *CABS* system would hence guarantee adaptability with regard to the situation at hand – persons and equipment would only be supplied at the spot where they are needed and in the moment when they are needed. In principle, the *modeling* of *systems*, such as a *CABS system*, should not be expected to differ a lot from the way of *modeling* any other *system*, using *SDBC* as long as *context-awareness* has adequately been addressed (see Chapter 2). Still, the Border Security domain assumes greater complexity because of numerous possible situations and prediction difficulties. Further, what is observed at the border is a "mixture" of personnel and devices, subject to numerous rules and functionalities, and it is not trivial approaching this in terms of technology-independent models, automation, and so on. This is because some (intuitive) tasks can only be realized by humans while other (surveillance) tasks can only be realized by devices, to give just an example. Hence, we need to "adapt" *SDBC* to the peculiarities of the Border Security domain. *SDBC* goes "top-down", from a "bird-view" enterprise model through delimitation with regard to the software system –to-be, to implementation. Nevertheless, for specifying a *CABS* system, we propose to

go "middle-out", as exhibited on Figure 6.35, and we adapt the application of *SDBC* accordingly.

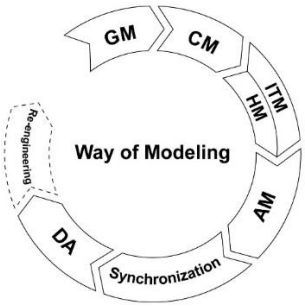

Fig. 6.35. CABS – Way of Modeling.

On the figure: "GM" stands for "*general model*", "CM" stands for "*conceptual model*", "ITM" stands for "*IT model*", "HM" stands for "*humans model*", "AM" stands for "*aspect model*", and "DA" stands for "*data analytics*".

We propose to go *middle-out* because in the Border Security domain, it seems most pragmatic to start with modeling "what is there" (a mixture of person-tasks, device functionalities, and so on to be seen at the border) – such a model we call a *general model* (GM). No other model that would inevitably be abstract, would allow for grasping everything correctly and also communicating it adequately with all relevant stakeholders – this is claimed to be of great importance particularly for the Border Security domain. Just as an example of GM, we consider a typical land border point and we take an 'imaginary' view on things that may be seen at a border point – see Figure 6.36:

Fig. 6.36. GM example.

As seen from the figure: there is a border fence and border police officers patrolling along the fence; there are cameras attached to the fence, which realize crowd monitoring and there are mobile cameras attached to drones; there are finger-print

devices that can be used by police officers for personal identification; there are (networked) computers running and streaming all sensor raw data, and also processing it by applying data fusion algorithms (for example), allowing 'higher-level' reasoning, and so on. Hence, we claim that such a model should be the starting point in specifying a *CABS* system.

We use the *GM* as basis for deriving a *CABS*-related *classification* of concepts – this we call a *conceptual model* (*CM*) – see Figure 6.35. This way of 'arriving' at the *CM* guarantees that our further *system* development activities would be 'grounded'. The `human agent` concept and the `device` concept appear to be essential within the *CABS conceptual model* (Figure 6.37). That is because the *CABS general model* suggests that anything that can be observed at the border either relates to a personal (human) role or to a functionality delivered by a device (equipment). Further, among the human agents at the border (besides the persons who are crossing the border and are thus left outside the scope of the CABS system) are `customs officers` and `police officers`, while among the devices one could observe at the border are `sensors`, `computers`, and `vehicles`. Sensors in turn could be `audio sensors` and `video sensors`, while computers could be `servers` and `personal computers`, and vehicles could be `cars` and `drones`. And so on. This is just as an example on how a *CM* can be derived, based on a *GM*.

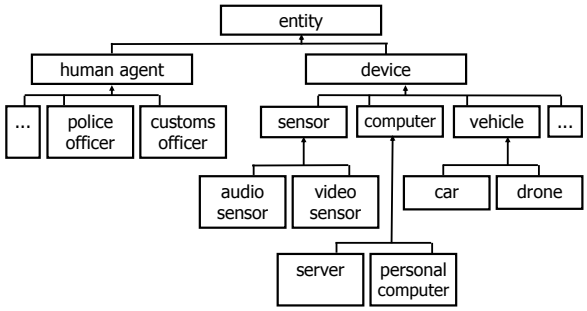

Fig. 6.37. Deriving a conceptual model.

Such a *conceptual model* is the necessary starting point in an *SDBC* software development but also if one would just need to build an *enterprise model*.

Hence, we demonstrated that not only *top-down modeling* but also *middle-out modeling* can be supported by *SDBC*.

IN SUMMARY, in the current chapter, we considered one case study and two illustrative examples, and demonstrated how *enterprise engineering* and *software engineering* can be brought together, supported by *SDBC* and enriched by an explicit consideration of *user-defined requirements*, and also how this can be extended to accommodate *service-orientation* and *middle-out modeling*.

EPILOGUE

I have written this book for software architects who want to improve the way they are developing enterprise information systems and I hope that reading the book was enjoyable.

Enterprise engineering and corresponding social theories have advanced and represent a good basis for modeling complex real-life (organizational) processes. Software engineering and corresponding computing paradigms have advanced as well and represent a good basis for developing software, starting from a computation-independent model of the software system-to-be. Nevertheless, bringing together enterprise engineering and software engineering is still a challenge even though this challenge has been acknowledged many years ago. For this reason, it is not surprising that we observe currently many software projects reaching failure, going over budget, bringing insufficient satisfaction to users, and so on. Hence, there is space for improvement in this regard - enterprise modeling and software engineering need to be better aligned. They should be considered as one integrated task. Otherwise, computation-independent software models, lacking adequate enterprise modeling background, would keep on leading to the development of software that would only partially fit its real-life (enterprise) environment.

The current book tells you how to sort this out - it gives the direction. Nevertheless, what it offers is not to be taken as an 'A to Z recipe', as in cooking. Instead, I believe that reading the book has inspired ideas and ways of thinking that you find useful with regard to YOUR way of developing software. I have not only presented social theories (Chapter 3) and computng paradigms (Chapter 4) but I have also introduced a common conceptual background for them (Chapter 2). Further, by introducing the SDBC approach (Chapter 5) and considering a case study and examples accordingly, I have brough forward some justification with regard to the integrated consideration of enterprise engineering and software engineering. It is up to you to reflect those ideas, guidelines, and examples in your work, such that you usefully enrich the software development approach you follow.

However, much more efforts are still needed in this direction and we should not forget that the alignment between enterprise modeling and software engineering has been challenging the software community for many years already. This is an interdisciplinary challenge that can only be solved by bringing together enterprise engineers and software developers, inspiring interdisciplinary project and discussions. I believe that the current book represents a small contribution in that direction and I am happy to further my efforts. If you want to join activities in exactly that direction, have a look at the website of the international symposium on business modeling and software design: www.is-bmsd.org.

I dedicate this book to the memory of my father, Blagovest.

BIBLIOGRAPHY

1. Abolhassani, M.: Business Objects: From Definition to Application. Delft University Press, Delft (2003)

2. Ahmed, M.A., M. Janssen, J. Van Den Hoven: Value Sensitive Transfer (VST) of Systems Among Countries: Towards a Framework. J. Electronic Government Research 8(1) 26-42 (2012)

3. Apostel, L.: Facts, Problems, Conjectures and Negations about Perception and Observations (Studies in Knowledge). J. Communication & Cognition (1980)

4. Atkinson, C.; J. Bayer; C. Bunse; E. Kamsties; O. Laitenberger; R. Laqua; D. Muthig; B. Paech; Z. Wust; J Zettel: Component-based Product Line Engineering with UML. Addison-Wesley (2001)

5. Atkinson, C. and D. Muthig: Enhancing Component Reusability through Product Line Technology. In the proceedings of the 7th International Conference on Software Reuse, April 15-19, 2002, Austin, Texas, USA (2002)

6. AWARENESS, 2008, Freeband AWARENESS project, http://www.freeband.nl/ project.cfm?id=494&language=en (2008)

7. Bertalanffy, L. von: General Systems Theory. Braziller, New York (1968)

8. BMSD, The International Symposium on Business Modeling and Software Design, http://www.is-bmsd.org

9. Brambilla, M., J. Cabot, M. Wimmer: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers (2012)

10. Bunge, M.A.: Treatise on Basic Philosophy, vol.4, A World of Systems. D. Reidel Publishing Company, Dordrecht (1979)

11. CCM, The OMG CORBA Component Model, http://www.omg.org/spec/CCM/

12. CLOSER, The International Conference on Cloud Computing and Service Science, http://closer.scitevents.org

13. Cobley, P. and L. Jansz.: Introducing Semiotics. Icon Books, Cambridge (2001)

14. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley (2000)

15. Cordeiro, J., J. Filipe, K. Liu: NOMIS: A Human Centred Modelling Approach for Information Systems. In the proceedings of the 4th International Workshop on Enterprise Systems and Technology (I-WEST), Athens, Greece. SCITEPRESS (2010)

16. Cordeiro, J., J. Filipe, K. Liu: Towards a Human Oriented Approach to Information Systems Development. In the proceedings of the 3rd International Workshop on Enterprise Systems and Technology (I-WEST), Sofia, Bulgaria. SCITEPRESS (2009)

17. DEMO, The EEI DEMO Methodology, http://www.ee-institute.org/en/demo

18. Dey, A.K..: Understanding and Using Context. J. Personal Ubiquitous Computing 5(1) 4-7 (2001)

19. Dietz, J.L.G.: Enterprise Ontology, Theory and Methodology. Springer, Heidelberg (2006)

20. Dietz, J.L.G.: Basic Notions Regarding Business Processes and Supporting Information Systems. In the proceedings of the CAiSE'04 Workshops in connection with the 16th International Conference on Advanced Information Systems Engineering, June 7-11, 2004, Riga, Latvia (2004)

21. Dietz, J.L.G.: The Atoms, Molecules and Fibers of Organizations. J. Data & Knowledge Engineering (2003)

22. Dietz, J.L.G.: Generic Recurrent Patterns in Business Processes. In the proceedings of the International Conference on Business Process Management (BPM), June 26-27, 2003, Eindhoven, The Netherlands. Springer - LNCS (2003)

23. Dietz, J.L.G.: Understanding and Modeling Business Processes with DEMO. In the proceedings of the 18th International Conference on Conceptual Modeling (ER), November 15-18, 1999, Paris, France. Springer - LNCS (1999)

24. D'Souza, D.F. and A.C. Wills: Objects, Components, and Frameworks with UML, The Catalysis Approach. Addison-Wesley (1998)

25. Eick, S.G., T.L. Graves, A.F. Karr, J. Marron, A. Mockus: Does Code Decay? Assessing the Evidence from Change Management Data. IEEE Transactions on Software Engineering, 27(1), 1-12 (2001)

26. EJB, The ORACLE Enterprise JavaBeans Technology, http://www.oracle.com/ technetwork/java/javaee/ejb/index.html

27. Filman, R., T. Elrad, S Clarke, M Aksit: Aspect-Oriented Software Development. Addison-Wesley (2004)

28. Freund, J.E.: Modern Elementary Statistics. Prentice-Hall International, New Jersey (1988)

29. Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton Mifflin, Boston (1979)

30. Goncalves da Silva, E.M.: User-Centric Service Composition, Towards Personalised Service Composition and Delivery. University of Twente, Enschede (2011)

31. Guareis de Farias, C.R.: Architectural Design of Groupware Systems: a Component-Based Approach. University of Twente, Enschede (2002)

32. Habermas, J.: Theorie des Kommunikatives Handelns. Erster Band, Suhrkamp Verlag, Frankfurt am Main (1981)

33. Hirschheim, R., H. Klein, K. Lyytinen: Information Systems Development and Data Modeling – Conceptual and Philosophical Foundations. Cambridge University Press, Cambridge (1995)

34. Holt, A.: Organized Activity and Its Support by Computer. Kluwer Academic Publishers, Dordrecht (1997)

35. Huysmans, P.: On the Feasibility of Normalized Enterprises: Applying Normalized Systems Theory to the High-Level Design of Enterprises. University of Antwerp (2011)

36. Ivanov, I.: Cloud Computing in Education: The Intersection of Challenges and Opportunities. In: Filipe, J., Cordeiro, J. (eds.) Web Information Systems and Technologies 2011. LNBIP, vol. 101, pp. 3 - 16. Springer, Heidelberg (2012)

37. Kotonya, G. and I. Sommerville: Requirements Engineering. John Wiley & Sons (1998)

38. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley (2003)

39. Lang, J., G. Pigozzi, M. Slavkovik, L. van der Torre: Judgment Aggregation Rules based on Minimization. In the proceedings of the 13th International Conference on Theoretical Aspects of Rationality and Knowledge. ACM (2011)

40. Lehman, M.M. and J.F. Ramil: Rules and Tools for Software Evolution Planning and Management. Ann. Soft. Eng. (2001)

41. Lewandowski, S.M.: Frameworks for Component-Based Client/Server Computing. J. ACM Computing Surveys (1998)

42. Levin, R.I., Rubin, D.S.: Statistics for Management. Prentice Hall (1997)

43. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)

44. Mannaert, H., J. Verelst, K. Ven: The Transformation of Requirements into Software Primitives: Studying Evolvability Based on Systems Theoretic Stability. Science of Computer Programming (2011)

45. Mannaert, H., J. Verelst, K. Ven: Towards Evolvable Software Architectures Based on Systems Theoretic Stability. Software: Practice and Experience (2011)

46. Mannaert, H., and J. Verelst: Normalized Systems – Re-Creating Information Technology Based on Laws for Software Evolvability. Koppa, Kermt (2009)

47. MDA, The OMG Model Driven Architecture, http://www.omg.org/mda

48. MOF, The OMG Meta-Object Facility, http://www.omg.org/mof

49. OASIS, Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 12 October 2006, http://docs.oasis-open.org/soa-rm/v1.0

50. Papazoglou, M.: Web Services: Principles and Technology. Prentice Hall (2008)

51. Pierce, C.S.: Principles of Philosophy. C. Hartshorne and P. Weiss, eds. Thoemmes, Bristol (1998)

52. Searle, J.R.: Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, Cambridge (1969)

53. Shishkov, B.: Methodological Support for the Design of Enterprise Information Systems with SDBC: Towards Distributed, Service-Oriented and Context-Aware Solutions. In the Proceedings of the 4th International Workshop on Enterprise Systems and Technology, July 2010, Athens, Greece SCITEPRESS (2010)

54. Shishkov, B.: Software Specification Based on Re-usable Business Components. Delft University Press, Delft (2005)

55. Shishkov, B.: Business Engineering Building Blocks. In the Proceedings of the 9th Doctoral Consortium of CAiSE – International Conference on Advanced Information Systems Engineering, May 27-28, 2002, Toronto, Ontario, Canada (2002)

56. Shishkov, B. and J.L.G. Dietz: Applying Component-Based UML-Driven Conceptual Modeling in SDBC. In the proceedings of the 7th International Conference on Enterprise Information Systems (ICEIS), May 24-28, 2005, Miami, Florida, USA. SCITEPRESS (2005)

57. Shishkov, B. and J.L.G. Dietz: Design of Software Applications Using Generic Business Components. In the proceedings of the 37th Hawaii International Conference on System Sciences (HICSS), January 5-8, 2004, Big Island, Hawaii, USA. IEEE (2004)

58. Shishkov, B. and J.L.G. Dietz: Deriving Use Cases from Business Processes, the Advantages of DEMO. In the proceedings of the 5th International Conference Enterprise Information Systems (ICEIS), April 23-26, 2003, Angers, France. SCITEPRESS (2003)

59. Shishkov, B. and D. Mitrakos: Towards Context-aware Border Security Control. In the proceedings of the 6th International Symposium on Business Modeling and Software Design (BMSD), June 20-22, 2016, Rhodes, Greece. SCITEPRESS (2016)

60. Shishkov, B. and M. Janssen: Towards a Service-Oriented Architecture for eVoting. In the proceedings of the 6th International Symposium on Business Modeling and Software Design (BMSD), June 20-22, 2016, Rhodes, Greece. SCITEPRESS (2016)

61. Shishkov, B. and M. Van Sinderen: On the Design of Context-Aware Applications. In the proceedings of the 2nd International Workshop on Enterprise Systems and Technology (I-WEST), May 23, 2008, Enschede, The Netherlands. SCITEPRESS (2008)

62. Shishkov, B. and M. Van Sinderen: From User Context States to Context-Aware Applications. In ICEIS'07 – Revised Selected Papers. LNBIP 12, Springer (2008)

63. Shishkov, B., M. Van Sinderen, B. Tekinerdogan: Model-Driven Specification of Software Services. In the proceedings of the ICEBE'07 IEEE International Conference on e-Business Engineering. IEEE (2007)

64. Shishkov, B., M. Van Sinderen, D. Quartel: SOA-Driven Business-Software Alignment. In the proceedings of the ICEBE'06 IEEE International Conference on e-Business Engineering. IEEE (2006)

65. Shishkov, B., M. Warnier, M. Van Sinderen: On the Application of Autonomic and Context-aware Computing to Support Home Energy Management. In the proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS), June 8-12, 2010, Funchal, Madeira, Portugal. SCITEPRESS (2010)

66. Stahl, T., M. Völter, J. Bettin, A. Haase, S. Helsen: Model-Driven Software Development - Technology, Engineering, Management. John Wiley & Sons, Heidelberg (2006)

67. Stamper, R.: Organizational Semiotics – Information Without the Computer? In: Information, Organization, and Technology – Studies in Organizational Semiotics, K. Liu, R.J. Clarke, P.B. Andersen, and R.K. Stamper, eds. Kluwer, Amsterdam (2000)

68. Stamper, R.: Organizational Semiotics. In: Information Systems: An Emerging Discipline?, J. Mingers and F. Stowell, eds. McGraw-Hill, London (1997)

69. Stamper, R.: Signs, Information, Norms and Systems. In: Signs of Work: Semiotics and Information Processing in Organizations, B. Holmqvist and P.B. Andersen, eds. Walter de Gruyter, New York (1996)

70. Stamper, R., K. Liu, M. Hafkamp, Y. Ades: Signs Plus Norms – One Paradigm for Organizational Semiotics. In the proceedings of the 1st International Workshop on Computational Semiotics, May 26-27, 1997, Paris, France (1997)

71. Stamper, R.: Language and Computer in Organized Behavior. In: Linguistic Instruments in Knowledge Engineering, R.P. v.d. Riet and R.A. Meersman, eds. Elsevier Science, Amsterdam (1992)

72. Stojanovic, Z.: A Method for Component-Based and Service-Oriented Software Systems Engineering. Delft University Press, Delft (2005)

73. Szyperski, C.: Component Software, Beyond Object-Oriented Programming. Addison-Wesley (1998)

74. UML, The Unified Modeling Language, http://www.uml.org

75. Unger, T., R. Mietzner, F. Leymann: Customer-Defined Service Level Agreements for Composite Applications. J. Enterprise Information Systems, 3(3): 369-391 (2009)

76. Van Sinderen, M.J.: From Service-Oriented Architecture to Service-Oriented Enterprise. In the proceedings of the 3rd International Workshop on Enterprise Systems and Technology (I-WEST), July 29-30, 2009, Sofia, Bulgaria. SCITEPRESS (2009)

77. Van Sinderen, M.J. and L.F. Pires: Protocols Versus Objects: Can Models for Telecommunications and Distributed Processing Coexist?. In the proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS). IEEE (1998)

78. Weinberg, G.M.: An Introduction to General Systems Thinking. John Wiley & Sons (1975)

79. Wieringa, R.J.: Requirements Engineering, Framework for Understanding. John Wiley & Sons (1995)

80. Wikipedia, The Free Encyclopedia, http://en.wikipedia.org

81. Williams, S. and C. Kindel: The Component Object Model: A Technical Overview. Microsoft Corporation White Paper, Microsoft (1994)

82. Winograd, T.A.: Language / Action Perspective on the Design of Cooperative Work. In Greif (Ed.), Computer Supported Cooperative Work: A Book of Reading, Morgan Kaufmann, San Mateo (1988)

83. XMI, The OMG XML Meta-data Interchange, http://www.omg.org/spec/XMI

84. XML, The W3C Extensible Markup Language, http://www.w3.org/XML

85. Yin, R.: Case Study Research: Design and Methods. Sage Publications (1994)