A CASE-BASED REASONING FRAMEWORK IN VALUE ENGINEERING

Francisco Loforte Ribeiro

Instituto Superior Técnico, Departamento de Engenharia Civil, Secção de Estruturas e Construção, Av. Rovisco Pais 1, 1096 Lisboa Codex, Portugal

Value Engineering (VE) is the title given to a set of value techniques applied during the design or 'engineering' phases of a construction project that has its origins in the US manufacturing industry. Currently, value engineers attempt to infer design rationale information from drawings and specifications for carrying out the value engineering analysis. Access to design rationales can help value engineers in different ways. Case-based reasoning is a technology for problem solving based on recall and reuse of specific experiences. Case-based reasoning offers techniques for representing and managing complex design rationale cases, augmenting a set of specific design experiences with generalized knowledge and formalizing a typically informal body of knowledge. Agent-based systems technology offers modularity and techniques for handling human-agent interaction, managing and searching information. This paper describes the utility of design rationales in value engineering. It examines the combination of case-based reasoning and agent-based technology to support value engineering analysis. Finally, it proposes a case-based framework in value engineering during design phase.

Keywords: case-based reasoning, design rationale, value engineering.

INTRODUCTION

Value Engineering (VE) was developed by Lawrence Miles in 1940s in the United States of America. The combined growth in VE practice through different US government agencies and private industry led to the establishment of the Society of American Value Engineers (SAVE) in 1958, a professional body formed to foster VE (Norton and McElligott 1995). Over the past decade, there has been a trend towards applying value techniques at all stages of the project life cycle. Therefore, the term value management (VM) has become a blanket term that covers all value techniques whether they entails Value Planning, Value Engineering or Value Analysis. A recent development in VM for construction industry (Connaughton and Green 1996) recommends that VM should incorporate VE.

A VE study during design attempts to identify unnecessary costs in design parts and components and suggest design alternatives (for those design items needing change) to reduce life-cycle costs without reducing the quality, function and performance of a building design (de la Garza and Alcantara 1997). Design involves the creation of a set of documents describing a yet unconstructed product. These documents contain implicit information about the rationale leading to the final form of the design.

Design rationale (DR) refers to the collection of information about the evolution of a design product. This body of information explicitly records design activity and the reasons for making choices and reasons for not making choices (de la Garza and

Ribeiro, F L (1999) A case-based reasoning framework in value engineering. *In:* Hughes, W (Ed.), *15th Annual ARCOM Conference*, 15-17 September 1999, Liverpool John Moores University. Association of Researchers in Construction Management, Vol. 1, 287-96.

Table 1: AI systems using DRs in construction

System Name	Domain	Service Provided by DRs
Shared–DRIM (Pena-Mora *et al.* 1995)	Design	Design support
Softda (Yamamoto and Isoda 1986)	Design	Reuse/redesign support
Cadre (Maher and Garza 1997)	Architectural design	Reuse/redesign support
Sibyl (Lee and Lai 1991)	Design	Reuse/redesign support
Janus (Shipman and McCall 1996)	Architectural design	Learning
DRIVE (Alcantara 1996)	Value engineering	Formulation of design alternatives
CAVE (de la Garza and Alcantara 1997)	Value engineering	VE support
ADD (Garcia *et al.* 1994)	Design –HVAC systems	Design support
DRARS (de la Garza and Alcantara 1997)	Building construction	Documentation support

Alcantara 1997). Capturing and storing a DR is synonymous with explicitly expressing the requirements, preferences, and reasoning implicitly embedded within the design drawings and specifications. Access to DR information can help value engineers to carry out the function analysis and properly formulate design alternatives of a particular design and, therefore to perform their tasks more efficiently (de la Garza and Alcantara 1997).

Several DR systems in design use case-based reasoning (CBR) as enabling technology (Maher and Garza 1997, Shipman and McCall 1996, Yamamoto and Isoda 1986). CBR formalizes a computational model of problem solving based on memory organization and reminding.

The main focus of this paper is on the use of CBR and agent-base technology to store, represent and make available DR information whenever the VE team needs them. Aiming to improve the reliability and effectiveness of the VE study during design phase is being developed by the author a CBR system here, designated by CAVA (Computer Aid Value Analysis). This paper describes some of the most representative DR systems in the construction industry using CBR and agent-based technology. It proposes a CBR framework to help value engineers in formulating, evaluating and developing cost-saving design alternatives of a current design.

SYSTEMS USING DESIGN RATIONALES

We briefly present some of the recent work on Artificial Intelligent (AI) systems using DRs in construction. In the last few years, interest in DR has grown. A number of systems in design and construction have been developed to date. They provide useful lessons for researchers and developers of futures rationale systems. Table 1 presents a selection of implementations that served as a base for CAVA.

Shared Design Rational Information Management (Shared–DRIM) system provides dependency management and collaboration support during design process. Shared–DRIM uses agent-based technology.

Softda system, for example, supports DRs reuse by acquiring the relationship about designs and requirements and using it to index documents and codes. Cadre is a CBR approach to architectural design. It stores DRs as cases for reuse support. Sibyl system is another example of DRs reuse support system. Sibyl allows designers reuse the rationales themselves. Janus combines CBR and agent-technology to store, represent and access DRs. In Janus, when an agent encounters a human decision that is sub-optimal according to its knowledge, it presents the designer with an appropriate recommendation. Design Rationale Information Phase of Value Engineering (DRIVE)

assists value engineers in generating suitable design alternatives by presenting rationale about an existing design. DRIVE is part of a much larger Computer-Aided Value Engineering (CAVE). Active Design Documents (ADD) system seeks to capture rationale by using the computer as a "designer apprentice". The domain of ADD is the preliminary design of HVAC systems for commercial office buildings. Design Rationale Authoring and Retrieval System (DRARS) uses the computer as a database search engine to present DR information. DRARS uses objects called views, goals, alternatives, claims, questions, answers, and versions.

MODELLING APPROACH

The modelling framework used to describe CAVA reflects a convergence of task-oriented approaches (Chandrasekaran *et al.* 1992), the ontological analysis methodology (Fernández *et al*. 1997), the Client Centred Approach (CCA) method (Watson *et al.* 1992) and the CASE-METHOD (Kitano and Shimazu 1996). The modelling framework describes the system into three levels. These levels are:

1. **The Object-level**: This level comprises manifold information and knowledge sources, ranging from machine-readable formal representation to human-readable informal representation.
2. **The knowledge-level description**: At knowledge level, the system is described in terms of the tasks, problem solving methods, primitive inferences, knowledge types and goals. This level also describes the structure of the knowledge needed by the problem solving methods through the ontological analysis of the domain. The product of this level is a task-method structure and the domain specific ontologies.
3. **The symbol-level**: where the system is implemented using a selected programming environment. The products of this level are; i) the *skeleton* system; ii) the *demo* system; and iii) the *working* system.

According to this framework, each level of description is a self-contained model of the system. In this methodology the $level_n$ implements the structure of the $level_{n-1}$.

Task method analysis

The task-method structure is the result of the task-method analysis carried out, in the context of the characterization of the domain. Its principal elements are tasks, problem solving methods, primitive inferences, and goals. The task-method structure provides a specification of the system in terms of *what* the system should do and *how* the system accomplishes its goals. Therefore, the task structure of CAVA consists of: *Tasks*. A task is specified by: i) the task definition; and ii) its initial and goal states; *Problem solving methods***.** A problem solving method is specified by: i) a problem space where the search for the solution takes place; ii) a set of sub-tasks or inferences that can be used to transform the initial state of a task to the goal state; (iii) types of knowledge it uses. At the highest level, the task structure sets up *new DR specification, find design alternatives* and *implement VE proposals* as sub-tasks of the VE task. Figure 1 presents a top view of task structure and describes how the case-based method accomplishes the finding design alternatives task. Rectangles represent tasks/subtasks and circles represent methods. The case-based method set up five sub-tasks of the finding design alternatives task: *design rationale recall*; *design alternatives recall*; *design alternatives adaptation*; *design rationale adaptation*; *design rationale storage*. At the lowest level of the task structure one method is used to perform a sub-task corresponding to another method.

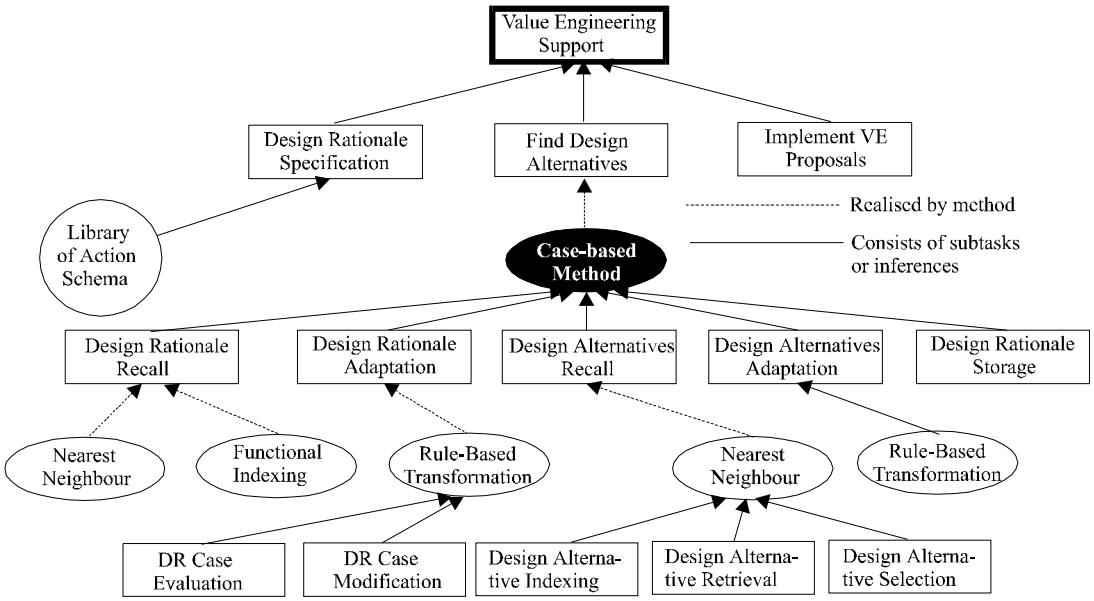

Figure 1: Task-method structure: a task-method sub-task decomposition and control strategy

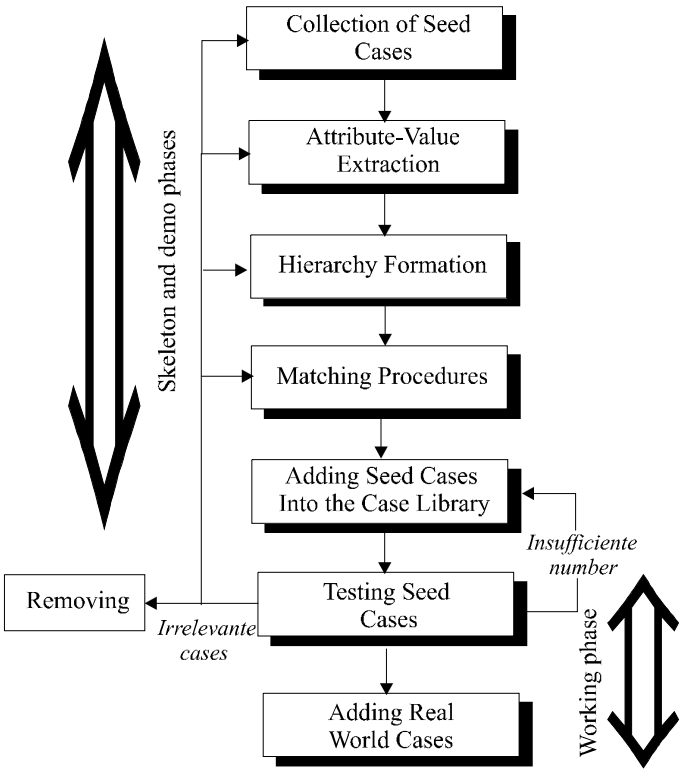

Figure 2: Case acquisition process

KNOWLEDGE ACQUISITION

Each one of the sub-tasks described by the task-method structure is performed by searching through a problem space from the initial state to the goal state. This search requires knowledge in order to map the initial state to the goal state of the task being solved. Acquiring and organizing this knowledge was an essential step in completing the task structure. A number of methods were employed for knowledge acquisition. Table 2 lists the knowledge acquisition methods employed at each stage of the implementation cycle.

Table 2: Methods employed at each system's development cycle

Knowledge acquisition method	**Development Cycle**			
Postal survey	Holistic picture			
Task-based method; text acquisition; DRs and design drawings and files acquisition		Skeleton system		
Task-based method; text acquisition; DRs and past design drawings and files acquisition			Demo system	
Interviewing; DRs and past design drawings and files acquisition				Working system

The task-method analysis, revealed that much of the knowledge, (domain knowledge, domain expertise and knowledge – know) required to perform each task of CAVA comes from a number of sources such as: sketches, drawings, specifications and abstract meetings of past designs; expertise and memory of designers and value engineers experts; regulatory and technical texts; design files; and computer databases. The acquisition of DRs in terms of cases was carried out in accordance with the following basic phases based on the CASE-METHOD (Kitano and Shimazu 1996) (Figure 2): (i) Collection of seed cases into full case reports; (ii) Attribute-value extraction; (iii) Hierarchy formation; (iv) Identification of the matching procedures; (v) Adding seed cases to the case library; (vi) Testing and refining the case representation and indexing scheme; (vii) Apply to new real world VE studies. The main benefit of the approach to KA is that it integrates the implementation with verification and validation of the case library.

IMPLEMENTATION

The implementation of CAVA was a staged process measured by the deliverables produced at each stage. The current implementation stage of the system corresponds to the *skeleton system* of the CCA method. The implemented system stores 10 DR cases in the case base.

Programming environment

The key concerns in the early stage of the development process of CAVA was: how represent the cases (DRs and modification plans), determining the similarity of the cases, retrieving the similar cases, adapting a case, integrating cases with other representational paradigms, and finally tailoring the interface for functionality and user friendliness. Taking into account these issues, ART*Enterprise Version 2.0 Beat R2 from Inference Corporation was chosen as the programming environment, and Microsoft Windows 95 and PC computer were chosen as the implementation platform. ART*Enterprise provides a very powerful programming environment that allows developers to build hybrid applications. ART*Enterprise offers a variety of representational paradigms including: objects supporting multiple inheritance, encapsulation and polymorphism; rules; and cases. The CBR components in ART*Enterprise provide facilities to quickly develop case-bases, the nearest neighbour matching method and the impressive text handling.

Case-base organization

A case base is a structure where the cases are stored. Case memory organization refers to the way cases are organized for access during retrieval. The system's CBR knowledge base comprises two case bases: the case base of DR cases and the case

base of action schema plans. Each case base consists of a memory of cases describing specific knowledge. We built the system's case bases using ART*Enterprise CBR tools. A case base is an instance of the object case-bases. The case-bases are organized into a manageable structure to support efficient searching and matching. To build a consistent structure for entering, retrieving and storing cases, the case bases had to become uniform. For each case base, a distinct structure was implemented. For the case base of DR cases gleaned from existing documentation (drawings and specifications). For the case base of action schema plans, this structure was engineered by interviewing designers and value engineers. We followed this structure in organizing cases into each case base memory. The DR case's size was an import consideration in choosing the organization of the case base for DR cases. Because a DR is a record of a complex set of experiences and decisions we divided DR cases into smaller chunks –sub-cases- and their parts linked as a hierarchy. Therefore, the case base for DR cases uses a hierarchically structured organization. The memory organization of the other cases base is a list of pointer to cases. Each case base is an ART*Enterprise object instance of the object case-bases. It comprises the case base object and a case base index. The case-base index stores the information needed for matching cases.

Representation of design rationale cases

Cases which comprise problems and solutions can be used to derive solution to new problems, as in Cadre and Janus systems. The system's CBR knowledge base contains four kinds of cases: *design rationales cases*; and *action schema plans*. A DR case is simply an ART*Enterprise object instance whose attributes constitute features of the case. It is represented in the CBR knowledge base as a *partonomic hierarchy* because a DR is a complex set of information and experiences. A partonomic hierarchy of a DR case is a decomposition of DR information into a hierarchy of sub-cases. Therefore, a DR case in the case base is represented as a hierarchy of sub-cases. This decomposition allows the search and match to focus only on the relevant parts of a DR. Processing only some knowledge linked to a case lets reasoning becomes more efficient. The development of a hierarchically structures case base requires defining a typical decomposition of a DR case. The system represents this decomposition explicitly through domain models. Thus, we defined classes containing attributes for those features and made the cases instances of those classes.

Case recall

ART*Enterprise gives one mechanism for retrieving cases: *nearest neighbour matching* method to find the best matching case. CAVA integrates search and match. It uses two retrieval algorithms for searching the case base: the *hierarchical search* for searching the case base of DR cases; and the *serial search* algorithm for searching the other case base. Both algorithms are combined with the nearest neighbour matching method to retrieve the best matching cases from the case base memory. CAVA uses the output of the new DR specification task as a probe into the case base to search for DR cases that match the current problem as closely as possible. In particular, it searches the DR cases using the hierarchical search algorithm, first looking for DR cases exactly matching the specified DR, and then for partial matches. Exactly matching are those whose indices (decision problem, element functions and element performances) are the same as those specified for a new DR. Partial matches, in order de preference, are DR cases matching one or two, or the three indices fully or partially. Given a description of the new DR, the hierarchical search algorithm, using the indices in the case base memory retrieves the most similar DR case to the current

situation by: (i) searching the case memory for a potentially similar DR cases; ii) matching and ranking the most similar DR cases; iii) retrieving the DR case with highest score.

Case adaptation
If case retrieval results in a DR case that only partially matches the specification of the given DR, then the case-based method attempts to adapt the retrieved the DR case to meet the current specification. The recall of best matching case provides the starting point for generating a new solution. The system uses the rule-base transformation method to perform the adaptation task. The rule-base transformation method has been employed by a number CBR researchers to implement the adaptation task, an in Cadre system. This method set up two sub-tasks: the *evaluation* of the selected case and the *modification* of the solution. The evaluation task checks the feasibility of the solution of the selected case. The modification task changes parts of the selected case. The system uses a set of heuristic rules: (i) to evaluate the differences between the selected DR case or sub-case and the new problem and the evaluation; (ii) and to modify the solution of the selected DR case or sub-case. These rules are used to tweak the parameters in a selected case to transform it into a solution to the new problem.

Case storage
Case storage is an important component of CAVA. It reflects the conceptual view of what is represented in the case and take into account the its indices. If the case-based method is successful in using previous DR cases to find the design alternatives to the current design, then it stores the new DR case (modified) in its case base memory. The new DR case is indexed using the data structure of the case-base index. In addition to complete DR case, the system also stores DR sub-cases in its memory. Thus, the system automatically acquires additional DR cases as it solves new problems.

Agent-based user interface
The user interacts through a form-based interface with a planning program –an *agent*- that, taking the user's goal as an input, search the agent's case base for an action plan to generate a sequence of actions achieving the goal. An agent is a problem solver that perceives and acts to achieve a narrow set of goals within a specific virtual or real environment.

The agent's case base contains plans (cases) that are indexed by user's goals and can be retrieved according to the input parameters. The agent basically interacts with the CBR components. Based on the retrieved sequence of actions, the agent, call the CBR components of the system to achieve the user's goal. The locus of agent automation is on the executive tasks. CAVA agent's model is based on the Norman's (Norman 1986) reference model of semiautonomous agents (see Figure 3). Norman characterizes the human computer interaction as the problem of bridging the twin gulfs of *execution* and *evaluation*. The execution side of the cycle involves translating a goal into a sequence of actions for achieving the goal. The evaluation side involves using feedback from the domain to compare the result of the action to the goal. Norman's model provides a useful reference to build human computer interfaces because it identifies the cognitive processes and the linkages between them that must be supported for agent-based interfaces to succeed. CAVA's agent model lies in automating only the execution side to achieve a goal, leaving evaluation as at task for the user. Thus, it comprises: (i) translating of a user's goal into a intention to act; (ii) translating this intention into a sequence of internal commands (action plan); (iii) executing this sequence of commands; (iv) presenting the results to the user for

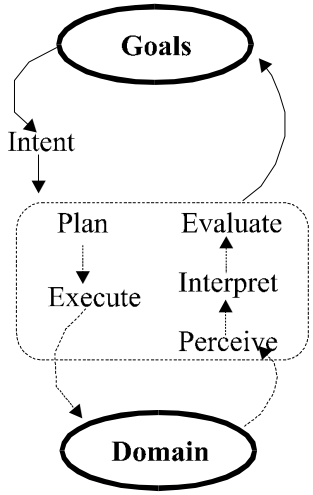

Figure 3: The Norman' reference model of semiautonomous agent

evaluation. The use of goals as input and action plans as agent primitives makes system's agent-based interface consistent with our reference model and consistent with the definition of an agent as a process automating stages of this model. The agent-based interface was implemented as a hierarchy of ART*Enterprise object instances of the object class user-interface. The action plans are stored as cases and indexed by user's goals into the case base of action plans. They were implemented as ART*Enterprise object instances of the object class case-bases.

LESSONS LEARNED

In modelling and implementing the system, a number of practical lessons were learnt about building the system. They were grouped into the following main headlines:

Lesson 1: It is difficult to acquire and represent and entire DR as "a case" : A DR is not simply "a case" but a complex set of information, experiences and decisions resulting in a complex system. Rationales are embedded not only in formal documents such as drawings and specifications, but also in informal media. In fact, almost anything in a design process may be part of a design rationale as long as it is represented and can be used to trace a reason for some aspect of VE study. The unstructured nature of the recording makes it difficult to access exactly what is needed and use that information in any quantitative way. This presents special requirements for representing, indexing and presenting a DR case. These questions can be addressed by using: i) a partonomic hierarchy of a DR case; ii) a flexible indexing format; iii) a hierarchically structured case base; and iv) a presentation format that enables the user to navigate both within and across the DR case.

Lesson 2: A DR case's size is an important consideration for its representation : Not every part of a DR is useful for a VE study. The task structure of the system and the ontological analysis determines what to represent and how to represent the rationales behind design process. DR can be divided into smaller chunks and their parts linked a hierarchy or a network. AI offers different decomposition schemes to achieve this.

Lesson 3: The availability of DRs which provide specific knowledge is vital for the success of a VE project using CBR: The development of system has shown that one of the things that must be considered in building a system using CBR is whether or not past cases containing specific knowledge are available in the domain. If these cases are not easily available or are very expensive to collect building a system will be

difficult. In problem solving tasks, cases must provide knowledge related to specific design experiences which can be used to build solutions for new problems in similar situations. The ability of the system to retrieve that match with a high similarity value and its performance improves with the number of appropriate cases.

CONCLUSIONS AND FUTURE WORK

The present research has indicated that DRs can help VE team in formulating, evaluating and developing design alternatives of a current design. Within the limits of this paper, a review of recent work on DR systems in construction has been carried out illustrating the different approaches advocated by several researches. The different frameworks to represent and access DR have been covered. A CBR framework in support of VE is developed. Within the development process of the system, a multiple level approach to model the system has been presented. The task-method structure provided a good specification for designing, implementing and evaluating the system. It shows how the case-based method solves the tasks of finding, evaluating and developing design alternatives. The case acquisition process, followed by validation and refinement, provided a vocabulary for representing, structuring and indexing DR cases in the case base. Much work remains in order to obtain a finished application. Future work include: acquiring and storing more DR cases; refining the representation structure for DR cases; refining the case indexing; improving the retrieval and matching mechanisms; and implementing the user-interface.

REFERENCES

Alcantara, P.T. (1996) *Development of a computer-understandable representation of design rational to support value engineering*. Unpublished PhD Dissertation, Virginia Tech, Blacksburg, Va.

Chandrasekaran, B., Johnson T. and Smith J. (1992) Task-Structure Analysis for Knowledge Modelling. *Communications of the ACM*, **39**(9), 124–138.

Connaughton, J.N. and Green, S.D. (1996) *Value Management in construction*: *a client's guide*. London: Construction Industry Research and Information Association

de la Garza, J.M. and Alcantara, P.T. (1997) Using Parameter Dependency Network To Represent Design Rationale. *Journal of Computing in Civil Engineering, ASCE*, **2**(11), 102–112.

Fernández, M., Gómez-Péres, A. and Juristo, N. (1997) METHONLOGY: from ontological art towards ontological engineering. *In: Procs. AAAI Spring Symposium Series*, Menlo Park, CA: AAAI Press, 33–40.

Garcia, A.C.B., Howard, H.C. and Stefik, M.J. (1994) Improving design and documentation by using partially automated synthesis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (*AIEDDAM)*, **8**(4), 335–354.

Kitano, H. and Shimazu, H. (1996) The experience-sharing architecture: a case study in corporate-wide case-based software quality control. *In*: Leak, D. (ed.) *Case-based reasoning: experiences, lessons, and future directions*, AAAI, 235–268.

Lee, J. and Lai, K.Y. (1991) What's in design rationale? *Journal of Human-Computer Interaction.* **6**(3), 251–280.

Maher, M. and Garza, A.J.S. (1997) Case-based reasoning in design. *IEEE Expert.* **12**(2), 34–41.

Norman, D. (1986) Cognitive engineering. *In*: Norman, D. and Draper, S. (eds.) *User-centred system design: new perspective on human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum.

Norton, B.R. and McElligott, W.C. (1995) *Value management in construction*: *practical guide*. London: Macmillan.

Pena-Mora, F., Sriram, D. and Logcher, R. (1995) Design rationale for computer-supported conflict mitigation. *Journal of Computing in Civil Engineering, ASCE*, **9**(1), 57–72.

Shipman, F. and McCall, R. (1996) *Integrating different perspectives on design rationale: supporting the emergence of design rationale from design.* Communication, Technical Report 96-001, Centre for the Study of Digital Libraries, Texas A&M University, College Station, Texas.

Yamamoto, S. and Isoda, S. (1986) SOFTDA: A reuse-oriented software design system. *In: Procs Compasac*, Los Alamitos, CA: IEEE Computer Society Press. 284–290.

Watson, I., Basden, A. and Brandon, P. (1992) The client-centred approach: expert systems development. *Expert Systems: The International Journal of Knowledge Engineering*, **9**(4), 181–188.