Towards a Rhetoric of Active Design Documents

Clarisse Sieckenius de Souza
and
Ana Cristina Bicharra Garcia

Departamento de Informática
Pontifícia Universidade Católica—PUC-Rio
e:mail: clarisse@inf.puc-rio.br
bicharra@inf.puc-rio.br

Monografias em Ciência da Computação — MCC05/94

Abstract

Design documentation is costly, often inconsistent, and generally incomplete. Active Documents offer an active approach to capturing and delivering design rationale. In ADD, an existing implementation of an active document, what is stored and manipulated by system and user is not just information, but knowledge. Delivering knowledge at the interface level is a complex task, involving communicative strategies that fully explore documentation resources and adequately meet users cognitive needs. The implemented system generates information that can be displayed as an explanation, but it lacks rhetorical organization and structuring. Users, on their own, cannot be expected to (1) explore the full wealth of knowledge resources available in an active document, and (2) diagnose their own misconceptions about the design model and history. This paper discusses the potential for improving the input and output code, as well as for introducing dialogue structure, so that the knowledge usability problem in (1) and the cognitive paradox problem in (2) are made smaller.

Keywords
active document, ADD, rationale, design knowledge, rhetorical structuring of discourse and dialogue, user-interfaces.

Resumo

A documentação de projetos é cara, muitas vezes inconsistente, e geralmente incompleta. Os Documentos Ativos são uma abordagem ativa tanto para a captura quanto para a transmissão da lógica de projetos. No ADD, implementação real de um documento ativo, o que é armazenado e manipulado pelo sistema e seus usuários não é apenas informação, mas conhecimento. A transmissão de conhecimentos a nível de interface é uma tarefa complexa requerendo estratégias comunicativas que possibilitem a exploração completa dos recursos de documentação e o atendimento adequado a necessidades cognitivas dos usuários. O sistema implementado gera informações que podem ser apresentadas a título de explicação mas carecem de estrutura e organização retórica. Não se pode esperar que os usuários, por sua própria conta, (1) explorem toda a riqueza de conhecimentos disponibilizados em um documento ativo e, (2) diagnostiquem seus próprios mal-entendidos acerca do modelo e do histórico do projeto. Este trabalho discute o potencial de aperfeiçoamento dos códigos de entrada e saída do ADD, bem como a introdução de uma estrutura de diálogo, capazes de minorar o problema da utilidade e utilização do conhecimento mencionado em (1) e do paradoxo cognitivo mencionado em (2).

I. Introduction

Design documentation is costly, often inconsistent, and generally incomplete. The paradox emerging from the ratio between documentation essence and documentation perspicuity and exactness in most engineering projects has been approached by a number of researchers [Lee & Lai'91] in recent years. Four major approaches to capturing and delivering design rationale (design documentation) have been proposed: to record the sequence of actions — history-based rationale [Lakin, Wambaugh, Leifer, Cannon & Sivard'89], to record the arguments and issues raised during a project — argumentation-based rationale [Conklin & Begeman'88; Fischer, Lemke, McCall & Morch'91], to record the device behavior model — model-based rationale [Baudin, Sivard & Zweben'89], and to record the design model—active design documents [Garcia'92].

Powerful documentation tools have been proposed, among which is the Active Design Document — ADD [Garcia'92]. ADD is a dynamic computational representation of design rationale based on an adjustable underlying model of the device conception process. The documentation system operates by checking designers decisions against a network of dependencies among design parameters, constraints, criteria and a utility function to guide the decisions. Whenever a discrepancy arises between decisions expected by the system and provided by the user, the user is prompted to evaluate the discrepancy. In this situation, either the system's suggestion is accepted or the system's knowledge base is changed by the user, so that the discrepancy is resolved. At the end of the process, the system contains a consistent and complete documentation which can be queried by documentation users for a wide variety of purposes.

It is clear that in ADD what is stored and manipulated by system and user is not just information, but knowledge. Delivering knowledge at the interface level is a complex task, involving communicative strategies that fully explore documentation resources and adequately meet users cognitive needs The mere disclosure of ADD's knowledge bases — namely, the design model, design history, and design parameter value details — does not warrant a successful documentation use (or reading), if there is no apparent signaling of the relevance of and the relationship between the knowledge pieces.

As an example, consider the following scenario. A documentation user examines the system output screen and notices that the value decided for the refrigeration system parameter in a building heating, ventilation and air conditioning (HVAC) design process was *DOUBLE DUCT*. She thinks that the best option would be a single duct VAV and asks **why-not** a *SINGLE DUCT VAV*. The question is asked by menu selection on the parameter-value screen space as shown in Figure 1. The answer she obtains is a contrastive display of values associated with the S*INGLE DUCT VAV* and the *DOUBLE DUCT VAV* options for the refrigeration system parameter.

The attention of the user is drawn to alternative parameter values, since this is the screen location in which she started to probe the system's knowledge base. The design history display location is open to manipulation at user's discretion, but no explicit connection between knowledge presented in the two locations is shown. Thus, the user may never realize that perhaps some relevant information could be obtained from the design history. To make the point clearer, suppose that instead of freely navigating through ADD's screen display areas, the user could participate in the following dialogue:

U: Why not selecting *SINGLE DUCT VAV* as refrigeration system?

S: The refrigeration system was first decided as a SINGLE DUCT VAV. However, a later change in the design model (see it in the Design History display), where the importance of criterion X was decreased to Y, caused the best alternative for the refrigeration system to be a DOUBLE DUCT.

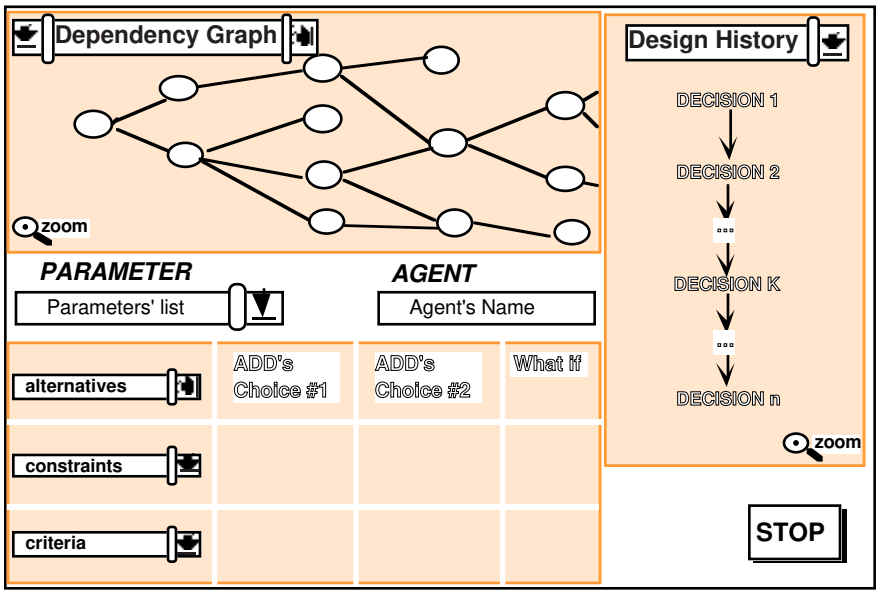


Figure 1: Current Active Document Explanation Display

The above example is intended to show that users cannot be expected to (1) explore the full wealth of knowledge resources available in an active document, and (2) diagnose their own misconceptions about the design model and history. These two problems will be focused in this paper, and a solution will be proposed on the basis of an underlying rhetoric of active design documentation. However, as pointed out by other researchers [Moore & Paris'93; Cawsey'93], rhetorical organization of information to be delivered, by itself, does not warrant effective communication. A continuous flow of interaction must be accommodated in an overall structure where discourse and dialogue phenomena can be used to generate optimized plans for knowledge transmission.

In order to face the challenge presented by (1), which we will coin as the KNOWLEDGE USABILITY PROBLEM, and (2), which we will coin as the COGNITIVE PARADOX PROBLEM, we propose that ADD's interface incorporates the following features.

- dialogue structure
- improved input code
- improved output code

In the next three sections of this paper, we will show an improved model of ADD's interface codes, and a detailed RST-based (Rhetorical Structure Theory [Mann & Thompson'87]) knowledge transmission planning. We will also point at important cognitive gains users can make by having insights about the system's reasoning strategies from integrated natural language and graphics displays. A discussion of the advantages of our approach to active design documentation reveals that the assets of model-based, history-based and argumentation-based design rationale documentation can be extracted from ADD by means of an adequate interface solution.

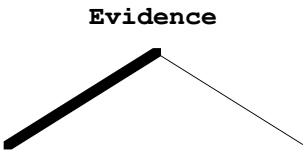
II. Turning Information into Knowledge

ADD's current interface includes text and graphics screen locations that can be manipulated by users via menu selection, mouse clicks, or window scrolling. Interface events are only loosely connected to each other, which results in potentially uncohesive screen displays that misguide users in search of design documentation. For example, the selection of a parameter in the Dependency Graph area (see Figure 1) causes display changes in the parameter alternatives, constraints and criteria value fields found in the screen area below the Dependency Graph, but causes no changes in the Design History display state. The example presented in section 1 illustrates that crucially important information related to parameter decision history may lie untapped in the history knowledge base. Thus, a structural integration of the various modes of information delivery in ADD is called for, in order to turn information into knowledge. The analogy we choose in order to express what we mean by this is a contrast between flat-surface building blocks which can only be put onto or beside each other (information-level) and Lego-like building blocks whose protuberant surfaces allow for and induce certain patterns of constructions (knowledge-level).

Although the relations between knowledge pieces can be treated at display state level only — for example, the Design History display automatically scrolls up or down as a parameter is selected in the Dependency Graph — this mechanism is still inadequate to express the causal relations expressed in the natural language dialogue above. Not even the animation of parameter changes along a temporal line on the Dependency Graph links and nodes could be expected to successfully get the message across to the documentation user. Consequently, we incorporate an additional expressive mode to ADD's interface — natural language text — in which perceived user's cognitive needs may be met by means of text planning techniques that maximize user's understanding abilities [Scott & de Souza'90]. Current solutions to intelligent multimedia applications emphasize the role of natural language in meeting communicative goals [Maybury'93]

As previously discussed [de Souza'93], unless input an output languages are not only co-referential [Draper'86], but also co-expressive (i.e. they have equivalent expressive power), breakdowns at interface level are likely to emerge frequently. Therefore, a finer quality in information delivery must be accompanied by a finer quality in information request. We do not claim that the input code must be full natural language text; it suffices that it can be mapped onto the same representation from which the output code extracts the relevant items to compose good natural language text and graphics. Textual explanations have global rhetorical organization and intentional and explicit redundancy with visualized items. Users are expected to (1) understand the system's interpretation of questions and the rationale for provided answers, and (2) navigate strategically throughout alternative information representation modes if the system's answers are not satisfactory.

Rhetorical Structure Theory [Mann & Thompson'87] has a number of attractive features for text planning. In particular, it is a belief-driven approach in which rhetorical relations between spans of text are specified in terms perlocutionary goals achievable through their inclusion in a message. This provides a powerful resource for reasoned information selection in a text generation system, as previously shown [Hovy'90, Scott & de Souza'90, Hovy'93, Moore & Paris'93]. In Figure 2, the specification schema for RST-relations is presented. Notice that by knowing the effect of a chosen relation on the reader's knowledge state, hypothesized sources of misconceptions may be directly addressed by a combination of remedial knowledge transmission in cohesive and coherent textual form.



N (Nucleus) S (Satellite)

```
Relation Name: Evidence

constraint on N: Reader might not believe N [...]
constraint on S: Reader believes S [...]
constraints on N+S:
Reader's belief in S increases his/her belief in N

effect: Reader's belief in N is increased
locus of the effect: N
```

Figure 2: RST relation specification schema [Mann & Thompson'87]

The text generation process adopted in ADD follows a traditional sequential approach [MacKeown'85] where text is first planned and then realized. Text planning in our application amounts to **deciding what to say and when to say it** based on (a) the dialogue flow (perceived focus and intention for information retrieval), (b) the system's knowledge base (inferred goals for documentation use), (c) screen layout data (reference to alternative modes of content representation), and (d) a mapping of (a), (b) and (c) contents onto RST schema. Text realization amounts to **deciding how to say it** by means of local morpho-syntactic rules that are necessary to instantiate a rhetorical schema (enriched by information items) as a natural language text span.

In Active Documents [Garcia'92], an underlying model of design provides the basis for representing adjusted design rationale. In other words, the active document model can be modified by users at design time, both at parameter-value level and at design-model level. The combination of the design dependency graph and history serves as a reasoning model from which users task-related beliefs (design background) and intentions (goals) can be reasoned about. Consequently, as users ask for explanation in a dialogue with the system, a mapping of questioned parameters onto the dependency graph and design history configures continuous or discontinuous tracks in the design territory. They provide focal information for text generation: continuous parameters are stacked on a focal structure, whereas discontinuous ones are held as candidates for a new focal structure.

We heuristically define a continuous track as a range of dependency of up to three nodes between parameters mentioned in two successive questions. Parameters which are more than three nodes apart from each other are defined as discontinuous. A finer and more precise decision about the definition of continuity is of course dependent on further experimental research. However, by construction, the Active Document underlying model incorporates designers perceptions of connectivity and relatedness among parameters, which we take as a trace of coherence and cohesion at dialogue level.

Users express their documentation needs by menu selection combined with screen display states and graphical object manipulation. Question types include: what, why, why-not, what-if, how, when and where. At the current state of ADD's interface redesign, only what, why-not, and what-about-if are being treated. The reason for it is that why, how, when and where questions can be directly derived from parts of rhetorical schema used in answering

what, why-not, and what-about-if questions. What-if questions are presently not treated in ADD's interface for lack of experimental data to support powerful generalizations about focus and presuppositions as represented in ADD's knowledge base. What-about-if questions, in turn, seem to provide clear focal information to support adequate text planning solutions.

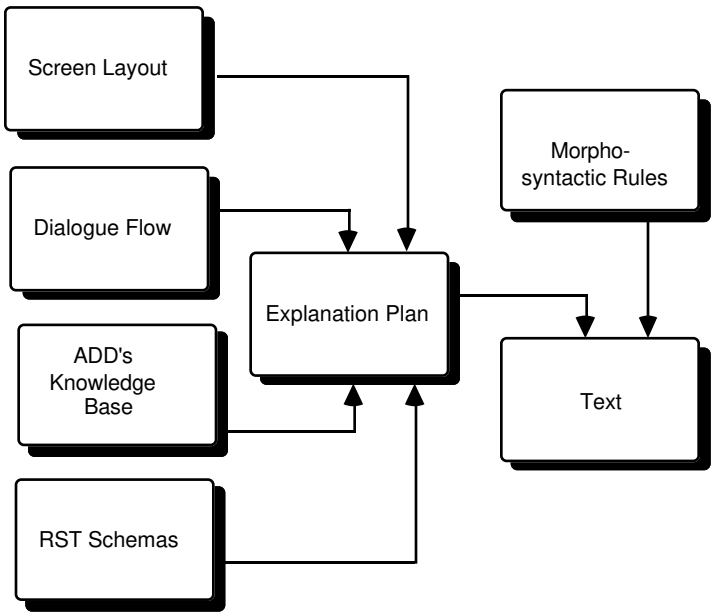


Figure 3: Information Sources involved in ADD's Text Generation Process

Direct manipulation of graphical objects and screen display states include possibilities that are summarized in Table I. The combination of menu selection and direct manipulation constitutes the improved input code, so that ADD's text generation process can be triggered (see Figure 3).

Element	Manipulation Options	Displayed Information
Dependency Graph	Zoom out	Complete parameter network + sub regions visited by all design agents (AMEBAS, [Garcia'92])
Dependency Graph	Zoom In selected Sub region	Detailed labeled parameter dependencies in specific design agent sub region
Design History	Zoom out	Temporal graph for design decisions, distinguishing patterns of backtrack or discontinuity between subsequent stages
Design History	Zoom In specific decision path	Detailed history of steps taken by design agents

Table I: Direct Manipulation of graphical objects and screen display states.

User's intentions and focus are inferred from heuristic rules summarized in Table II. Intentions correspond to belief states of the type used in RST specifications. For example,

if a user asks a what-about-if question like "What about Overall Cost if refrigeration system is DOUBLE DUCT?", as seen in Table II, we assume that the user's intention is to gauge the impact for X (Overall Cost) if Y (refrigeration system) had different value (DOUBLE DUCT). From the underlying design model we know that the choice of the refrigeration system is constrained by available ceiling space. If the focal stack does not include the ceiling space parameter, the system infers that the user believes that DOUBLE DUCT is a possible value for refrigeration system. Documentation shows, however, that ceiling space is insufficient for DOUBLE DUCT. Knowledge transmission must then dispel this potential misconception by displaying information about the impossibility of the user's hypothesis coming to be. The selection of RST relations in which user's belief that DOUBLE DUCT is a better alternative than the SINGLE DUCT one is replaced by a compare & contrast textual schema of the following kind.

Relation: Contrast
Nucleus 1: **Relation: Evidence**
Nucleus: Single Duct VAV was chosen
Satellite: Ceiling space is 40"
Effect of Evidence: Reader's understanding of S increases reader's belief in N.
Nucleus 2: **Relation: Condition**
Nucleus: Double Duct was Chosen
Satellite: Ceiling space should be 80"
Effect of Condition: Reader's recognizes that the situation in N depends on the situation in S.

Effect of Contrast: Reader recognizes the comparability and differences yielded by the comparison being made.

Moreover, a crucial linguistic mechanism needed for the integration of multimodal information delivery under natural language text is deixis — the ability to designate objects by pointing at or referring to them, instead of naming, describing, or characterizing them. In the example provided in section 1, we read the expression "see it in the Design History Display". The referent of it is to be encountered in extra linguistic *medium:* a visualization display segment on screen. The inclusion of such deictic references in natural language text is not a costly one in ADD, since the system has full control of knowledge sources and knowledge display modes.

An important advantage of including this mechanism in natural language answers to users questions is that there is substantial support for decisions about intentional redundancies in visual and textual codes. This feature is a means to reinforce users perceptions about relevant knowledge sources in the system. By the same token, users can learn connections between different modalities of expression and make guesses about some unexpressed but possible ones. This feature encourages them to probe the system's resources to a greater extent.

Question Type	Parameter(s)	Continuity	Intention	Focus
What	X	false	User wants to know the value of X	X
What	X	true	User wants to know the value of X and follow rationale	a stack where X is at the top, and previous foci are pushed down
why not	X	false	User wants to contrast current value with another value believed to be possible	X
why not	X	true	User wants to contrast current value with other values suggested by previous parameter values	a stack where X is at the top, and previous foci are pushed down
what about X if Y	X, Y	false (X is not continuous with previous parameters)	User wants to gauge the impact for X if (a number of) Y(s) had different value(s)	X, with sub foci on Z's (the set of parameters affected by changes in Y(s))
what about X if Y	X, Y	true (X is continuous with previous parameters)	User wants to gauge the impact for X if (a number of) Y(s) had different value(s)	a stack where X is at the top, Z's (the set of parameters affected by changes in Y(s)) are the sub foci immediately below X, and previous foci are pushed down

Table II: A Summary of Heuristic Rules to Decide on User's Intentions and Focus

III. Some Effects of Rhetorical Organization in ADD

In this section we will contrast ADD's current loosely structured interface codes with the ones proposed in this paper and show the effects of rhetorical active design documentation in terms of the KNOWLEDGE USABILITY PROBLEM and the COGNITIVE PARADOX PROBLEM mentioned in section 1.

The scenario for our example is one in which an architect is examining design documentation of a HVAC system. The project consists of a Direct Expansion cooling system using a Double Duct refrigeration and Boilers to produce heat. The HVAC system serves a 5-story office building located in San Jose. The building is of rectangular shape, whose dimensions are 200' X 300'.

Now, the architect thinks a SINGLE DUCT VAV for the refrigeration system would be a better option than what she sees on the screen — i.e. DOUBLE DUCT. So, she double-clicks on the refrigeration system parameter in the Dependency Graph display area and a dialogue box pops up with questions she can ask about that parameter's value. She selects a why-not question and types in the value she believes is better than the chosen one. (Figure 4)

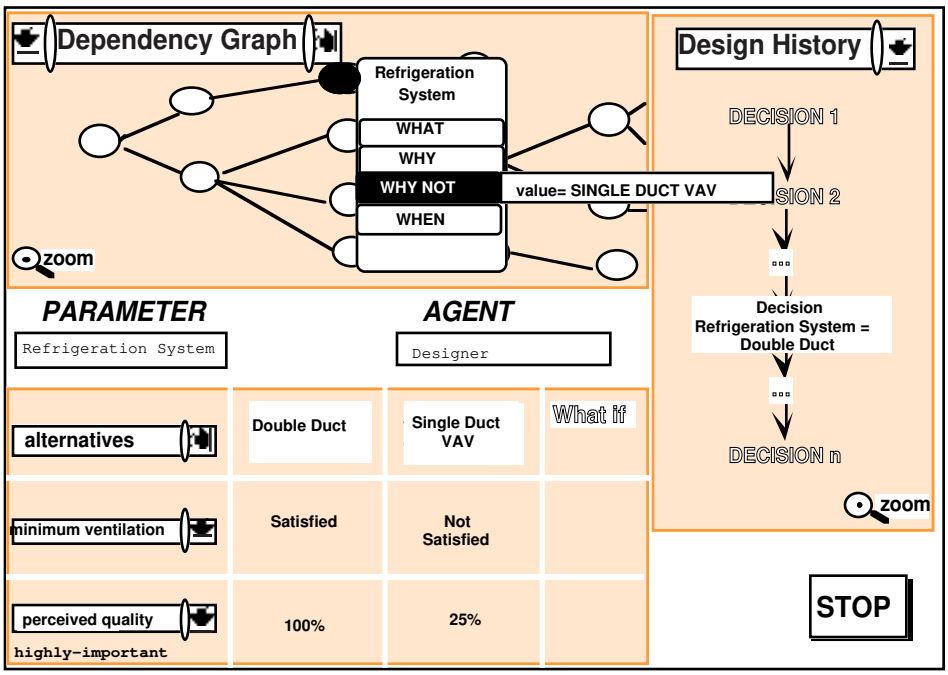


Figure 4: Architect's interaction with ADD

ADD's response is currently a visual contrast of actual and supposed values. However, the mere contrast of alternative values does not reveal any degree of commitment from the designer to the chosen value. This happens because, being a knowledge base system, ADD fills in the gaps left by designers with decisions inferred from the underlying parametric model of design which is constantly evaluating decisions as the documentation is constructed. Consequently, a value may appear in the documentation by way of three different paths:

- the designer has taken the decision about the parameter value, and has never returned to it;
- the designer has never taken any decision about the parameter value, which has been provided by the system;
- the designer has considered a number of alternative values for the parameter and the final choice is a result of changes imposed to the underlying parametric model of design initially fed into ADD.

When the question is asked, ADD knows what is the decision path associated to the parameter value choice at hand. It also knows if the alternative value suggested by the user in a why-not question has or has not been considered by the practitioner at design time. However, the current interface protocol is restricted to a very local evaluation of context, in which the disclosure of other relevant pieces of related information is neither needed, nor encouraged, in a successful question-answer interchange block.

Suppose that the real context of decision for the refrigeration system is such that the parameter was originally computed by the system as SINGLE DUCT (which, by construction, reflects the fact that this is the most usual choice for a similar project). But, for overriding reasons, the designer has chosen a DOUBLE DUCT, going as far as to change the underlying values of restrictions and design criteria, which have caused structural changes in the system's model.

If the architect knows that her choice for a SINGLE DUCT VAV was actually the system's first choice, some very important effects are attained. First, the architect understands that the designer's commitment with the DOUBLE DUCT alternative is very high, since even the underlying model has been modified to accommodate his preferred choice. Second, the confidence the architect has in ADD as an effective documentation tool is considerably increased, since she realizes that in the initial model the parameter choice was the same as she would have chosen herself. Third, knowing that a model-level change has been made at design time, the architect is aware that other discrepancies found during ongoing interaction may be due to this change. As a result, the architect may decide to probe the depth of the modification by querying the Design History, directly, or by just checking critical parameter values she wouldn't like to see changed for her own purposes.

In the opposite situation, if the architect wants to change the refrigeration system parameter value and discovers that the designer hasn't even bothered to evaluate it, she may directly jump into what-about-if questions to see if the system accommodates her own choice for that parameter — a SINGLE DUCT VAV. Further on, the architect may decide to evaluate which parameter values have been decided by the designer, before going into details about parameter-associated values, constraints and evaluation criteria.

RST-based natural language text incorporated to ADD's interface may achieve the above suggested degree of cooperativity by following a simple heuristic rule:

RULE 1: Answers to WHY-NOT questions must include TYPE of parameter value.
IF TYPE = decided THEN include HISTORY in compare & contrast schema
IF TYPE = {specified, inferred by ADD} THEN include TYPE in compare & contrast schema

The resulting RST schema is an expansion of the one shown in section 2. An elaboration about the parameter value (containing its type) or the circumstances in which it was decided (design history stages) are embedded into the evidence text span in natural language.

Relation: Contrast
Nucleus 1: **Relation: Evidence**
Nucleus: **Relation: Elaboration**
Nucleus: value chosen
Satellite: value type if {specified, inferred by ADD}
Satellite: known or believable constraining value
Effect of Evidence: Reader's understanding of S increases reader's belief in N.

Nucleus 2: Relation: Condition
Nucleus: value suggested
Satellite: constraining value
Effect of Condition: Reader's recognizes that the situation in N depends on the situation in S.
Effect of Contrast: Reader recognizes the comparability and differences yielded by the comparison being made.

Relation: Contrast
Nucleus 1: Relation: Evidence
Nucleus: Relation: Circumstance
Nucleus: value chosen
Satellite: value decision history
Satellite: known or believable constraining value
Effect of Evidence: Reader's understanding of S increases reader's belief in N.
Nucleus 2: Relation: Condition
Nucleus: value suggested
Satellite: constraining value
Effect of Condition: Reader's recognizes that the situation in N depends on the situation in S.
Effect of Contrast: Reader recognizes the comparability and differences yielded by the comparison being made.

IV. Concluding Remarks

It was demonstrated that an active document is able to record and deliver the content of design rationale [Garcia'92]. Its adjusted design model allows the generation of relevant pieces of information to justify design decisions. Nevertheless, the lack of an organized and rhetorical structure in the explanation transfers to the user the responsibility of interpreting and connecting the pieces of information displayed as rationale.

ADD's current implementation displays rationale content in three parts: parameter dependency graph, sequence of decisions, and parameter local evaluation. Users browse the displays to figure out the explanation. In spite of its enormous potential of use, due to the lack of explicitness about the underlying explanation model, active documents are destined to be underused. Besides, before requesting rationale, users actually need to understand what is the extent of their own misconceptions about a design.

This paper has shown the potential of improving the input and output code, as well as introducing a dialogue structure to diminish the cognitive paradox problem and to increase knowledge usability.

We improve the input code by providing an input language, not necessarily natural language, in which users can clearly express their needs. The output code receives one more layer of explanation structuring. A text explanation that organizes and integrates the pieces of information spread in the other display modes impacts the quality of the interaction. The deictic mechanism applied in the text allows this integration. It guides, but does not constrain, the users in indicating where to look for information. Consequently, documentation users can derive more information spending less time browsing the document.

Additionally, the introduction of a rhetorical structure reveals the designer's rationale for a project. Instead of making sense of the different pieces of information, users receive an explicit interpretation that reflects the way the system processed what the designer

developed. Therefore, a rhetoric of explanations minimizes the user's misconceptions about what has happened during design. Moreover, documentation users and active documents benefit from a dialogue structure that allows for a continuous flow of information in the rationale explanation.

The introduction of an explanation text that incorporates a rhetoric structure in active documents accumulates the gains of argumentation- and history-based rationale approaches without the drawbacks of both approaches. ADD, with this improved interface, provides documentation users with clearly stated arguments, without the static record and display behavior of argumentation-based models. Even in the current state of active documents, since the sequence of decisions has been stored, history-based models are included in active documents. The proposed enhancement provides one more level of interpretation to the decisions' sequence. For example, the textual explanation emphasizes the importance and complexity of some decisions, as a function of the frequency to which they have been considered by the designer. The sequence of decisions, in a higher abstraction level, can explain a designer's strategy. And the designer's strategy may help documentation users to understand not only a local level rationale (decision level), a global view of rationale (design level).

V. References

Baudin, C., Sivard, C. and Zweben, M. (1989) Using device models and design goals for design rationale capture. NASA Ames Research Center, *Technical Report* M.S.244-17

Cawsey,A. (1993) *Explanation and Interaction: the Computer Generation of Explanatory Dialogues*. Cambridge,Ma. MIT Press.

Conklin, J. and Begeman, M. L. (1988) gIBIS: A hypertext tool for exploratory policy discussion. *Proceedings of the 1988 Conference on Computer Supported Cooperative Work (CSCW-88)*, ACM, Portland, Oregon, pp. 140-152

de Souza, C.S. (1993) The Semiotic Engineering of User Interface Languages. *International Journal of Man-Machine Studies*. 39. pp. 753-773

Draper,S.W. (1986) Display Managers as the Basis for User-Machine Communication. in D.A. Norman and S.W. Draper (eds.) *User Centered System Design*. Hillsdale,NJ. Lawrence Erlbaum. pp.339-352

Fischer, G., Lemke, A. C., McCall, R. and Morch, A. I. (1989) Making argumentation serve design. *Human Computer Interaction*, 6(3).

Garcia, A. C. B. (1992). *Active Design Documents: A new approach for supporting documentation in preliminary routine design*. Ph.D. thesis, civil engineering department, Stanford University, CA, USA.

Hovy,E. (1990) Unresolved issues in paragraph planning. in Dale,R.; Mellish,C.; and Zock,M. (eds.) *Current Research in Natural Language Generation*. London Academic Press. pp. 17—46

Hovy,E. (1993) Automated discourse generation using discourse structure relations. *Artificial Intelligence* 63(1-2), pp. 341-386

Lakin, F., Wambaugh, H., Leifer, L., Cannon, D. and Sivard, C. (1989) The eletronic design notebook:Performing medium and processing medium. Visual Computer: *International Journal of Computer Graphics*, 5(4). pp. 214-226

Lee, J. and Lai, K. Y. (1991) A Comparative analysis of design rationale representations, *Technical Report* 91-121, MIT Center for Coordination Science.

Mackeown,K.R. (1985) *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge. Cambridge University Press.

Mann,W.C. and Thompson,S.A. (1987) Rhetorical Structure Theory: a Theory of Text Organization. *Technical Report* ISI/RS-87-190. Information Science Institute.

Moore,J.D. and Paris,C.L (1993) Planning text for Advisory Dialogue: Capturing Intentional and Rhetorical Information. *Computational Linguistics* 19(4). pp. 651-694

Maybury,M.T. (1993) *Intelligent Multimedia Interfaces*. Menlo Park, Ca. AAA Press / MIT Press.

Scott,D.R. and de Souza,C.S. (1990) Getting the message accross in RST-based text Generation. in Dale,R.; Mellish,C.; and Zock,M. (eds.) *Current Research in Natural Language Generation*. London Academic Press. pp. 47-74