Explaining Conceptual Models —
Using Toulmin's argumentation model and RST

Hercules Dalianis
Paul Johannesson

Department of Computer and Systems Sciences (DSV)
Royal Institute of Technology (KTH) and Stockholm University
Electrum 230, S-164 40 Kista, Sweden
email: {hercules, pajo}@dsv.su.se

Abstract

An important activity in requirements engineering is validation, which is the process of checking whether a model correctly represents a piece of reality and the users' requirements. One technique for supporting validation is explanation generation, which combines paraphrasing of a specification with question–answer facilities that interactively support a user in exploring a model. In this paper, we propose an architecture and design principles for constructing explanation generation systems for conceptual models. The architecture is partly based on Toulmin's argumentation model, which provides a framework for structuring arguments and Rhetorical Structure Theory. We argue that Toulmin's argumentation model needs support from a discourse theory like RST to create customized explanations to the user. To each type of explanation there exists a set of appropriate RST relations to be used.

1. Introduction

Requirements engineering is the branch of systems engineering concerned with eliciting and specifying the functionality of large and complex software-intensive systems. Put simply, requirements engineering focuses on the early phases of the systems development life cycle. The result of requirements engineering is a requirements specification, which plays several roles in the process of systems engineering. First, it is the basis for and part of a contract between requirements holders and systems developers. Secondly, it is an architectural, implementation independent drawing of the future system to be built. Thirdly, it provides an explicit basis for reasoning with and among the clients about various qualities of the system.

Requirements engineering consists of several related activities. First, the acquisition of requirements through observation, interviews, analysis of texts, reverse engineering, group work, co-development, etc. Secondly, the modelling and specification of requirements. Thirdly, the analysis of

requirements, which includes validation and verification. Finally, communication and documentation of the results of requirements engineering, where traceability and information management are needed.

By validation is meant the process of checking whether a model correctly represents a piece of reality and the users' requirements. Verification, on the other hand, concerns formal properties of a model, such as syntactical correctness, consistency, and completeness. While verification often can be automated, validation is inherently an informal process requiring subjective, human judgement. In this informal process, it is essential that different types of stakeholders participate, including people with limited knowledge of modelling and systems design. However, people who are unfamiliar with modelling languages may have severe difficulties in understanding and validating a model. Furthermore, the sheer size and complexity of a model may make it difficult even for experienced designers to validate a model.

In order to ease the validation process, several different techniques have been proposed. One approach is to introduce *graphical symbols*, [Kung93], or *user defined concepts*, [Mylopoulos90]. Another approach is to *paraphrase* parts of a conceptual model in natural language, [Rolland92], [Dalianis92], [Dalianis96]. In order to validate large models, a technique called *complexity reduction* has been proposed, which presents views of a conceptual model and hides irrelevant details, [Seltveit93]. *Model simulation* is still another technique, which can be used for observing and experimenting with the dynamic properties of a model, [Harel92], [Zave84]. A validation technique similar to model simulation is *model planning* that allows users to explore a model by constructing plans, i.e. operation sequences that result in states where certain conditions hold, [Costal96]. *Explanation generation* techniques have been used to integrate the techniques mentioned above and provide a uniform interface to the users. Explanation generation extends paraphrasing by including question–answer facilities that interactively support a user in exploring a model. Model simulation can be complemented by explanation generation that guides a user through the execution and explains the system's behaviour in more detail. Explanation generation has previously been used mainly for expert systems, e.g. the Explainable Expert System (EES), [Moore91], which provides natural language explanations of rule-based systems based on Rhethorical Structure Theory, RST, [Mann88]. Recently, there has also been an interest in using explanation generation for validating conceptual models. The most comprehensive work on explanation generation for validating conceptual models seems to be [Gulla96], which discusses an explanation component for CASE environments. This component provides a natural language interface to complex model information and integrates model simulation by explaining a model's behaviour by relating trace information to properties of the model. We have previously shown in [Dalianis97] that Toulmin's argumentation model is an excellent tool to create customized explanations during a validation scenario.

The purpose of this paper is to outline an architecture and design principles based on a combination of an argumentation model and a discourse theory for an explanation component for object-oriented conceptual models. The work reported upon in this paper extends previous work by providing explanation generation not only for static parts of a model, history traces, and simulation, but also for planning. The structure of the paper is as follows. Section 2 introduces the modelling

language used throughout the paper. Section 3, discuss Toulmin's argumentation model and RST. Section 4, includes a set of examples with corresponding discourse structure analysis. Section 5, summarizes the paper and gives suggestions for future work.

2. The Delphi Language

The modelling concepts and notation used in this paper are based on the language Delphi, [Höök93]. Delphi is a conceptual modelling language developed by Ericsson Utvecklings AB, where the language has been used for defining specifications of telephone services. The language is supported by a development environment including tools for verification and validation. The formal semantics of Delphi is given by means of first order logic and state machines, which makes it possible to carry out advanced analyses on Delphi specifications. We give here only an informal overview of Delphi as this is sufficient for the purposes of the paper.

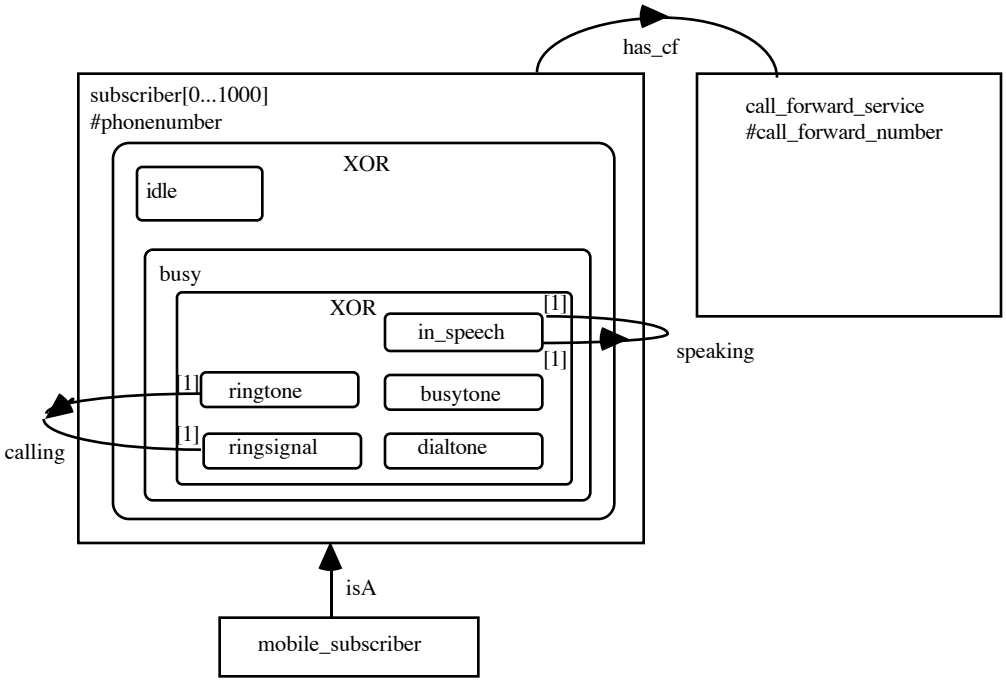


Figure 1. A Delphi static model

A Delphi specification consists of two parts: a *static model* describing static, structural aspects and a *dynamic model* describing behavioral aspects. The static model describes the *entity types* that exist in a system, their properties, and their relationships. The static model can also include static constraints, called *invariants* that put restrictions on allowable states. The dynamic model consists of a set of *dynamic rules* that define the state transitions of the system. A dynamic rule consists of a

stimulus, conditions, and conclusions. The informal semantics of a dynamic rule is that if a stimulus occurs in a state and the conditions are true in this state, then the conclusions will be true in the next state.

An example of a Delphi specification is shown in Figure 1 (the static model) and Figure 2 (the dynamic model). This specification defines POTS (Plain Old Telephone Service) extended with the service Call Forward. The static model includes three entity types: `subscriber` with the property `phone_number`, `mobile_subscriber` that is a subtype of `subscriber`, and `call_forward_service` with the property `call_forward_number`. The model specifies the set of states of a subscriber, e.g. a subscriber may be `idle`, have `ringtone`, or have `dialtone`.

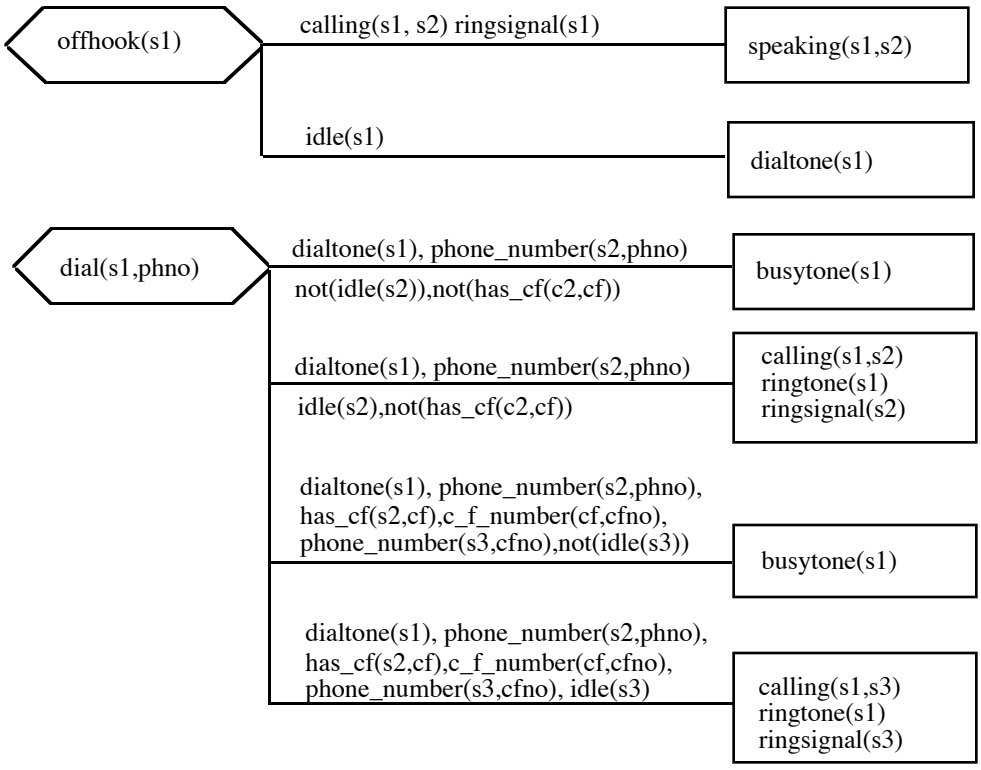


Figure 2. A Delphi dynamic model

The static model also graphically defines several invariants. For example, if a box is included in another box, the state corresponding to the first box is a substate of the state corresponding to the second box. This means that the static model defines that any subscriber who has `ringtone`, `ringsignal`, `dialtone`, `busytone`, or is `in speech` is also `busy`. The XOR notation

indicates that the substates are exclusive, e.g. a subscriber cannot simultaneously have `ringtone` and `busytone`, nor can she be both `busy` and `idle`. Furthermore, the static model defines a number of relations between subscribers, e.g. that one subscriber can be `calling` another. The model also expresses the invariant that if one subscriber calls another, then the first subscriber must have `ringtone` and the second `ringsignal`. Furthermore, the model specifies cardinality constraints for the relations.

The dynamic model consists of two dynamic rules. For each rule, the stimulus is shown within a hexagon, the conditions are placed along a line, and the conclusions are shown within a rectangle. The second rule contains four branches; the first branch states that dialling to a busy subscriber without call forward results in busytone; the second branch states that if one subscriber dials a phone number and the subscriber with this phone number is idle and does not have call forward, then the dialling subscriber will get ringtone and the other one ringsignal; the third and fourth branches state that dialling to a subscriber with call forward results in busytone if the subscriber forwarded to is busy, and otherwise in ringtone at the dialling subscriber and ringsignal at the other one.

A Delphi specification can have instantiations, or *fact bases*, describing a number of entities and their properties and relations at a particular point in time. Some examples of fact bases, which will be used in later sections, are given in Figure 3.

Fact base A	Fact base B
subscriber(john) phone_number(john,625118) idle(john) subscriber(mary) hone_number(mary, 500198) idle(mary)	subscriber(john) phone_number(john,625118) subscriber(mary) phone_number(mary,500198) calling(mary, john) ringtone(mary)
Fact base C	Fact base D
subscriber(john) phone_number(john,625118) subscriber(mary) phone_number(mary,500198) speaking(mary, john) in_speech(mary) in_speech(john)	subscriber(john) phone_number(john,625118) has_cf(cf1) call_forward_service(cf1) idle(john) subscriber(mary) phone_number(mary,500198) idle(mary)

Figure 3. Fact bases in Delphi

3. Architecture for an Explanation Generation System

A basic problem in explaining conceptual models is that of identifying forms of explanations and dialogue structures that effectively support users in understanding the models. Experience has shown that it is not particularly effective to simply paraphrase large fragments of a conceptual model, as people easily get lost in a long text and may even find it unfocused or irrelevant, [Gulla96]. Much more effective are short texts, or visualizations, that answer focussed questions about the structure or behaviour about a model in the context of a particular use case. Especially effective are answers that combine information from different parts of the conceptual model, e.g. from the fact base as well as from the dynamic rules, [Dalianis92]. In order to design an adequate explanation component, two fundamental issues have to be addressed. First, what types of questions and explanations should be supported? Secondly, how should the dialogue between user and system be structured?

When exploring a conceptual model, a user should be able to ask several different types of questions about constructs in the model. There seem to be, at least, four different types of questions in this context. The first type of question is the structural question that asks about the structure of a construct, its properties and relationships to other constructs as well as the constraints pertaining to the construct. The second type of question is the causal question that asks about the cause of a construct, i.e. the events that have made it to come into existence. The third type is the reason question that asks about the reason for a construct, i.e. why the construct has been introduced and what purpose it serves. Finally, there is the procedural question that asks how something is carried out.

A user should not only be allowed to ask isolated questions about a model and get answers, but also to engage in a more structured dialogue with the system involving follow-up questions and more detailed explanations. In order to design such dialogues, some kind of argumentation model is needed as foundation. An attractive candidate for this purpose is Toulmin's argumentation model, [Toulmin59], which describes the structure of arguments. The starting point of an argument is a *claim*, which is a sentence asserting some proposition. This claim is related to *grounds* supporting the claim; the grounds should provide the required basis for accepting the claim. If the grounds given for the claim do not convince a listener to the argument, further grounds can be given in support. If the listener is still not convinced, the argument may continue with a *warrant*, i.e. a relation between a ground and a claim showing that the grounds are indeed relevant for the claim. A warrant has usually the form of a general rule that is applicable to the case at hand. If grounds and warrant should not be sufficient to convince the listener, further supportive arguments can be given in the form of *backing*. Backing is used to show that the warrant is reliable and relevant in the context. Normally, backing takes the form of rules at a higher level than the warrant. However, grounds, warrant, and backing may not invariably lead to the required claim – *qualifications* such as "usually" and "possibly" may then be used to clarify the relationships between grounds and claim. If the claim does not follow with certainty from the grounds, an argument may describe the circumstances where the claim does not hold – these are called *rebuttals*. An argument can thus schematically be described as shown in Figure 4 below.

A major benefit of Toulmin's argumentation model for the purpose of explaining conceptual models is that it can guide the gathering of information from different components of the conceptual model. For example, in an interactive approach, if the user questions the grounds for a specific claim, the warrant can include additional information from the conceptual model and form the basis for a more detailed explanation.

In this way, Toulmin's model provides a simple and informative explanation structure that gives an adequate amount of information at the right time and level. Explanations designed according to Toulmin's model are probably most useful for claims on the instance level, i.e. claims about particular objects and their relationships. If a claim is on the instance level, it is natural to structure the argument about the claim in such a way that the grounds are also on the instance level, the warrant on the schema level, and the backing on the meta schema level.

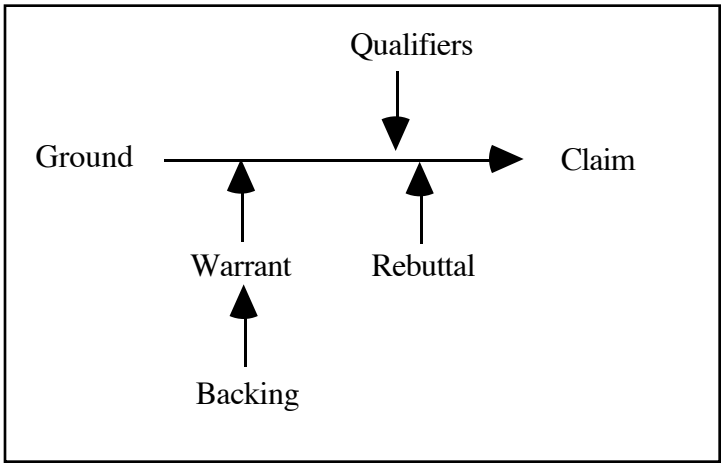


Figure 4. Toulmin's argumentation model

This structure gives a correspondence between the central parts of Toulmin's model and the components of a conceptual model. This correspondence can be used to organize the explanation of model constructs as shown in the following sections.

We have previously, in [Dalianis97], shown the use of Toulmin's argumentation model to give explanations in a validation scenario. One drawback of using solely Toulmin's argumentation model, specifically when the explanations tend to become long, is that the explanation itself looks rather unstructured. Then RST would be a candidate for structuring the explanations. Rhetorical Structure Theory (RST), [Mann88], is a model for describing the structure of coherent text. The basic idea of RST is that a segment of text has a central item, called a nucleus. This nucleus is supported by other segments of text, called satellites that are related to the nucleus in different ways. For example, a satellite may be a causal link or express evidence for the nucleus, or state a condition that must hold

if the nucleus is to be accepted. The purpose of RST is to use it for analyzing the discourse structure of texts. A text might have different discourse trees depending on how the reader interprets the text.

A text (discourse) consists of a number of coherent clauses. One or more RST relations can describe the inter clausal relations, i.e. within a set of RST relations one can fit in a text. Below, in Figure 5, follows a list of rhetorical relations from [Mann88] with a short explanation.

- **circumstance** - presents a realized situation and the circumstances.
- **solutionhood** - presents a problem and a solution.
- **elaboration** - the content of the first clause is elaborated in the second clause.
- **background** - the final clause is not comprehended unless the backgound is read.
- **enablement** - presents an action which is enabled by the enablement.
- **motivation** - presents an action in which the motivation increases for the actor to act.
- **evidence** - the first clause is proven by the second clause.
- **justification** - the first clause is justified by the second clause.
- **volitional cause** - presents a situation that is a volitional cause
- **volitional result** - presents a situation that is a volitional result.
- **non volitional cause** - presents a situation that is a non volitional cause.
- **non volitional result** - presents a situation that is a non volitional result.
- **purpose** - presents an activity and a connected situation which is unrealized.
- **antithesis** - the antithesis in second clause strengthens the thesis in clause one
- **concession** - the clause does not claim that the situation does not hold.
- **condition** - a unrealized condition is expressed.
- **otherwise** - if A does not happen then C.
- **interpretation** - the second clause gives an interpretation of the first clause.
- **evaluation** - the second clause evaluates of the content of the first clause.
- **restatement** - the second clause is a restatement of the first clause.
- **summary** - the last clause makes a short summary of what in previous clauses was said.
- **sequence** - a number of clauses are succeeding each other.
- **contrast** - shows that two clauses are contrasting each other.

Figure 5. Overview of RST Rhetorical relations

A main advantage of using RST for explaining conceptual models is that it provides a basis for constructing more coherent natural language output, [Gulla96]. However, using RST has the drawback that it requires a customization for each domain or application, since the typical RST relations do not have any immediate correspondences to conventional conceptual modelling constructs. This is in contrast to Toulmin's model where there are natural correspondences between the model's components and conceptual modelling constructs. These correspondences make it easy to

apply Toulmin's model for any specification expressed in an object oriented conceptual model. RST may, therefore, be less attractive for these types of models, but can be useful for semantically richer models that provide, for example, constructs for time and intentionality.

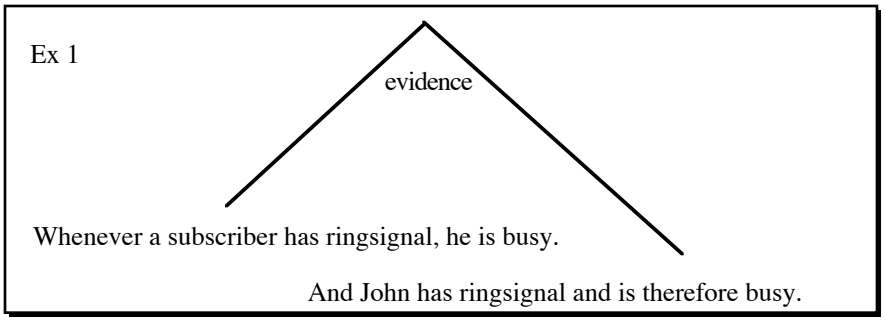
In [Dalianis92], Hobbs coherence relations [Hobbs90] were used to explain a conceptual model. The relations used where *elaboration, examplification* and *parallel*. Hobbs coherence relations are more general than the RST relations, e.g. Hobbs coherence relation *examplification* corresponds to the RST relation *evidence*, and *parallel* to *sequence*. Cue words or cue phrases are sometimes used to mark or augment the discourse relation, see examples below and their conclusion text.

We can see that in [Dalianis97] the system answers become less and less structured the longer they become. Using some of the RST relations from the list above will increase the readability of the explanations. From [Dalianis97] we can distinguish five types of questions/answers that will be enhanced using RST structures: Structural, Causal, Procedural, and Procedural with rebuttal and Reasonal questions.

4. Enhancing explanations using RST

In this section follows a set of examples showing how the RST relations can be used for structuring the answers that are longer than a clause. The text after the SYSTEM: shows how the text appeared in [Dalianis97] and the graph shows how the text can be structured based on the RST relations. The RST relations will make the generation of the texts computable, i.e. having knowledge of the text structure one can write a program performing the same task. In the real program the clauses will become attached to the correct leaf depending of what category the clause will have and what relations are available depending on what explanations are gives. To each type of question we can find a unique RST relation. When there is more to say such that the clauses does not fit within that RST relation then extend the RST structure with the RST relation Sequence.

Structural question Combining static model and fact base B.
USER: Why is John busy?
SYSTEM: Whenever a subscriber has ringsignal, he is busy. And John has ringsignal and is therefore busy. (Warrant)

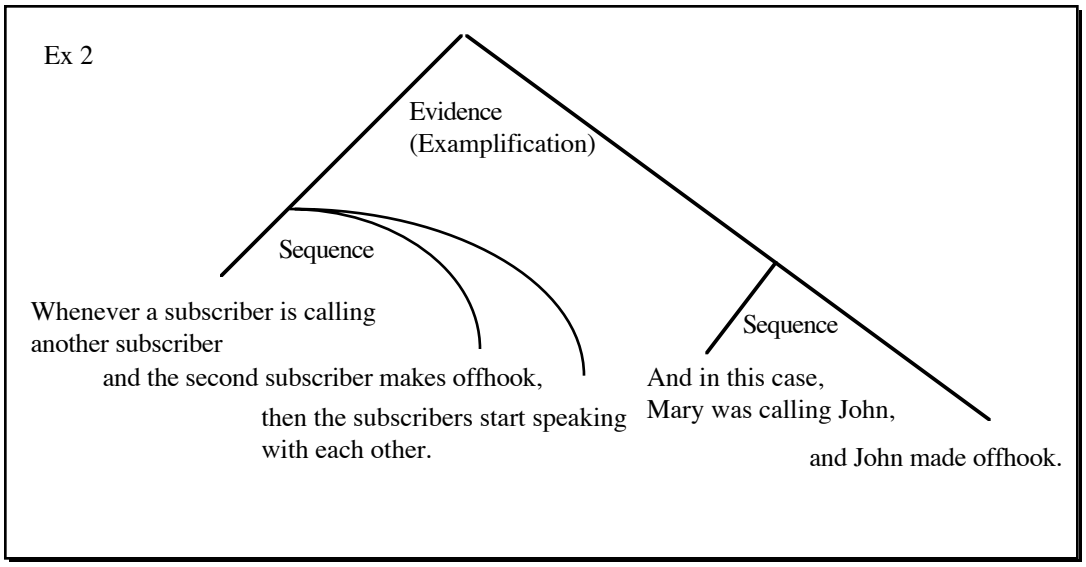


Conclusion: For Structural questions use the RST relation Evidence. Evidence proves the first clause with the second clause. We could for example use the cue phrases, for example, or and in this case (which is used in the next example) to increase the strength of the Evidence relation.

Causal question combining dynamic model and fact base C.

USER: *Why is John speaking with Mary?*

SYSTEM: Whenever a subscriber is calling another subscriber and the second subscriber makes offhook, then the subscribers start speaking with each other. And in this case, Mary was calling John, and John made offhook. (Warrant)

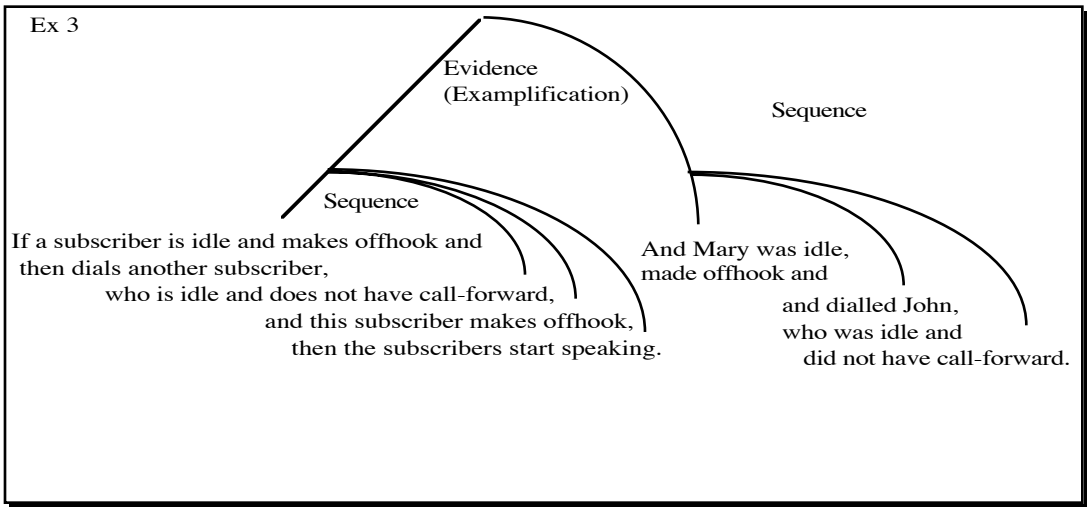


Conclusion: For Causal questions use RST relation Evidence extended with Sequence. Evidence proves the first clause with the second clause which contains a sequence with two supporting clauses. The Evidence relations is marked with the cue phrase And in this case. The Sequence relation can be marked with cue words increasing the temporal nature of Sequence. For example First and then.

Causal question with longer history trace combining dynamic model and fact base C.

USER: *Why is John speaking with Mary?*

SYSTEM: If a subscriber is idle and makes offhook and then dials another subscriber, who is idle and does not have call-forward, and this subscriber makes offhook, then the subscribers start speaking. And Mary was idle, made offhook and dialled John, who was idle and did not have call-forward. (Warrant)

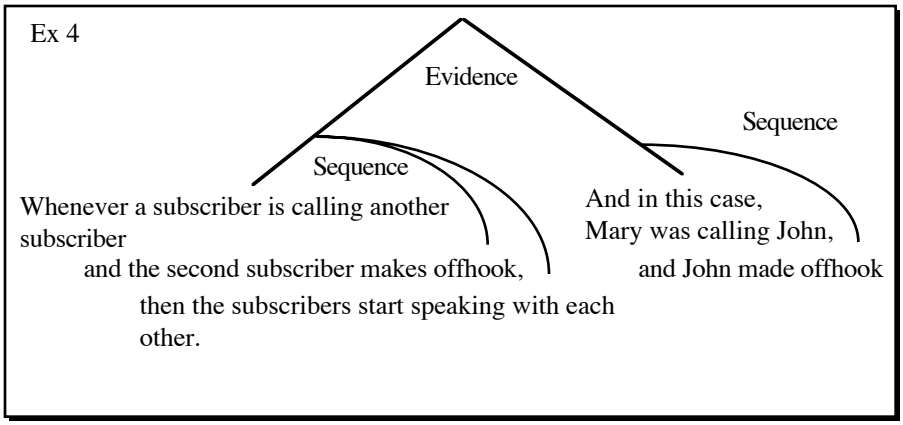


Conclusion: For Causal questions (with history trace) use RST relation Evidence extended with Sequence. Evidence creates a prove relation and Sequence orders the different clauses to be proven. Finally the proof is elaborated to give more details. Here we could augment the RST relation Sequence with the cue words. First ... Second....Then ... Finally

Procedural question combining dynamic model and fact base B.

USER: *What happens if John makes offhook?*

SYSTEM: Whenever a subscriber is calling another subscriber and the second subscriber makes offhook, then the subscribers start speaking with each other. And in this case, Mary was calling John, and John made offhook. (Warrant)

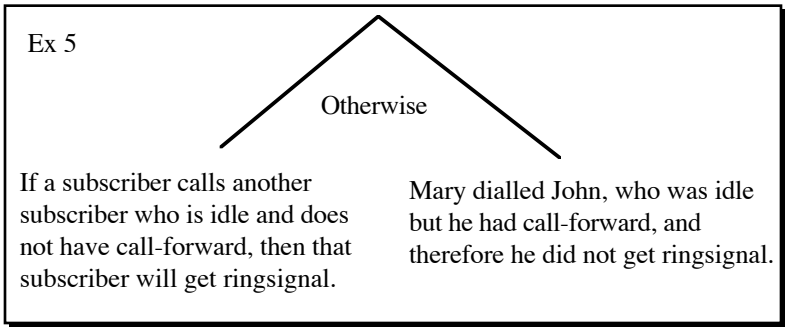


Conclusion: For Procedural questions (with history trace) use RST relation Evidence. (See above). Again the cue phrase And in this case is used to mark the Evidence relation.

Procedural question with rebuttal Based on fact base D.

USER: If Mary dials John, will he get ringsignal?

SYSTEM: If a subscriber calls another subscriber who is idle and does not have call-forward, then that subscriber will get ringsignal. Mary dialled John, who was idle but he had call-forward, and therefore he did not get ringsignal. (Warrant)

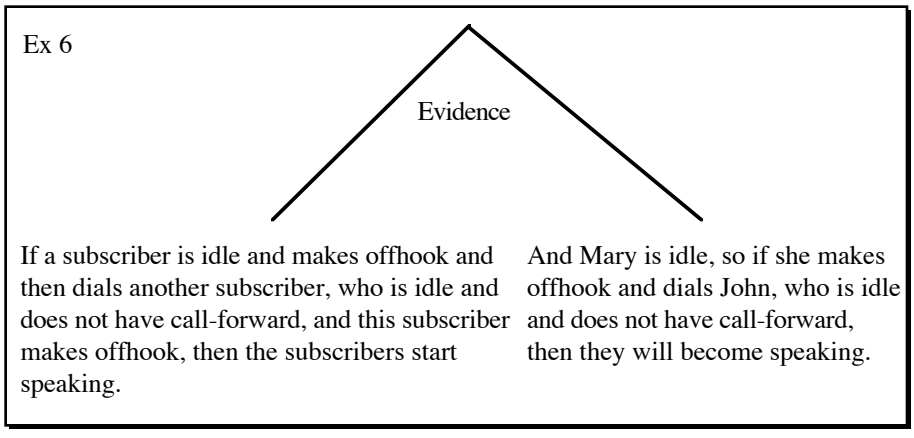


Conclusion: For Procedural questions with rebuttal, use RST relation Otherwise (extended with Sequence if necessary). The otherwise relations says "if A does not happen then C happens". Here we could use the cue word/phrase but or on the contrary, to stress the fact that the conditions of the first clause were not fulfilled.

Causal question combining dynamic model and fact base A.

USER: *How can Mary become speaking with John?*

SYSTEM: If a subscriber is idle and makes offhook and then dials another subscriber, who is idle and does not have call-forward, and this subscriber makes offhook, then the subscribers start speaking. And Mary is idle, so if she makes offhook and dials John, who is idle and does not have call-forward, then they will become speaking. (Warrant)

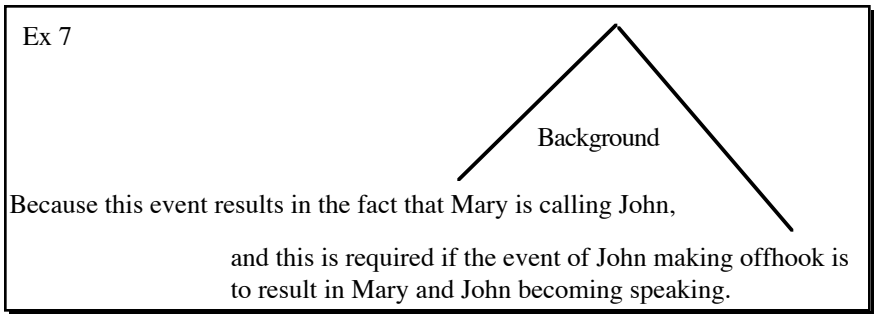


Conclusion: For Causal questions, use RST relation Evidence (extended with sequence). Cue phrases for the Evidence relation: For example or And in this case

Reasonal question combining dynamic model and fact base A.

USER: *How can Mary become speaking with John?*

SYSTEM: Because this event results in the fact that Mary is calling John, and this is required if the event of John making offhook is to result in Mary and John becoming speaking. (Grounds)



Conclusion: For Reasonal questions, use RST relation Background. The RST relation Background is marked by the cue phrase this is required if, one could also use the cue phrase: This is the reason that .

Summary from the examples

Based on the examples above, we can distinguish a set of useful RST relations i.e. evidence, sequence, otherwise and background.

- For Structural questions and for Procedural questions, use the RST relation Evidence.
- For Causal questions, use RST relation Evidence extended with Sequence when necessary.
- For Procedural questions with rebuttal, use RST relation Otherwise (possibly extended with Sequence).
- For Reasonal questions, use RST relation Background.

To implement these ideas practically each clause would need tagging information, marking what role it has and how it can be used together with RST to generate explanations. One heuristic for tagging is that rule based information must be placed on the left side of the RST relation evidence, and its corresponding fact base (instance) information must be placed on the right hand side.

We know from RST that the RST relations can be expanded following the RST structure. The RST relation Sequence could be added whenever the clauses fit into the over reaching main RST relation. Expanding for example Evidence with Elaboration and Sequence will preserve the coherence of the text as long as the fact base is not too large.

5. Concluding Remarks

In this paper, we have discussed explanation generation systems for conceptual models and proposed an architecture as well as design principles for constructing such systems. We have demonstrated how to integrate Toulmin's argumentation model together with a set of RST trees containing the RST relations: Evidence, Sequence, Otherwise and Background.

The obtained set of RST relations will support the designer of a natural language generation system to implement the correct type of explanations and to put e.g. cue phrases in the right places that strengthens the argumentative structure of the discourses.

The generated discourse structures will provide the requirements engineer and other user categories with an adequate amount of information and support a wide range of validation techniques.

In the examples in the previous sections, we have shown only explanations expressed in natural language. However, an adequate explanation component should combine natural language with graphical representations. These representations should provide a user with a graphical view of a system and allow her to obtain explanations of its various components by means of point and click interaction. When a user clicks at a component to have it explained, the system should display a menu of the types of explanations applicable for this particular component – these types are given by the five types of questions introduced in Section 4. The system will then display an explanation combining a graphical view with text.

What has not been shown in the paper is an implementation of the above ideas, which most likely will reveal new findings.

References

[Costal96] D. Costal, E. Teniente, T. Urpi, and C. Farre, "Handling Conceptual Model Validation by Planning", *7th International Conference on Advanced Information Systems Engineering, Springer*, 1996.

[Dalianis92] H. Dalianis, "A Method for Validating a Conceptual Model by Natural Language Discourse Generation", *CAISE-92 International Conference on Advanced Information Systems Engineering,* Loucopoulos P. (Ed.),Springer LNCS 593, pp. 425 - 444, 1992.

[Dalianis96] H. Dalianis, *Concise Natural Language Generation from Formal Specifications*, Ph.D. thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm, 1996.

[Dalianis97] H.Dalianis & P. Johannesson: Explaining Conceptual Models -- An Architecture and Design Principles. In David W. Embley and Robert C. Goldstein, (Eds.), ER'97, 16th International Conference on Conceptual Modeling, Lecture Notes in Computer Science No. 1331, pp. 215-228, Springer Verlag, 1997

[Gulla96] J. A. Gulla, "A General Explanation Component for Conceptual Modelling in CASE Environments", *ACM Transactions on Information Systems*, vol. 14, no. 2, pp. 297 - 329, 1996.

[Harel87] D. Harel, "Statecharts: a Visual Formalism for Complex Systems", *Science of Computer Programming*, vol. 8, no. 3, pp. 231 - 274, 1987.

[Hobbs85] J.R Hobbs: On the Coherence and Structure of Discourse, Report No. CSLI-85-37, October 1985.

[Höök93] H. Höök, *A General Description of the Delphi Language*, Ellemtel internal report, 1993.

[Kung93] D. Kung, "The Behavior Network Model for Conceptual Information Modelling", *Information Systems*, vol. 18, no. 1, pp. 1 - 21, 1993.

[Mann88] W. Mann and S. Thompson,"Rhetorical Structure Theory: Towards a Functional Theory of Text Organization". In *Text*, Vol. 8. No. 3, pp. 243-281.

[Moore91] J. Moore and W. Swartout, "A Reactive Approach to Explanation: Taking the User's Feedback into Account", in *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pp. 1 - 48, Dordrecht, 1991.

[Mylopoulos90] J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis, "Telos: Representing Knowledge about Information Systems",*ACM Transactions on Information Systems*, vol. 8, no. 4, pp. 325-362, 1990.

[Rolland92] C. Rolland and C. Proix, "Natural Language Approach to Conceptual Modeling", in *Conceptual Modeling, Databases and CASE: An Integrated View of Information Systems Development*, Ed. P. Loucopoulos and R. Zicari, pp. John Wiley, New York, 1992.

[Seltveit93] A. Seltveit, "An Abstraction-based Rule Approach to Large-scale Information Systems Development", in *5th International Conference on Advanced Information systems Engineering*, Ed. pp. 328 - 351, Springer Verlag, 1993.

[Toulmin59] S. Toulmin, *The Uses of Arguments,* Cambridge University Press, 1959.

[Zave84] P. Zave, "The Operational versus the Conventional approach to Software Development", *Communications of ACM*, vol. 27, no. 2, pp. 104 - 117, 1984.