From a File-Oriented View to an Object-Oriented View

P.J.M. Frederiks, Th.P. van der Weide

Department of Information Systems, University of Nijmegen
Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands
{paulf,tvdw}@cs.kun.nl

Published as: P.J.M. Frederiks and Th.P. van der Weide. From a File-Oriented View to an Object-Oriented View. Technical Report CSI-R9601, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, January 1996.

Abstract

The last three decades the architecture of information systems has evolved from file-oriented, via data-oriented and communication-oriented towards an object-oriented view. Hand in hand with this architectural evolution the way a user communicates with the information system is changed. In this paper we discuss the relation between the different architectures and their associated man-machine communication. We introduce the concept of information grammar, and show that this grammar can be seen as a common point of convergence. Together with the evolution, we can identify an evolution of methods for information system development. The construction of an information grammar is also addressed in this paper.

Keywords: object-orientation, natural language, information grammar, verbalization, validation, information system architecture, man-machine communication

Classification: *AMS-1991* 68P15, 68P99, *CR-1991* H.1.2, H.2.1

1 Introduction

The evolution of information systems has been shifted from machine-oriented into more and more human-oriented approaches. Rather than focusing only on the structure of the information, the way the information is used to capture the pragmatics of the communication language within the application domain is discussed. This is reflected by a change of the architecture, man-machine communication and analysis methods. Central for all these aspects is the so-called information grammar, This grammar is the basis for all communication with the information system but also describes the structure of all information to be processed.

In this paper we first introduce information grammars in section 2. In section 3 we discuss the evolution of information system architectures. Section 4 contains an overview of a way of working to build the information grammar. Finally we give a number of conclusions in section 5. Appendix A contains a small example of an implementation of an information grammar.

2 Information Grammars

This section discusses the nature of information grammars. Furthermore, some related work, with respect to the linguistic object-oriented information modeling approach, is addressed.

2.1 Underlying perspective

Almost all modern modeling techniques have as point of departure a description in natural language of the world to be modeled. This description is referred to as the *initial specification*, and is provided by so-called *informed users* (domain experts). Ideally, this initial specification is a precise specification from which system developers can derive the required information system.

Using natural language for problem specification is not new in the field of computer science. Already in the late fifties COBOL was introduced as a semi-natural programming language in order to allow programmers to build systems in their 'own language'. In the early seventies syntactic-oriented programming methods, like step-wise refinement, were introduced, see e.g. [Dij76, Mee78]. The step-wise refinement method verbalizes the problem top-down using simple control structures like `IF THEN ELSE FI`. Also a natural language approach to information modeling is not new. The conceptual data modeling techniques NIAM [DO90], EER ([EGH+92], [BCD+95]) and PSM [HPW94, CW93] are based on such an approach. For these techniques the goal of the modeling process is to derive the grammar that governs the communication within the application domain (Universe of Discourse, UoD for short), the so-called *information grammar*. This grammar can be depicted as an information structure diagram. The verbalization of the elements of the information structure diagram forms the base for the terminal symbols for this grammar.

A major advantage of the linguistic approach is that (intermediate) results can be validated by the informed user readily, as each intermediate result corresponds to a (partial) grammar for sentences in the language spoken in the UoD. The rationale behind the linguistic approach to information modeling is a scheme of base axioms describing the cognitive identity of domain experts and system analysts. Base axioms as such fall outside the scope of this paper, for more information see [Nij89], [Win90] and [FKW95].

2.2 Related work

For reasons of readability an overview of the related work presented in this section is divided in two parts. The first section describes some related work with respect to the analysis of informal specifications, whereas the second section gives information about some related work concerning the verbalization part.

2.2.1 Analysis of informal specifications

The linguistic approach to object-oriented modeling has been introduced before. For the OOA method ([CY90]) a simple linguistic approach is described, which only considers *nouns* and *verbs*. Although this method offers the system analyst not more than a rule of thumb, it is still very useful for the determination of the global structure of the system. The analysts can find object types and action types with this way of working.

In the SACIS method (see e.g. [Gra94]) the verbs are further refined to: *doing verbs* (expressing an action type), *being verbs* (expressing a classification), *having verbs* (expressing composition), *modal verbs* (expressing conditions) and *stative verbs* (expressing an invariance-condition). The nouns are categorized to *proper nouns* which stand for instances (population) and *improper nouns* which can be used for the determination of classes and object types. Besides these two refinements the SACIS method provides grammatical views for attributes, operations, associations and events.

The KISS method ([Kri94]) is a further refinement, even though some parts of speech which are considered in the SACIS method are left out, because it allows for a deep analysis of the informal specification. Besides finding object types, via direct objects and indirect objects in the sentences, the *connection* and *direction* between object types is investigated focusing on verbs (predicates) and prepositions. *Adjectives* and *adverbs* are used to recognize properties (attributes) of object types and action types. A real novelty is the use of *gerunds*. A gerund is a substantive noun that is derived from the infinitive form of a verb by suffixing this initial with '-ing'.

All three methods are using in some way grammatical analysis of the informal specification but lack a verbalization mechanism. The way of working of these methods, during analysis, consists of collecting lists of candidate object types, action types, etc.

2.2.2 Verbalization of models

As pointed out in section 2.1 verbalization mechanisms are very useful for validation purposes. A first attempt for both the grammatical approach to an informal specification as well as the verbalization of conceptual schemata for requirements engineering is proposed by [RP92]. By recognizing different types of linguistic patterns the grammatical analysis is carried out. For the generation of natural language out of a conceptual schema the *Chomsky theory* [Cho65] is used. The basic Chomsky assumption is the existence of an underlying structure, to any sentence in any human language. In addition, there is an infinite number of ways, namely the surface structures to represent the deep structure in different languages. The deep structure expresses semantics of a sentence by means of semantic elements and relationships among them. The proposed framework is supported by an expert design system, known as OICSI (French acronym for intelligent tool for information system design). Note that this approach is not particularly focused on object-oriented analysis. Furthermore, as a result of using the Chomsky theory this approach has a semantic nature.

A similar, and as the authors ([MG94]) say less sophisticated, tool is LOLITA which supports a linguistic approach to developing object-oriented systems. This approach is also semantically based as LOLITA uses a data dictionary to find related classes, or closely related, for the classes found in the informal specification. Besides this feature, LOLITA is able to automatically identify ambiguities, inconsistencies, correcting misspellings and guess new words.

The LIKE project (Linguistic Instruments in Knowledge Engineering) has resulted in a method called COLOR-X which is an abbreviation for COnceptual Linguistically based Object-oriented Representation language for Information and Communication Systems (where ICS is abbreviated to X).The static and dynamic part of object-oriented modeling is described in [BR95b] and [BR95a], respectively. The informal specification is captured in graphical models which can be translated to an intermediate language called Conceptual Prototyping Language (CPL). CPL ([Dig89]) is a formal modeling technique, based on *functional grammars* ([Dik89]), which can be used for specification as close as possible to the informal specification. As a result the verbalization of the graphical models is based on CPL.

The main difference between the above mentioned approaches and ours is that they are semantically based whilst ours is syntactically based. In other words, the meaning of the words (in their context) of the informal specification is taken into account, using lexicons like Wordnet, while our approach is focused on particular parts of the sentences, e.g. nouns, verbs, etc. Another difference is that our approach models the *communication* between user and information system instead of modeling the information system. An advantage of this approach is that the resulting information grammar can be used as a base for the query language of the user.

3 Architectures

In this section we discuss several typical architectures for information systems. Coupled on these architectures, we focus on man-machine communication. Before we discuss a general architecture of an object-oriented information system the evolution of information system architectures is discussed shortly. This discussion is partially adopted from [Bub86] and is augmented with our own view on information system architectures.

3.1 File-oriented architecture

Figure 1 shows the view on an information system in the sixties. An information system consisted of a collection of application programs, i.e., data definitions and procedures (non-transparent), communicating with each other by passing files.

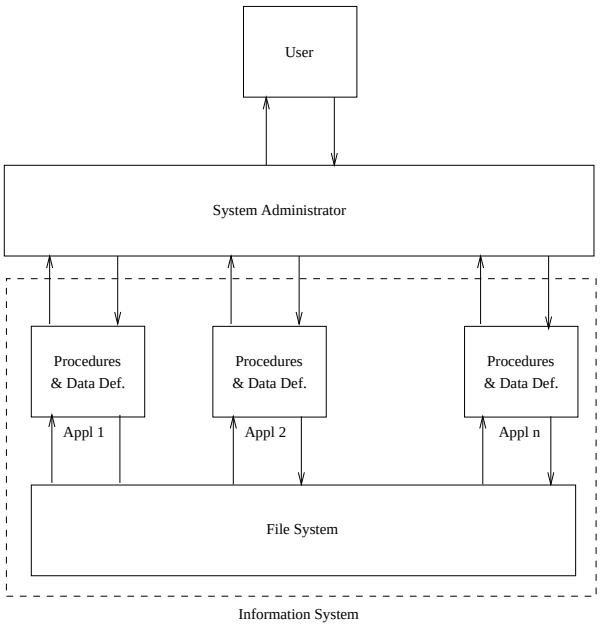

Figure 1: A file-oriented architecture of an information system

The operation mode of such an information system was mainly batch processing, usually dominated by a *system administrator*. The task of the system administrator can be summarized as to administrate and guarantee the system at an operational level. The input data for a batch run were gathered in a file (set of cards), the results consisted of files and printed output. The format of data files was rather straightforward. It was the responsibility of the end user to transform real data into this strict format, and to interpret the resulting data within the application domain. The communication between man and machine thus was mainly on the side of the machine, the human had to adapt to the machine.

Building an information system concentrated around the construction of the programs in the collection. In other words, the programs had a central place, the data definition was of secondary importance ([Dat91]). For the construction of such systems thus the techniques for program construction were used (such as step-wise refinement).

3.2 Data-oriented architecture

In the seventies the view (figure 2) on information systems became data-oriented and was dominated by the information systems data definitions. The essential difference with the file-oriented architecture is the separation of data definition (in data bases) and data processing (in application programs). This separation was achieved by explicit data definitions in data base schemata at several levels: *external*, *conceptual* and *internal* ([Dat91], [Gri82]). Another difference with the file-oriented architecture is the absence of a system administrator. However, in the data-oriented architecture a database administrator is responsible for the structural data aspects on all levels. The database administrator also governs authorization aspects via the external user views.

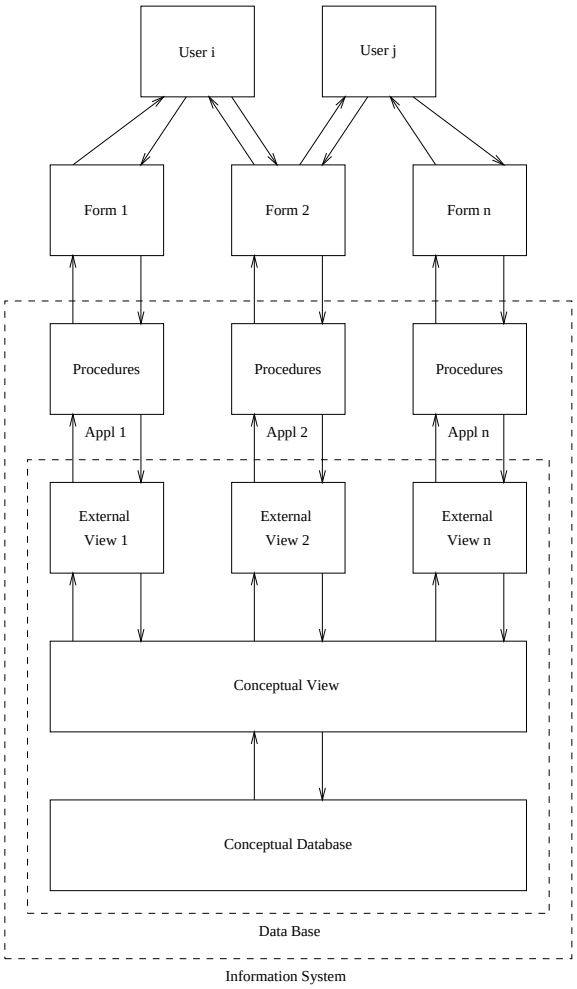

Figure 2: A data-oriented architecture of an information system

Note that the data-oriented architecture is not the same as presented in [Dat91]. In our architecture the aspects of user communication are also involved, while details at the level of data storage are omitted.

The communication in the data-oriented architecture typically is guided by filling (digitized) form. Forms are a very widespread communication mechanism within organizations and therefore very acceptable for human beings. In [PSB+94] it is argued that form filling is suitable from an

ergonomic point of view. In practice forms are usually very well designed, and close to the mental model of the (intended) users ([Sch92]).

In file-oriented architectures, it was the responsibility of the user to translate the data from (real) forms into the format described by the data files. In the data-oriented architecture, real forms are mapped onto digitized forms, enabling the user to simply fill in this form. The process of data extraction from the form now is performed by the computer rather than the user. As a result, the man-machine communication is more directed towards the human being.

As a result of this approach, a separation of concerns is possible. Basically, an application program is divided into the following parts:

- the interface with the user.
- the interface with the data base.
- the processing in terms of these interfaces.

Central is the construction of the data models, which are not dependent of any application program. Several methods and techniques have been designed to support this construction process, e.g. relational model ([Cod70]) and ER ([Che76]). Special utilities are usually available to build forms. However, most of them do not support the designer at the level of making well-designed forms.

3.3 Communication-oriented architecture

During the eighties the data-oriented architecture was still often used. However, this view was augmented by all kinds of fourth-generation language mechanisms (query language facilities, form definitions and processing definitions, report generators, dictionaries, all kinds of graphics and spread-sheets, etc.). The role of the information system changed from a dedicated system into a general purpose system. The system now keeps track of the definition of the data, and also administers the application programs (the *procedures base*) and the actual forms (the *forms base*).

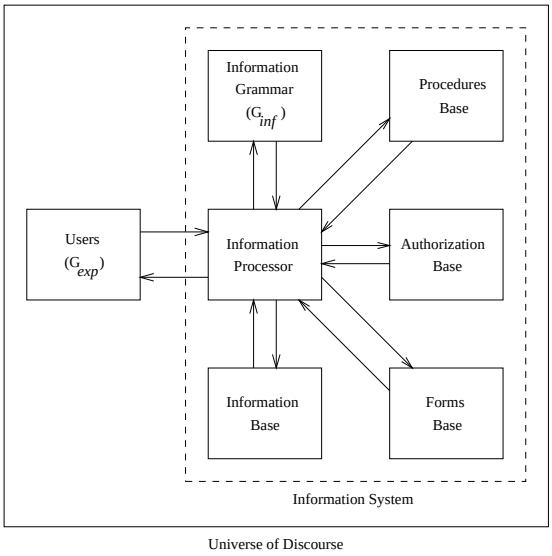

Figure 3: A communication-oriented architecture of an information system

A general communication-oriented architecture for information systems is depicted in figure 3 (adapted from [Win90]). In this architecture the user sends update or retrieval requests, in accordance with a certain information grammar, to the *information processor*. Each request is checked against the user's access right from the *authorization base*. If a user is authorized to perform a request, the information processor performs the request on the *information base*. In [BW94] this topic is discussed extensively.

The communication-oriented architecture offers the user much more freedom for manipulation and interpretation of data. Due to the extra facilities, the system now is capable of providing the user with interpreted data for example in the form of a pie chart. Therefore the communication with the information system became the main topic for improvement. A typical communication-oriented system is an SQL system. The communication between human and machine is in terms of the elements that describe the application domain, according to the syntax rules of SQL. In order to be able to communicate with the system, the user should know:

- vocabulaire: the names of tables and their columns,
- grammar rules: the syntax rules of SQL.

These components are part of the information grammar. Experience has shown that users have difficulties formulating SQL statements. There are two main reasons:

1. The syntax rules for SQL are too complex, especially their recursive nature.
2. The relation between a relational model and the corresponding application domain is usually not clear for users.

The first problem may be solved by allowing a semi natural language format for the query language. An example of such a language is Lisa-D ([HPW93]). The second problem is addressed by conceptual modeling techniques such as NIAM ([NH89]), PSM ([HW93]) and FORM ([HO92]).

Data modeling techniques such as NIAM, PSM and FORM focus on the communication within the application domain rather than the required data definition. As they model the communication, (partial) results can be validated by verbalization end paraphrasing. Furthermore, many methods and techniques are supported by case tools and case shells ([VH95]).

3.4 Object-oriented architecture

In an object-oriented approach, the centralistic view of the communication-oriented architecture is completely modified (see figure 4). The information system will now provide the user with the opportunity to address objects more or less directly. The information system consists of subsystems (complex objects) which communicate by messages. Each such communication is governed by a corresponding local communication grammar. The local communication grammars together form the (global) information grammar.

In figure 4 subsystem i sends messages to subsystem j according to grammar $\mathcal{G}_{ij}$. Subsystem j responds with messages to subsystem i according to grammar $\mathcal{G}_{ji}$. A subsystem in its turn may consist of a number of other subsystem. The smallest subsystem is an (elementary, i.e. non-composed) object. The information system is a (possible infinite) network of objects.

If we take a closer look at the subject/object communication, see figure 5, we see that the user (a so-called *subject*, i.e. an object outside the information system) makes update and retrieval requests by sending corresponding messages to an appropriate object. Just like the communication between subsystems, the communication between subject and object is governed by a pair

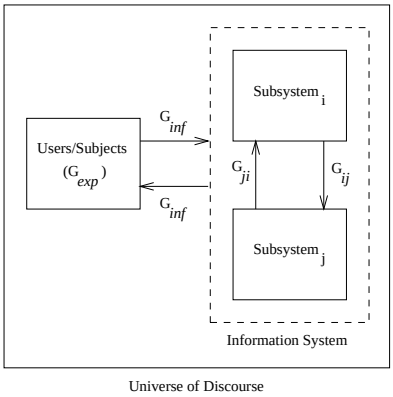

Figure 4: An object-oriented architecture of an information system

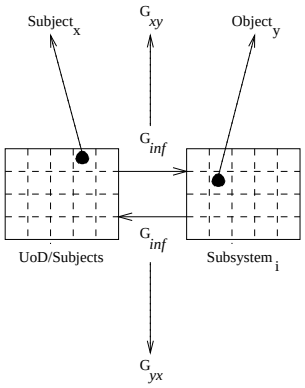

Figure 5: Subject/object communication

of grammars $\langle \mathcal{G}_{xy}, \mathcal{G}_{yx} \rangle$ which is suitable for the subject and the object respectively. An authorization mechanism is therefore not explicitly mentioned as the authorization is enclosed in the subject/object grammars.

A subject activates objects to perform the corresponding actions, or creates new objects for this purpose in correspondence with its communication grammar. Activated objects may, in turn, send new requests in accordance with some object/object communication grammar (see figure 6) to other objects. The result of such a request to another object is interpreted by the object, which was activated by the subject, and communicated to the user. The strict separation between data

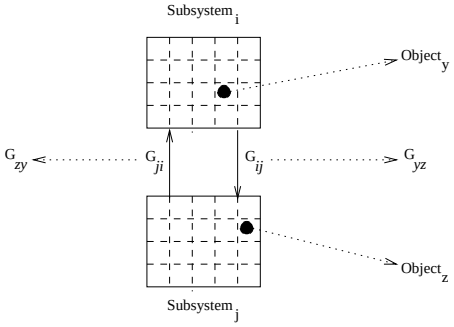

Figure 6: Object/object communication

and control structures is dropped in an object-oriented architecture of an information system and is replaced by a strict encapsulation of data and operations related to an object type.

In the object-oriented architecture, the communication between human and machine becomes a communication between subjects and objects. An advantage over the communication-oriented approach is that the object-oriented approach provides the opportunity to differentiate between cases of communication. For example, a jurist may address the information system quite differently from a car seller. There is no a-priori need that they know their language, but they may still want to communicate with the same information system.

In [Joh95] the author identifies related subsystems using Galois connections. In order to verify that the proposed correspondences between subsystems are consistent with the semantics of the conceptual schemas, the modeling formalism utilizes linguistic instruments. The instruments used are in particular *case grammars* and *speech act theory*. The modeling formalism described in this paper can be used to support schema integration.

The techniques of the communication-oriented architecture are still valid in this case. However, this will lead to a number of local grammars, that have to be integrated to form the global information grammar. For such integration there seems to be no good solution at the moment. This requires an integration between OOA techniques ([FKW95]) and cognitive models ([HVV91]).

4 Building the Information Grammar

As stated in [Wij91] the philosophy behind a method does affect the way of working and the way of modeling of this method. Traditionally, three aspects of a (natural or artificial) language are distinguished: *syntax*, *semantics* and *pragmatics*. Our approach is to exploit syntactical aspects as much as possible, although some semantics may be necessary for the modeling process. The domain expert is held to be responsible for the semantic part.

The way of working during the analysis phase can be divided in the following stages:

1. Collect significant information objects from the application domain.
2. Verbalize these information objects in a common language (expert language formulations).
3. Reformulate these formulation in a unifying format (reformulated expert language formulations).
4. Abstract rules and facts from reformulated expert language formulations (information grammar and population).
5. Interpret and visualize the information grammar (analysis models).
6. Verbalize analysis models.

These stages are repeated until the verbalization of the formal specification (the analysis models) matches the informal specification. Stage 1 and 2 of this way of working are performed by the domain expert solo. The reformulation of the expert language formulations, stage 3, is based on interaction between analyst and domain expert. The tasks of stage 4 and 5 are performed by the analyst, who is appointed for this job mainly! Finally, the verbalization phase in stage 6 is performed by the system analyst and validated by the domain expert. Figure 7 resumes the analysis phase schematically. Each stage of this analysis phase is marked with the number associated above.

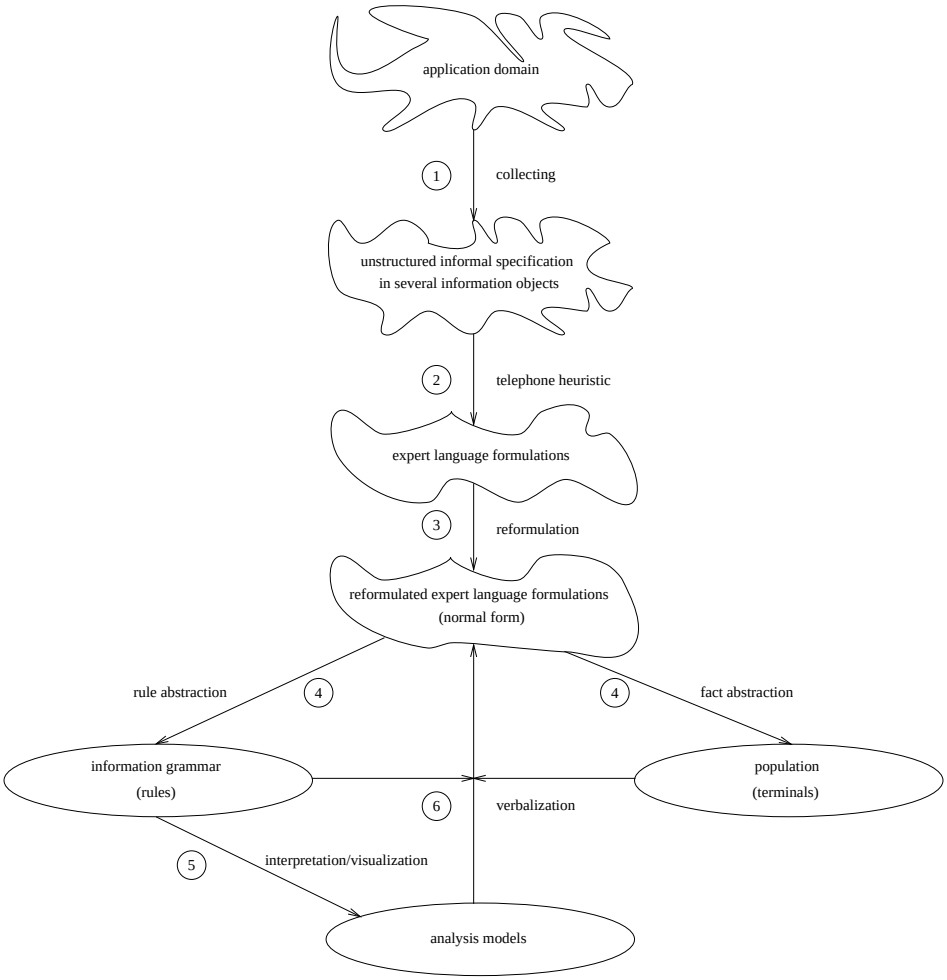

Figure 7: Overview of the analysis phase

4.1 Verbalizing information objects

The communication in the universe of discourse may employ all kinds of information objects, for example text, graphics, etc. However, a textual description serves as a unifying format for all different media. The conversion to a textual description can be done using the *telephone heuristic* ([NH89]):

> *Explain your observations to a non-expert via a telephone.*

The verbalization principles discussed in [FKW95] require the ability to express the information via readable sentences. Other ways of visualizing may also be employed. For example, graphic techniques have shown to be most useful for representing the structure of an application domain, e.g. ER and NIAM. For the purposes of this paper, we will restrict ourselves to textual representations.

The language which results after the telephone heuristic is called the *expert language*. The underlying grammar is referred to as $\mathcal{G}_{\text{exp}}$. The information modeling problem then can be described as:

> *Find, within a certain class of sufficiently efficiently computable grammars, a grammar* $\mathcal{G}_{\text{inf}}$ *which best approximates* $\mathcal{G}_{\text{exp}}$.

Note that this problem is known to be NP-complete. The quality of the information grammar $\mathcal{G}_{\text{inf}}$ can be measured by:

1. *completeness*:
   each sentence from the expert language can be expressed in terms of sentences from the information grammar.
2. *naturalness*:
   each sentence from the information grammar has a meaning within the UoD.

The completeness property guarantees that each expert sentence can be communicated to the information system. If the sentence is not part of the information grammar language, then the user has a formulation problem. It is expected that the formulation problem in this approach is smaller than when using 4th generation languages as SQL.

The information modeling procedure can now be formalized:

> *An information modeling procedure M can be seen as a mapping between grammars, such that $M(\mathcal{G}_{\text{exp}}) = \mathcal{G}_{\text{inf}}$.*

A first restriction, called the *stabilization requirement*, is that the procedure is stable:

$$M(M(\mathcal{G}_{\text{exp}})) = \mathcal{G}_{\text{inf}}$$

4.2 Unifying format

The main goal in object-oriented analysis is to find the objects and their *responsibilities*, i.e. *which* object type is responsible for *what* action type. In a next stage the properties of the object types are investigated and the order in which action types occur is analyzed. By activating the expert language formulations the analyst can find by parsing these sentences, object types and their corresponding responsibility for performing action types. Or, in more technical terms, activated sentences give clues for object classes, the methods and the way they are communicating by messages. For example, the sentence *a song is recorded by a band*, is an example of a passive sentence. This sentence can be rewritten into the following active sentence: *a band records a song*. However, there are sentences which cannot be activated, like *a band has a name*. These sentences are nevertheless very useful to find candidate properties.

One more type of sentences may be considered. These are the sentences containing *is a*, like: *a musician is a person who plays an instrument*. From such sentences an initial class hierarchy may be deduced. In many cases, such abstractions have to be elicited from domain experts by the system analyst.

The order in which action types occur can be deduced from the connections between the sentences of the reformulated expert formulations. Whereas the other reformulations are focused on one sentence, this step requires insight in the total specification. An example is the following fragment of an informal specification:

> *... After a band has been set up, persons may join this band. ... Several bands have the opportunity to record some of these songs. ...*

From the example we learn that the action *to record* is preceded by an action *to set up*.

4.3 Rule and fact abstraction

The task from the analyst is to recognize two different aspects from the expert language formulations:

1. What are the general *rules*? In this step the structure of the information system is set up. As stated in the previous section the object types and their responsibilities are collected.
2. Which *facts* are important? In this step instances of the properties of object types are collected.

As an example consider the fact: *the band Rolling Stones records the song Paint it black.* This fact has as rule *band records song.* The rules found in this step are the production rules of information grammar. Facts also provide clues for a *sample population* of the information system. This sample population is useful for validation.

4.4 Communicating the information grammar

The important parts of the information grammar (i.e. nouns) can be depicted using all kinds of graphical models, called *analysis models.*

In order to communicate optimally, and to validate the analysis models properly, with the informed user a verbalization mechanism for the analysis models is necessary. For this verbalization mechanism the information grammar (the production rules) and the population is used. The verbalization mechanism must be able to produce sentences spoken by the domain expert.

Appendix A shows an implementation of a small information grammar within the AGFL formalism ([Kos91]). The Grammar Workbench ([DKNZ92]) can be used to generate sample sentences randomly according to this grammar. It is also possible to generate a parser for the information grammar. Both the GWB and the parser generator (GEN) are useful tools for verbalization and validation of the information grammar.

5 Conclusions

In this paper we presented a small survey of the evolution of architectures, man-machine communication and development methods in the context of information system specification. During this evolution the man-machine communication has come more and more in prominence. As a result development methods shifted from a technical line of approach (*how* to build an information system) to a communication-oriented point of view (*what* communication is desired with the information system). A central role in this shift of views has been played by the information grammar. Both the philosophy behind this grammar and a way to build this grammar has been presented.

The way to (partially automatically) build an information grammar is a first topic for more research. Combining both parsers (syntax-driven) and lexicons (semantics-driven) seems to be a promising way to achieve the information grammar. On the other hand the models to build this information grammar should be reconsidered. In [FW96] a closer inspection of analysis models will be presented. The growing complexity of man-machine communication requires more advanced validation mechanism. In [DFW96] more experience will be gained by using the Grammar Workbench for validation of the information grammar.

Acknowledgment

The authors would like to thank Patrick van Bommel and Kees Koster for their constructive criticism and suggestions on earlier versions of this paper. Furthermore, we like to thank the other members of the Senter project, i.e. Hogeschool Heerlen, KISS B.V., Open Universiteit and Shell Research B.V..

A Implementing the Information Grammar

Consider the following part an information grammar as presented in [FKW95]:

$$
\begin{aligned}
\langle \text{to join} \rangle &\longrightarrow \langle \text{to join} \mid r \rangle \mid \langle \text{to join} \mid i \rangle \\
\langle \text{to join} \mid r \rangle &\longrightarrow \langle \text{Musician} \rangle \text{ joins } \langle \text{Band} \rangle \\
\langle \text{to join} \mid i \rangle &\longrightarrow \langle \text{Band} \rangle \text{ is joined by } \langle \text{Musician} \rangle \\
\langle \text{Band} \rangle &\longrightarrow \text{The Rolling Stones} \mid \text{The Beatles} \\
\langle \text{Musician} \rangle &\longrightarrow \text{Keith Richards} \mid \text{John Lennon}
\end{aligned}
$$

This grammar is translated into a two level grammar in the AGFL formalism as follows:

```
inv :: RESP ; INV.
start: TO JOIN (inv).
TO JOIN (RESP): MUSICIAN, "joins", BAND.
TO JOIN (INV): BAND, "is joined by", MUSICIAN.
BAND: "The Rolling Stones" ; "The Beatles".
MUSICIAN: "Keith Richards" ; "John Lennon".
```

GWB in this case may generate randomly the following sample sentences:

```
Keith Richards joins The Rolling Stones
John Lennon joins The Beatles
The Rolling Stones is joined by Keith Richards
John Lennon joins The Rolling Stones
```

Note that this set is not complete, i.e. it does not contain all possible sample sentences. As syntax rules might be recursive, GWB is equipped in such way that it avoids coming into an infinite loop. It is also possible to generate a parser for the information grammar using the parser generator (GEN). In order to accept new person names, the rule for `MUSICIAN` is modified as follows:

```
MUSICIAN: PROPER NAME.
PROPER NAME: $MATCH("[A-Z][a-z]*").
```

Applying GEN to this simple grammar leads to a parser that allows sentences like:

```
Nell joins The Rolling Stones
```

This parser will generate the following parse tree, augmented with some statistics:

```
parsing 1       (0.002 sec.)

start
  TO JOIN(RESP)
    MUSICIAN
      PROPER NAME
        "Nell"
    "joins"
    BAND
      "The Rolling Stones"

printing time was:      (0.010 sec.)

all:     1      (prescan: 0.011 s., parsing: 0.003 s., printing: 0.010 s. )
```

References

[BCD+95] E. Buchholz, H. Cyriaks, A. Düsterhöft, H. Mehlan, and B. Thalheim. Applying a Natural Language Dialogue Tool for Designing Databases. In *Proceedings of the First Workshop on Applications of Natural Language to Databases (NLDB'95)*, pages 119–133, Versailles, France, June 1995.

[BR95a] J.F.M. Burg and R.P. van de Riet. COLOR-X: Linguistically-based Event Modeling: A General Approach to Dynamic Modeling. In J. Iivari, K. Lyytinen, and M. Rossi, editors, *The Proceedings of the Seventh International Conference on Advanced Information System Engineering*, Lecture Notes in Computer Science, pages 26–39, Jyvaskyla, Finland, 1995. Springer-Verlag.

[BR95b] J.F.M. Burg and R.P. van de Riet. COLOR-X: Object Modeling profits from Linguistics. In *Proceedings of the KB&KS'95, the Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, Enschede, The Netherlands, 1995.

[Bub86] J.A. Bubenko. Information System Methodologies - A Research View. In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, editors, *Information Systems Design Methodologies: Improving the Practice*, pages 289–318. North-Holland, Amsterdam, The Netherlands, 1986.

[BW94] E. Bertino and H. Weigand. An approach to authorization modeling in object-oriented database systems. *Data & Knowledge Engineering*, 12:1–29, 1994.

[Che76] P.P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[Cho65] N. Chomsky. *Aspects of the theory of syntax.* MIT Press, Cambridge, Massachusetts, 1965.

[Cod70] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, 1970.

[CW93] M.A. Collignon and Th.P. van der Weide. An Information Analysis Method Based on PSM. In G.M. Nijssen, editor, *Proceedings of NIAM-ISDM.* NIAM-GUIDE, September 1993.

[CY90] P. Coad and E. Yourdon. *Object-Oriented Analysis.* Yourdon Press, New York, New York, 1990.

[Dat91] C.J. Date. *An Introduction to Data Base Systems*, volume 1. Addison-Wesley, Reading, Massachusetts, 5th edition, 1991.

[DFW96] C.F. Derksen, P.J.M. Frederiks, and Th.P. van der Weide. Paraphrasing as a Technique to Support Object-Oriented Analysis. Technical Report CSI-R9603, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, January 1996.

[Dig89] F.P.M. Dignum. *A Language for Modelling Knowledge Bases.* PhD thesis, Free University, Amsterdam, The Netherlands, 1989.

[Dij76] E. W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[Dik89] S.C. Dik. *The Theory of Functional Grammar. Part I: The Structure of the Clause.* Floris Publications, Dordrecht, The Netherlands, 1989.

[DKNZ92] C. Dekkers, C.H.A. Koster, M.-J. Nederhof, and A. van Zwol. The Grammar Workbench: A First Step towards Lingware Engineering. In *Proceedings of the second Twente Workshop on Language Technology, Memoranda Informatica 92-29*, pages 103–115, Enschede, The Netherlands, April 1992. University of Twente.

[DO90] L. Dunn and M. Orlowska. A Natural Language Interpreter for the Construction of Conceptual Schemas. In B. Steinholz, A. Sølvberg, and L. Bergman, editors, *Proceedings of the Second Nordic Conference CAiSE'90 on Advanced Information Systems Engineering*, volume 436 of *Lecture Notes in Computer Science*, pages 175–194, Stockholm, Sweden, 1990. Springer-Verlag.

[EGH+92] G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H-D. Ehrich. Conceptual modelling of database applications using an extended ER model. *Data & Knowledge Engineering*, 9(4):157–204, 1992.

[FKW95] P.J.M. Frederiks, C.H.A. Koster, and Th.P. van der Weide. Object-Oriented Analysis using Informal Language. Technical Report CSI-R9516, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, December 1995.

[FW96] P.J.M. Frederiks and Th.P. van der Weide. Properties and Design of Information Architectures. Technical Report, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, January 1996. (in preparation).

[Gra94] I. Graham. *Object-oriented Methods.* Addison-Wesley, Reading, Massachusetts, 1994.

[Gri82] J.J. van Griethuysen, editor. *Concepts and Terminology for the Conceptual Schema and the Information Base.* Publ. nr. ISO/TC97/SC5-N695, 1982.

[HO92] T.A. Halpin and M.E. Orlowska. Fact-oriented modelling for data analysis. *Journal of Information Systems*, 2(2):97–119, April 1992.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HPW94] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Grammar Based Information Modelling. Technical Report CSI-R9414, Submitted for publication, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, October 1994. Electronically available as: http://www.icis.qut.edu.au/~erikp/articles/Grammar.ps.Z.

[HVV91] G. de Haan, G.C. van der Veer, and J.C. van Vliet. Formal modelling techniques in human-computer interaction. *Acta Psychologica*, 78:27–67, 1991.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[Joh95] P. Johannesson. Supporting Schema Integration by Linguistic Instruments. In *Proceedings of the First Workshop on Applications of Natural Language to Databases (NLDB'95)*, pages 41–56, Versailles, France, June 1995.

[Kos91] C.H.A. Koster. Affix Grammars for natural languages. In *Attribute Grammars, Applications and Systems, International Summer School SAGA*, volume 545 of *Lecture Notes in Computer Science*, pages 469–484. Springer-Verlag, Berlin, Germany, June 1991.

[Kri94] G. Kristen. *Object Orientation, the KISS Method: From Information Architecture to Information System.* Addison-Wesley, Reading, Massachusetts, 1994.

[Mee78] L.G.L.T. Meertens. Program text and program structure. In P.G. Hibbard, S.A. Schuman, editor, *Constructing quality software*, pages 271–283, Amsterdam, May 1978. North-Holland. IFIP WG 2.1/WG 2.4 Working Conference, Novosibirsk.

[MG94] L. Mich and R. Garigliano. A Linguistic Approach to the Development of Object Oriented Systems using the NL System LOLITA. In E. Bertino and S. Urban, editors, *Proceedings of the International Symposium, ISOOMS '94: Object-Oriented Methodologies and Systems*, volume 858 of *Lecture Notes in Computer Science*, pages 371–386, Palermo, Italy, September 1994. Springer-Verlag.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, Sydney, Australia, 1989.

[Nij89] G.M. Nijssen. An Axiom and Architecture for Information Systems. In E. D. Falkenberg and P. Lindgreen, editors, *Information System Concepts: An In-depth Analysis*, pages 157–175. North-Holland/IFIP, Amsterdam, The Netherlands, 1989.

[PSB+94] J. Preece, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human Computer Interaction.* Addison-Wesley, Reading, Massachusetts, 1994.

[RP92] C. Rolland and C. Proix. A Natural Language Approach For Requirements Engineering. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 257–277, Manchester, United Kingdom, 1992. Springer-Verlag.

[Sch92] B. Schneiderman. *Designing the User Interface: Strategies for Effective Computer Interaction.* Addison-Wesley, Reading, Massachusetts, 2nd edition, 1992.

[VH95] T.F. Verhoef and A.H.M. ter Hofstede. Feasibility of Flexible Information Modelling Support. Technical Report CSI-R9510, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, August 1995.

[Wij91] G.M. Wijers. *Modelling Support in Information Systems Development.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1991.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice.* Kluwer, Deventer, The Netherlands, 1990.

Contents