Design Rationale for Collaboration: The Active Document Approach

Ana Cristina Bicharra Garcia
Mark Stefik
H. Craig Howard

e-mail: bicharra@inf.puc-rio.br

Submitted to Research in Engineering Design Journal.
March 1994

We present a new approach for supporting documentation in routine design tasks using computationally active documents. This approach substantially reduces the costs of collaborative design. The major costs are in the time required to check and maintain consistency between design decisions and requirements, the time to explore a large design space, and the time lost in communication delays between people with different expertise. By reducing these costs, the active design document approach supports the creation, use and revision of design documents. We demonstrate the feasibility of the approach with a detailed case study of its application to the preliminary design of heating ventilation and air conditioning (HVAC) systems. Evaluation of a prototype system confirms that where the approach is applicable, it can significantly improve the quality and reduce the cost of design documentation.

1. Introduction

Design documentation is expensive, often inconsistent, and generally incomplete. Even for ordinary designs, the quantity of information that might be included as design rationale is large. But just what kinds of information should be included in design rationale? To this question we offer an answer that is radical in its departure from most previous work on design rationale, but elegant in its simplicity. We use the term rationale to refer to the contents of design documents. In any specific context, it means the information that a user needs to answer a question. Our approach to defining rationale is open-ended. Rather than beginning by specifying exactly what counts as rationale in abstract terms, we expand the term to encompass whatever information arises in design documents or is used in the communication about design.

Design is a collaborative process. All of the design collaborators are faced with costs, and many of those costs are related to communication. Design rationale is best understood as supporting the information needs of collaboration, not only for the generation of designs, but also for the other collaborative processes such as testing whether specifications are met, testing whether alternatives are feasible and have been explored, and negotiating changes. To recognize opportunities for computer support we must understand those needs, both in their breadth and in their economics. We must identify what the collaborators are doing and what they need to communicate about. We evaluate the effectiveness of computer support for design rationale in terms of the economics of both the production and the use of documentation.

For manually produced documents, consistency of the design is maintained by the designer. As designs get larger, it becomes more difficult to check them for internal consistency. A small heating, ventilation and air conditioning (HVAC) system design may involve two hundred parameters. Due to this large number of parameters and the dependencies among them, it is common to find explanations or even decisions that contradict the set of design specifications.

In addition, designers and clients do not start out knowing exactly what is needed. They develop and negotiate the set of specifications for a project and record it in design documents. Design documents are constantly revised to incorporate new designed parameters and to update the project with new specifications. Documents are created, used, and revised in what we might call the document lifecycle.

Most automatic documentation systems do little to support the **communication** needs of **collaborative design**: to reveal design assumptions, to provide a consistent explanation of the design, to maintain consistency between the explanation and the design, and to provide feedback on changes in specification

The active design document (ADD) approach uses a computer-based design model to drastically alter the costs of producing and using design documentation including:

1. the cost of checking consistency: The parametric design model (with its constraint checking) shifts the work of checking consistency from manual work by the designer to automated work by a computer.
2. the cost of exploring a design space: The initial parametric design model guarantees that most decisions can be automatically documented. Consequently, the designer has more time to try different alternatives.
3. the costs of communication: The active design approach makes engineering constraints and design specifications both explicit and active. Because this information is in the active document, participants can more readily answer questions that would otherwise require communication delays for meeting with their collaborators.

In this paper we discuss the ADD approach as an opportunity to improve engineering practice. The fundamental difficulties that underlie the quality of design documentation, both in terms of its quality and usability, have to do with the interacting costs of design and communication, which are addressed by our approach.

Section 1 describes the ADD approach to design documentation. Section 2 presents an active design document showing that the approach is feasible. Section 3 explains the experiments and our evaluation of the approach. Section 4 discusses related work in the area of design rationale. Section 5 presents our conclusion and shows how the results depend on certain properties of the domain.

2. The Active Design Document Approach

The active design document approach (ADD) uses a parametric model of design decisions. This model defines the terms both for design and explanation, and determines what is meant by design consistency. In this approach, design is not a separate activity from documentation. Instead, design consists of establishing parameters in the model. Because the model is an active document, explanations are generated automatically on demand and consistently with the model. Because the model allows people other than the designer to simulate and try other design alternatives, it sidesteps some of the costs and delays of communication.

Four assumptions about documentation use are important to the active design document approach (Garcia, 1993):

- “Explanation Completeness” Hypothesis: The same model that generates a design should be able to explain it.
- “Explanation Visibility” Hypothesis: This hypothesis says that the appropriate terms for describing design rationale are accessible through document analysis and interviews techniques.
- "Explanation Kernel" Hypothesis: A small set of types of parameterized document suffices to provide form and context for most document queries.
- "Explanation Diversity" Hypothesis: Design explanations vary according to the perspective and goals of the person requesting it. However, all explanations can be derived from the same model that originated the design.

Our approach to design rationale is specific to routine preliminary design. We applied our approach to the preliminary design of HVAC systems. Before presenting the approach, this section describes the documentation lifecycle emphasizing the role of documents in the preliminary design stage of building designs. We also describe the design model for the domain of heating, ventilation and air conditioning systems. Finally, we describe the active design document approach in detail.

2.1. The Documentation Lifecycle

Documentation is an issue for all areas of engineering. However the costs and demands of creating and using documents are particularly critical for designs of unique artifacts such as the design of a building. Because building designs are almost always one-of-a-kind (one client and one building), there is no opportunity to amortize designs over thousands of copies. It is in this extreme setting that the active document approach was developed, and in which we have tested it using a prototype system. In this section we describe the building design process (Luth, 1991) emphasizing the creation, development, use, and revision of design documents.

Building designs are usually developed in four phases: conceptual design, preliminary design, design development, and construction drawings. Each design stage produces documents that address different issues. During the conceptual design phase, the design team develops the building's concepts and requirements. This phase produces a set of documents containing the requirements governing the design. In the preliminary design phase, each design trade develops a set of alternatives that satisfy the set of design requirements developed during conceptual design. After negotiation among design participants and approval by city inspectors, the design is detailed. Therefore, the document generated during preliminary design represents the design that is to be detailed. The detailed design drawings developed during design development and approved by the city inspectors are further specified to contain construction instructions. This last document also needs to be approved by the city.

In preliminary design, HVAC system designers receive a document with initial requirements to be addressed. These requirements are frequently modified during the project by the various design participants including owners, architects and structural engineers. The changes reflect the evolving aspect of design. This creation, use, and revision of documents, repeated for each design phase, consist the document lifecycle.

Based on this evolving set of requirements, HVAC system designers are requested to propose a design in a very short period of time varying from one week to a month. Even for quite ordinary designs, the quantity of information that might be included as design rationale is large due to the large number of parameters that need to be documented. In addition, designers need to keep track of the dependencies among these parameters. For example, a typical HVAC system design for a small commercial building contains 150 to 200 parameters to be identified. Most parameters depend on three or more parameters. This information would be expensive to document completely. In practice, only a small portion of it is documented. For designs developed for bidding on contracts, it is typical that only a fraction of the designs will actually lead to funded projects. In such situations, there is even less incentive to document.

Later, whenever questions about the design are raised, designers regret not having spent time properly documenting the design because they must spend time recalling and explaining the rationale for the design to documentation users. This perception of value and attitude presents an important requirement on documentation systems: any system supporting documentation should not require much time from designers. Design must remain the designers' main activity.

Figure 1 presents a piece of the design space containing the parameters, their dependencies, and the design space portions explored by different users. Each portion of the design space explored by a user is called the AMEBA diagram of the interaction. The name AMEBA comes from the amorphous shape of the diagram. The diagrams as well as the network represent the design space explored or to be explored to generate a preliminary design of HVAC systems.

Figure 1: AMEBA diagram illustrating the portion of the design model being discussed. Nodes in this figure correspond to design parameters in the HVAC parametric model. Links represent dependencies, such as derivation paths and constraints. See the legend for the interpretation of the shaded regions.

2.2. The active design document approach

The active design documentation approach addresses the documentation overhead, instability of specifications, and consistency issues of preliminary design documents by using an initial design model able to generate design decisions and rationale most of the time. This section describes the approach and the way it addresses these documentation issues.

The design model we created for the HVAC domain does not always generate the same decision as a particular designer. This can happen for any of several reasons. The case may be outside the coverage of the model, the designer's experience may differ from that encoded in the model, or there may be an error in the model. In such cases, designers alter the design model.

Whenever a user changes specifications, the model propagates the effects. The active document then receives the new HVAC parameter values and evaluates them in terms of the active constraints and criteria. Consequently, it checks the local impact caused by the change in the design specification, such as a violation of an architectural constraint. In addition, it forces the new value and propagates it to the set of influenced parameters.

ADD allows documentation users to discover the designer's assumptions, as well as to support some exploration of design space even when designers are not available. ADD's available design model can simulate what would be the design under different circumstances. Consequently, the design choices can be shared by designers and clients. Answers for questions about a design are *generated* instead of *recorded* by the document.

In summary, an active design document has the following attributes:

- has an initial model of the domain;
- has a decision-making model for the domain;
- has a mechanism for checking consistency;
- supports changes to specialize or correct the model;
- can generate design decisions;
- allows specification changes maintaining design consistency;
- formulates explanations for the design decisions;
- has low documentation overhead for designers;
- integrates design and documentation activities;

We tested the feasibility of the active design document approach by modeling and implementing an active document for the preliminary design of HVAC systems. The prototype systems, presented in the next section, was tested and we obtained encouraging results.

2.3. An HVAC design model

Our empirical studies identified elements and processes in the design task that became the basis for defining the representational language for design. The design of HVAC systems requires selecting heating, ventilation and air conditioning systems. It also includes the selection, specification and location of equipment of each subsystem. The characteristics of the task suggest some type of constraint-satisfaction (Sussman, Holloway, and Knight, 1979) approach for doing the design. ADD’s parametric design model is similar to the one used in VT's model for elevator's design (Marcus, 1989). Figure 2 illustrates the kinds of information we include in our design representation.

Figure 2: *Semantic network of the elements of HVAC system design process model.*

This representation emphasizes the micro level of the decision-making. *Decision*, *alternatives*, *evaluation criteria*, *constraints*, *parameters*, *fixes*, *impacts*, *previous cases*, *goal*, *design context* and *design agents* are the nodes represented in this network. The set of relations in this representation includes *generate*, *constrain*, *evaluate* and *select*. The process starts with a design goal to be achieved within a specific design context and considering a set of design agents. Design agents (architect, code, standards, mechanical engineer, etc.) are the sources for the constraints and criteria that should be considered in the design. Given the design goals, ADD generates design parameters to be instantiated. When a parameter is instantiated, there may be enough information to carry out some other step. Alternative values for parameters must be tested against constraints. Agents generate the evaluation criteria that evaluate each alternative.

Constraint violations lead to fixes that, in turn, may lead to the introduction of new constraints.

The parametric model contains information about all parameters in the design process. A parameter can be derived or primitive. Primitive parameters are the part of the initial specification for a design case such as the coefficient of conversion from BTU to TONs parameter. Derived parameters are parameters that need to be either calculated or decided to define the design. Derived parameters can be calculated through mathematical equations, heuristic rules or tradeoff analysis.

The HVAC design parameters are connected to each other through dependency links. The dependencies effect the order in which ADD determines the values for parameters. In addition, the dependency links determine the direction of propagation in case of changes in a design parameter value. For example, if parameter A depends on parameter B, then before solving parameter A, ADD needs to determine a value for parameter B. In addition, any change in parameter B needs to be propagated to parameter A. Figure 3 illustrates a dependency diagram containing ADD’s main derived parameters and the dependencies among them. This figure represents about 30 percent of the design parameters represented in ADD’s model of the HVAC domain. On the other hand, it represents around 80 percent of the total decided parameters needed in the preliminary HVAC design phase. The parametric model also contains heuristic rules for generating, evaluating and selecting alternatives. The dependencies between parameters represent constraints imposed either by the device behavior, the environment, or the design participants’ preferences.

Figure 3: ADD’s dependency diagram containing a representative set of the main design parameters.

The existence of an explicit design model (that is, the domain and decision-making models) suggests the feasibility of a computational assistant to aid the process of designing and documenting an artifact. In ADD, this “assistant” takes the form of an active document that keeps track of dependencies, propagates constraints, and checks parameter values.

3. Using ADD for the Preliminary Design of HVAC systems

As mentioned earlier, HVAC system designers work under time pressures for developing preliminary designs. They focus on generating a design rather than documenting it. Documentation is often seen as taking time away from the design process; consequently, designers try to minimize their documentation work. For designs developed for bidding on contracts, it is typical that only a fraction of the designs will actually lead to complete projects. In such situations, there is even less incentive to document.

This perception of value and attitude by designers presents an important requirement on documentation systems: any system supporting documentation should require minimal time from designers. Design must remain the designers' main activity. This section shows how an automated system can change the cost structure of creating documentation and thereby improve the delivery of information.

3.1. An Architecture for Active Documents

To test the active design document approach, we built and evaluated a prototype system as an active document for the preliminary design of HVAC systems.

We created and implemented an architecture, shown in Figure 4, that addresses the requirements presented in Table 1. Figure 4 shows the architecture we used for developing an active design document. This architecture shows:

- Reasoning Components. These are responsible for generating design decisions, comparing these decisions with the designer's decisions, preparing design reports and controlling the documentation process (the Anticipator, Reconciler, Rationale Generator and Controller respectively);
- Design Knowledge Base. This contains knowledge about the HVAC system docmain and knowledge about a specific case;
- Interfaces for creating documentation. These are active only when creating a design document. These interfaces allow designers to develop their projects and to adjust ADD's design model (the design and justification interfaces, respectively);
- Interfaces for Retrieving documentation. These allow documentation users to query and question the design (the Explanation interface).

Figure 4: An architecture for implementing the Active Design Documentation model.

3.2. Reasoning for Documentation Capture and Retrieval

The purpose of ADD's reasoning process is to generate a consistent and complete design document with low overhead for designers. ADD increases the quality of design documents by recording the design model that generates the design instead of just recording the parameter values. In addition, ADD's initial design model decreases designers' effort in the documentation activity. Designers add specific knowledge only when ADD is not able to generate a certain design value. Furthermore, the parameter values are guaranteed to be consistent relative to ADD's model. This section describes ADD's reasoning modules, which generate the consistent design document at low cost for designers. These modules are responsible for:

- generating the design (Anticipator),

- reconciling ADD’s design with the user’s (Reconciler),
- preparing rationale reports (Rationale Generator), and
- controlling ADD’s actions.

The role of the Anticipator is to predict a value for a decision topic given by the designer considering the current state of the design and the active requirements. To make a prediction, the Anticipator uses the domain knowledge base (the parametric design and engineering decision-making models) and information about the specific design case. It is a constraint-based reasoner; i.e., given some constraints and a set of evaluation criteria, it is able to generate and analyze alternatives and propose a set of solutions. Figure 5 illustrates the Anticipator procedure to decide the value for a parameter.

Figure 5: A simplified version of the Anticipator’s procedure.

Given a parameter, if the value is already known, the Anticipator returns the value. Otherwise, the Anticipator checks whether it has all necessary information for determining the parameter value, i.e., if all subparameters are already known. The Anticipator first determines any subparameter following the same procedure. After determining the values of all subparameters, the Anticipator is ready to choose a value for the parameter being studied. First, the Anticipator generates the alternative values for the parameter. Then, it applies the active constraints on the alternative values. If there is no alternative that satisfies all constraints, the Anticipator returns all alternatives with their evaluations (in an overconstrained situation, ADD notifies the designer and goes to the acquisition mode). All alternatives that satisfy all constraints are evaluated considering the set of active criteria and their importance in the design. ADD selects the best alternatives and returns this set.

The Anticipator proposes a set of parameter values to be compared to the value proposed by the user. The module responsible for this comparison is the Reconciler. In the typical cases, ADD agrees with user and thus determines that it understands the rationale for the decision. If there is a mismatch, the Reconciler diagnoses the type of match or mismatch that occurs between the designer's and ADD's proposed values.

Whenever a mismatch is diagnosed, the rationale generator and the justification interface are activated. The *Rationale Generator* is activated to prepare ADD’s rationale for its expectation. The Justification Interface is activated to elicit changes to ADD’s model from designers.

The Knowledge Elicitor is activated whenever a mismatch is diagnosed. The Knowledge Elicitor module works closely with the Justification Interface. This module interprets the information provided by the designer. The elicitation is guided by the user, who triggers a procedure to do one of the following:

- Change Requirements. The Reconciler creates a new version of the design containing the changes. Consequently, the old information remains available.
- Change Design Constraints. The Reconciler guides the user in changing the model. The Knowledge Elicitor module translates the new constraint into either LISP functions or heuristic rules depending on the type of parameter (deduced or decided parameters).
- Change Design Criteria. The reconciler changes the criteria for evaluating a design.

ADD's last reasoning module is the Controller that supports the overall interaction cycle in anticipatory design. The Controller defines ADD's sequence of actions, but not the order of designer actions. It is often in idle mode. As soon as the designer updates the design case by proposing a new parameter to be evaluated, the Controller activates the Anticipator to generate an expectation for the parameter value. Then, the Controller sends ADD's expectation and the user's value to be evaluated by the Reconciler. If the Reconciler diagnoses a match between the values, the Controller updates the Design Case and returns to its idle mode. Otherwise, it activates the Justification Interface to acquire more information for ADD's design model. The Controller also propagates the changes to any parameter influenced by those changes checking whether the changed parameters still comply with the values proposed previously proposed by the user.

In summary, the Anticipator generates the document, the Reconciler determines whether the document reflects the designer's idea, the Justification Interface allows the user to adjust the document, and the Controller guarantees design consistency. Consequently, the designer's effort for generating design documentation is heavily transferred to ADD.

3.3. User Interfaces for Documentation Capture

Designers use the *Design Interface* to develop design cases and to understand the design specification. ADD proposes values for design parameters, adjusts initial requirements and generates more requirements or design parameters. ADD's Design Interface supports opportunistic design by offering a series of design actions.

Concepts from conventional graphical user interfaces were adequate for our prototype system and the HVAC domain. For concreteness, we describe them here briefly. As illustrated in Figure 5, the *Design Interface* offers two types of actions: Specification Retrieval Decision-Making actions. Designers retrieve the initial specification on the project that includes:

- the set of specific requirements for the building (such as the budget limits),

- the design specification drawings (blueprints) containing structural and architectural information about the building,
- the job specification containing the name of the design participants; and
- the set of criteria and the importance they play in the case being developed.

In addition to studying the initial specification, designers develop their projects by selecting parameters, choosing parameter values and changing requirements. ADD's design interface supports this decision-making activity by allowing designers to select parameters, that become the focus of the decision-making process, and values for those parameters. The parameters are logically organized in groups. For example, *HVAC system configuration, HVAC degree of centralization, HVAC equipment location, equipment quality,* and *Building Heating Loads* parameters are classified under the HVAC_PARAMETERS.

Figure 5: *ADD’s Design Interface Anatomy.*

Designers can propose values for design parameters in any order. Even when parameter dependencies require a certain order of actions, ADD does not impose this order on designers. ADD creates assumptions for the dependent parameters to avoid imposing decision ordering.

The *Justification User Interface* complements the Design User Interface by allowing the system to output its decision reasoning and the user to insert rationale for justifying contradictory decisions. This interface also does not impose any special order of actions.

The *Justification User Interface* starts working as soon the Reconciler detects a conflict between the designer’s and the Anticipator’s decisions. In this case, the Justification User Interface presents the user with its decision and rationale as illustrated in Figure 6. A designer can either accept ADD’s proposed value and explanation, force his/her own solution, or go to the acquisition mode to adjust ADD’s model.

Figure 6: *ADD’s Justification Interface: the lower right window presents the options for changing ADD’s design model while the remaining windows present ADD’s rationale for its expectation (evaluation of alternative values for a parameter considering active constraints and criteria, dependency network containg the parameter being discussed and history of the decisions).*

In case of a expectation failure, designers can either 1) change any parameter values, 2) change the set of specification or 3) engage in a thorough model adjustment. The Justification Interface offers three different types of actions (as illustrated in Figure 7): *Change Parameter Values, Change Design Data* and *Change Parametric Model* actions. The last set of actions allows designers to actually alter ADD's design model by changing:

- the parameters settings (by adding or deleting parameters from the model),
- the parameter's dependencies (by adding, modifying or deleting dependencies between parameters in the parametric model),
- the constraints definition (by editing, deleting, or adding constraints);
- the criteria definition (by adding a new criterion, changing a criterion evaluation for the parameter alternative values, or changing the importance of a criterion in the design).

Figure 7: Hierarchy of the options offered by the Justification User Interface.

3.4. User Interfaces for Documentation Retrieval

The appearance of ADD's interfaces are modeled after current engineering documents. This approach to supporting the retrieval of information is in contrast with the conventional use of query languages. Facts are not found by describing appropriate names and indices to a query language. Rather, they are presented in a regular integrated way in the reports that make up the design documentation. To find information, one requests the appropriate kind of report. In this way, information is always presented in a context, and pieces of information that are generally used together are reported together.

The user requests design explanation by interacting with a menu-based interface. ADD processes two types of questions: questions that involve a simple retrieval of data and questions that involve rationale generation. As illustrated in Figure 8, the data retrieval options include:

- retrieval of the initial design specifications,
- retrieval of ADD's adjusted parametric model,
- retrieval of a parameter value,
- retrieval of the final design specification, and
- retrieval of the design history;

while the rationale generation options include:

- rationale generation for a parameter value,
- rationale generation for comparison among options, and
- simulation of a design under different circumstances (sensitivity analysis).

Figure 8: Hierarchy of the explanation options offered by the Explanation Interface.

ADD generates reports to answer the questions selected through the Explanation Interface. ADD contains a definition of the parameters that need to be retrieved to formulate such reports.

The Rationale Generation options instigate more complex procedures. These options accommodate questions such as:

(1) why does a parameter have a specific value?,
(2) why does a parameter not have a given value? or
(3) what would happen to a design if the specifications were slightly different?

Answering these questions requires a design action such as generate alternatives, evaluate alternatives considering a set of constraints and criteria, propagate changes through the parameter dependencies and evaluating the changes.

Once the data for the reports are available, the Rationale Generator’s effort orients towards filtering the information to be presented to documentation users. ADD considers two types of explanation filters: breadth and design view filters. The breadth filtering is defined by the user's selected explanation emphasis and by the information clustering observed in the domain reports. The breadth filtering determines the amount of information ADD displays to the user for a given design parameter (illustrated in Figure 9).

Figure 9: Data Input for helping focusing an explanation.

ADD presents the parameter evaluation in terms of the criteria selected by the user for explanation emphasis. A user can check other criteria by changing the explanation emphasis. ADD’s objective is to provide only the information that interests the user at a given time. This is called a **breadth filtering** because ADD is defining the issues influencing a parameter.

The depth filtering determines what view of the design model should be displayed. The depth filtering is defined by the user background. For example, a question concerning the equipment capacity is answered in terms of the specific building loads calculations if asked by a HVAC system designer, however the same question may be answered in terms of the building location, and building exposure if asked by the owner. The answer is the same, but the view of the answer varies.

The main role of design documents in preliminary design is to allow a two-way dialog between the designer and documentation users. HVAC designers need to understand the design requirements requested by the other building design participants including the owners and the architects. Since designers do not have perfect knowledge of these requirements, they make guesses about the requirements during design. In addition, during the preliminary stage of the design requirements often change.

ADD’s dynamic document allows documentation users to discover the designer’s assumptions, as well as to support some exploration of design space even when designers are not available. ADD’s available design model can simulate what would be the design under different circumstances. Consequently, the design choices can be shared by designers and clients.

Whenever a change in the building specification or in the design requirements is proposed by a documentation user, the Rationale Generator retrieves all parameters influenced by the changes. For each of these parameters, the Controller invokes the Anticipator to obtain an expectation considering the new design conditions. The Reconciler is activated to check the match between the old and new parameter values. If the new specification produces a different parameter value, the Controller records and propagates the changes to the influenced parameters. At the end, the Rationale Generator contains a list of parameters that need to be changed to adjust to the new design specifications. The changed parameters correspond to the impact on the HVAC system design given changes in the design specification. As soon as the impacts are calculated, the Rationale Generator returns the design to its original specification.

The same process occurs when the change affects the HVAC system design. In this case, the user is probably interested in checking the impact on the other design trades (or even in other aspects of the HVAC system design) if a HVAC system design parameter changes. The Rationale Generator receives the new HVAC parameter value and evaluates it in terms of the active constraints and criteria. Consequently, it checks the local impact caused by the change in the design specification, such as a violation of an architectural constraint. In addition, it forces the new value and propagates it to the set of influenced parameters. At the end, the Rationale Generator reports the local evaluation of the change and a list of other HVAC parameters affected.

4. Evaluating ADD’s Performance

To establish viability of the architecture for building active documents, we developed a pilot study using ADD to build, revise and use an active document for a realistic problem. Our studies have helped us to understand how an active document can affect the cost of creating and using documentation.

First, we conducted two tests with experienced designers interacting with ADD in the preliminary design of a modest HVAC system. Then we had two typical users of HVAC documentation interact with ADD to understand one of the designs developed in the first tests. In both cases, we kept statistics on ADD’s performance to measure its degree of interference with the designers and to understand how its internal architecture performed. We also videotaped the sessions. In addition to the statistics and videotapes, we interviewed the designers and documentation users, asking them to evaluate the potential usefulness of ADD in their work environment.

4.1. Pilot Study in Creating Documentation

We selected two experienced designers from different HVAC system companies to take part in our experiments. Both have more than 10 years of professional experience in generating HVAC system designs. They had no previous experience with ADD or with the problem case. Each asked to develop a design proposal (preliminary design document) for the given case. That case consisted of developing and documenting a design for a 5-story office building located near San Francisco using ADD. (To minimize one source of possible bias, the realistic project used for the test case was not considered in the development of ADD's initial knowledge base.)

Procedure

Each designer was given an initial set of specifications including the building blueprints, design criteria, the list of design participants, and the owner's requirements for the design. Then the designer was asked to prepare a preliminary design meeting these specifications. The preliminary design of an HVAC system involves the instantiation of about 150 parameters. Each designer explicitly decided between 15 and 20 parameters. All other parameter instantiations were considered implicit decisions, but the designers also checked their values interactively with ADD at the end of the session. There was no imposed order for the designers' decisions or actions.

The designers interacted with ADD though Dr. Garcia because we didn't have enough time from the experts to completely train them in ADD's user interface. The designers asked for information about the case, selected parameters to be instantiated, and provided values for those parameters. Dr. Garcia did not provide any additional verbal input about the project because we wanted to verify that the information shown by ADD was sufficient.

The sessions lasted about 2 hours each. In addition to the material collected observing designers interacting with ADD, we interviewed them to verify the usefulness of a tool like ADD, the adequacy of the approach to support design and documentation, and the need for such a tool to assist the documentation process.

Data Analysis

We analyzed the data in the protocols to verify the adequacy of the architecture for building an active document and the impact of an active document on the cost of creating documentation. We recorded the frequency with which each module was activated in each session as well as the number of right expectations generated by ADD. The purpose was to measure the percentage of automatic documentation as a measure of the time saved.

Results

The statistical results of this pilot study are shown in Table 1. We first measured how the ADD architecture responded to each designer's actions. For example in session 1, the designer explicitly decided values for 17 parameters (indicated by the number of activations of the Reconciler, which compares designer's decisions with ADD's decisions). Those 17 parameters include 4 that ADD had to recompute after knowledge elicitation to handle conflicts between designer choices and ADD's choices (measured in the number of activations of the Knowledge Elicitor). On its own, ADD determined values of a total of 74 parameters (the number of activations of the Anticipator), a number that includes many parameters that the designer handles implicitly in preliminary design.

In analyzing those numbers, we note that ADD produced an incorrect expectation only 4 times for 74 parameters decided in session 1. The fraction of the parameters correctly and automatically documented is what we call the *anticipation hit ratio*.

	User	**Session 1** **Project Manager A**	**Session 2** **Project Manager B**
Number of Activations per ADD module	Design Interface	33	13
	Anticipator	74	59
	Reconciler	17	12
	Knowledge Elicitor	4	6
	Justification Interface	10	20
	Anticipation Hit Ratio	0.95	0.90
	Informal Evaluation	Excellent	Excellent

Table 1: Pilot study results for creating documentation.

The correct expectation is extremely important in guaranteeing low overhead in the documentation process. Equation 1 is the computation of the time required to produce an active design document, including the time for uninterrupted design and the time required to enter extend ADD's knowledge base to justify the user's decision. It highlights the importance of having a high percentage of right expectations in the active document approach.

$$T_e = T_h * r + T_i * (1 - r) \qquad \text{(Equation 1)}$$

where:

T_e is the time to enter rationale;
T_h is the time to document a parameter automatically;
T_i is the interaction time—the time spent adjusting ADD's design model to match the user's decision; and
r is the anticipator's hit ratio.

Equation 1 tells us that the expected design time depends on the anticipator's hit ratio. If the knowledge base is tuned to the user's practice, then r is close to 1. In such cases, ADD's approach to documenting additional parameters imposes very little overhead on the designer. In our test cases, even though the initial design model was not tuned, the anticipation hit ratio was above 90%. This suggests that the domain is mature and that the strategies used for designers do not have a big variance.

In our interviews, the designers confirmed our numerical results. They saw substantial potential for the active documents in HVAC design.

4.2. Pilot Study in Retrieving Documentation

We conducted two tests to evaluate ADD’s performance in delivering rationale. We selected two natural users of HVAC system documents: an owner’s representative and a mechanical engineer. The owner’s representative had more than 15 years of experience analyzing design documents and making sure that the owner’s requirements were satisfied by a project. The other user was a university professor who teaches the design of HVAC systems. Neither had no previous experience with ADD or with the problem case, which consisted of understanding and approving the design proposal developed in the documentation creation experiment.

Procedure

Again, Dr. Garcia served as the interface with the system. Her role was to input the user questions to ADD. Both users easily verbalized their questions about the design and analyzed the answers presented by the system. Dr. Garcia did not add any information to that displayed by ADD. There was no time restriction on the interaction; however both took 1 to 1 1/2 hours interacting with the system.

Data Analysis

We analyzed the data in the videotapes produced by the two experiments. We were looking for

- the adequacy of the architecture to provide answers to users’ questions, and
- the adequacy of the answers to satisfy documentation user’s needs.

Results

Table 2 presents the number of times each module was activated to generate design explanations. All modules were activated during both sessions, indicating that they are all were necessary in explanation. Even in retrieval sessions, the Anticipator is activated, illustrating the fact that explanations are generated and not just retrieved from active documents. During session 1, the user retrieved design facts, such as initial requirements, and decision history. He questioned the value of 4 parameters and verified the value of other 6 parameters. In both sessions, the documentation users changed specifications to check the design response to those changes. ADD was able to answer all of their questions, except for the ones related to parameters outside its design model or related to parameters not addressed by the designer during the development of the project.

User		**Session 1 Project Manager**	**Session 2 Mechanical Engineer**
Number of Activations per ADD Module	Explanation Interface	33	13
	Anticipator	74	59
	Reconciler	17	12
	Rationale Generator	4	6
Answer Acceptance		0.9	1
Informal Evaluation		Excellent	Excellent

Table 2: Pilot study results for retrieving documentation.

The documentation users reported that the case study was a good representative of the cases they normally solve in their offices. They also stated that without ADD they would take much longer to find information in a project and analyze it. They also commented that they missed the blue prints at first till they realized that the blue prints were also available inside the tool. They were surprised with the potential of the tool, especially the ability to check impacts on a design given changes in the requirements. Both agreed that a design tool with ADD’s capabilities would have a major impact on design speed, quality and flexibility.

The results from the initial test cases indicate that ADD offers the following benefits:

- increasing the number of questions that can be made on a design;
- providing focus for the explanation, and
- allowing documentation users to challenge the design, changing requirements and assumptions

In summary, the same design model that originates a design can be used for constructing an explanation. The same model allows documentation users to explore new alternatives. Consequently, the designer is no longer the only one doing design. The acquired design model can also be used as the index of information that can be filtered considering the user’s interests and profile.

5. Related Work

Previous work on design rationale follows three major approaches: to record the sequence of actions (history-based rationale), to record the arguments and issues raised during design (argumentation-based rationale), and to record the device behavior model (device model-based rationale).

This section presents the three other approaches to documentation; i.e., history-based, argumentation-based and device model-based rationale, emphasizing their performance related to the preliminary, routine design issues. At the end of this section we present a table with the summary of this comparison.

5.1. Argumentation-based Rationale

The argumentation-based approach is an outgrowth of research on hypertext. The primary goal of this research has been to provide a uniform structure for creating arguments, so as to make arguments more directly comparable. The emphasis in this work has been on expressiveness of the argumentation language together with domain and task independence.

According to the argumentation-based rationale approach, design rationale is an informal representation of the arguments that defines a design. This approach offers a fixed, domain-independent vocabulary for describing the reasons for a design. Rationale is recorded and retrieved as uninterpreted chunks of text classified using the argumentation-based vocabulary. Previous approaches, such as IBIS (Kunz, 70), PHI (McCall, 87), QOC (MacLean, 91), and DRL (Lee, 90), differ in the vocabulary and relations defining rationale.

The quest for generality in argumentation-based rationale comes at the expense of attention to the requirements of the design task. Thus, argumentation systems are not especially well tailored to engineering design, routine design, preliminary design, engineering decision models, trade-offs, particular domains, or the different needs of different documentation users. Although these systems have goals for generality, they are unspecialized and weak.

Argumentation-based models can not support preliminary design documentation because the models (1) impose overly general and unfamiliar notation for designers; (2) force designers to have an even more active role in the documentation task (increasing the documentation overhead); (3) can not guarantee consistency of the recorded rationale, consequently the recorded rationale might not explain the design; (4) offer a static view of design, consequently they can not support the 2-way communication needed in preliminary design; and (5) can not go beyond recording and retrieving information, since rationale is uninterpreted text.

5.2. History-based Rationale

The history-based rationale approach represents rationale as a sequence of events that take place during design. According to this approach, the “design log” is sufficiently rich to reconstruct the design and, consequently, to explain it. Many systems, such as Lakin's electronic notebook project (Lakin, 1989) and EDN (Karinthi, 92), aim to provide an environment that can construct the design log. In this model documentation should contain all information and actions that take place during the design.

The history-based approach focuses primarily on low overhead methods for recording design decisions. This approach has sought generality in this recording without commitment to any particular design domain or phase of design. To this end, it relies primarily on uninterpreted design records. Furthermore, this approach focuses on the creation of design records, but not their use. Consequently, it pays little attention to the processes of design tradeoffs or negotiation over specification that is characteristic of preliminary design.

One of the assumptions of this approach is that the record of the actions explains the design. However, many design decisions and criteria are never written down so that the relevant rationale is inaccessible to a document user. The designers' actions might reflect some of their design process; however, there is no way to guarantee that the actions can reproduce the designers' reasoning process. The history helps to understand a design, but it is not enough to explain why the designer selected a specific alternative. In short, this approach saves on documentation creation payments but mortgages document retrieval futures. Recording of documentation is cheap, but there is nothing to assure that what is recorded will be useful or even meaningful.

Another problem with the adequacy of the history-based approach relates to the consistency of the recorded document. Since rationale is recorded as non-interpreted actions, there is no way to guarantee the document's consistency and accuracy to explain the design. In comparison with the active design document approach, this reduces the incentive for the designer to use the approach.

The design-history approach plays a very passive role in the documentation. Consequently, besides recording a huge amount of information (including irrelevant information), the approach can not guarantee that the recorded information is able to explain the generated design. The approach is not practical for supporting preliminary design because design participants need fast access to information that reveals and explains the assumptions considered in the design.

5.3. Device Model-based Rationale

The device-model approach for rationale uses a device model to explain a design, and assumes that explanations in terms of devices are sufficient for all explanation needs. This machine interpretation for rationale is more powerful and domain specific than the argumentative and history-based models proposed above.

The device-model approach is based on many of the techniques and assumptions of device-model based expert systems, especially those for diagnosis. The basic motivation for that work is to provide more automatic methods for reasoning about a design by using detailed knowledge about the devices that comprise it. In this research branch, exemplified in work by Gruber (Gruber, 1992; Gruber and Russell, 1990) and Baudin (Baudin, Sivard and Zweben, 1990), a deep model containing form, function and behavior information about domain concepts supports the acquisition of rationale in a specific domain.

This approach has been more successful for diagnosis than for design. In diagnosis the model for the system composed of devices is considered to be complete. However, throughout most of a design process, some parts of the the system are always unspecified. Indeed, the work of design necessarily involves exploration of the space of possible designs. This brings requirements for modeling specifications, tradeoffs, and social aspects of the design such as an analysis of the different needs of different documentation users. By their nature, device models do not make use of this information, and therein lies the source of their weakness.

Although the device model-based approach models devices, it does not really model design. For example, there is no place in this approach to talk about design choices and evaluation criteria. The approach is not motivated by an understanding of design as a collaborative process. The approach pays no attention to the information and communication needs of collaborating designers.

5.4. Comparing the Approaches

The documentation problems identified in our field studies were the basis for comparing the approaches. Table 3 summarizes the evaluation of each approach considering whether the approach (1) handles the problem successfully, (2) does not handle it successfully, or (3) does not address it at all. Each cell in table 2 contains either a +, - or ? sign meaning:

- + addresses issues and substantially contributes to a solution;
- — contributes little to a solution; and
- ? does not address the issue.

parameterized

	Argumentation -based Rationale	History-based Rationale	Device Model-based Rationale	Active Design Documents
Low Overhead	—	+	—	+
Consistency	—	—	+	+
Completeness (adequacy)	—	—	—	+
Access Information	—	—	?	+
Allow Negotiation	—	—	+	+
Common Terminology	—	+	+	+

Table 3: Comparing design rationale approaches to the documentation problems identified during preliminary routine design.

In summary, the active document approach is a radical departure from previous work on design rationale. These other approaches to design rationale have not been so focused on the issues of preliminary routine designs. Thus, although those approaches are the closest related work, the active document approach represents a significant departure both in methods and evaluation criteria. The other approaches suffer in the comparison in part because they were never so precisely aimed. Nevertheless, this very aiming is part of the point of the approach. We believe that the sharper aim is necessary for a higher payoff.

6. Conclusions

In this paper, we have described the active design documents approach to support documentation lifecycle. We presented the implementation and experiments of an active design document for the domain of HVAC preliminary design. The test cases indicate that is feasible to build an active design document that measurably aids designers in the documentation process. We observed that the performance of the documentation system depends crucially on its ability to correctly anticipate design decisions. We propose a measure to check the utility of the approach for a domain: the anticipation hit ratio. The AMEBA diagram of each design session showed the fraction of the parameter space actually designed and documented by designers compared with the portion of the design space being covered by the automatic documentation agent.

The active document approach reduces the costs for a designer to document design assumptions, especially as they relate to communicated specifications. It reduces the perceived cost of design exploration by making it easier for users to try design variations both for users and designers. Designers benefit because of the automatic propagation of changes and checking of constraints. In additions, users exploring the design space benefit because they do not have to wait for a designer to be present. This also enables users such as owner representatives to check that their specification are being correctly understood. It also enables them to explore the sensitivity of cost to design and specification variation.

Design is collaborative. The ADD approach directly affects the nature of communication between design participants. It gives them a rich and active medium for referring both to the design and elements of the design process. We believe that this approach can improve design practice both in the short run and in the long run. In the short run it can improve the consistency of designs by its automatic checking, and the quality of design by lowering the cost of exploring more of the design space in detail. In the long run, we believe it makes the design practice more visible, and thereby, more accessible for evolution and tuning.

References

Baudin, C., C. Sivard, et al. (1990). "Using device models and design goals for design rational capture." Technical Report. NASA Ames Research Center, CA.

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

Fischer, G., A. C. Lemke, et al. (1991). “Making Argumentation Serve Design.” Human-Computer Interaction **6**(3/4).

Garcia, A. C. B., H. C. Howard, and J. Stefik (1993). "Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design." Technical Report #85, Center for Integrated Facility Engineering, Stanford University, CA.

Garcia, A. C. B., H. C. Howard, and J. Stefik (1994). “Improving Design and Documentation by Using Partially Automated Synthesis.” Submitted to AIEDAM special issue on Design Research Methodology.

Gruber, T. R. and P. P. Nayak (1992). Computer-assisted Formulation of Engineering Models: Issues for Knowledge Acquisition. Stanford University, Knowledge Systems Laboratory.

Gruber, T. R. and D. M. Russell (1992). Generative design rationale: Beyond the record and replay paradigm. Design Rationale. Lawrence Erlbaum.

Karinthi, R. (1992). "Capturing Design Rationales for Use in a Concurrent Engineering Environment." AAAI '92-Design Rationale Workshop, san Jose, CA,

Kunz, W. and H. W. J. Rittel(1970). "Issues as Elements of Information".Technical Report. Center for Planning and Development Research, University of California, Berkeley.

Lakin, F., H. Wambaugh, et al. (1989). “The electronic design notebook: Performing medium and processing medium.” Visual Computer: International Journal of Computer Graphics **5**(4): 214-226.

Lee, J. (1990). "SIBYL: A tool for managing group decision rationale." Proceedings of the conference on Computer Supported Cooperative Work (CSCW-90), Los Angelos,

Luth, G., H. Krawinkler, K. H. Law (1991). "Representation and Reasoning for Integrated Structural Design." CIFE Technical Report #055, Center for Integrated Facility Engineering, Stanford University, CA.

MacLean, A., R. Young, et al. (1991). “Questions, Options, and Criteria: Elements of A Design Rationale for User Interfaces.” Human Computer Interaction **6**(3/4):

McCall, R. (1986). “Issue-serve systems: A descriptive theory for design.” Design Methods and Theories **20**(8): 443-458.

Marcus, S. (1989). “Understanding Decision Ordering from a piecemeal collection of Knowledge” Knowledge Acquisition **1**: 279-298.

Sussman, G, J. Holloway, and T. F. Knight (1979). “Computer aided evolutionary design for digital integrated systems.” MIT Artificial Intelligence Laboratory Memo 526, Cambridge, MA.