ADD-GHS: A Structured Annotations Based Proposal for ADD's Extension

Flávio Miguel Varejão
email: fvarejao@inf.puc-rio.br

Ana Cristina Bicharra Garcia
Computer Science Department, UFF
email: bicharra@inf.puc-rio.br

Clarisse Sieckenius de Souza
email: clarisse@inf.puc-rio.br

PUC-RioInf.MCC38/96

Abstract: Many approaches have been proposed to document the design activity. ADD is a system that is based on the idea of transforming documents from a repository of data to a model of the design. This system provides a clearly delimited expressive language where a previous set of domain abstractions and active and communicative intentions is **fixed**. However, design activity is **flexible** by nature, in consequence of its own exploratory character. To solve this problem, we propose an ADD's extension that aims to increase its expressiveness as a design rationale tool. This extension allows the designer to include structured annotations to design documentation. These annotations are free text annotations organized by an index structure captured from the context where the annotation is made. This structure supports retrieval of annotations in different contexts from that the annotation is made.

Keywords: Design Rationale, Active Design Documents, ADD, Annotation

Resumo: Várias abordagens têm sido propostas para documentar a atividade de design. ADD é um sistema que se baseia na idéia de transformação dos documentos de design de repositórios estáticos de dados em modelos computacionais do próprio design. Este sistema fornece aos usuários uma linguagem de expressão, claramente demarcada, onde um conjunto prévio de abstrações sobre o domínio e intenções de ação e comunicação é fixo. Contudo, a atividade de design é por natureza flexível. Para contornar este problema, apresentamos uma proposta de extensão do ADD que visa torná-lo ainda mais expressivo como ferramenta para captura e recuperação de design rationale. Esta extensão se baseia na incorporação de anotações estruturadas à documentação de design. Anotações estruturadas são anotações feitas pelo usuário em linguagem natural organizadas por uma estrutura de indexação capturada do contexto onde ela foi feita. Esta estrutura suporta a recuperação de anotações em contextos diferentes daquele no qual a anotação foi capturada.

Palavras Chave: Design Rationale, Documentos Ativos de Design, ADD, Anotação

1 - Introduction

Typically, the design activity's products are detailed descriptions of the artifact and indications of how to build it. Considering the usual difficulties in the design of artifacts, documenting not only the design's product, but also how and why an artifact was designed can be extremely useful to:

- promote more reflection about the design's questions,
- make future changes in the product easy,
- permit the acquired knowledge to be reused in the automatic design of other products,
- increase the quality of future projects by providing basis for errors' reduction,
- support collaborative design [PSL95, GV96], since the different agents involved can understand and verify the reasons of a conflict among them,
- keep the institutional memory,
- improve the instruction of new employees.

The knowledge and reasoning underlying a design is known by the general term of *Design Rationale.* Aiming to help designers to document this knowledge, many approaches have been proposed to assist the process of design rationale capture and retrieval, such as:

- Record a history of design decisions [GGF94],
- Use an argumentative model to elicit the design rationale [MYBM91],
- Infer design rationale from design models [IFVC93].

An interesting approach for acquiring design rationale is adopted by active design documents. This approach is based on the idea of transforming design documents from a static repository of data to a computational model of design itself. From this model, an active document can generate rationale for explaining a design.

ADD (Active Design Document) [Gar92] is a computational system that uses this approach. It is based on the metaphor of having a computerized apprentice following the designer in the design process. The design process model is represented in ADD by a parametric dependency network. Each parameter in this network represents a design entity used in the artifact's design process. The parameters are linked according the dependency between them. However, these networks can not anticipate the entire knowledge necessary in a particular design because design activity is exploratory by nature.

In this paper, we describe an ADD's extension proposal that aims to increase its expressiveness as a design rationale tool. The idea underlying this proposal is to combine the advantages of argumentative models of design rationale capture [Shu96] with the advantages of a model based capture [GR96]. The extension supports the acquisition of opportunistic knowledge by allowing the designers to make structured annotations. Such annotations can be easily retrieved during the use of the documentation due to the proposed structure.

2 - Motivations for ADD's Extension

ADD's model can represent different kinds of methods and knowledge that are frequently used in design. As the design activity is a creative exploration process,

ADD's formal model can be changed to reflect new solutions and new design methods during the design process. Although ADD allows this type of change, there are circumstances where the designer may want to provide complementary information to ADD's model and history, without directly changing the model. These circumstances arise when:

a. The user wants to reinforce what is represented in the model. In this case, the model is appropriated to the design, but he wants to informally explain aspects not expressed by the model.

b. Even though the model is representative of the design decision (i.e., ADD's calculated value is equivalent to the designer's proposed value), it does not reveal an user's intention when proposing this value. For instance, the user can propose a value to a parameter to adjust another parameter. This information would not be explicitly represented in the model.

c. It is not convenient to express the new knowledge in the parametric network. In these cases, although the knowledge is important for the particular case and can be expressed by the network, it is completely opportunistic (usually, it will not be repeated in other designs). Examples are episodic knowledge [Vis95] and the knowledge coming from prototypes' results.

d. The new knowledge was not foreseen in the model and its inclusion in the parametric network would be hard and complex. For instance, a new type of electronic component could completely change part of a computer configuration model. Temporarily, the new component information and how it affects the design could be provided informally till the adjustment of the formal model.

e. The user wants to justify a model modification. In this case, he could intend to explain why a new calculation procedure is better than the former.

ADD's knowledge representation and reasoning provide a clearly delimited expressive language where a previous set of domain abstractions and active and communicative intentions is **fixed**. However, design activity is **flexible** by nature, in consequence of its own exploratory character. Therefore, the circumstances listed above denote an expansion necessity of ADD's expressive media, as a meta language (like (a) or (e)) or as an expressive extension in the same level as ADD's language (like (b) or (c)). Still, cases like (d) are constrained by basic computational limits, since no artificial language has a creative and innovator component that enables itself to generate new types of domain objects and their respective expressive modes and communicative constraints.

A way of capturing these types of complementary information is permitting that free text annotations be made by the designer. However, free text annotations are very restrict because they disregard important contextual information that can be captured from ADD's model and history. Such information can be used, jointly with the contextual information provided by the designer, as indexes of a structured annotations mechanism that will allow broad annotations' retrieval and utility. Hence, structured annotations are free text annotations organized by an index structure captured from the context in which the annotation is made.

In fact, the great advantages of integrating structured annotations to ADD are to provide an orthogonal information structure to ADD's formal model and to make it possible to capture the new knowledge informally.

Integration of structured annotations to ADD will allow it to capture, express and infer other types of relationships (structural, functional, temporal, organizational and others) besides dependency relationships. These relationships will be expressed by correlated annotations that provide an ADD's orthogonal information network. For instance, structured annotations will permit to explicitly represent the relationships between artifact's structure and function that are essential in many design systems [RG96, IFVC93]. Another example of using this type of annotation is to reflect the organizational structure of a multi-agent design environment (the institutional hierarchy and the process negotiation), which is extremely important in concurrent design [FKS95].

Moreover, ADD will be able to informally capture rationale that is not expressed in the formal model, in particular, the opportunistic design knowledge that can not be previously formalized in the model.

For all the above reasons, ADD's expressive power of representing, capturing and retrieving design rationale will be increased by incorporating the structured annotations mechanism.

On the other hand, the advantages that ADD's extension has to entirely argumentative models are the designers liberation of the arduous task of recording the complete rationale, since most part of this knowledge is embodied in ADD's model, and the easiest and broadest retrieval, since the informal knowledge can be structured according to the capture context and accessed by indexing and inference mechanisms.

3 - Annotations in ADD

We can make a parallel between our approach and GHS (General Help Systems) [Sou92, Sou93] approach. GHS is a structured annotation based system that provides support to search, navigation and learning in information networks, such as Internet. As in our problem, it is not possible to foresee, previously and completely, what are the users' needs in these networks.

GHS uses the structured annotation mechanism to make it easy for the users to provide and retrieve information about the available resources and, thus, to solve their problems by themselves. This is also the objective of our proposal. Understanding GHS, therefore, gives us indications about how design rationale capture and retrieval based on structured annotations should be done. Considering this fact, we called **ADD-GHS** the proposed ADD's extension.

Figure 1 shows a scheme of the GHS Annotation Structure. We can identify the central question considered by de Souza [Sou92, Sou93] relative to GHS. The structured part of the annotations executes certain linguistic functions such as indexing, minimal discourse setting and metalanguage (as indicated in italic in the figure). On the other hand, the free text part attends to the creative and innovative character of all human intelligent activity. These factors just go along with the aspects mentioned in section 2 about the contributions of structured annotations to ADD's environment.

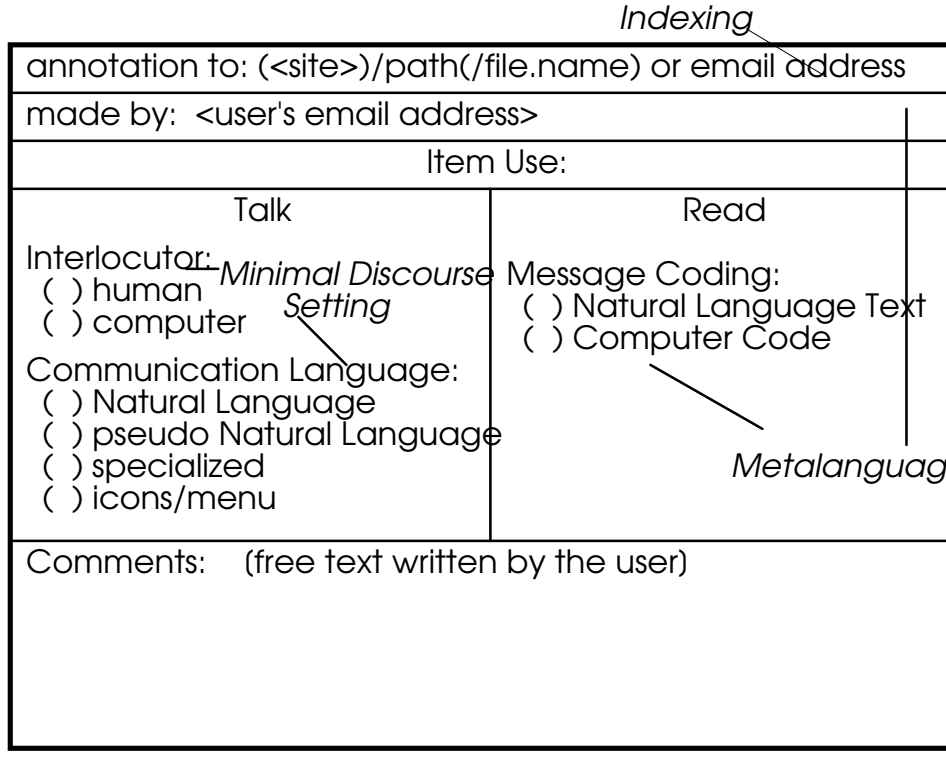

Figure 1 - GHS Annotation Structure

3.1 - Objects Referred by Annotations

Annotations in ADD can be classified according to the objects that are referred by the annotation. When the annotation refers to a single parameter, it is **specific** and, therefore, related to a single decision. This type of annotation expresses an extension of ADD's model in the sense of increasing the perspicuity of a domain element.

If the annotation refers to a group of parameters, it is **collective**. Usually, collective annotations express relationships that are not captured by the dependency network. This type of annotation expresses an extension of ADD's model in the sense of increasing the abstraction levels represented in the domain.

At last, if the annotation refers to a part of the model, that is not expressed by parameters, or to the model as a whole, it is **extra**. This type of annotation provides information that affects the use of part or of the entire model. Thus, they express an extension of ADD's model in the sense of increasing the perspicuity of the whole model or in the sense of augmenting the set of domain objects represented.

3.2 - Use of Annotations

An annotation can be used to attend different purposes. In **explanation** [CS96], contextualized annotations can be shown in answers of why, why not and how certain decision was taken. Another type of use is **to support designers' decision**. During the development of distinct projects, annotations are recorded in an annotation database that can be accessed whenever necessary. Thus, before taking a decision, the designer can retrieve knowledge related to that decision using a retrieval mechanism that infers from the context what is relevant to be presented to the user. Other way of using it is as a **knowledge acquisition source**. As the annotation database grows, it may be interesting to make an analysis of the annotations in order to identify what needs to be changed in ADD's formal model to reflect the design process in a more appropriated

way. This use intends to reduce the designers' necessity of using annotations to express design rationale. At last, it is still possible to imagine using annotations as a **machine learning** source of knowledge, even though this use would be very difficult to implement because it would involve text understanding.

4 - Representation of Annotations

We will describe in this section the representation model that is used to record annotations. We propose distinct indexing structures to specific, collective and extra annotations that will be presented during this section.

Before describing the indexing structures of annotations, it is important to mention that the capture of these information involves the composition between automatic acquisition from ADD's model and direct acquisition from the user. These mechanisms will be shown in section 5.

4.1 - Representation of Specific Annotations

Figure 2 introduces a scheme of the fields that compose a specific annotation register. The scheme is divided in three main fields: Context, Meaning and Comments. Although all these fields involve information about the context of the annotation, we called Context the field that represents only the contextual information directly related to ADD's model and history. The Meaning field provides knowledge about the contents of the annotation and is separated in the Focus and Objectives subfields. At last, the Comments field is the one where the free text can be inserted and is also divided in the Text and Keywords subfields.

The Context field is divided in many subfields. The first three fields identify the parameter referred by the specific annotation, its value at the moment the annotation is made and the number of times its value has been changed. The following field represents the subparameters that are active when the annotation is made and their respective values. The Class field identifies the domain category of the parameter. All these fields are important because they record the formal model configuration when the annotation is made. In particular, the number of times the value has been calculated indicates the agreement the designer has with the value. If it has been changed many times, it means that this parameter was quite evaluated before being adopted. Therefore, we presume that the compromise of the designer with this value is large.

The Author field identifies the person who made the annotation. This information is essential when there is more than one responsible for executing the design of the same system, in a multi-agent system, or when we are examining past projects. Then, we may recognize if the person who made the annotation is a renowned expert or an apprentice and, therefore, know the belief the annotation deserves.

The History field shows what was the designer's procedure (a record of the actions done over that parameter) before deciding to make an informal annotation. This record provides precise information about the sequence of events until the annotation.

The next field, Motivation, presents an indication of the designer's intent when making the annotation. This field demands a heuristic process of interpretation and classification during the annotation capture. In the case of Motivation, it is necessary to interpret the model and history context in order to classify the annotation within a set of possible motives. Since this is a heuristic process, the information expressed in the Motivation field is much more evidential than exact. On the other hand, it provides a more flexible annotation retrieval than the one obtained with the comparison between history fields only.

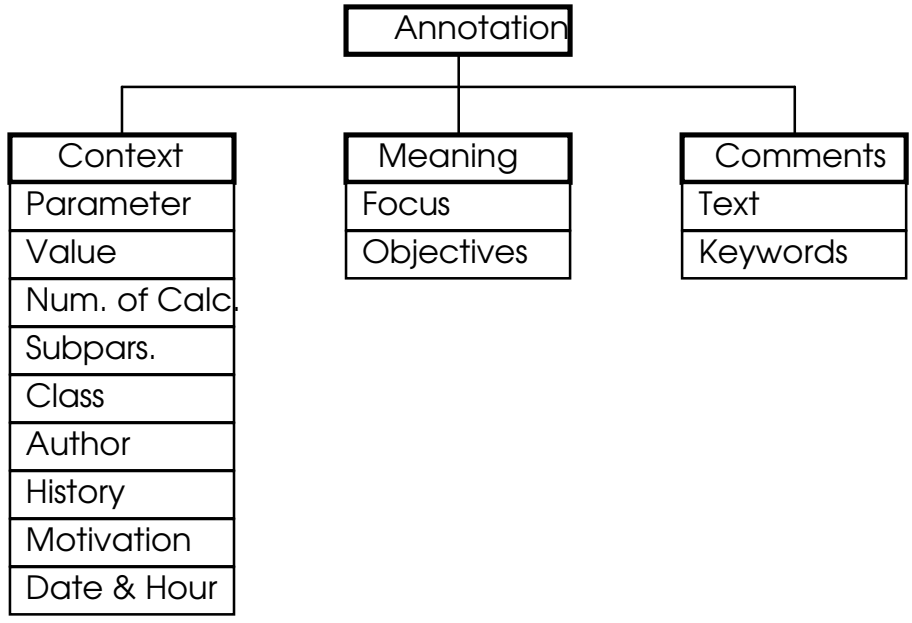

Figure 2 - Specific Annotation Scheme

The last Context field retains the date and hour when the annotation was made. This field provides a temporal organization of annotations. In case of annotations with conflicting information, it allows us to verify which one is more recent and, presumably, the prevalent. Furthermore, it also makes it possible for ADD to show the parameter sequence of annotations in chronological order. This type of information can be important to follow the evolution of designers' interpretation about the parameter.

The purpose of the Meaning field is to permit the designer to provide more information about the contents of the annotation. This information must be structured in such a way that ADD-GHS can use it formally to analyze the annotation relevance. The Focus subfield, therefore, allows the designer to express what are the topics, within a group of parameter's properties, that will be emphasized in the annotation. In the same direction, the Objectives subfield allows the designer to directly express, within a list of possible objectives (such as, *add information to the parameter*, *suppress information of the parameter*, *rephrase the information about the parameter, reinforce proposed value*, *refuse other values*), what are his intentions of doing an annotation.

Ultimately, the Comments field is where the designer writes the free text in natural language. The free text is written in the Text subfield and in the Keywords subfield are kept the morphological patterns that occur, over a delimited frequency, in the text. These patterns are used as indexes to annotation retrieval.

4.2 - Representation of Collective Annotations

As can be seen in figure 3, the collective annotation scheme has a similar structure to the specific annotation. Such as the former, the collective annotation scheme is divided in the fields Context, Meaning and Comments. However, the main differences are in the subfields of the Context field and in the contents of the subfields of Meaning.

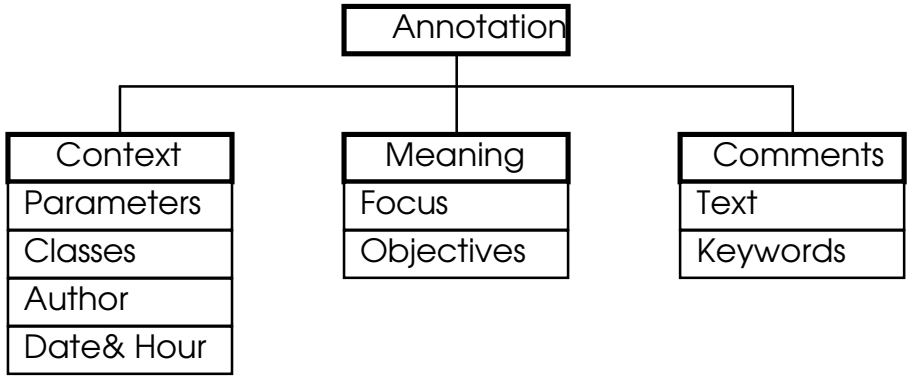

Figure 3 - Collective Annotation Scheme

In the collective annotations scheme the Context field is divided in the Parameters, Classes, Author and Data and Hour subfields. The Parameters and Classes subfields represent, respectively, the parameters and classes referred by the annotation. The Parameters subfield allows the designer to express precisely which parameters are related to the annotation. The Classes subfield allows that annotations refer to groups of parameters related by the parameters classification hierarchy.

The Meaning field is also divided in Focus and Objectives. The Focus subfield expresses the feasible relationships between the parameters referred by the annotation. These relationships can be:

- *Dependency*: when the annotation supports or contradicts the dependency relationship between parameters, or justifies a parametric network change.
- *Abstract*: in the case where parameters are aggregated to represent a new abstraction.
- *Behavioral*: if the annotation justifies a designer's procedure. For instance, it can indicate that a sequence of parameter's values was settled to define another.
- *Functional*: when the parameters are related according to a function of the artifact or of its parts.
- *Structural*: in the case where the parameters are related according to the artifact's structure.
- *Organizational*: when the parameters are related according to organizational aspects of the institution or of the design team.
- *Contextual*: when the parameters are correlated by an environmental context that affects the design, construction or use of the artifact.

The Objectives subfield allows the designer to express if the annotation aims to *add information to the model*, *suppress information of the model* or *rephrase the information about the model*.

4.3 - Representation of Extra Annotations

The only differences of the extra annotation scheme (figure 4) to the collective one are the absence of the Parameters and Classes subfields in the Context field and in the meaning of the options in the Focus subfield.

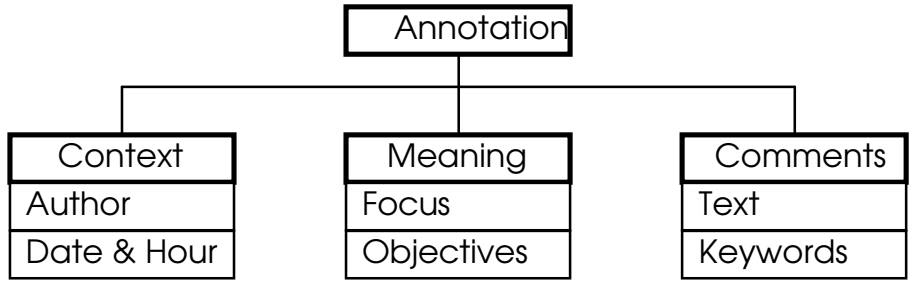

Figure 4 - Extra Annotation Scheme

The Focus subfield expresses the feasible topics that an extra annotation can emphasize. These topics can be:

- *Causal*: in the case of insertion of new knowledge about the design method.
- *Abstract*: if the designer wants to present arguments in more abstract levels than the model's level.
- *Validity*: when the annotation supports or contradicts the validity of the entire model.
- *Functional*: when the annotation introduces a new artifact's or component's function.
- S*tructural*: when the annotation inserts a new structural element to the artifact.
- *Organizational*: if the annotation emphasizes organizational aspects of the institution or of the design team.
- *Contextual*: when the annotation shows a situation where the environmental context affects the design, construction or use of the artifact.

5 - Capture of Annotations

The annotations' capture in ADD-GHS is started by the designer at the moment he needs to make an annotation. The capture consists of acquiring the text of the annotation from the designer and the information that will be used as indexes of this annotation. ADD-GHS executes the annotation's indexing using three methods. The first method consists of acquiring, from ADD's model and history, information about the context where the annotation was made. The second is based on the requirement of information from the designer (in a kind of template that should be filled). At last, the third method automatically acquires indexes from the text written by the user. These indexes are morphological patterns that happen in the text over a delimited frequency (as, for example, in WAIS program).

It is important to emphasize that the designer's required information is optional (the user can not fill the template or part of it). Moreover, the template intentionally requires a small effort from the user. So that, the user will not be discouraged to provide the information. In a particular case when the designer does not provide any additional information, the annotation will be only indexed by the data extracted from

ADD's model and history, and by the morphological patterns found in the text. Consequently, this annotation can only be retrieved through a search for these data and patterns.

In general, the subfields of the Context field can be directly obtained from ADD's model and history. The exception is the Parameters and Classes subfields of the collective annotations. In these subfields, the designer selects from lists containing the parameters and classes of the model. A similar procedure is used to define the Focus and Objectives subfields. The designer can specify from lists what are the topics and objectives of the annotation. However, in these subfields, instead of selecting options from the list, it is still possible to provide keywords to define the focus and the objectives of the annotation. At last, the Text field is freely provided by the designer and the field Keywords is automatically obtained by a frequency analysis of the morphological patterns found in the text.

6 - Retrieval of Annotations

As already mentioned, annotations can be retrieved to explain the design or to support designer's decision. At any moment the user can invoke the retrieval mechanism to request annotations. ADD-GHS allows the user to optionally provide information on the contents of the annotation he wants to retrieve. This information can be provided by partially or totally filling a template with the subfields of the annotations. This information permits ADD-GHS to use the annotation structure and the design context to retrieve the annotations that are relevant to the current context. If he does not provide any information, ADD-GHS will only use the current configuration of the model and the history as indexes to retrieve the annotations.

The ADD-GHS contextual retrieval mechanism is composed of:

- a database containing the annotations,
- a mechanism to query this database,
- a knowledge base composed of the parametric network and the parameters classification hierarchy,
- an inference mechanism that navigates through the knowledge base and identifies the annotations that are relevant to the intended use.

The database is indexed by the information obtained during the annotation's capture. The query mechanism permits the database to be accessed by the usual search procedures. The knowledge base has references to the annotations (parameters and classes know which are the annotations that refer to them). The inference mechanism consists of three procedures:

- *Relevance Identification*: procedure that compares, using an evaluation function, the annotations' contexts of capture and use, and selects the ones that are relevant to presentation.
- *Generalization*: procedure that expands the use of an annotation that refers to a specific parameter to a different parameter.
- *Specialization*: procedure that extends the use of an annotation that refers to a class of parameters or a set of parameters to its subclasses or instances.

Likewise analogy based design [QG96], the generalization and specialization processes can occur at superficial and deep knowledge levels. When they occur at the superficial level, the basis of the process is the categorical relationships between the parameters.

Thus, the process only considers the parameters classes and their relations in the domain hierarchical structure. On the other hand, when they occur at the deep level, the processes are more elaborated and involve an analysis of the parameters decision process.

7 - Conclusion and Future Work

In this work, we proposed an extension of the ADD's system that combines the ADD's formal model and history of design process with informal arguments obtained through structured annotations. The great advantages of this extension (called ADD-GHS) over ADD is to permit the documentation of the opportunistic knowledge and to provide an information network, orthogonal to ADD's model, that can capture non dependency relations. Besides, we identify that ADD-GHS has linguistic roles that permit the expansion of ADD's expressive language. It acts like an expressive extension in the same level as ADD's language or in the metalanguage level. On the other hand, the use of structured annotations extends the use of informal arguments in comparison with argumentative models. Furthermore, the combination with ADD's formal model reduces the work the designer should realize to document his rationale.

Although we have no certification about the entire appropriateness of our proposal, ADD's model gives us positive evidences regarding the feasibility of the used representation in case of single agent systems. Notice, however, that the same can not be said about multi-agent systems. In this case, as we use a fairly generic representation, only the analysis of cases generated by ADD-GHS will determine the appropriateness of the proposal and give us indications about what should be changed in the model.

A preliminary version of ADD-GHS was already implemented in the ADDProc system. ADDProc is a multi-agent system that aims to support and document the design of process plants of oil platforms. Even though the implemented version of ADD-GHS has only part of the whole proposed here, it contributes as a demonstrative of the feasibility of the proposal.

The next step in this research will be to complete the implementation of ADD-GHS and to execute some tests to evaluate the appropriateness of our proposal. In particular, with these tests we intend to analyze:

1. the interaction between ADD-GHS and the users - verify if the users can understand the functionality of ADD-GHS and how to use it to capture and retrieve annotations.
2. the expressiveness of the used representation - compare the usability rate of the option Keywords in the Focus and Objectives fields, during the capture of annotations, with the rate of the other fields. This comparison permits us to verify if the options of our representation are appropriated.
3. the appropriateness of the retrieval mechanism - verify if ADD-GHS retrieves the information requested by the user and existent in the annotations' database .
4. the usability rate of ADD-GHS - verify how much the designer uses ADD-GHS in comparison to ADD's formal model.

In the future, the structured annotations' mechanism could be used as a source of feedback to ADD's model. With the increasing use of this mechanism, a lot of knowledge will be additioned to ADD through the annotations. From time to time, it will be convenient to analyze these annotations to identify what are the needs of the

users and what can be updated in ADD's parametric model. During this analysis, we can also use the morphological pattern matching mechanism to identify cultural patterns of design. As a result, the structured annotations' mechanism will also work as a source of knowledge acquisition for ADD.

In the same way, we understand that the overuse of this mechanism can imply in a low suitability of the parametric model of ADD. In case of this overuse be evident, it should be considered to entirely remodel ADD's parametric model.

8 - References

[CS96] - da Cunha, C.K.V. & Souza, C.S. *"The role of Explanation Systems in Multi-Agent Systems"*. **Submitted to International Conference on Intelligent User Interfaces'97, Orlando, Florida,** 1996.

[FKS95] - Finger, S., Konda, S. & Subrahmanian, E. *"Concurrent design happens at the interfaces"*. **AIEDAM**, 9, pp. 89-99, 1995.

[Gar92] - Garcia, A.C.B. *"Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design"*. **PhD Thesis**, Department of Civil Engineering, Stanford University, August, 1992.

[GGF94] - Ganeshan, R., Garret, J. & Finger, S. *"A framework for representing design intent"*. **Design Studies**, 15, 1, January, 1994.

[GR96] - Gruber, T.R. and Russel, D.M. *"Generative Design Rationale: Beyond the Record and Replay Paradigm"*. In Moran T.P. & Carrol, J.M. (eds.) , **Design Rationale: Concepts, Techniques, and Use**, Lawrence Erlbaum Associates: Hilsdale, NJ, in press.

[GV96] - Garcia, A.C.B. & Vivacqua, A.S. *"The Use of Active Design Documents to Assist Conflict Mitigation in Concurrent Engineering"*. To appear in the **Third Conference on Concurrent Engineering Research and Applications (CE96/ISPE)**, Toronto, Ontario, Canada, 1996.

[IFVC93] - Iwasaki, Y., Fikes, R., Vescovi, M. & Chandrasekaran, B. *"How Thimgs Are Intended to Work: Capturing Functional Knowledge in Device Design"*. In **Proceedings of XIII IJCAI** - Chamberry, France, Vol. 2, pp. 1516-1522, 1993.

[MYBM91] - MacLean, A., Young, R., Bellotti, V. & Moran, T. *"Questions, Options and Criteria: Elements of a Design Rationale for User Interfaces"*. **Humam-Computer Interaction**, 6, pp.201-250, 1991.

[PSL95] - Peña-Mora, F., Sriram, R.D. & Logcher, R. *"Conflict mitigation system for collaborative engineering"*. **AIEDAM**, 9, pp. 101-124, 1995.

[QG96] - Quian, L. & Gero, J.S. *"Function-Structure-Behavior Paths and Their Role in Analogy-Based Design"*. **Technical Report**, Key Centre of Design Computing, Department of Architectural and Design Science, University of Sidney, 1996.

[RG96] - Rosenman, M. & Gero, J.S. *"The Role of Functional Reasoning in Design"*. **Technical Report**, Key Centre of Design Computing, Department of Architectural and Design Science, University of Sidney, 1996.

[Shu96] - Shum, S.B. *"Design Argumentation as Design Rationale"*. In Kent, A. & Williams, J.G. (eds.), **The Encyclopedia of Computer Science and Technology**, New York, Marcel Dekker, Inc., 1996.

[Sou92] - de Souza, C.S. *"Augmenting Informativeness and Learnability of Items in Large Computer Networks"*. **Technical Report** No. CSLI-92-164, Center for the Study of Language and Information, Stanford University, May, 1992.

[Sou93] - de Souza, C.S. *"Communicative Indexing in Network Resource Discovery Systems"*. **Technical Report** No. CSLI-93-182, Center for the Study of Language and Information, Stanford University, December, 1993.

[Vis95] - Visser, W. *"Use of episodic knowledge and information in design problem solving"*. **Design Studies**, 16, 2, April, 1995.