Assistance for the domain modeling dialog

Sander Bosman Theo van der Weide
sanderb@cs.kun.nl tvdw@cs.kun.nl
Computing Science Institute, University of Nijmegen
The Netherlands

Abstract

This paper considers the domain modeling dialog between domain expert and system analyst. In this dialog, the system analyst interprets the domain knowledge provided by the expert, and creates a formal model that captures this knowledge. As the expert may not express knowledge in a very precise way, the system analyst has to find the correct interpretation out of many possible interpretations.

In order to improve the quality of the modeling dialog, we propose a mechanism that enables the system analyst to have a better idea of the intentions of the domain expert, especially where the expert expresses these intentions poorly.

1 Introduction

Domain modeling is the process of creating a domain model in some formal language, from domain knowledge available through domain experts. The resulting domain model is a precise, complete and consistent description of the relevant domain knowledge, at the right level of abstraction, denoted in a formal language such as ER (Chen, 1976) or PSM (Hofstede and Weide, 1993).

Two important roles can be distinguished in the modeling process. The *domain expert* is responsible for providing domain knowledge, but does not (need to) have modeling skills. The *system analyst* does not (need to) have domain knowledge, but has the skills to create a formal, well-abstracted model from the domain knowledge provided (Frederiks and Weide, 2004).

Several methods exist that help system analysts in structuring their task. Of the more elaborate methods, NIAM (Nijssen, 1989) and its descendants such as ORM (Halpin, 1995) are good examples. These methods are language oriented, using a *domain description* as input for analysis. The domain description, consisting of a set of elementary sentences, is assumed to be created by the domain expert and available when the method starts.

We consider the modeling process as a *dialog* between domain expert and system analyst. Here, the complete domain description is not available before the analysis starts, but grows with each statement that is expressed by the domain expert. The domain description can be seen as the minutes of the dialog. This situation is depicted in figure 1.

The methods mentioned earlier assume the domain expert can express domain knowledge in a precise, consistent and well-abstracted way. However, domain experts often can *not* express domain knowledge in this strict way. Rather, the knowledge is often expressed

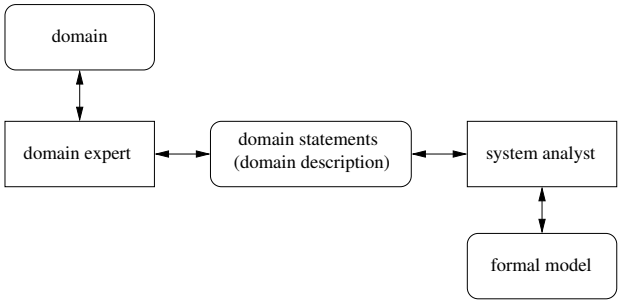


Figure 1: *Modeling process: dialog between domain expert and system analyst*

as a collection of informal pieces of information, which leaves room for many possible interpretations.

This paper proposes a mechanism to improve the quality of the modeling dialog. Instead of rejecting imprecise expressions from the domain expert, we allow them as they may contain valuable domain knowledge. In addition, *interpretation knowledge* is built up that has the goal of reducing the possible interpretations (ultimately to a single correct interpretation).

Figure 2 shows how we consider the interpretation of domain knowledge by the system analyst. The domain description D is the domain knowledge expressed by the domain expert. This domain description is interpreted by the system analyst, resulting in the formal model M. The extra information needed to interpret D and produce M is the *interpretation knowledge* I.

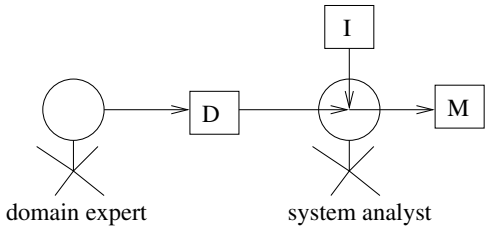


Figure 2: *Interpretation of domain knowledge*

The interpretation knowledge often remains implicit, in the mind of the system analyst. Making this knowledge *explicit* may improve both the modeling efficiency as well as the quality of the final model. The explicit interpretation knowledge can be seen as a motivation of the modeling decisions taken, based on the dialog minutes. Using this, the system analyst may become more conscious and rational about how the domain knowledge is handled (Veldhuijzen van Zanten et al., 2003).

Section 2 describes the basic interpretation knowledge needed to interpret a semi-formal domain description as is expected by NIAM, where informality does not play a role. Then, section 3 discusses the implications of allowing informal domain knowledge.

2 Interpretation of semi-formal domain descriptions

This section discusses the interpretation of semi-formal domain descriptions, providing the basis for the next section where informal domain descriptions are discussed.

2.1 Domain descriptions

The domain description is the collection of information provided by a domain expert as domain knowledge. The domain description changes during the modeling dialog: new information is added and information found to be false or irrelevant is changed or removed.

For our purposes, we assume a domain description can be seen as a set of *sentences*, each being the representation of an elementary *statement*, denoted in some *language* called L_D. Both textual and graphical descriptions can be viewed this way, but we will limit ourselves to textual descriptions in this paper.

For now we consider semi-formal domain descriptions: descriptions which are consistent and unambiguous, such that only one single interpretation can result.

Example 1 The set of elementary sentences resulting from CSDP Step 1 of the ORM method (Halpin, 1995) is such a domain description. For example:

```
Person with name Lee has age of 38 years.
Person with name Mary has age of 19 years.
```

The sentence structure of the first sentence will be shown in example 3.

2.2 Formal model

The formal model can also be seen as a description, consisting of a set of statements, denoted in a formal language L_M. Many modeling languages display a model as a graphical schema, such as conceptual schemas. These schemas can also be seen as a set of statements.

Example 2 Let L_M be a formal language, used in the following examples, based on a subset of PSM (Hofstede and Weide, 1993). The following statements are included in L_M, and can be used to create a formal model with:

Statement	Description
entity(x)	x is an entity, such as *'person with name Lee'*
entity-type(x)	x is an entity type, such as *'person'*
label(x)	x is a label, such as *'Lee'*
label-type(x)	x is a label type, such as *'name'*
relation-type(x)	x is a relation type, such as *'<person> has <age>'*
relation(x)	x is a relation, e.g., *'(person with name Lee) has (age of 38 years)'*
instance-of(x,y)	x is an instance of type y. E.g., *'person with name Lee'* is instance of type *'person'*.

In the following section we will show how, from this set of formal statements, a model is built using the sentences from a domain description.

2.3 Interpretation of domain descriptions

For semi-formal domain descriptions, the interpretation information I is agreed upon by both domain expert and system analyst, before the dialog starts. It specifies the correct interpretation for each domain statement $s \in D$.

Let Interp be the interpretation function that produces the formal model M, given a domain description D and interpretation knowledge I:

$$\mathsf{Interp}(D, I) \rightarrow M$$

In this situation, only two types of interpretation knowledge are needed to interpret D:

1. The specification of a Parse function, needed to recognize the sentence structure of sentences in D;
2. The specification of a translation Trans, translating statements from D into statements in M.

Together, these functions fully specify the interpretation information:

$$I = \langle \mathsf{Parse}, \mathsf{Trans} \rangle$$

The interpretation function can now be specified as:

$$\mathsf{Interp}(D, \langle \mathsf{Parse}, \mathsf{Trans} \rangle) = \mathsf{Trans}(\mathsf{Parse}(D))$$

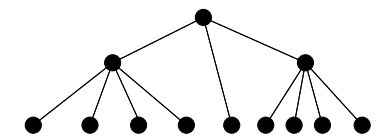
The following sections discuss the functions Parse and Trans in more detail.

2.3.1 Recognizing sentence structure

The first step to interpret a domain description is parsing it into meaningful concepts and structures. We already assumed a domain description can be seen as a set of sentences. For our purposes, we make additional assumptions on the structure of sentences in the domain description.

Assumption 1 A sentence can be seen as a hierarchical composition of parts, called *terms*. Atomic terms, or labels, can not be decomposed any further. Terms which are not atomic are called complex terms.

Example 3 The first statement of example 1 may have the following structure:



Assumption 2 Terms represent *concepts*. Concepts, or conceptions, are abstract things that reside within the mind of the domain expert. As we assume the concepts in a mental model are related to the domain, we can say that terms represent things in the domain (Bosman and Weide, 2003).

The Parse function takes a sentence from L_D, and produces a *parse expression* from the set of possible parse expressions PExpr:

$$\mathsf{Parse} : L_D \rightarrow \mathsf{PExpr}$$

Parse expressions represent the hierarchical structure of a sentence. Their form is a reflection of assumption 1 (using EBNF notation):

```
PExpr ::- Label
PExpr ::- TermType (PExpr+)
```

Example 4 This example considers Natural language Normal Form (NNF), which closely resembles the NIAM normal form. In NNF sentences, entities are uniquely described by a *standard name* (such as *'person with name Lee'*), consisting of of the following three elements:

Entity type	the type of which the entity is an instance (e.g., *"person"*).
Label type	the type of the concrete object used to identify the entity (e.g., *"name"*)
Label	the concrete object that identifies the entity (e.g., *"Lee"*)

A sentence is in NNF if it is unsplittable and all entities are described by their standard name.

The following context-free grammar G_{NNF} specifies the language of NNF sentences:

(sentence construct)	S → S c S
(standard name)	S → ET c LT L

Here, `c` = verb or preposition, `ET` = entity-type, `LT` = label-type, `L` = label. The grammar is a variation of the one described in (Collignon and Weide, 1993). Textual labels are to be surrounded by quotes to avoid ambiguity.

Example 5 The following sentence is in NNF:

```
CD with title Urk 'is created by' artist with name 'The Nits'
```

Parsing the sentence yields:

```
S(
  SN(ET('CD'), P('with'), LT('title'), L('Urk')),
  'is created by',
  SN(ET('artist'), P('with'), LT('name'), L('The Nits'))
)
```

2.3.2 Translation to a formal language

The second interpretation step is to translate a parsed sentence from the domain description into the formal language. This is specified by the Trans function:

$$\mathsf{Trans} : \mathsf{PExpr} \rightarrow \wp(L_M)$$

Translation of a sentence in L_D results in a *set* of sentences in L_M, since more than one statement may be needed to describe the sentence to be translated.

Example 6 Let L_D be a language conforming to the NNF structure from example 4. Let L_M be the formal language from example 2. The translation function Trans, that translates from L_D to L_M, can be given by the following recursive specification. The two Trans rules directly correspond to the two grammar rules of example 4.

(sentence construct)
Trans(S(s_1,c,s_2)) =
{ relation-type(rt), relation(s), instance-of(s,rt) } $\cup$ Trans(t_1) $\cup$ Trans(t_2)
where
s = S(s_1,c,s_2)
t_1 = getType(s_1)
t_2 = getType(s_2)
rt = RT(t_1, c, t_2)

(standard name)
Trans(SN(ET(*et*), P(*p*), LT(*lt*), L(*l*))) =
{ entity-type(*et*), entity(*e*), instance-of(*e*,*et*), label-type(*lt*), label(*l*), instance-of(*l*,*lt*) }
where
e = SN(ET(*et*), P(*p*), LT(*lt*), L(*l*))

The help-function getType is needed by Trans: it returns the type of a given term. This function is defined recursively by:

getType(SN(ET(*et*), P(*p*), LT(*lt*), L(*l*)) = *et*

getType(S(s_1, c, s_2) = RT(t_1, c, t_2)
where t_1 = getType(s_1), t_2 = getType(s_2)

Note that the parse expression of the form RT(t_1, v, t_2) is introduced to represent a relation type with role types t_1 and t_2, and connector c.

The precise nature of the domain description allows a straightforward translation function. The function is also suitable for any domain, since no domain knowledge is present in the specification of the translation function.

3 Interpreting informal domain descriptions

In the previous section, it has been discussed how a negotiated domain description may be transformed into a formal model, using conventional techniques. In this section, we shift focus and take the negotiation process also into account.

A main issue encountered is the interpretation of *informal* domain descriptions. Intuitively, an informal domain description is vague or incomplete in some way: there is uncertainty about how to interpret it correctly. The base method from the previous section can not handle informal knowledge, as it requires each statement *can* be interpreted with certainty.

Instead of rejecting informal knowledge, as done in the previous section, we would rather like to accept *every* piece of available knowledge. Any piece of knowledge, even when it is informal, may help the system analyst in constructing a correct formal model.

The intention of this section is to propose a model to handle informal domain descriptions. We will not go into details what heuristics and strategies are best to support a system analyst during the dialog, but focus on a general model in which heuristics and strategies can be 'plugged in'.

3.1 The nature of informal descriptions

A description is informal when there is uncertainty about the correctness of the domain description and/or the interpretation of that domain description. This uncertainty may appear at various places:

- A statement $s \in D$ may be *invalid*, although the domain expert may believe the statement is valid at the time he expresses it.
- The *interpretation* $\mathsf{Interp}(s)$ of a statement $s \in D$ may be invalid or ambiguous.
- The interpretation $\mathsf{Interp}(s)$ of a statement $s \in D$ may be not known at all.

Uncertainty is a fundamental property of informal descriptions. As a result, a system analyst can *never* be sure he has the right interpretation as is intended by the domain expert. Any domain description and any interpretation of it is based on uncertain knowledge, and therefore their validity can never be established with certainty.

However, there is still a big difference between *plausible* and *implausible* interpretations. Absolute certainty about an interpretation may never be achieved, but becoming *almost certain* about the correctness of an interpretation can still be aimed at.

The approach for handling uncertainty we will use in the next sections is based on these ideas:

- The system analyst tries to find and work with the most plausible interpretation of the domain description.
- This interpretation is assumed to be valid, until proven otherwise. In that case, another more plausible interpretation is to be chosen.
- At any point within the dialog, the system analyst may ask the domain expert for information that can diminish uncertainty. This information may lead to improved confidence in the current interpretation, or it may lead to another, more plausible interpretation.

This approach assumes that invalid interpretations can be detected:

Assumption 3 *(invalidity of interpretation is detectable)*
Invalidity of the interpretation of an informal description can eventually be detected, when the size of the domain description grows ($|D| \to \infty$). Invalidity can be detected when:

- There is no interpretation of a statement: $\mathsf{Interp}(s) = \emptyset$. In this case, either there should be an interpretation, or the statement should not be part of the domain description.
- The model M, the result of the interpretation, is invalid. The most common reason for a model to be invalid is *inconsistency*, but other reasons may also cause it to be invalid.

The following section illustrates the dialog approach in more detail.

3.2 An approach for handling uncertainty

In this section, we will discuss our basic approach for handling uncertainty. In the next sections, two types of uncertainty will illustrate the approach.

Let $Dialog_t$ be the sequence of actions that have taken place as part of the dialog, up to moment t. Actions include asking questions, giving answers, etc. At any point during the dialog, the domain description D is the result of actions that occurred in the dialog. Let Rd be the function that produces the domain description from a dialog:

$$\mathsf{Rd}_t : Dialog_t \rightarrow D_t$$

We will omit the time t when the current state of the dialog is meant.

The interpretation of a domain description, $\mathsf{Interp}(D_t, I_t)$, results in a model M_t. For informal descriptions, this interpretation is now the *most plausible* or *most probable* interpretation at time t, from the system analyst's point of view. All the choices and assumptions made to reach this interpretation are part of the interpretation knowledge I_t.

The following procedure illustrates how the system analyst can structure the dialog such that informal descriptions can be handled:

```
dialog:
   Dialog = ∅
   repeat
      D := Rd(Dialog)
      M := Interp(D, I)
      if invalid(M):
         find new plausible interpretation, by adjusting I to I'
         M := Interp(D, I')
      choose:
         ask domain expert for any new input, or,
         ask domain expert for information that reduces uncertainty
   until ready
```

In each iteration, a choice is made about what to do next. The first choice asks the domain expert for any input. This input may be a new statement that should be added to the domain description, or, that a statement should be removed from D.

```
ask new input:
   ask domain expert for new input
   answer: action A_{t+1}
   Dialog += A_{t+1}
```

The second choice is to ask the domain expert for information that may reduce uncertainty about the current interpretation. This may be triggered by low confidence in the current interpretation, or, it may be that more interpretations are equally plausible, and the system analyst wants more information on which one to choose.

The result can be a *validation* of the current interpretation, when the current interpretation remains the most plausible one and its confidence is increased. An other result can be an *interpretation shift* to a new interpretation that is more plausible than the current one.

It is important for the system analyst to have a good strategy for when to reduce uncertainty. If asking too quickly, the domain expert might be disturbed in his current line of thought because he has to explain details of how to interpret the expressed knowledge. On the other hand, waiting to diminish uncertainty for too long lets the system analyst work with a current interpretation that he is not confident about.

3.3 Validity of domain statements

We illustrate the approach by lifting the requirement that domain statements need to be valid. Although validity of domain statements can still initially be *assumed*, any domain statement may prove to be invalid.

Assessing whether domain statements are valid becomes part of the system analyst's task. For the current interpretation, he needs to keep track of which domain statements he believes are valid and which are not.

Let $SV \subseteq D$ be the set of domain statements of which the system analyst believes they are valid. This set becomes part of the interpretation information I:

$$I \triangleq \langle \mathsf{Parse}, \mathsf{Trans}, SV \rangle$$

Interpretation of a domain description is only based on the statements that are believed to be valid:

$$\mathsf{Interp}(D, \langle \mathsf{Parse}, \mathsf{Trans}, SV \rangle) = \mathsf{Trans}(\mathsf{Parse}(SV))$$

At any point within the dialog, an interpretation may become invalid caused by a wrong choice of SV. Following the dialog procedure illustrated in the previous section, the system analyst will need to find a new plausible interpretation that *is* valid. This involves finding a new SV.

A new interpretation can be found by first creating a set of possible interpretations, and then selecting the most plausible one from it.

Let S be a set of possible plausible interpretations, where each interpretation is fully defined by the interpretation information I. A possible interpretation is considered to be plausible when it is consistent and maximal:

$$\mathsf{S} = \{I | I.SV \subseteq D \wedge \mathsf{IsValid}(\mathsf{Interp}(D, I)) \wedge I.SV \text{is maximal}\}$$

An interpretation is maximal when no other domain statement can be assumed to be valid, without making that interpretation invalid. This reflects the assumption that domain statements must be assumed to be valid unless they 'cause problems'.

For any possible interpretation, the statements thought to be invalid are:

$$\mathsf{InvalidStatements}(D, I) = D - I.SV$$

The *conflict set* CS is the minimal set that is assumed to contain conflicts:

$$\mathsf{CS}(\mathsf{S}) \triangleq D - \bigcap_{I \in \mathsf{S}} I.SV$$

Example 7 Let the following domain statements be specified:

```
s1: John lives in Nijmegen.
s2: John has a bike.
s3: John lives in Maastricht.
s4: A person lives in one city.
```

Interpretation of the first 3 sentences causes no inconsistency, therefore $SV_3 = \{s_1, s_2, s_3\}$. However, statement 4 causes inconsistency. Now, $D_4 = \{s_1, s_2, s_3, s_4\}$. The set of possible plausible interpretations $\mathsf{S} = \{I_1, I_2, I_3\}$ where

$$I_1.SV = \{s_1, s_2, s_3\}$$
$$I_2.SV = \{s_1, s_2, s_4\}$$
$$I_3.SV = \{s_2, s_3, s_4\}$$

The conflict set $\mathsf{CS} = \{s_1, s_3, s_4\}$; statement s_2 never causes inconsistency.

Eventually, the assumptions made in SV should be reflected in the domain description itself. When a current interpretation (which includes SV) is validated with the domain expert, the domain expert may realize the statements assumed to be invalid by the system analyst are *indeed* invalid. This leads to removal of these invalid statements from D. At the end of the modeling process, this should lead to $\mathsf{InvalidStatements}(D, I) = \emptyset$.

3.4 Conceptual ambiguity

Handling conceptual ambiguity is another illustration of the approach.

Conceptual ambiguity arises when a term is used to refer to more than one concept (homonimity). A similar situation is when a concept is referred to by more than one term (synonimity). Homonyms and synonyms disrupt the assumption made in assumption 2 that there is a one-to-one relation between term and concept. With conceptual ambiguity, terms and concepts are not interchangeable things any longer.

Example 8 Let G_{RNNF} be the grammar G_{NNF} from example 6, extended with the following production rule:

```
S -> L
```

With this extra rule, entities can be identified by either a standard name (SN) or a simple label (L). No restriction is put on simple labels, therefore labels may be ambiguous.

Example 9 Using G_{RNNF}, the following domain description can be parsed and interpreted:

```
s1: John lives in Nijmegen.
s2: John has a bike.
s3: John lives in Maastricht.
s4: A person lives in one city.
```

In the previous section, the terms $John$ in s_1 and s_2 were required to identify the same person. However, now they might refer to the same person, or they might refer to two different persons.

Interpretation of ambiguous terms is handled by adding an extra step in interpretation. Let DA be a function that assigns an ambiguous parse expression to an unambiguous parse expression:

$$\mathsf{DA} : \mathsf{PExpr} \rightarrow \mathsf{PExpr}$$

The interpretation information I is augmented with the disambiguating function DA, which disambiguates terms where needed:

$$I \triangleq \langle \mathsf{Parse}, \mathsf{Trans}, \mathsf{DA} \rangle$$

and

$$\mathsf{Interp}(D, \langle \mathsf{Parse}, \mathsf{Trans}, \mathsf{DA} \rangle) = \mathsf{Trans}(\mathsf{DA}(\mathsf{Parse}))$$

Note that we can also combine this extension with the one from the previous section (adding SV). However, we do not do this here for clarity.

Example 10 An instance of the term $John$ may be disambiguated to the concept $John_2$:

$$\mathsf{DA}(John) = John_2$$

Which disambiguations to use in a specific dialog is often unknown or at least uncertain. As done in the previous sections, a plausible interpretation is worked with, based on a plausible disambiguation function DA.

When at some time during the dialog the current interpretation becomes invalid, it becomes apparent that the interpretation knowledge I must be wrong. Therefore, I has to be changed such that a new plausible interpretation will result. This involves changing DA. The new interpretation can be selected from the set of plausible possible interpretations:

$$\mathsf{S} = \{I | I.\, \mathsf{DA} \in \mathcal{DA} \wedge \mathsf{IsValid}(\mathsf{Interp}(D, I))\}$$

Here, $\mathcal{DA}$ is the set of possible plausible disambiguation functions DA.

Example 11 Interpretation of the four sentences of example 9 causes inconsistency, when DA specifies that the three instances of the term $John$ refer to one single person $John_1$. Another interpretation, where the terms from s_1 and s_3 are assumed to refer to $John_1$ and $John_2$ respectively, is one of the plausible possible interpretations from S.

4 Conclusion

This paper discussed the information modeling process as a dialog between domain expert and system analyst. It was shown how the formal model, being the result of the modeling process, can be seen as the interpretation of the domain description provided by the domain expert. The *interpretation knowledge* specifies how this interpretation is to be performed. For conventional techniques, where certainty is required about how to interpret the domain description, it was shown what the interpretation knowledge consists of.

Then focus shifted to *informal domain descriptions*, where uncertainty about how to interpret a domain description *is* accepted. A model was introduced which may help the system analyst in handling these uncertainties. Using this model, the system analyst works with the most plausible interpretation, until this interpretation causes inconsistency. When this occurs, another plausible interpretation is to be chosen. The assumptions and choices that lead to the current interpretation are again part of the interpretation knowledge. Two types of uncertainty were discussed, and it was shown how they are handled by the approach.

References

Bosman, S. and Weide, T. v. d. (2003). A case for incorporating vague representations in formal information modeling. In *Conferentie Informatiewetenschap 2003*, TU Eindhoven, The Netherlands.

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

Collignon, M. and Weide, T. v. d. (1993). An Information Analysis Method Based on PSM. In Nijssen, G., editor, *Proceedings of NIAM-ISDM*. NIAM-GUIDE.

Frederiks, P. and Weide, T. v. d. (2004). Information modeling: the process and the required competencies of its participants. In *Proceedings of the International Workshop on Applications of Natural Language to Databases (NLDB'2004)*.

Halpin, T. (1995). *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition.

Hofstede, A. t. and Weide, T. v. d. (1993). Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100.

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

Veldhuijzen van Zanten, G., Hoppenbrouwers, S., and Proper, H. (2003). System Development as a Rational Communicative Process. In Callaos, N., Farsi, D., Eshagian-Wilner, M., Hanratty, T., and Rish, N., editors, *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, volume XVI, pages 126–130, Orlando, Florida, USA.