Towards Entropy-Based Requirements Elicitation

Arkadiy V. Kostanyan, Vladimir A. Shekhovtsov

Department of Computer-Aided Management Systems
National Technical University "KhPI", Kharkiv, Ukraine
kostanyan@datasvit.net, shekvl@kpi.kharkov.ua

Abstract: This paper presents our views towards constructing the universal parsing technique for the software requirements texts and the requirements elicitation technique based on the output of this parsing process. With the proposed parsing technique, it should be possible to achieve language-independent processing of the requirements texts. Source sentences are treated as systems with words as elements and with a state determining the set of links between elements. The main goal of the parser is to find the state of the system with the highest organization level by minimizing its entropy. The requirements elicitation technique extends trained indicator approach by Cleland-Huang treating inter-word links as indicators. The output of the elicitation process is the set of requirements information ready to be processed further using Conceptual and Aspectual Predesign techniques.

Key words: NLP methods, parsing, aspectual predesign, conceptual predesign, genetic algorithm, entropy, training, indicator weights

1 Introduction

The input for requirements analysis phase of the software process consists of customer requirements expressed in various textual, graphical, or verbal forms. For large-scale systems, the volume of requirements makes it difficult to understand and analyze them in full detail. Therefore, an organization or requirements information becomes necessary. To organize the requirements information obtained from the requirements texts into special intermediate models and convert these models into the conceptual model, the Conceptual Predesign [KM02] and Aspectual Predesign [SK05, She06] techniques were proposed (the former aimed at processing functional requirements, the latter enhancing it to cover non-functional requirements).

Prior to using these techniques, however, it is necessary to fill the intermediate models with requirements information. One of the possible approaches of doing this is to use natural language processing (NLP) techniques for analyzing the requirement specifications texts and eliciting the necessary information. For each NLP method, the first stage is a parsing of text. This article describes our suggestions towards constructing the universal requirement specification parser aimed at determining the set of links

between the sentence's words. We also propose the requirements elicitation technique using the set of links determining by the parser. The output of the elicitation process is the set of classified requirements information suitable for use with Conceptual and Aspectual Predesign.

The remainder of this work is organized as follows. Section 2 gives brief background information. Section 3 states the problem. Section 4 describes the parsing technique, section 5 – the elicitation technique. Section 6 describes the related research. Section 7 includes our conclusions and recommendations for future research.

2 Background

2.1 Conceptual and Aspectual Predesign

The main goal of the Klagenfurt Conceptual Predesign [KM02, MK02] is to implement requirement gathering that could be controlled and verified by the end-user. This goal is achieved with the help of KCPM (*Klagenfurt Conceptual Predesign Model*) – intermediate semantic model that resides between the requirement specification and the conceptual model. This model consists of semantic concepts that are more general than conceptual modeling concepts and could be more easily understood and verified by the end user, such as *thing-type* (generalization of entity/class and attribute), *connection-type* (representing all kinds of relationships among the concepts), *operation* (generalization of the method), and *event*. KCPM could be presented in tabular form (as glossaries) and as the semantic network. The KCPM is supposed to be retrieved from the functional requirement specifications. This can be done manually or with the help of Natural Language Processing (NLP) tools. The latter approach is used in the project NIBA [Nib02, FKM03] that will be described later in this section.

The main goal of Aspectual Predesign technique [SK05, She06] is to elicit the non-functional (quality) requirements from the software requirements specifications into enhanced model based on KCPM (we call this model Aspectual Predesign Model, APM). In this model, thing-types are used to represent quality requirements types, crosscutting behavior implementing quality requirements (advices, interceptors) is represented via operation-types; pointcuts (rules that connect advices to some places in model where they are supposed to be called) are represented via the modified connection-types. Aspectual predesign can be seen both as an extension to the Klagenfurt conceptual predesign that allows mapping the non-functional requirements and as an intermediate step of the AOSD [Asp05] residing between aspect-oriented requirements engineering and aspect-oriented modeling.

As for KCPM, it is necessary to perform the elicitation of the non-functional requirements to obtain the natural language requirements specifications before starting the aspectual predesign process and to capture the artifacts of these requirements directly from the requirements specifications via NLP methods. For this purpose, one can use the results of existing projects (e.g. NIBA) or develop special-purpose approaches.

2.2 NIBA project

NIBA project represents tools and methods to get ready conceptual model directly from requirements texts. The NIBA project workflow consists of the following phases [Nib02]:

1. Textual analysis using an NTS based parser.
2. Interpretation of the parser output and transformation of requirements related patterns into KCPM schema entries.
3. KCPM schema validation and modification by the end users and their consultants.
4. Transformation of the KCPM schema into conceptual model by application of heuristic transformation rules.
5. First cut cost estimation using function point analysis based on the knowledge gathered in phases 2) and 4).

For the purpose of our paper, we need to look at the first step of NIBA workflow in more detail.

NTS (Naturlichkeitstheoretische Syntax) [MFW98] is a grammar model based on generative syntax. Descriptions of grammatical phenomena are represented by trees expressing both constituency and dependency relations. These trees are unfoldings / projections of lexical base-categories. Basic lexical categories are V0 (verb), N0 (noun), A0 (adjective), P0 (preposition), Q0 (quantifier), ADV0 (adverb), PT0 (particles), and SPZ0 (auxiliaries and determiners).

NL-Parsing is done using a Prolog engine. The Prolog Parser is written in the Definite Clause Grammar (DCG) format. The lexicon of the parser comprises 20 verb classes, to which class-specific phrase structures have been assigned. The input for the parser could be any German sentence related to the chosen domain.

3 The Problem Statement

Investigating the current state of affairs, we see the following open problems for NLP techniques (in particular those implemented for NIBA) applying to requirements texts.

1. It is necessary to have parser capable of processing sentences with arbitrarily complex structure. Such parser should have open system of rules making it easy to define new sentence structure cases if necessary. Current version of NTS parser uses DCG that restricts the available sentence structures and lexical constructs to some predefined set not intended for extension.

2. Large software projects are multinational and often multi-language in nature. Therefore, the parser should be capable of handling sentences expressed in different languages (at least for Latin-based language group). Current version of NTS grammar uses lexical categories tightly tied to German and not intended for reuse as other languages' categories.

3. The output of the parser should be used as an input for requirement elicitation techniques. For further processing, these techniques need to have access not only to the linguistic information related to the parsed texts, but also to the requirement-related information. Such information currently is not included in the output of the NTS-based parser.

4. Currently, the NTS-parsed information can only be transferred into KCPM (on the Step 2 of the NIBA workflow); functional requirement-specific actions are "hardwired" into this process. It is not possible currently to transfer non-functional (quality) requirements information into APM.

The purpose of this paper is to make a first step and propose requirements parsing and classification techniques addressing these problems.

4 Entropy-Based Parser Concept

To solve the parsing problems described above we propose the new concept of parser building that we call *entropy-based parser*. In this section, we present the basic concept of this parser, its foundations, and implementation issues.

4.1 A Sentence System

We consider each source sentence as a *system* with the set of *elements* (words) and the set of *links* between the elements (the *sentence system*). The link between the words is defined as a semantic connection between the concepts corresponding to these words in a sentence's context. We follow the opinion of [Rup06] that the words by themselves (out of context) are meaningless. Consequently, the set of links between the words makes sense of the sentence from the knowledge point of view and generates the sentence system from its elements from the system analysis point of view. To put it briefly, we assume that *the current state of the sentence system is determined by the current set of links between its words*.

Let us formalize this idea further and present some notions and definitions. First we denote a sentence system by W, its elements by w_i, $i \in I(W)$, where $I(W) = \{1, 2, ... |W|\}$ is a set of all indexes of W elements; $|W|$ is a cardinality of W.

We define a *link state* of the sentence system as a presence of the link between the particular pair of the sentence elements (w_i, w_j). A link state is denoted by $s(i,j) \in S(W)$, $i,j \in I(W)$ where $S(W)$ is the set of all possible link states of the sentence system W. Later on we will often omit pair indexes defining link state simply as $s \in S(W)$. Every link state is supplemented with *link state probability* $P(s) \in [0, 1]$, $s \in S(W)$ indicating the probability of the corresponding link to appear in a sentence. We outline the ways of calculating this probability later in this section.

We define an *enhanced sentence system A(W)* as a sentence system where all word elements and link states are augmented with additional sets of properties. Actually, in A every element $x \in W \cup S(W)$ is replaced with the following tuple:

$$x' = < x, \mathrm{Prop}(x), \mathrm{Req}(x) >, \; x \in W \cup S(W) .$$

Here Prop(x) is the set of *linguistic properties* of x which is either word or link state; Req(x) – the set of *requirement-related properties* of x. Further we will also use the notion of *w'* (augmented word), and *s'* (augmented link).

Linguistic properties are language-dependent (they can represent such characteristics as declension, conjugation, case, number etc.); they are used to calculate linguistic probabilities of links further in this section.

Requirement-related properties are defined in a process of requirement elicitation; establishing them for all the words and links in a sentence forms the basis for eliciting the requirement out of this sentence. They will be described in detail in Section 5.

4.2 Entropy Calculation

The main task of the parser is to define exact structure of the sentence (a set of words and the links between them). We assume this structure corresponds to the state of the system with *the highest degree of organization* and use well-known notion of *entropy* as a measure of this degree. We calculate the system entropy $H(W)$ as follows:

$$H(K) = -\sum_s P(s) \cdot \log(P(s)), \; s \in K \subset S(W), \quad K \in U(W) \subset \Omega(W). \tag{1}$$

Here K is the particular subset of link states of a sentence, $U(W)$ – set of all subsets of the link state set in W leaving no words of W unlinked, $\Omega(W)$ is a set of all possible subsets of the link state set in W. We assume that it is not feasible to consider sentences containing unlinked words.

The system state with the highest degree of organization has the minimum of entropy. Therefore, the main task of our parser is to select *entropy-minimizing subset of link states* $K^* \subset S(W)$ among all subsets of the link state set qualified to calculate entropy:

$$\left\{ K^* \middle| \; H^* = \min_K \left(-\sum_s P(s) \cdot \log(P(s)), \; s \in K \subset S(W) \right), \; K \in U(W) \; \subset \; \Omega(W) \right\} .$$

We calculate link state probabilities $P(s)$ for elements of link state set K using the following formula:

$$P(s) = P^L(s) \cdot P^S(s), \quad s \in K .$$

Here $P^L(s)$ and $P^S(s)$ are, correspondingly, *linguistic* and *statistical probabilities* of the link state s. Here P^L represents a priori probability based only on common linguistic knowledge; whereas P^S represents a posteriori probability calculated on a basis of statistical experiments with training corpora.

We calculate the linguistic probability of the link state $s(i, j)$ using the following formula:

$$P^L(s(i,j)) = \frac{1}{|j-i|} \cdot \frac{L(i,j)}{L(i)+L(j)}, \quad i, j \in I(W) \cdot$$

Here $L(i, j)$ is a number of linguistic properties of a pair of elements (w_i', w_j') making possible the presence of a link between w_i and w_j; $L(i)$ – total number of linguistic properties defined for an element w_i'. When w_i and w_j are adjacent (j is next to i) and all the w_i' and w_j' properties make possible the presence of a link, the value of this characteristic is equal to one (the link certainly exists). In the worst case (no properties make possible the presence of a link), it will be equal to zero (the link certainly not exists).

Example of linguistic properties making possible the presence of a link between w_i and w_j can be the property "w_i is an adjective" together with the property "w_j is a noun". Exact rules should be defined for the particular language.

We calculate the statistical probability of the link state $s(i, j)$ using the following formula:

$$P^S(s(i,j)) = \frac{C(i,j)}{C(i)+C(j)}, \quad i, j \in I(W) \cdot$$

Here $C(i, j)$ is a number of sentences in the training corpora where both elements of the link state pair (w_i, w_j) are present; $C(i)$ – number of sentences in the training corpora, were element w_i is present. This value also changes in range from zero to one.

4.3 Sentence Parsing Algorithm

Now we should define the method for searching the optimal set of links for the described criteria (minimal entropy value). In our case, the main task is to select K^* from a predefined number of $K \in U(W) \subset \Omega(W)$. It is possible to solve this combinatorial problem by exhaustive search, but if the number of words in the sentence is large, this could take a long time due to combinatory explosion effect.

To avoid exhaustive search, we propose to use a genetic algorithm (for general theory of these algorithms, their convergence, performance advantages over the exhaustive search etc. see e.g. [Mic99]). This algorithm (**Algorithm 1**) uses links as individuals and

subsets of possible links $K \in U(W) \subset \Omega(W)$ as populations. It is applied to shorten the search among all the subsets of the same size. Its sequence of steps is as follows:

Step 1. Randomly select initial population $K \in U(W) \subset \Omega(W)$ of size $|K| \in \left[|W|-1, \binom{|W|}{2}\right]$.

Step 2. Perform the crossover operation swapping first words of two random links.

Step 3. Perform the mutation operation replacing both words of the randomly chosen link with two random words from the same sentence.

Step 4. Calculate entropy for the modified population according to (1).

Step 5. Perform the selection operation generating new population based on values of $P(s) \cdot \log(P(s)),\ s \in S(W)$ (selected are top $|K|$ individuals).

Step 6. Terminate if entropy remains unchanged through *MG* iterations; in this case, K^* for the particular subset size is found; otherwise go to step 2.

As mentioned, this algorithm considers only subsets of the same size. To look through all the $K \in U(W) \subset \Omega(W)$ we propose the following **Algorithm 2**:

Step 1. Let $G = |W|$-1.

Step 2. Apply Algorithm 1 to subsets of size G, find $K^*(G)$, calculate $H(K^*(G))$, set $K^{\min} = K^*(G), H^{\min} = H(K^*(G))$ if $H(K^*(G)) < H^{\min}$ or $G = |W|$-1.

Step 3. Let $G = G$+1.

Step 4. Terminate if entropy is not decreasing through *ML* iterations or if $G = \binom{|W|}{2}$+1 (all possible subsets are processed); in this case $K^{\min}$ is the entropy-minimizing set of links; otherwise go to step 2.

It is important to note that for most requirement texts (where sentences are not large), exhaustive search can be feasible so this algorithm should be applied only if performance problems are expected.

As an output of the parsing process (which consists of applying Algorithm 2 or exhaustive search sentence-by-sentence), we obtain the set of sentence structures $(W, K^*(W))$, each structure containing the set of the words and the set of the links between these words. These structures form the input for the requirement elicitation process, which we describe in the next section.

5 Requirements Elicitation

An input for the requirements elicitation process is the set of parsed sentence systems. Every such sentence system has the set of its words and the set of links between them.

5.1 Requirement-Related Properties

Requirement-related properties are represented by the following tuple:

$$\mathrm{Req}(x) = < \mathrm{Kind}(x), \mathrm{Repr}(x), \mathrm{Satis}(x), \mathrm{Role}(x) >, \; x \in W \cup S(W)$$

Every Req(x) element represents particular *facet* of the requirement [Gli05] connected to the element x (which, as earlier, is either word or the link). Following a classification by Glinz [Gli05], enhanced in [SK06] according to the recommendations by ISO [ISO01], we briefly outline the set of possible values for each facet.

1. Kind(x) represents the *kind* of requirement connected to x. Possible kinds according to [Gli05, SK06] can be selected from the following two-level hierarchy:

a. **Functionality**, decomposed into:
   Suitability, Accuracy, Interoperability, and Security;
b. **Reliability**, decomposed into:
   Maturity, Fault tolerance, Recoverability;
c. **Usability**, decomposed into:
   Understandability, Learnability, Operability, and Attractiveness;
d. **Efficiency**, decomposed into:
   Time behavior and Resource utilization;
f. **Maintainability**, decomposed into:
   Analyzability, Changeability, Stability, and Testability;
e. **Portability**, decomposed into:
   Adaptability, Installability, Co-existence, and Replaceability.

Other requirements hierarchies exist (see, for example, [Kas06] for an extensive hierarchy of the quality aspects corresponding to the quality requirements); one can choose which is better for a particular case.

Our hierarchy does not separate functional and non-functional (quality) requirements, they are listed alongside each other (see [Gli05, Kas06] for more in-depth discussion). It is important for our purpose, as we target both Conceptual and Aspectual predesign so the parser should parse all the requirements regardless of their types.

2. Repr(x) denotes the *representation* of the requirement. Possible values are "operational", "quantitative", and "qualitative".

3. Satis(x) denotes the *degree of satisfaction* of the requirement. Possible values are "hard" (the requirement must be satisfied) and "soft" (the requirement is desirable to be satisfied).

4. Role(x) denotes the *role* of the requirement. Possible values for roles are "prescriptive" and "assumptive".

These four facets are our preliminary attempt to outline a set of requirements-related properties. Elaborating detailed definition of these properties is a target for further investigation.

5.2 Setting the Sentence Properties

Requirement-related properties can be set by the process similar to one proposed by Cleland-Huang [Cle06]. Following [Cle06] it is possible to define the set of indicator terms (keywords) and train this set on the sample sentences already classified into requirements categories. Here the training means determining the weights indicating the relevance of the terms for the requirements categories. After the set is trained, it is possible to classify newly presented document sentences into requirements categories based on the total relevance weight of the indicator terms found in every sentence.

We propose to adopt this technique by maintaining the *trained element set* $A^T = W^T \cup S^T$, where W^T is the set of trained words and S^T is the set of trained links. The element of this set (*trained sentence element*, based on either word or the link) is denoted by $x^T = < x, \mathrm{Req}^T(x) >,\ x \in A^T$, where $\mathrm{Req}^T(x)$ is the following tuple:

$$\mathrm{Req}^T(x) = < V^{\mathrm{kind}}(x), V^{\mathrm{repr}}(x), V^{\mathrm{satis}}(x), V^{\mathrm{role}}(x) >,\ x \in A^T.$$

Here $V^{\mathrm{kind}}(x) = \{v^{\mathrm{kind}}(x,f) \in [0,1], f \in F^{\mathrm{kind}}\}$ is a set of indicator weights $v^{\mathrm{kind}}(x,f)$ for the set F^{kind} of all possible values f of the *kind* facet (indicating relevance of x for the categories corresponding to these values, i.e. categories corresponding to the requirement kinds). For brevity, we will use the *kind* facet in the following description; other facets are processed exactly the same way.

The weights $v^{\mathrm{kind}}(x, f)$ are calculated using simplified formula from [Cle06] as

$$v^{\mathrm{kind}}(x, f) = \frac{1}{N_f} \sum_{W_f} \frac{N(x, W_f)}{|W_f|} \cdot \frac{N_f(x)}{N(x)},\ f \in F^{\mathrm{kind}},\ x \in A^T.$$

Here N_f is a number of sample sentences falling into "kind *f*" category (*f-sentences*), $|W_f|$ is a cardinality of *f*-sentence system; $N(x, W_f)$ is a number of occurrences of x in f-sentence W_f; $N_f(x)$ is a number of *f*-sentences containing x, $N(x)$ is a total number of sentences containing x. See [Cle06] for more discussion regarding calculation of these weights and adjusting them based on sample sentence systems used in training.

We propose to go through the parsed sentence system element by element. Every element x of the parsed sentence system is matched against W^T if x is a word or against S^T if it is a link (the match succeeds if an element is found in a set). Then, if x is a word, all the links involving this word are matched against S^T. If x is a link, both words comprising this link are matched against W^T. We stop the process at this point, but in theory it is possible to recursively explore the links further, this process can be controlled by specified recursion depth.

After all the comparisons are made, the matched elements form the basis for obtaining property facet values for the current element. To do this, we calculate total weights z_f^{kind} of all the requirement facet values:

$$z_f^{\text{kind}}(x) = \prod_{y \in AM^T(x)} v^{\text{kind}}(y, f), \quad f \in F^{\text{kind}}, \quad AM^T(x) \subset A^T \cdot$$

Here $AM^T(x)$ is the set of trained elements matched against x. The facet value $f^{\text{kind}*}(x)$ possessing the largest total weight is chosen as the value of this facet for the current element.

$$\text{Kind}(x) = \{ f^{\text{kind}*}(x) \mid z^{\text{kind}*}(x) = \max_{f \in F^{\text{kind}}} z_f^{\text{kind}}(x) \}$$

For cases when no exact match is found, we plan to implement calculating the proximity between the current element and the trained set elements and set facet values based on the weights calculated by combining this proximity and the trained weights. In this situation, current element with all its facet weights can be added to the trained set (so the training can continue during the elicitation process). An approach to proximity calculation is a target of future research.

5.3 Eliciting Requirements from Extended Sentence Systems

The result of the described process is the set of extended sentence systems containing requirement-related properties supplementing all sentence elements (both words and the links between them). Actually, these properties define the *classification of sentence elements according to the requirement facets*. This classification can be used to convert the elements into the representation of KCPM and APM. It is especially well suited to eliciting the quality requirements ready to be filled into APM:

1. Links between the words can be converted into APM connection types representing rules describing crosscutting in the system (quality requirements usually crosscuts different functionality of the system). Corresponding requirement-related properties (belonging to both the link and the words it connects) define the type of corresponding quality requirement. In AOSD and APM terminology, these rules are called pointcuts.

2. Words connected with these links can be converted into operation-types describing crosscutting behavior of the system (the implementations of quality requirements). In AOSD and APM terminology, these operations are called advices or interceptors.

See [SK05, She06] for more discussion of the APM and possible types that can be represented by this model.

6 Related research

Our work can be put under an umbrella of the requirements discovery techniques. These techniques process free-text or structured requirements specifications using NLP, Information Retrieval, and similar methods with a goal of obtaining the requirements from these documents in automated way.

According to the Theme/Doc method [BC04] the requirements specifications are processed in search for keywords that are used to generate diagrams for the design phase. This keywords search is mostly template-based so the parsing is simple. Approach in [Ros04] uses IR-based techniques (e.g. regular expression matching) for finding the requirements related to the particular system quality. Only simple regular expressions are considered, nothing in this approach can be compared to the real requirements parser.

In Section 5, we have already shown how it is possible to use an approach of Cleland-Huang [Cle06] to supplement our parsing technique. In its initial form, this approach classifies requirements documents into categories related to quality aspects (our *kind* facet). It differs from our technique in that it does not include parsing step (the input for classification process is pure text) and it does not consider the properties of links between the words.

7 Conclusions and future work

Described entropy-based parsing and elicitation technique has some important properties making it suitable for use with Conceptual Predesign and Aspectual Predesign.

1. It allows working with linguistic information on the abstract level using universal notions. As a result, it is possible to use such parser for any language sharing common set of specific linguistic properties.

2. It reveals word-linking information that can be used to describe the requirements that cannot be easily described by pure words such as quality requirements.

3. It augments parsed words and links with requirement-related properties that can be used to elicit the requirement information ready to be filled into KCPM or APM.

We plan to test the viability of the ideas presented in this article using real-life set of requirements specifications texts. We also consider investigating the structure of requirement-related properties in more detail and create rules for converting parsed and property-augmented information into KCPM or APM.

We also plan to investigate the possibility to use neural network-based approaches to implement the training process applied to define the requirement-related properties.

Bibliography

[Asp05] Aspect-Oriented Software Development, Addison-Wesley, 2005.

[BC04] Baniassad, E.; Clarke, S. Finding Aspects in Requirements with Theme/Doc. In: Proceedings of Early Aspects 2004, Lancaster, UK, 2004.

[Cle06] Cleland-Huang, J; Settimi, R.; Zou, X.; Solc, P. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In: Proceedings of RE'06 Conference, 2006.

[FKM03] Fliedl, G.; Kop, Ch.; Mayr, H.C.: From Scenarios to KCPM Dynamic Schemas: Aspects of Automatic Mapping. In: (Düsterhöft, A.; Thalheim, B. Eds.): Natural Language Processing and Information Systems - NLDB'2003. Lecture Notes in Informatics P-29, GI-Edition, 2003, pp. 91-105.

[Gli05] Glinz, M. Rethinking the Notion of Non-Functional Requirements. In: Proc. of the Third World Congress for Software Quality (3WCSQ 2005), Munich, Vol. II, 55-64.

[ISO01] ISO/IEC 9126-1 (2001). Software engineering – Product quality – Part 1: Quality model. International Organization for Standardization, 2001.

[Kas06] Kaschek, R.; Pavlov, R.; Shekhovtsov, V.; Zlatkin, S.: Towards Selecting among Business Process Modeling Methodologies. In: Proc. of BIS 2006, Klagenfurt, Austria, 2006.

[KM02] Kop, Ch.; Mayr, H.C.: Mapping Functional Requirements: From Natural Language to Conceptual Schemata. In: Proc. International Conference SEA 2002, Cambridge, USA, Nov. 4-6, 2002, pp. 82-87.

[MFW98] Mayerthaler, W.; Fliedl, G.; Winkler, Ch. Lexikon der Natürlichkeitstheoretischen Syntax und Morphosyntax, Stauffenburg Verlag, Brigitte Narr, Tübingen, 1998.

[Mic99] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1999.

[MK02] Mayr, H.C.; Kop, Ch.: A User Centered Approach to Requirements Modeling. In: M.Glinz, G. Müller-Luschnat (eds.): Proc. Modellierung 2002. Lecture Notes in Informatics P-12 (LNI), GI-Edition, 2002, pp.75-86.

[Nib02] Niba, L.C.: The NIBA workflow: From textual requirements specifications to UMLschemata. In: Proc. ICSSEA'2002, Paris, December 2002.

[Ros04] Rosenhainer, L. Identifying Crosscutting Concerns in Requirements Specifications, Workshop on Early Aspects, Vancouver, Canada, Oct. 2004.

[Rup06] Rupp, C. Clairvoyance for Intermediate Learners. Keynote for International Conference NLDB 2006.

[She06] Shekhovtsov, V.A.; Kostanyan, A.V.; Gritskov S.; Lytvynenko Yu.: Tool Supported Aspectual Predesign. In: Information Systems Technology and its Applications - ISTA'2006. LNI P-84, GI-Edition, 2006, pp. 153-164.

[SK05] Shekhovtsov, V.A.; Kostanyan, A.V.: Aspectual Predesign. In: Information Systems Technology and its Applications - ISTA'2005. LNI P-63, GI-Edition, 2005, pp. 216-226. URL: http://apmtool.sourceforge.net/apm/apmpaper.pdf

[SK06] Shekhovtsov, V.A.; Kostanyan, A.V.: Universal software system requirements classification. In: Proc. CSIT'2006, September 2006, Lviv, Ukraine.