From Scenarios to KCPM Dynamic Schemas: Aspects of Automatic Mapping

Günther Fliedl, Christian Kop, Heinrich C. Mayr
University of Klagenfurt
[fliedl | chris | mayr]@ifit.uni-klu.ac.at

Abstract: Scenarios are a very popular means for describing and analyzing behavioral aspects on the level of natural language. In information systems design, they form the basis for a subsequent step of conceptual dynamic modeling. To enhance this step, linguistic instruments prove applicable for transforming scenarios into conceptual schemas of various models. This transformation usually consists of three steps: linguistic analysis, component mapping and schema construction. Within this paper we investigate to which extent these steps may be performed automatically in the framework of KCPM, a conceptual predesign model which is used as an Interlingua between natural language and arbitrary conceptual models.

1. Introduction

Usually information system development projects start with textual requirements descriptions which are, in a subsequent step, transformed into a conceptual schema based on a particular conceptual model (e.g. using the UML [BRJ99]). For analyzing behavioral aspects on the natural language level so-called scenarios are very popular (see e.g. [Ro98]) and used for various purposes (see e.g. [We98]). Much effort has been spent, therefore, on systematically mapping such scenarios to conceptual schemas in order to retain the intended semantics and to provide a basis for an automatic transformation. Linguistic instruments prove useful for that purpose ([AR97], [Bu97], [BDT96]). Within this paper we investigate such instruments in the context of the framework of KCPM (Klagenfurt Conceptual Predesign model [KM98]), a conceptual predesign model which is used as an Interlingua between natural language and arbitrary conceptual models. In particular, we concentrate on those transformation aspects that may be performed automatically by focusing the two main transformation steps: component mapping and schema construction.

The first step consists in finding the appropriate target schema primitive for a given scenario component (see section 5.1). This can be achieved by some mapping heuristics which are based on linguistic models (e.g. [Di80], [Fi77]). We use the NTMS (Naturalness theoretical morpho syntax) [MFW98]) as our primary linguistic model.

1 The work presented within this paper is part of the NIBA project ("Natürlichsprachliche Informationsbedarfsanalyse", "Natural Language Based Requirements Analysis") which is funded by the Klaus Tschira Stiftung, Heidelberg
Within the second step, decisions have to be made about where and how a new schema primitive has to be integrated into the target schema. We will show, that the question where to integrate a new primitive can be answered at a very early phase by analyzing sentences in relation to their linguistic context and by generating a successor graph from the sentences in a text. This is supported heuristically by the linguistic model NTMS. To answer the question of how to integrate a given primitive, we developed a set of integration guidelines (see section 5.3).

To cover the objectives of this paper, we first introduce the interlingua KCPM shortly in chapter 2. Then the linguistic model NTMS is presented (chapter 3). In chapter 4 we describe how the NTMS can be used to construct the successor graph. Chapter 5 discusses the afore-mentioned mapping and construction heuristics. In chapter 6 the presented work is summarized. We do not demonstrate the transformation of a KCPM schema to a conceptual model (e.g. an activity chart) since this would extend the paper enormously. For this purpose the interested reader is referred to [KM02].

2. KCPM concepts for dynamic modeling

Comparing the different approaches to conceptual behavior modeling in UML [BRJ99] (e.g. state charts, sequence diagrams, use cases, activity diagrams etc.), there seems to be a set of basic principles:
- there are actors capable to execute tasks/services,
- task execution is coupled with the manipulation of some objects (including message sending),
- the execution of a task depends on some pre-conditions,
- the execution of a task results in some post-conditions,
- objects are related to pre- and post conditions.

In the context of requirements elicitation and analysis for business information systems it seems to be useful to concentrate on these principles. Therefore, KCPM is restricted to two main concepts for behavior modeling covering these principles: operation type and cooperation type.

Operation types are used to model functional services that can be called via messages (service calls). As such they may be seen as a generalization of the notions use case, activity, action, method, service etc. Each operation type is characterized by references to so-called thing types, which model the actors (calling and executing instances of the resp. operation type [HM98]) and service parameters. The notion of thing type is the central KCPM concept for structure modeling [KM98] and may be seen as a generalization of the usual conceptual notions class (or entity set) and value type\(^2\). Typical things (instances of

---

\(^2\) Value type corresponds to value set in the ER-model or to typed expression in UML. A KCPM value type may be user defined (e.g. Color, Customername etc.). It is not restricted to simple types (e.g. Number, Date etc.).
thing types) are natural or juristic persons, material or immaterial objects, abstract notions. In textual requirements specifications they are usually referred to by noun phrases.

UoD dynamics emerge from actors (thing types) performing operations under certain circumstances (pre-conditions) and thus creating new circumstances (post-conditions). This is captured by the KCPM concept of co-operation type\(^3\) which is formally (simplified) defined as a triple \(<\text{pre},\{(A,O)\},\text{post}>\) where pre and post are sets of conditions (possibly combined by logic expressions), and \(\{(A,O)\}\) is a set of actor, operation type pairs. Cooperation types related by pre- and post-conditions establish a cooperation type schema. Figure 1 illustrates this definition: Co-operation types are represented by rectangles, operation types by ellipses. The latter are drawn into the rectangle of those co-operation types they are contributing to. Conditions are represented by circles. The fact that a condition \(c\) is pre-(post-)condition of a co-operation type \(CT\) is represented by arrows pointing from \(c\) to \(CT\) (from \(CT\) to \(c\)). Each condition may be pre-and/or post-condition of one or more co-operation types. In this way, a network of co-operation types may be constructed. For each operation type and condition we depict the ‘involved’ thing types.

KCPM also offers abstraction mechanisms by nesting schemas: a complex operation type again may be defined using a cooperation-type (sub-)schema.

3. NTMS

The NTMS (Natürlichkeitstheoretische Morphosyntax) [Fl99] is a grammar model based on generative syntax. Linguistic phenomena are represented by trees expressing constituency and dependency. These trees are projections of lexical base-categories. We distinguish between dominant and subdominant percolation. Every tree has just one head and accordingly just one dominant heritage line. The head of a construction usually is a lexical category. Given our framework, we do not use abstract theta-grids, but specify predicate-argument-structures filled with concrete semantic roles; see [Fl00b].

Predicate-Argument-Structures (PAS) as defined in [Fl00b] are an important part of lexicon entries. Lexicon elements bear a code valence and the argument position of the indicated theta-roles. Fine differentiations are made by means of morphosyntactic

---

\(^3\) This term has been taken from object orientated business process modeling [KKM95]. A co-operation in that sense is an elementary step of a business process to which one or more actors contribute by (concurrently) executing operations (services).
features. Thematic roles can be considered as semantic parts of verbs. Mainly we use the following theta-roles for verb classification: AG, EXP, TH, GO, SO, LOC [Fl00b].

AGENT (AG) denotes the executive of an action, e.g. a natural or juristic person carrying out or intending that action. EXPERIENCER (EXP) is a person perceiving something or realizing a psychologically specifiable experience to which it has a certain attitude. THEMA (TH) denotes an entity that is involved in an action or event without being able to influence or control that action. GOAL (GO) is a concrete or abstract place where an event terminates or that is reached when the action is considered successful. SOURCE (SO) is the starting point of a process, an action or an event. LOCATION (LOC) is the place put in direct or indirect relation with the entity or object involved in the event encoded by the verb. Verb classes are defined with respect to specific theta-role configurations like the PAS in the list given in table 1, which are related to the abbreviations on the left. Verbs are heads of verbal phrases and sentences. Therefore, NTMS proposes to project the semantics expressed by the respective verb class to the phrase and sentence level. We will take advantage of that when defining the graph in section 5.2.

<table>
<thead>
<tr>
<th>Class</th>
<th>Abbreviation</th>
<th>PAS</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AUX</td>
<td>&lt;∅&gt;/V_fin</td>
<td>can, must shall ...</td>
</tr>
<tr>
<td>2</td>
<td>eV</td>
<td>[TH]</td>
<td>to go off, to arrive, to come in</td>
</tr>
<tr>
<td>3</td>
<td>iV</td>
<td>AG/TH[]</td>
<td>to work, to exist</td>
</tr>
<tr>
<td>4</td>
<td>LocV</td>
<td>TH[LOC]</td>
<td>X remains in P</td>
</tr>
<tr>
<td>5</td>
<td>PossV</td>
<td>GO[TH]</td>
<td>X posseses Y</td>
</tr>
<tr>
<td>6</td>
<td>PsychV</td>
<td>TH[GO/TH]</td>
<td>X overtaxes Y</td>
</tr>
<tr>
<td>7</td>
<td>tVag/2</td>
<td>AG[TH]</td>
<td>X activates Y, X checks Y</td>
</tr>
<tr>
<td>7.1</td>
<td>tVag/2d,g</td>
<td>AG[GO]</td>
<td>X contradicts Y</td>
</tr>
<tr>
<td>7.2</td>
<td>tVag/2pp4</td>
<td>AG[SO/GO]</td>
<td>X argues for Y</td>
</tr>
<tr>
<td>7.3</td>
<td>tVag/2sk</td>
<td>AG[THabstr]</td>
<td>X avoids Y</td>
</tr>
<tr>
<td>7.4</td>
<td>tV/ppsL</td>
<td>AG[GO/SOabstr]</td>
<td>X insists on something</td>
</tr>
<tr>
<td>8</td>
<td>tV/3</td>
<td>AG[GO/TH,GO/SO]</td>
<td>X gives Y something</td>
</tr>
<tr>
<td>8.1</td>
<td>tV/3ti</td>
<td>AG[TH,GOabstr]</td>
<td>X promises Y something</td>
</tr>
<tr>
<td>8.2</td>
<td>tV/3tda</td>
<td>AG[THacc,GOabstr]</td>
<td>Y asks Y to do something</td>
</tr>
<tr>
<td>8.3</td>
<td>tV/3sk,ak</td>
<td>AG[GO,THabstr]</td>
<td>X convinces Y of something</td>
</tr>
<tr>
<td>9</td>
<td>SentV</td>
<td>EXP[TH]</td>
<td>X sees Y</td>
</tr>
<tr>
<td>10</td>
<td>Vcop</td>
<td>TH[N2/A2]</td>
<td>X is something</td>
</tr>
<tr>
<td>11</td>
<td>tV/2</td>
<td>-AG/-EXP[TH]</td>
<td>X contains Y</td>
</tr>
<tr>
<td>12</td>
<td>ReflV</td>
<td>AGi/Thi[i,GO/LOC/TH]</td>
<td>X engages himself in sth.</td>
</tr>
</tbody>
</table>

Table 1: Verb classes of the NTMS (Abbreviations5)

For reasons of transparency we note “pp” instead of p2.

1 For reasons of transparency we note „pp“ instead of p2.

5 (1) AUX = auxiliary; (2) eV = ergative verb; (3) iV = intransitive verb; (4) locV = locative verb; (5) possV = possessive verb; (6) psychV = psychological verb; (7) tVag2 = bivalent agent verb; (7.1) tVag2d,g = bivalent agent verb with dative-object or genitive-object; (7.2) tVag2pp = bivalent agent verb with prepositional object; (7.3) tVag2sk = bivalent agent verb with sentential object; (7.4) tVag2pp = bivalent agent verb with sentential
AUX(1) stands for α-modal auxiliaries. They do not govern θ-roles and form together with the main verb a verb-complex, whose verb status they govern. Instead of a PAS we note the empty θ-grid and the infinitive governed by the auxiliary. Since it is the auxiliary that carries inflectional endings/finite features in the context of a main verb, the status of the main verb can only be non-finite. Verb classes (2)–(12) determine the syntactic and semantic structure of sentences. Their elements commonly are called head verbs. Their bracketing structures represent in addition to syntactic relations those features which are relevant for an interpretation in the field of conceptual modeling. In [Fl99] all head verb class bracketing structures occurring in our model are listed.

Note that our approach has been initially developed for German, however, for a better understanding, we use English examples.

For the purpose of requirement engineering it is now necessary to filter out those verb classes that can be head verbs of potential requirement sentences and mapped to KCPM notions from the ones listed above. In the case of parsing structured sentences, a parse tree defining phrases as unfoldings/projections of lexical entities is generated. These unfoldings are defined with regard to the respective word categories. Then these linguistic concepts are mapped onto conceptual predesign notions according to the semantic features and categories of the linguistic concepts concerned.

Modeling dynamics, the question arises whether a phrase decodes a condition or an operation-type (based on these, cooperation-types may be put together). It proved advantageous to answer that question in two steps, namely by first associating the phrase with one of the following five basic notions and second mapping the associated notion to a KCPM notion. To find out the answer, in a first step we relate verb classes to a reduced set of basic modeling notions that cover from numerous lexical classifications. These notions are activity, completion of activity, property/state, event and restriction. They allow a simple classification of words and phrases on the base of linguistic criteria.

4. Linguistic Support

In this section we will shortly introduce the linguistic support we are taking advantage of for performing the mapping process. To exemplify the concepts we will use the following business rules given in form of a scenario text (taken from [FS98] and slightly modified):

S1  The order comes in.
S2  The order department checks the each article on the order.

prepositional object; (8) tV/3 = trivalent verb; (8.1) tV/3ti = trivalent verb with infinitival complement sentence and thematical identity of the antecedent and the logical subjects of the infinitival group; (8.2) tV/3tdd = trivalent verb with infinitival complement sentence and thematical difference of the antecedent and the logical subject of the infinitival group; (8.3) (tV/3sk,ak) = trivalent verb with infinitival complement and subject control or arbitrary control; (9) sentV = verbum sentiendi; (10) Vcop = copula verb; (11) tV/2 = transitive verb, whose subject does not carry the AG-Role nor the EXP-role; (12) reflV = reflexive verb.
S3 If each article is available in stock, then the order department relates the article to the order.
S4 If the stock quantity of one article is less than the minimal stock quantity, then the stock clerk must order additionally this article.
S5 If the order comes in, also the bookkeeping department checks the payment.
S6 If the payment is authorized and we have all articles in stock, then the order department releases the order.
S7 If payment is authorized but there are not all articles in stock, then the order department must put the order to the pending orders.
S8 If the payment is not authorized, then the order department must reject the order.

Clearly, this example is far from the complexity of textual descriptions in real world situations. This, however, does not mean that our concepts are restricted to such simple cases.

4.1 Basic Notions

Our basic assumption is that for the purpose of dynamic modeling, the twelve verb classes defined in the NTMS model can be related to the following notions ([Fl00], [Ko02]):

- **activity/action** (someone/something does something): relates to all kind of verb classes which have an actor role (iV, tVag/2, tVag/2d,g, tVag/2pp, tVag/2sk, tV/pps,k, tV/3, tV/3ti, tV/3tda, tV/3tdd, tV/3sk,ak, ReflV).
- **property/state** (someone/something is characterized by something): relates to the verb classes LocV, PossV, PsychV, SentV, Vcop, tV/2.
- **event** (something that happens [to someone/something] and results in a state/property): relates to ergative verbs (eV) having a terminative aspect.
- **completion of activity**: relates to activity/action verbs in past participle form like “the order is signed”.

Auxiliaries like want, wish, may usually change the categorization of the main verb they are related to. In principle they express a property/state (e.g., “the clerk wants to book an order”) and, therefore, are related to that category.

A specific basic notion is **restriction** which covers various temporal or causal conditions that do not have a thing type involved, e.g. “on Friday 13th”, “in the evening”).

Actually, we are restricting our research to simple sentences, sentences that are coordinated by and, and if/then constructions. In the latter two cases again there will be (component) phrases that can be related to our basic notions.

4.2 Successor dependencies between sentences

For mapping the source (a natural language sentence or phrase) to the target (an operation, co-operation, pre- or post condition) it is also important to know where to integrate a new
schema part into the existing schema. For this purpose we extend the linguistic model of the NMTS with a precedence indicator $pi$ [F199]. This indicator is part of the bracketing expressions of a sentence and indicates the position of a specific sentence within a text segment. Using this information we can construct a successor graph in which each sentence is presented as a node. A directed edge from one sentence to another symbolizes that the sentence at its starting point is a predecessor of the sentence at its end point (see figure 2). Interpreting this graph we can now determine the position at which a mapping result of a sentence has to be inserted in the target schema (see chapter 5.3).

The precedence indicator ($pi$) is determined by applying the following heuristics:

R1: If a non conditional sentence $S_i$ is headed by an ergative verb, and if its subject is the subject or (part of the) object of a non conditional sentence $S_j$ which headed by a binary agentive verb, then there is a successor relationship $S_i < S_j$. This rule is motivated by the fact that in the context of dynamic modeling, $S_i$ expresses a possibly missing pre-condition.

R2: If a non conditional sentence $S_i$ is headed by a binary agentive verb, and if its subject or (part of the) object is the subject of a non conditional sentence $S_j$ which headed by an ergative verb, then there is a successor relationship $S_i < S_j$. This rule is motivated by the fact that in the context of dynamic modeling, $S_j$ expresses a possibly missing post-condition.

R3: If a sentence $S_i$ is part of the if-phrase of a conditional sentence $S_j$ then there is a successor relationship $S_i < S_j$.

R4: If there are identical nominal phrases (NP) or identical heads of the nominal phrases, then the sentences build a syntactical successor relationship which can be defined as follows: If $NP_{subj}$ (NP with syntactical feature subject) of sentence B is identical with $NP_{obj}$ (NP with syntactical feature Object) of sentence A, then there is a successor relation $A < B$ (sentence A before sentence B).

R5: If there is a sentence A with a head of $NP_{subj}$ and if there exists another sentence B where the head of $NP_{subj}$ modifies the head of $NP_{subj}$ then there exists a successor relationship $A < B$ (e.g. sentence A: my computer does not work properly; sentence B: the screen of my computer is flickering).

Currently we have limited our approach in that we only examine the predecessor sentences of a given sentence according to the rules R1 to R5. Thus, cycles will not be generated although they might be useful in practice to specify loops in a process.

In our example text we find non conditional sentence S1 "The order comes in" in the first position. Sentence S2 has the following specific linguistic characteristics:
- the head of the sentence is a binary agentive verb,
- the sentence is non-conditional, i.e. there is no keyword "if",
- a part of the object ("order") is identical to the subject of S1.
According to the heuristic rule R1 we conclude, that S1 is a predecessor of S2 (S1 < S2).

In sentence S5 the phrase “the order comes in” is part of the if-phrase. Applying heuristic rule R3 we receive S1 < S5.

Analyzing sentence S3 we observe, that the subject (NP_{subj}) in the if-phrase is identical with the object (NP_{obj}) of sentence S2. According to R4 we conclude S2 < S3. The head of the subject in sentence S3 (“article”) is a modifying phrase of the new subject in sentence S4 (“quantity”). This causes S3 < S4 according to R5.

The head of the subject phrase (NP_{subj}) (“payment”) refers to the head of the object phrase in sentence S5. The configuration of S7 and S8 looks similar with respect to the head “payment” of the subject phrase “the payment”. Therefore the filtered sequences are: S5 < S6, S5 < S7 and S5 < S8. Concerning “article”, the other head of a subject phrase in the sentences S6 and S7, sequences from sentences S2, S3 or S4 can be derived with R4. In this case the designer manually decides which of the sentences S2, S3, S4 is the predecessor sentence of the sentences S6 and S7.

The resulting successor graph of the sentences S1 to S8 is shown in figure 2:

![Successor graph for the example](image)

**5. Construction Process**

The KCPM approach is open for any kind of textual sources (free text or structured sentences). However, in the context of our project NIBA (see footnote 1) we restrict ourselves to structured sentences as given in section 4 in order to enhance automatic parsing and analysis (nevertheless the type of structured text which is automatically processed is continuously enriched).

The schema construction process consists of three steps, that are to be iterated by the designer for each sentence:
5.1 Mapping

The mapping of the previously defined concepts (activity/action, property/state phrase, event-phrase, completion of activity phrase and restriction phrase) to a KCPM schema primitive is defined as follows:

- an activity/action is mapped to an operation type in KCPM,
- a property/state, an event-phrase and a completion of activity is mapped to a KCPM condition,
- a restriction phrase is mapped to a pre-condition.

In our example, sentence S1 is an event-phrase according to our basic notions. Therefore, it is mapped to a pre-condition. On the other hand, the verb in sentence S2 encodes an activity/action phrase. Therefore sentence S2 is mapped to an operation type.

A co-operation type primitive is derived from if/then-constructions. In [Fl00] we have shown, that if there is a “valid pattern”, i.e. a possible combination of notions,
- a co-operation-type primitive with a pre-condition part and an operation-type part can be derived (IF <prc> THEN <O>) or
- a co-operation-type primitive with a operation type section and post-condition section can be derived (IF <O> THEN <poc>).

Thus the sentences S3 to S8 are mapped according to these mapping heuristics.

5.2 Completion

Often, textual requirements specifications are incomplete, i.e. information necessary for schema completion is missing. In the second step, therefore, the designer has to complete

---

6 Note that this mapping is not the only information we extract from natural language requirements sentences: e.g., they are also subject to mappings when extracting model elements describing static UoD aspects.
a newly inserted schema primitive. For instance, if an operation type primitive has been selected, where the actor is not referred to within the related activity phrase, or where some parameters are missing, then this has to be supplemented. Another example is the completion of a co-operation type primitive, where the post conditions are not yet known. Sometimes this may be derived from the related activity phrase (e.g. \textit{IF <pre> THEN X releases order} implicitly contains the post-condition \textit{order is released}).

5.3 Integration

This step is supported by the construction guidelines depicted in figure 5 and by the “successor information” derived from the successor graph in section 4.2 (figure 2).

The construction guidelines give hints on how to integrate a new mapped schema primitive to the existing schema: We start from a given primitive $G_i$ and investigate the schema information available so far: $C_1$ corresponds to the situation where only a related precondition exists, $C_2$ to all other situations (i.e., there is already a related co-operation/operation type entry). Then, the matching case is selected from the table.

The successor graph of section 4.2 helps to decide where the new schema primitive must be integrated. The main idea here is: If in the successor graph a sentence $S_j$ is a direct successor of sentence $S_i$ ($S_i < S_j$) and both sentences can be transformed to modeling primitives (e.g. operation type, co-operation type, pre- or post-condition), then the schema primitive resulting from $S_j$ can be connected to the schema primitive resulting from $S_i$, which is already part of the existing schema.

Treating these steps in detail would mean going far beyond the scope of this paper. We therefore restrict ourselves to illustrate the construction based on the simple example already presented in section 4.

- Sentence S1 is an event phrase due to the ergative verb “comes in”. Therefore it is mapped to a pre-condition which forms the first schema primitive.
- Sentence S2 represents an activity phrase and is thus mapped to an operation type. Furthermore, from the successor graph we know that sentence S2 follows sentence S1. Hence, the resulting primitive (operation-type) of S2 must be integrated to the primitive derived from sentence S1 (pre-condition). Therefore, we can now use construction guideline $G_4/C_1$ to complete the previously found schema primitive by forming a co-operation type.
- Sentence S3 is an \textit{IF <pre> THEN O} construct to which we assign a co-operation type primitive consisting of a pre-condition and an operation type. Once again a look at the successor graph shows us that S3 follows S2. That means, that the integration point of the primitive resulting from S3 is right after the position where we have already integrated the primitive derived from S2. We therefore may apply the construction guidelines of $G_5/C_2$. Since there are two alternatives (A, B), it is necessary to make a decision. Choosing alternative B means that we accept the pre-condition of the new schema primitive also to be the post-condition of an existing one.
Sentence S4 is transformed and integrated similarly to sentence S3 using again guideline G_5/C_2 with alternative B. Following the successor graph, the integration position of S4 is right after the position the primitive derived from S3, which was previously integrated to the co-operation type schema.

Sentence S5 leads to a co-operation type primitive, the pre-condition of which is matched according to G_5/C_1,B with the initially found one ("order comes in"). The successor graph reflects that fact by a successor relationship (S1 < S5). Thus, a new path within the co-operation type schema starts. The primitive derived from sentence S5 is integrated directly after the pre-condition primitive of sentence S1.

From sentence S6 we can derive a co-operation type with two pre-conditions. We again may apply G_5/C_2,B to extend the schema. However at this point, the situation occurs that, according to the successor graph, S6 is a successor of more than one predecessor sentence (S5 < S6, S2 < S6, S3 < S6, S4 < S6). We can partially solve this problem because for the term "payment" only the successor relationship (S5 < S6) exists. Therefore, we can integrate the primitive derived from S6 right after the primitive resulting from S5. Note that this is only the integration position for "payment". For the other involved term "article" there still exist several relationships (S2 < S6, S3 < S6, S4 < S6). In this case the designer still has to resolve the conflicting situation manually. For our purpose, let us assume that he prefers the relationship (S2 < S6) as the correct one. Then the primitive derived from S6 has to be integrated after the primitive resulting from S2.

This process is continued exploiting sentences S7 and S8 an applying guideline G_5/C_2,B, as well as analyzing the successor graph. For S7 the situation is similar to the situation described in S6, thus it has to be resolved analogously. For sentence S8 the integration point is right after the mapping result of S5 according to the successor graph.

Usually textual requirements specifications do not contain “trivial” post-conditions explicitly (e.g. “the order is rejected” as a post-condition of rejecting). However, we might add such post-conditions for a more complete description. In particular, such post-conditions might again be matched with pre-conditions of further co-operation types, thus contributing to a stronger connection of the schema components. The completed schema resulting from that process is presented in figure 4.

6. Conclusion

In this paper heuristics were presented that support the automatic generation of conceptual dynamic schemas based on textual scenarios. By introducing an interlingua instead of the common well known models for dynamic conceptual modeling (activity diagrams, state charts, sequence diagrams) our approach is independent from such models while collecting functional requirements. However, it is powerful enough to map to the above mentioned conceptual UML models by applying mapping heuristics ([KM02], [Ko02]).
Means for automatic schema mapping may enhance requirements collection and analysis significantly. Whereas our NIBA Toolset (see [Fl02]) is rather powerful for treating requirements concerning static UoD aspects (by transforming natural language texts into UML object models), the transformation of dynamic aspects are in its beginnings. The paper introduced the following heuristic techniques for that purpose:

1) mapping based on a set of basic notions (section 4.1): to support the association of a conceptual primitive to a given natural language phrase,
2) heuristics to derive successor relationships (section 4.2): to support the decision where (at which position) a new schema part can be integrated into an existing schema,
3) integration guidelines (section 5.3): to support the decision how a new schema part should be integrated.

However, the heuristics to derive successor relationships between sentences have still to be extended. They have some limitations noted explicitly or implicitly in the paper:

- The construction of the successor graph is done linearly from the first to the last sentence of the text. For each sentence only it’s predecessor sentences are examined to find the right successor/predecessor relation according to the rules.
- Whereas the presented heuristics, being developed originally for the German language, may also be applied for English texts, it is quite likely that additional ones will be more language specific.

Apart from these limitations we think that the construction of a successor graph can also help other approaches to integrate schema elements to an existing schema (e.g. a new activity to a activity diagram).
References


[Fi00b] Fliedl, G.; Kop, Ch.; Mayerthaler, W.; Mayr, H.C.; Winkler, Ch.: From Natural Language to Conceptual Predesign (Static and Dynamic Aspects of Language Engineering); ÖGAI Journal 20/1


### Appendix

<table>
<thead>
<tr>
<th>Event</th>
<th>Property</th>
<th>Completion of Activity</th>
<th>Activity</th>
<th>IF (&lt;\text{pre})&gt; THEN (&lt;\text{op}&gt;)</th>
<th>IF (&lt;\text{pre})&gt; THEN (&lt;\text{poc}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G1) Event, Restriction</td>
<td>(G2) Property</td>
<td>(G3) Completion of Activity</td>
<td>(G4) Activity</td>
<td>(G5) IF (&lt;\text{pre})&gt; THEN (&lt;\text{op}&gt;)</td>
<td>(G6) IF (&lt;\text{pre})&gt; THEN (&lt;\text{poc}&gt;)</td>
</tr>
<tr>
<td>R/E</td>
<td>Prop.</td>
<td>Col/A</td>
<td>Not a valid case!</td>
<td>A)</td>
<td>B)</td>
</tr>
<tr>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
<td>(&lt;\text{pre}&gt;) \rightarrow \text{Element}</td>
</tr>
</tbody>
</table>

E(xisting schema to which the new element (primitive) is added.

- G1/C1
- G1/C2
- G2/C1
- G2/C2
- G3/C1
- G3/C2
- G4/C1
- G4/C2
- G5/C1
- G5/C2

Construction Guidelines

Figure 5: co-operation type schema construction guidelines