COMPLIANCE TO QUALITY CRITERIA OF EXISTING REQUIREMENTS ELICITATION METHODS

Peter Bollen
Maastricht University, Faculty of Economics and Business Administration, Maastricht, the Netherlands

John Simons
Groningen University, Faculty of Management and Organization, Groningen, the Netherlands

Abstract. In this article we define a requirements elicitation method based on natural language modelling. We argue that our method complies with synthesized quality criteria for RE methods, and compare this with the compliance of traditional RE methods (EER, ORM, UML). We show limited empirical evidence to support our theoretical argument.

Keywords: Requirements engineering, requirements determination, requirements elicitation, criteria for RE approaches.

1. Introduction

The development of information systems requires a thorough requirements elicitation process [10]. Improvements in this process will lead to improved systems development efforts [9].[1]

Most requirements elicitation methods do not assess the compliance to objective criteria in the literature, partly due to the absence of criteria that can operationally be applied by RE analists. In [7] an overview of criteria for RE approaches is given. These criteria are subsequently synthesized into 4 overarching quality criteria for RE methods.

This paper is organized as follows. In the next paragraph we discuss and summarize quality criteria for RE methods. These criteria are *completeness*, *efficiency*, *formality* and *domain richness*. They apply to three different aspect of RE, being the way of modeling (the product of RE), the way of working (the operational process of RE) and the way of controlling (the managerial process of RE). In the paragraph on (our) natural language modeling RE method we show that this method does comply with all four quality criteria. We proceed by

[1] Requirements Elicitation (RE), Requirements Determination (RD) or Requirements Analysis (RA) is considered to be one of the most critical activities in an information systems development project [19]. Requirements elicitation, -determination or -analysis contributes to a large extent as a source of information systems failures [10, 18, 28]. In the remainder of this article we will refer to RE, RD, or RA as *Requirements Elicitation*

showing empirical evidence for our theoretical argument. For a detailed discussion on the applicability of RE methods, see [6].

We conclude with a reflection on our theoretical and practical approach to the quality assurance problem for RE methods.

2. Summary of quality criteria from the literature

In [7] four quality criteria for specifications, (software and information systems) requirements and conceptual schema's are presented. Two dimensions for criteria found in the literature are distinguished, being the applicability in the business UoD and the applicability in the requirements elicitation UoD.

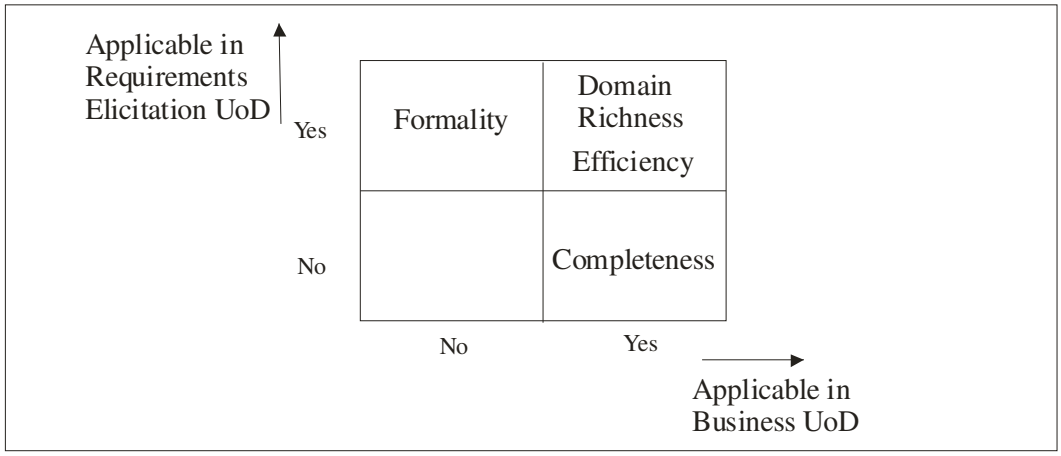

Fig. 1. Applicability of Quality Criteria for the RE and Business Application UoD (this figure is taken from [7: p. 9])

In this section we will summarize the research findings from [7] were it is illustrated how we can map the partial and non-operationalized criteria that were found in a literature survey onto four operationalized criteria for REM's: *completeness, efficiency*, *formality* and *domain richness.* In figure 1 the relevance of these criteria as a function of the RE product and the RE process are illustrated (taken from [7: p.9]).

2.1 The completeness criterion

In [7] the completeness criterion for a requirements elicitation method is operationalized, e.g. *what* must be incorporated in a requirements specification for an application domain. In this section we will give a summary of those findings.

Three perspectives are distinguished: the *data-oriented* perspective, the *process-oriented* perspective and the *behaviour-oriented* perspective. The data-oriented perspective focuses on the business data: the domain concepts, the definitions and naming conventions for those domain concepts, the relationships

between the domain concepts and other 'static' and 'structural' knowledge in the enterprise. In the process-oriented perspective the business activities and user perceivable tasks, are the focal areas of interest. The process-oriented perspective describes what procedures exist for the creation of instances of semantic relationships. The behaviour-oriented perspective [7: p.10] contains a description of how 'events' can be cross-referenced to procedures in the process-perspective and relationships in the data-oriented perspective [7: p.10]. In addition two types of rules that are distinguished within the former perspectives: *state* rules and *action* rules. A combination of these two frameworks leads to the following category of rules that exists in a UoD: *data model*, *static constraints*, *static derivation rules*, *dynamic constraints* and *dynamic rules*. In [7] the availability of (modeling) procedures is considered to be of great importance for the way of working and the way of controlling.

Table 1. The definition of the completeness criterion (taken from [7: p. 11])

	Way of modeling	**Way of working**
Definition of completeness criterion	The availability of conceptual modeling constructs for the data model, the static constraints, the static derivation, the dynamic constraints and dynamic rules that allow the modularization of the requirements specification.	The availability of procedures for instantiating the data model, the static constraints, the static derivation, the dynamic constraints and dynamic rules for the lowest to the highest level of specification completeness.

2.2 The efficiency criterion

In [7] it is concluded that the existence of 'equivalent' modeling constructs in a RDM's way of modeling might lead to modeling when additional information about the requirements specification becomes available

With respect to the way of working it is explained that the availability of a procedure that guides an analyst in the elicitation process will minimize the required number of analysis steps and potential rework. It is concluded that an efficient procedure must contain a role definition for the analyst and must make a distinction between the responsibilities of the domain user and the responsibilities of the analyst.

With respect to the way of controlling *quality* and *project management* efficiencies are defined. Quality deficiencies have to be 'repaired' by the process that is responsible for creating it. This implies that the REM's way of working must contain a number of 'quality-checking' procedures. Secondly, the efficiency of the project management is measured as *performance,cost* and *time.*

Table 2. The definition of the efficiency criterion (taken from [7: p. 12])

	Way of Modeling	**Way of Working**	**Way of Controlling**
Definition of efficiency criterion	Average number of modeling constructs in a requirements specification language that serve the same purpose must be as low as possible for a given minimum required level of specification organization and for a given minimum required level of semantic stability. The modeling constructs should be easily learned and remembered	Availability of procedure(s) that can be easily applied by an analyst and that will result in a maintainable specification.	Availability of quality assurance steps and the extent in which performance, cost and time can be optimized by having validation mechanisms for domain experts and the presence of go/no go controls in combination with communication milestones built into the modeling procedure(s)

2.3 The formality criterion

The third criterion that is given in [7] is *formality*. A REM must lead to an *(internally) consistent* and *precise* requirements specification. To achieve this the modeling constructs that are used for the specification of requirements in the different perspectives, firstly must be formally defined.

Table 3. The definition of the formality criterion (taken from [7, p. 13])

	Way of Modeling	**Way of Working**	**Way of Controlling**
Definition of formality criterion	Extent in which modeling constructs in language are formally de-fined and can be used to create consistent and unambiguous specifications.	Extent in which procedure is formal in terms of its ability to provide internal verification support or closure and its ability to facilitate external validation.	Extent in which activities can be formally planned. Extent in which quality management is contained in formal (sub)procedure Extent in which provisions that enable traceability are contained in REM. Extent in which results of the RE process can be verified.

Secondly, the way of working, must be formalized in algorithm(s) that contain a precise description of how the formal modeling constructs must be instantiated to obtain consistent specifications. Finally the REM should contain provisions that enable traceability of modeling outcomes between stages

2.4 The domain richness criterion

The literature review presented in [7] gives four dimensions that characterize the application domain .

The first of these dimension is the dimension *perception.* An application domain can range from a uniform perception for all users one hand to an application UoD in which perceptions can be different for different user groups.

The second dimension is the dimension *turbulence.* Here an application domain can range from a domain in which no change occurs to a domain in which continuous changes take place.

Another dimension is concerned with the extent in which the domain knowledge is 'tacit' versus 'explicit'. This is the *tacitness* dimension. In explicit domains all domain requirements are already available in explicit form. In tacit domains the domain requirements are not readily available, but are implicitly contained in routines, experience and ways of working of the users in the domain.

The fourth domain richness dimension is the way in which the requirements elicitation process is anchored. This can range from tangible anchors (for example user examples) to abstract domains in which requirements can only be obtained by means of interviews, directed-questions and/or what-if analysis The four 'domain richness' dimensions that characterize application domains are summarized in table 4.

Table 4. Dimensions that characterize the application domain (taken from [7, p. 14])

Dimension	**Low extreme**		**High extreme**
Perception	uniform for all users	-	Different for all users
Turbulence	no change	-	continuous change
Tacitness	fully tacit	-	fully explicit
Anchoring	tangible	-	abstract

In this section we have summarized the findings of [7] that contain a synthesis of the quality criteria found in the literature in the fields of software engineering, (information) systems development methodologies, conceptual modeling and requirements elicitation.

3. The Natural Language Modeling Requirements Elicitation Method

In this section we give an overview of a RE approach that is documented in Bollen [5, 6] called *Natural Language Modeling* (NLM). The NLM approach for requirements elicitation is based upon the natural language axiom that states that all verbalizable information can be expressed as declarative natural language sentences. The main purpose of the NLM approach for RE is to capture the complete set of abstracted natural language sentences for an application domain. This complete set of abstracted natural language sentences (or sentence group

templates) then will serve as the anchor on which additional business rules can be defined.

3.1 Evolution of NLM

Since the early 1970's a number of semantic modeling approaches have emerged. From the pioneering work of Abrial [1] on the binary relationship model, followed by Falkenberg's object-role model [14] which was subsequently extendend by the popular 'circle-box' notation and an accompanying modeling methodology (ENALIM) [20]. The ENALIM methodology provided the foundation for control data's (binary) NIAM [27]. In the late 1980's binary NIAM evolved into N-ary fact oriented information modeling [17, 21] and the acronym NIAM became a shortcut for *natural language information analysis methodology*. The most recent text book on this approach are [15, 16] in which the NIAM methodology is renamed Object Role Modeling (ORM). NLM has its ancestor in NIAM but has evolved into an approach that can be used for knowledge structuring in general. In this article we will embed NLM in a modeling context for requirements elicitation.

3.2 Basic Modeling Constructs in NLM

A name in human communication is used to refer to a concept or a thing in a real or abstract world. A *name* is a sequence of words in a given language that is agreed upon to refer to *at least* one concept or thing in a real or abstract world, for example, *Jake Jones*, *567893AB*, *General electric*

The choice of names used in communication is constrained by the reference requirement for effective communication. For example, a chemical company will use a *customer code* for referring to an individual *customer*. The use of names from the name class *customer name* in the customer management registration subject area for referring to individual customers, however, will not lead to effective communication because in some cases *two or more* customers may be referenced by *one* name instance from this name class. This is one of the reasons why not all names can be used for referencing entities, things or concepts in a specific part of a real or abstract world. On the other hand it is evident that knowledge workers that are involved in activities in an application subject area have knowledge on the reference characteristic of the potential name classes for the different groups of 'things' in a real or conceived world. This means that they should be able to tell an analyst whether a name from a specific name class can be used to identify a thing or concept among the union of things or concepts (in a specific part of a real or conceived world).

Now we have agreed on the naming conventions for referencing entities, things or concepts in a 'real' or 'conceived' world, we will postulate the main principle (or axiom) in NLM. This principle states that: all appearances of verbalizable information (e.g. forms, note-books, web-pages) can be expressed as declarative natural language sentences. This important principle underlies the

Natural Language Modeling for business applications and we will call it the 'Natural Language Axiom':

In every (business) organization examples of verbalizable information can be found. These examples can be materialized as a computer screen, a world wide web page, a computer report or even a formatted telephone conversation. Although the outward appearance of these examples might be of a different nature, their content can be expressed using natural language.

Before we will introduce further NLM modeling constructs, we will give a common example (of verbalizable information) in figure 2.

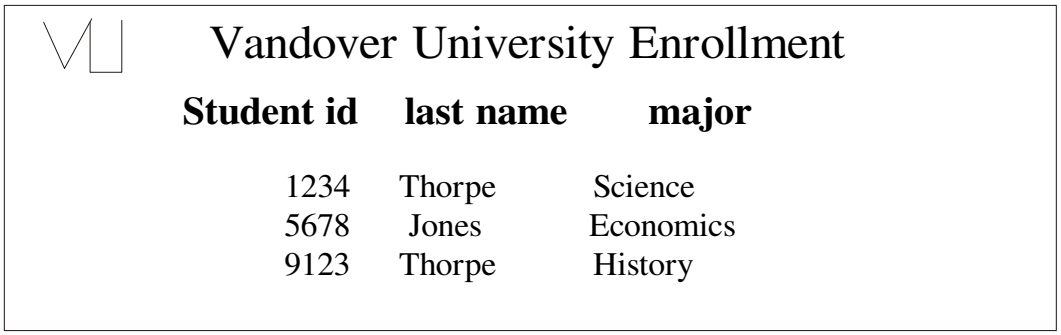

Fig.2. Example Vandover University Enrollment.

The starting point for the RE in NLM is (a number of) real-life verbalizable examples of communication in the subject area.

Verbalizing and abstracting the example of an University Enrollment application area in figure 2 results into two groups according to the type of sentence predicate *(..majors…,* respectively, *..has last name..*). Two sentence group templates for the first sentence group can be derived in which we denote the predicate as text and the variable parts as text between brackets: *Student <enrolled student> majors in major <chosen major>* and *Student <enrolled student> has chosen the major <chosen major>*. Figure 3 shows a graphical representation of the two sentence groups in the University Enrollment example. Each role is represented by a 'box', e.g. *enrolled student*. Each sentence group is represented by a combination of role boxes. Sentence group SG1 is represented by the combination of role boxes *enrolled student* and *chosen major*. Sentence group SG2 is represented by the combination of 'role' boxes *registered student* and *last name*.

For each sentence group one or more sentence group templates are positioned underneath the combination of role boxes that belong to the sentence group. In the diagram of figure 3 sentence group templates 1 and 2 belong to sentence group *Sg1*. Sentence group template 3 belongs to sentence group *Sg2*.

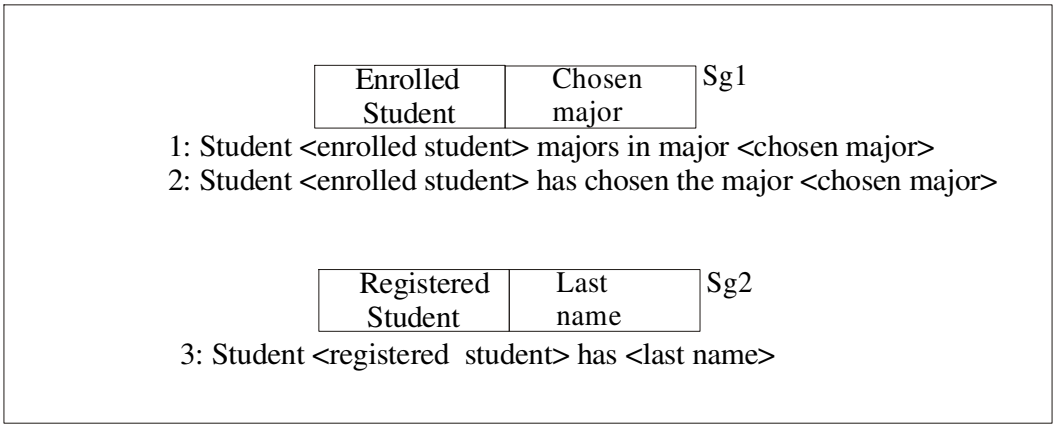

Fig. 3. Roles, sentence group and sentence group template(s) for university enrollment example.

For every application area the relevant concepts and their definitions must be recorded in a list of concept definitions. Such a list of concepts and their definitions should contain a definition for *each* intention in the UoD. A defining concept should either be an intention or a different concept that must be previously defined in the list of concepts or it should be defined in a common business ontology or it must be a trivial generally known concept (for example, sun, moon). For example, the definition of the concepts *Student* and *Major.*

Student: *A student is a person that studies at Vandover University.*
Major: A major is a course program offered to students by Vandover University

The definition of the concept types in the list of concept definitions must specify how the knowledge forming the concept (*definiendum*) is to be constructed from the knowledge given in the definition itself and in the defining concepts (*definiens*). A defining concept should either be a different concept that must be previously defined in the list of concepts or it should be defined in a *generic* business ontology. Brasethvik and Gulla use such a list of concept definitions in the context of a 'shared' or 'common information space' in which the semantics of information is locally constructed and '.. reflects the 'shared agreement' on the meaning of the information' [8: p.47].

Concept	***Definition***
Student	*a person that studies at Vandover University*
Student ID	*a name class, instances of which can be used to identify a <student> among the union of <student>s that have ever been, are or will be enrolled at Vandover University*
Major	*a course program offered to <student>s by Vandover University*
Major name	*a name class, instances of which can be used to identify a <major> among the union of <major>s offered by Vandover University*
Last name	*a name class*

Fig. 4. List of concept definitions for university enrollment application (example 1).

In figure 4 we have given an example of such a list of concept definitions for the Vandover University enrollment UoD.

Naming convention sentence groups

In this section we will further formalize the outcome of the process of the selection of a name class for referring to things in a real or abstract world. The outcome of such a naming process will result in the utterance of sentences, for example sentences 2.1, 2.2, 2.3 and 2.4 (based upon the example in figure 2).

1234 is a name from the student ID name class that can be used to identify a student within the union of students at Vandover University……(sentence 2.1)
5678 is a name from the student ID name class that can be used to identify a student within the union of students at Vandover University…….(sentence 2.2)
Science is a name from the major name name class that can be used to identify a major within the union of majors at Vandover University……(sentence 2.3)
Economics is a name from the major name name class that can be used to identify a major within the union of majors at Vandover University(sentence 2.4)

Sentences 2.1 through 2.4 express that a certain *name* belongs to a certain *name class* and that instances of the name class *student ID*, can be used to identify an instance of a *student,* and an instance of the name class *major name*, can be used to identify an instance of a *major* within the UoD of Vandover University. We can give, for example, the definition of the concept *Student ID*: *Student ID is a name class.* The 'intension' of the names in sentences 2.1 through 2.4 is a *name class* and NOT a type of *thing, entity* or *concept* in the real world. We will, therefore, refer to sentences 2.1, 2.2, 2.3 and 2.4 as *naming convention sentences*. The corresponding sentence group will then be called a *naming convention sentence group.*

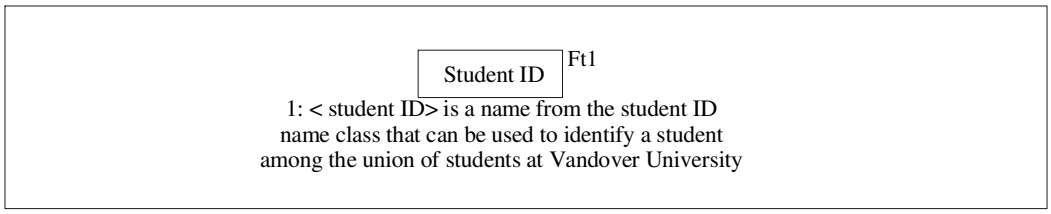

Fig.5. Naming convention sentence group for *student*

Compound reference schemes

In the Vandover University example the intension student has a "simple" reference scheme, namely: the single role "enrolled student" or "registered student". In many cases, however, a *simple* reference scheme will not be sufficient for referencing instances of a given intension. In those cases we will need *compound* reference schemes.

In the predecessor methodologies of NLM, there exists a number of ways in which compound reference schemes can be modeled. In Halpin [15] two of these reference scheme types are illustrated: *nesting* and *co-referencing*. We can apply compound reference schemes in NLM in the same way as the simple reference schemes. To illustrate this we will first adapt our example UoD. We will assume that Vandover University has merged with Ohao University. In order to streamline the enrollment operations, it is decided to centralize them. This means that after the merger, a student can no longer be identified by the existing student ID because a given *student ID* can refer to a student in the (former) Ohao University, *and* to a different student in the (former) Vandover university. To capitalize on the existing naming conventions it is decided to add the qualification *O* (for Ohao) or *V* (for Vandover) to the existing student ID. This extension is the *university code.* The sentence group templates and the corresponding sentence groups in which such a compound reference scheme is implemented are given in figure 6.

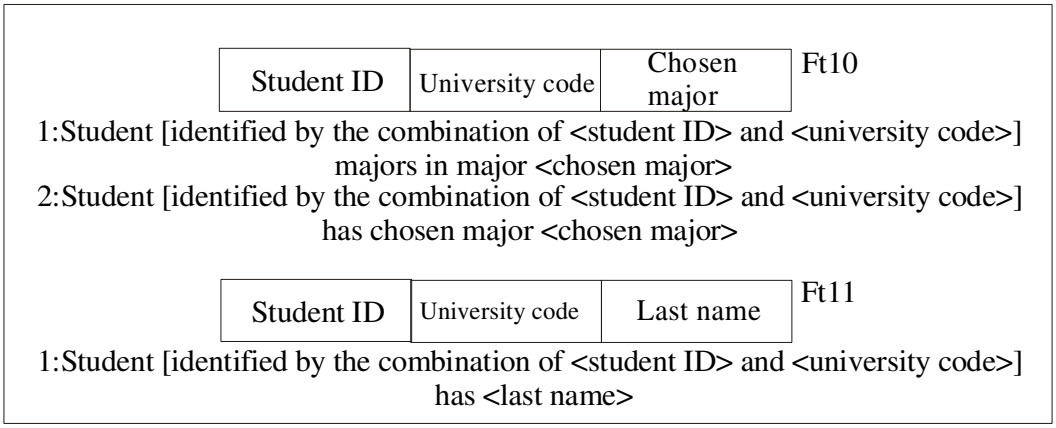

Fig.6. Sentence group(s) template(s) with compound reference scheme for *student*

We have introduced the [ ] ('brackets') symbol for capturing the definition of the compound reference scheme (see figure 6). For example, the reference scheme for student in sentence group Ft10 consists of the roles *student ID* and *university code* and is defined as follows: *Student [identified by the combination of <student ID> and <university code>].* The case of a simple reference scheme is actually a special case of the compound reference scheme in which the brackets and description within (except for the role name used in the reference) are left out. In addition to this we need to adapt the naming convention sentence groups for the constituting intensions of the compound reference scheme. For example, the naming convention sentence group for *student* should be adapted to reflect the application subject area in which it can be used to identify a specific student. In this case a student can be identified by his/her student ID within a *specific* University (Ohao or Vandover).

The unification of simple reference schemes and the different types of compound reference schemes into one uniform way of referencing, and the

capability to capture the precise semantics of naming conventions are improvements in NLM to the predecessor methodologies.

3.2 Business Rule Modeling Constructs in NLM

In this section we will give the NLM modelling constructs that allow us to capture the business rules that can be expressed as propositions on the extension of a basic information model. These business rules can be expressed as (combinations of) *population state constraints*, *population state transition constraints*, *derivation rule constraints* and *impulse type constraints*.

Population state constraints

In this section we will introduce the modeling constructs that express that some extensions of a *basic information model* are **not** allowed to exist. In order to make a distinction into an extension of a basic information model regardless of the fact whether it is allowed to exist and an extension of a basic information model that is allowed to exist, we will introduce the concept of *population state*. A *population state* is an extension of a basic information model that is allowed to exist.

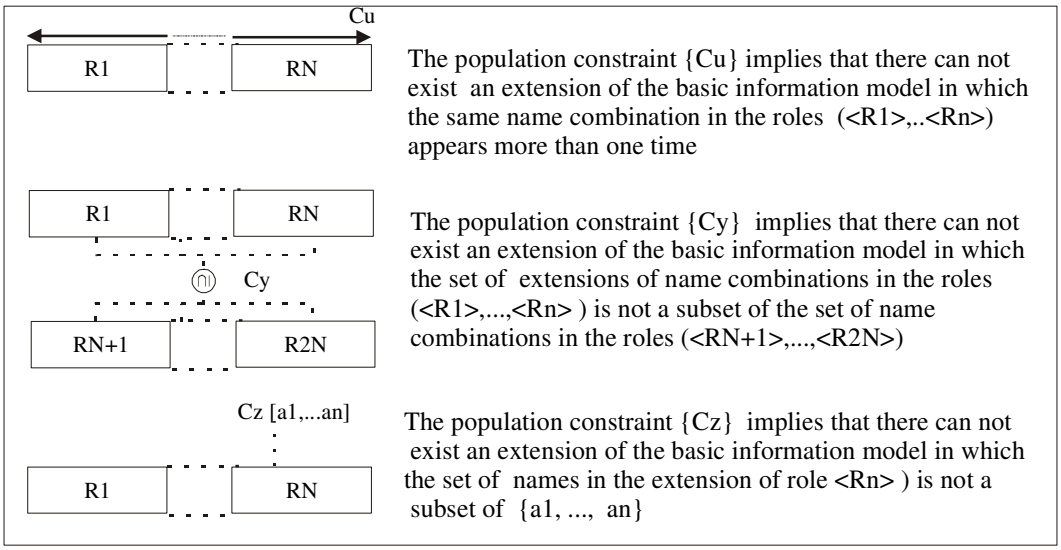

Fig.7. Legend for uniqueness, subset and value population state constraints.

NLM further restricts the extensions that are allowed to exist by incorporating specific *domain knowledge* or *business rules* that can be expressed as propositions on the basic information model that must be true for ***every*** population state. We will call such a proposition a *population state constraint* (see figure 7 for the definition of a number of population constraint types). A *population state constraint p* in a basic information model *BIM* limits the allowed extensions of the basic information model *BIM* to those extensions that comply to the proposition specified in the population state constraint *p*.

The business rule: *a student can be enrolled in at most one major*, can be expressed as the following constraint instance from the constraint legend in figure 7: *The population constraint C1 implies that there can not exist an extension of the basic information model in which the same name combination in the role combination* ***University code*** *and* ***Student ID*** *in sentence group FT10 appears more than one time.* If we inspect this example we can conclude that the addition of a population state constraint onto a (basic) information model actually eliminates those extensions from the set of extensions that do **not** comply to the proposition.

Population state transition constraints

The population state transition constraints specify the limitations on subsequent extensions of a basic information model. A *population state transition constraint* q in a basic information model BIM is a proposition that limits the before-after extension combinations of the basic information model BIM to those combinations for which the proposition of q is true.

The population state transition constraints constrain the possible state sequences of the extension of the basic information model. Even if an extension of the BIM complies to the population state constraints, the allowed before/after combinations are further constrained by these state transition constraints. Constraint *C14* in figure 9 is an example of a state transition constraint that reflects some business rule from our university enrollment example.

Derivation rule constraints

In addition to the population state- and population state transition constraints that limit the possible extensions of a basic information model in terms of for example uniqueness and set-comparison restrictions, a different group of constraints is needed that is able to specify limitations on how values of roles from the basic information model can be derived. We will call this type of constraint: *a derivation rule constraint*. A *derivation rule (constraint)* specifies that instances of a given sentence group can not be inserted or updated freely, but their value is restricted to the pre-conditions and derivation formula of a derivation rule constraint. In the university enrollment example, we have derived two derivation rule constraints: *C15* and *C16* (see figure 9b).

Impulse type constraints

In this section we will give a definition of the *event*, *event type* and *event occurrence* concepts and the group of constraints that constrain the behaviour within a UoD: the *impulse type constraints*.

In order to define the impulse type of constraints we need to define the concept of event occurrence first.

An *event occurence* is a happening at a certain point in time in the application subject area that can lead to the execution of one or more derivation rules and/or the insertion or deletion of sentence instances into/from the

application's information base. For example the event occurrence: *student 'V 2345' wants to enroll for major 'science' at '12:45:56' hours on day '01/12/2004'*. A different event occurrence is: *student 'V 2345' wants to enroll for major 'science' at '18:45:56' hours on day '03/06/04'*. We can group the former two event occurrences into the following event: *student 'V 2345' wants to enroll for major 'science*'.

An *event* is one or a number of potential happenings in the application subject area that can lead to the execution of one or more derivation rules and/or the insertion or deletion of sentence instances into/from the application's information base.

An *event type* is a class of events in the application subject area, each of these events can lead to the execution of one or more derivation rules (of the same type) and/or the insertion or deletion of sentences (of the same sentence groups(s)) into/from the application's information base.

An *impulse type (constraint)* is an ordered triplet that contains an event type, a condition type[2] under which the occurence of an event of an event type can lead to the execution, of a specified derivation rule or inserte/delete operation and a derivation rule or insert/delete operation. The impulse type constraints explicitly model the temporal relationships between 'happenings' or events in the application subject area and information system events and enforces them upon the derivation rules and information base update operations. In figure 9b we have given the instances of the impulse type constraint: *C17, C18* and *C19* for our extended University Enrollment example.

3.3 The Modeling Procedure in NLM

The most distinguishing feature of the NLM requirements elicitation method is in the existence of explicit algorithms for every requirements elicitation step [5, 6].
In figure 8 we have given an example of such an algorithm that shows how uniqueness constraints can be detected whenever a basic information model is given (for the complete set of algorithms see Bollen [6: p.131-162)]). The bold-fonded part of this algorithm depicts the user input in the RE process. In NLM such algorithms are in principle defined for every activity in creating the basic information model and for every type of population constraint that is defined in NLM.

[2] Including the 'empty' condition type, which means that the occurrence of an event will unconditionally lead to the execution of a derivation rule and or/insert delete operation(s)

```
Algorithm 6. Uniqueness constraint derivation
BEGIN UNIQUENESS((I)BIM ,UoD ,G {(I)BIM is basic information
model that refers to an (integrated UoD)}
WHILE not last sentence group of arity >1
DO take a random sentence instance from a complex sentence
   type template for this sentence group from the example
   UoD: (a1,...., aN): ft (I)BIM
   Take the first role from this sentence group (m:=1)
   WHILE not last role in sentence group
    DO  Create an example sentence where the instance
        of role m is altered. Determine whether the
                   combination of this sentence with the first
        sentence is allowed
        IF the existence of such a sentence is allowed
          together with (a1,.... aN)
        THEN add this sentence to the uniqueness
             significant population
        ELSE  define a uniqueness constraint UC on
              roles {1,...,N}\m of sentence group ft
        ENDIF
     Go to the next role in sentence group (m:=m+1)
   ENDWHILE
         Take next sentence group
 ENDWHILE
 {N-1 law check}.Apply the N-1 law on each sentence group
END
```

Fig.8. Algorithm for detecting uniqueness constraints.

An information model referring to a universe of discourse is a *basic information model* for that UoD together with all *population constraints* that reflect the business rules in that UoD and that can be defined on the roles of the basic information model for that UoD.

In figure 9 the resulting information model for the student enrollment UoD application area is shown.

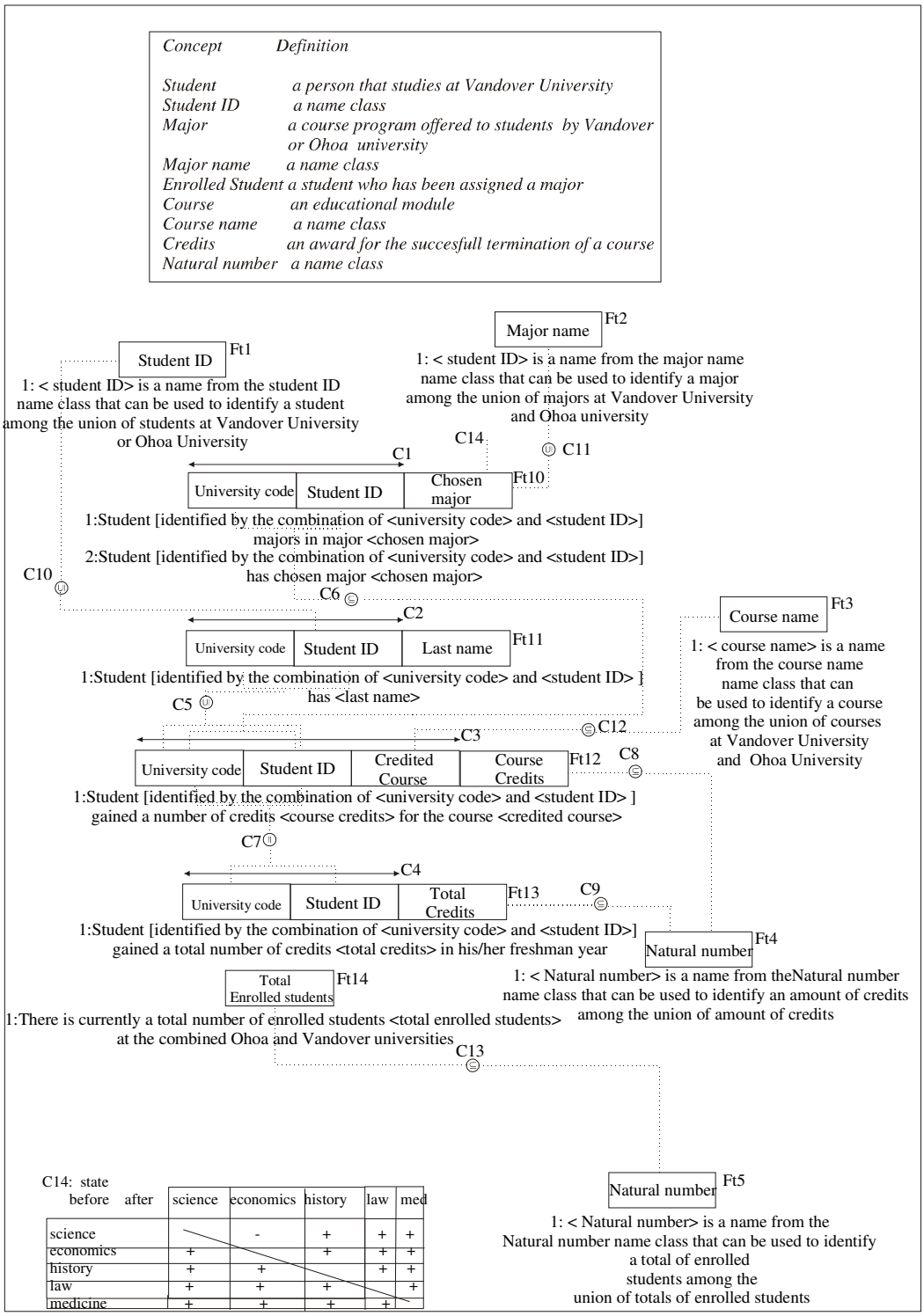

Fig.9 (a). Information model and population constraints for extended university enrollment

```
C15: Create total number of credits<{(arg„student)}>
IF  there exist an instance of FT12
SUCH THAT FT12.<university code>.<student ID>=arg1
THEN create an instance of fact type FT13
     SUCH THAT
      FT13.<university code>.<student ID>:= arg1
     FT13.<total credits>:=DF1
               DF1:=    FT12.<credits> [where FT12.<university code>. <Student ID>='arg1']
ENDIF
```

```
C16: Create total number of enrolled students
IF  there exist an instance of FT10
THEN create an instance of fact type FT14
     SUCH THAT          FT14.<total enrolled students>:=DF2
     DF2:= COUNT(Ext(FT10))
ENDIF
```

```
C17
ON E T2:  Insert(Student'x' wants to enroll in Major 'y') into application data  base has
            succeeded (arg1:'x'; arg 2: 'y')
DO  Create total number of enrolled students
```

```
C18
ON ET3: Delete(Student'x' wants to enroll in Major 'y') from application data  base  has succeeded (arg1:'x'; arg 2: 'y')
DO Create total number of enrolled students
```

```
C19
ON ET1: student requests enrollment in major(arg1: student, arg2:major )
IF[FT13.<total credits>
        (Where FT13.<university code>.<Student.ID>='ET1.arg1')] > 24
AND   [ IF ET1.arg2='science' THEN( mathematics EXT (FT12.<credited
       course>[where FT12.<university code>.<Student.ID>='ET1.arg1'] AND
       FT12.<course credits>[where FT12.<university code> .
       <Student.ID> ='ET1.arg1' AND where FT12.<credited course >
        ='mathematics' ]>8)
                                  OR
      [ IF ET1.arg2='history' THEN( language and culture EXT (FT12.<credited
       course>[where FT12.<university code>.<Student.ID>='ET1.arg1'] AND
       FT12.<course credits>[where FT12.<university code> .
       <Student.ID> ='ET1.arg1' AND where FT12.<credited course >
        ='language and culture' ]>5)
OR
      [ IF ET1.arg2='economics' THEN(macro econ. EXT (FT12.<credited
       course>[where FT12.<university code>.<Student.ID>='ET1.arg1'] AND
       FT12.<course credits>[where FT12.<university code> .
       <Student.ID> ='ET1.arg1' AND where FT12.<credited course >
        ='macro econ.' ]>8)
OR
      [ IF ET1.arg2='medicine' THEN(biology. EXT (FT12.<credited
       course>[where FT12.<university code>.<Student.ID>='ET1.arg1'] AND
       FT12.<course credits>[where FT12.<university code> .
       <Student.ID> ='ET1.arg1' AND where FT12.<credited course >
        ='biology' ]>5)
OR
      [ IF ET1.arg2='law' THEN(biology. EXT (FT12.<credited
       course>[where FT12.<university code>.<Student.ID>='ET1.arg1'] AND
       FT12.<course credits>[where FT12.<university code> .
       <Student.ID> ='ET1.arg1' AND where FT12.<credited course >
        ='finance' ]>5)  ]
DO Insert (student'Et1.arg1' has chosen major 'ET1.arg2').
```

Fig.9 (b). Derivation rule and impulse type constraints for extended university enrollment example

In figure 10 we have given an outline of the 'overarching' NLM modeling procedure that must be followed in an enterprise wide application development program to capture the 'enterprise-wide' business ontology into a list of

definitions, an integrated NLM information model and set of population constraints defined on this integrated model.

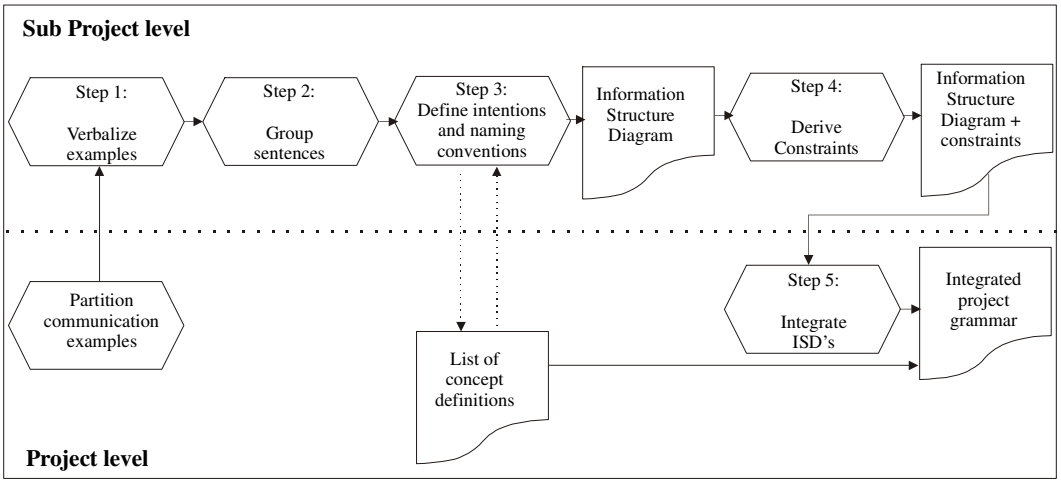

Fig. 10. Outline of NLM modeling procedure

We note that the activities step 1 through step 4 should be applied on a sub-project level. The list of definitions that needs to be maintained for the recording of the concept definitions for the domain, however, must be managed on a project level or should even be considered an enterprise-wide knowledge management tool. Activity step 5 of this modeling procedure is the integration step in which the (partial) information models from the (sub)projects in for example, an ERP implementation program are harmonized and in which the sentence groups and population constraints are merged into an integrated specification. To facilitate this merging process, we recommend to assign specific ranges for sentence group (template) codes and constraint codes to the sub-projects. If, necessary, additional inter-sub schema constraints can be specified.

4. Assessing NLM in terms of the derived criteria for RE approaches

4.1 Conclusions for the way of modeling

The NLM requirements specification language contains only one information bearing construct: the sentence group (template). The introduction of the sentence group template construct and the application concept repository in NLM allows us to capture the complete domain semantics of the UoD as the union of the relevant sentence groups and the accompanying *population state-*, *population state transition-*, *derivation rule-* and *impulse type constraints*. NLM therefore fulfills the *completeness* requirement to the highest possible extent. It was also shown

that NLM leads to requirements specifications that can easily evolve with changing application domains because a change in business rules can be easily accommodated by deleting/replacing an existing constraint or adding a new constraint to the NLM requirements specification. The way of modelling in NLM provides an advantage in terms of its modelling efficiency in comparision to traditional requirements specification modelling languages like for example (extended) entity-relationship modelling [11, 25] and the structural diagrams in UML [23] mainly due to the absence of multiple fact-encoding modeling constructs.

4.2 Conclusions for the way of working

The application of algorithms: *verbalization, grouping, classification and qualification* and *atomization* in NLM (e.g. see [6: p.131-145]) will lead to the detection of all semantic relationships and naming conventions in the application subject area. The application of the static constraint derivation algorithms: *uniqueness constraint derivation* and *set comparison constraint derivation* (see [6: p.152-156]) will lead to all uniqueness and set comparison constraints that govern the application subject area. In order to derive all instances of the dynamic constraints NLM has specified the *transition constraint derivation* algorithm in NLM's way of working [6: p.157-158]. In order to derive all instances of the derivation rule constraints NLM has specified the *derivation rule constraint* algorithm in which the precise specification (or derivation formula) can be established [6: p.159-160]. In the *impulse constraint* derivation algorithm [6: p.161-165] the question in which an internal event can lead to the execution of a derivation rule or another information base event is incorporated. Furthermore, this algorithm systematically confronts the users with derivation rules and tries to elicit the potential 'external' events that might invoke such a derivation rule. In the *integration of basic information models* algorithm, a view integration algorithm has been defined [6: p.149-151].

The application of the natural language axiom in an organizational setting in which domain users are enabled to make implicit knowledge, explicit allows us to apply NLM in many organizational settings, ranging from abstract to tangible UoD's and from natural language descriptions to other descriptions that can only be understood by users.

The sub-division of the modeling procedures in NLM's way of working into a number of formal algorithms has been done in such a way that the amount of analysis steps that have to be performed by (an) analyst(s) is minimized. The precise specification of the NLM modeling procedure in a number of algorithms with built-in formal quality assurance checks and external validation steps complies to the definition of formality for the way of working.

The way of working in NLM provides an advantage in terms of its completeness, formality and efficiency in comparison to traditional requirements specification modelling approaches (extended) entity-relationship modelling [11, 25] and UML [23] mainly due to the presence of modelling algorithms.

4.3 Conclusions for the way of controlling

The way of working in NLM has a work breakdown structure that consists of 5 activities or transformations that are laid down as (sets of) formal algorithms and therefore can be formally planned as activities in a requirements determination project. Furthermore, NLM contains provisions that enable traceability in the requirements determination processes, by forcing an analyst to use naming conventions for the concepts that he/she uses in the process of requirements elicitation. The *reconstruction* check in the *verbalization* algorithm, the *completeness* check in the *grouping* algorithm, the *consistency* check in the *classification and qualification* algorithm , the *reference* check in the *atomization* algorithm, the *ontological equivalence* check in the *integration* algorithm, and the *N-1 law* check in the *uniqueness constraint derivation* algorithm are explicit quality-assuring verification sub-procedures that are built-into NLM's modeling procedure. In table 5 we have summarized the significant properties of NLM in the light of the 4 (sets of) criteria that were given in [7].

Table 5. NLM's compliance to the criteria from [7].

Criterion	Way of Modeling	Way of Working	Way of controlling
Domain richness		The NLM requirements. Elicitation approach can handle different perceptions on an application domain by recording the different perceptions in different sentence group templates. The NLM approach allows for 1-on-1 adding and/or deleting of business rules in a turbulent and explicit (to tacit) application environment. NLM supports initial specifications ranging from verbalizable 'real-life' examples to requirements expressed in natural language.	
Completeness	The modeling constructs in NLM cover all relevant conceptual perspectives and types of rules of an application area.	The algorithms defined within the context of the NLM modeling procedure will always guarantee that all instances of the constraint types that are known to exist in the application domain, will be captured. The algorithms can be applied on an incomplete RS.	
Efficiency	A RS expressed as a NLM information diagram is a well organized set of specifications in which sentence group (templates) and roles constitute the basic model. The different constraint types can be defined as propositions on the roles in the sentence groups.	The easy executable algorithms will always lead to the minimum required number of modeling steps. The algorithms allow for easy maintenance of RS.	The NLM approach has a clear demarcation of stages in the RE process in which the deliverables are clearly defined. Furthermore each sub-procedure in NLM has built-in quality checking procedures
Formality	The definition of the modeling constructs that encode constraints is fully consistent with the unambiguous definition of the modeling constructs for the basic information model	The NLM modeling (sub)-procedures are expressed as formal algorithms that transform the pre-defined input document into an pre-defined type of output document and contains provisions for validation by end users	The formal NLM modeling procedure assures traceability and correctness of the specifications in which in-between results are verified by the user

5. Empirical validation of NLM

The NLM approach and its predecessors have been applied in student's masters projects [4, 12, 13, 29] and in many RE projects in large organizations, e.g the ABP pension fund [24] and the dutch railways corporation [26]. A 'sugarized' version of NLM: *kenniskunde* [22] is used in a number of curricula for systems engineering [2] and as a 'learning accelerator' for (in-company) education programs on a poly-technic level [3]. In all these environments users and analysts were asked to give a first judgement on the applicability of our NLM-method. Although no formal interviewing and hypothesis testing has been done, the majority of them indicated that NLM was easy to use and was perceived as a quality method compared to "traditional" methods. Further research is needed to back up this limited empirical evidence.

6. Conclusions

Quality criteria from the literature were synthesized into a coherent and consistent set of quality criteria for requirements elicitation methods. There are four criteria *completeness, efficiency, formality and domain richness* that apply to three aspects of RE, being the way of modeling, the way of working and the way of controlling.

Traditional RE methods (EER, ORM) only partially comply to these quality criteria [6, 7]. UML does comply to the completeness criterion for the way of modeling, but lacks compliance to the efficiency and formality criteria for the way of modeling and the way of controlling.

Theoretically NLM complies to these quality criteria, because our RE method based on natural language modeling comprises the required set of modeling constructs and accompanying procedures

Limited empirical evidence supports this hypothesis, however further research is needed to make this statement methodologically significant.

References

1. Abrial, J. Data Semantics. In: Klimbie, J., Koffeman, K. (eds.): *Data Base Management*, North Holland, Amsterdam, (1974),1-59
2. AMBI. http://www.academicservice.nl/onderwijs/OnderwijsSeries/AMBI.jsp , visited on april 21st 2005. (2004)
3. Bastiaens, E., Nijssen, S., Manintveld, J. Werken aan behoud van kennis en behalen van een HBO-diploma in één. *Opleiding & Ontwikkeling 1 /2* (2005), 28-30 (in dutch)

4. Bogget, M. *Implementation of a Management Reporting System & Preparation of MRP model at PTZ Nelahiozeves Unilever*. Final Thesis Business Economics. University of Maastricht, (1994)

5. Bollen, P. The Natural Language Modeling Procedure', in: A. Halevy and A. Gal (eds.). proceedings *Fifth Workshop on Next Generation Information Technologies and Systems. Lecture Notes in Computer Science 2382*. Springer-Verlag Berlin, (2002), 123-146

6. Bollen, P. *On the applicability of requirements determination methods*. Ph.D thesis, faculty of Management and Organization, Rijksuniversiteit Groningen. (2004) (can be downloaded from http://www.ub.rug.nl/eldoc/dis/management/p.w.l.bollen/)

7. Bollen, P., Simons, J. *Quality criteria for Requirements Elicitation methods*. research memo. Faculty of Economiocs and business Administration. University of Maastricht. (2005)

8. Brasethvik, T., Gulla, J.A. Natural language analysis for semantic document modeling. *Data & Knowledge Engineering 38,* (2001), 45-62.

9. Browne, G., Rogich, M. An empirical investigation of user requirements elicitation: comparing the effectiveness of prompting techniques. *Journal of Management Information Systems, 17*, 4 (2001) ,223-249

10. Byrd, T., Cossick, K., Zmud, R. A synthesis of research on requirements analysis and knowledge acquisition techniques. *MIS Quarterly*, (march 1992), 117-138.

11. Chen, P. The entity-relationship model: Towards a unified view of data. ACM *Transactions on Database systems,* 1 (1) (1976), 9-36

12. Clayes, C. *Final thesis*. International business studies. University of Maastricht, (1996) (in dutch)

13. Enter, N. *The semantics of the CIC SAP R/3 core*. Final thesis. International business studies. University of Maastricht, (1999)

14. Falkenberg, E. Significations: the key to unify data base management. *Information Systems, 2*, (1976),19-28

15. Halpin, T. *Information modeling and relational databases: from conceptual analysis to logical design.* Morgan Kaufmann, (2001)

16. Halpin, T., Evans, L., Hallock, P., macLean, B. *Database modeling with Microsoft visio for enterprise architects,* Morgan Kaufmann (2003)

17. Halpin, T., Orlowska, M. Fact-oriented Modelling for Data Analysis. *Journal of Information Systems, 2*, (1992), 97-118

18. Hevner, A., Mills, H. Box-structured requirements determination methods. *Decision Support Systems, 13*, (1995), 223-239

19. Hickey, A., Davis, A. A unified model of requirements elicitation. *Journal of Management Information Systems*, *20*, 4, (2004),:65-84

20. Nijssen, G. On the gross architecture for the next generation database management systems. In: B. Gilchrist (ed.) *Information Processing 1977*, IFIP. (1977), p. 327-335.

21. Nijssen, G., Halpin, T. *Conceptual schema and relational database design: A fact based approach*, Prentice-Hall, Englewood Cliffs, (1989)

22. Nijssen, G. *Kenniskunde 1A*. PNA Publishing, Heerlen, (2001)

23. OMG. UML 2.0 superstructure specification. (2004) http://www.omg.org/cgi-bin/doc?ptc/2004-10-02, visited on 1 september 2005.

24. Spijkers, P. A manager's experience with NIAM-ISDM in a large scale critical project. in G.M. Nijssen and J.Sharp (eds.), *Proceedings second NIAM-ISDM working conference*., (1994), A1-A27.

25. Teorey, T., Yang, D., Fry, J. A logical design methodology for relational databases using the extended E-R model. *ACM Computing Surveys* , 18 (2), (1986), 197-222

26. Twisk,F. How to make an elephant dance or; Putting good theory into practice, *Proceedings of the NIAM-ISDM 1993 conference*, (1993)

27. Verheijen,G., van Bekkum J. NIAM: An Information Analysis Method. In: Verrijn-Stuart,A., Olle T., Sol H., (eds.): *Proceedings of IFIP TC-8 CRIS-1 conference*, North- Holland Amsterdam, (1982), 537-590

28. Wetherbe, J. Executive Information Requirements : getting it right. *MIS Quarterly*, *15*, 1 (1991), 51-65

29. Wolthuis, M. master thesis: *Boekhouden op basis van Universele Informatiekunde*. University of Limburg. (1997) (in dutch)