In Proceedings of the 10^{th} International Conference on Advanced Information System Engineering, CAiSE'98, pages 363-380. Pisa, Italy, June 1998.

Building Quality into Case-Based Reasoning Systems

Igor Jurisica[1] and Brian A. Nixon[2]

[1] University of Toronto, Faculty of Information Studies
140 St. George St., Toronto, ON M5S 3G6, Canada
jurisica@fis.utoronto.ca

[2] University of Toronto, Dept. of Computer Science
Toronto, ON M5S 3H5, Canada
nixon@cs.toronto.edu

Abstract. Complex decision-support information systems for diverse domains need advanced facilities, such as knowledge repositories, reasoning systems, and modeling for processing interrelated information. System development must satisfy functional requirements, but must also systematically meet global quality factors, such as performance, confidentiality and accuracy, called non-functional requirements (NFRs).

To build quality into an important class of decision support systems, case-based reasoning (CBR) systems, this paper presents "QualityCBR," a goal-oriented, knowledge-based approach for systematically dealing with NFRs for CBR systems. With the idea that similar problems have similar solutions, CBR systems store cases (problems with solutions) and solve new problems by retrieving and reusing similar past cases. QualityCBR integrates existing work on CBR and NFRs. It helps developers state and refine NFRs, consider tradeoffs, make decisions and evaluate their impact on NFRs. We illustrate the approach in a complex medical domain, *in vitro* fertilization, where CBR suggests therapy for patients, predicts the probability for successful pregnancy, and determines patient's characteristics that can improve pregnancy rate.

1 Introduction

Complex information systems in both the public and private sectors need a number of advanced facilities, including decision support systems (DSSs), repositories, reasoning systems, and facilities for modeling and processing large amounts of complex information. Many DSSs have been built for individual domains in an *ad hoc* manner. However, to effectively build families of DSSs for complex domains, such as medical, governmental or industrial applications, we need a *systematic* knowledge-based approach to: (1) empower expert user to make effective decisions using a DSS, and (2) address concerns for quality and performance requirements.

Case-based reasoning (CBR) systems [25] are an important class of DSSs that represent experiences (problems with solutions) as cases. Cases are used for solving new problems by accessing past cases and comparing their similarity to a given problem. In this paper we use a generic CBR system called $\mathcal{TA}\mathbf{3}$ (pronounced tah-tree) to build a complex medical DSS, which can be used to advise physicians who prescribe treatment plans for *in vitro* fertilization (IVF) patients [24].

CBR systems must meet *functional requirements,* including retrieving past cases, selecting and reasoning about relevant ones, interactively exploring cases, and adapting them to produce a solution, which is then evaluated. In addition, CBR and other large and complex information systems must meet *non-functional requirements* (NFRs or quality requirements), which are global requirements for quality factors such as performance, accuracy and confidentiality. NFRs are important for the success of complex systems. In the case of medical systems, confidentiality is crucial.

Dealing with NFRs systematically is difficult, because a developer must consider not only requirements, but also implementation alternatives and tradeoffs. In addition, requirements can be in conflict with each other (e.g., it may not be possible to have both expressive representation and fast access time). There are many implementation alternatives, with impact on different NFRs (e.g., having entry clerks type information twice might improve accuracy, but decrease user friendliness). Decisions are interrelated, with a global impact on the target system. For these reasons, one can't simply use "canned alternatives" to meet the quality requirements. Instead, we use an approach where the developer considers the characteristics of a particular system being developed and application needs in a systematic way. This provides a process that helps producing customized systems that meet quality requirements.

Simple applications can usually be built in an *ad hoc* manner, and dealing with requirements may not be difficult. However, a distinguishing aspect of large and complex information systems, whether medical, governmental or industrial, is that characteristics including data, algorithms, domains, requirements, priorities and workload must all be considered. Furthermore, these characteristics interact in complex ways. Hence it is important to deal with them in a systematic way.

We deal with the complexity of these kinds of systems by: using a knowledge-based approach which catalogues expertise, offering competent and efficient CBR facilities, and using a structured approach to deal with NFRs. These facilities are combined in our approach, called *QualityCBR.*

To provide a development process that addresses NFRs for CBR, and is goal-oriented, systematic developer-directed and qualitative, we draw on the "NFR Framework" [4, 6, 31]. The NFR Framework supports this process of building quality in to a system interactively, while considering NFRs *throughout* the development process. Quality requirements are treated as goals to be systematically achieved during the design and development process. The NFR Framework, with its associated tool, helps a developer state and refine NFRs, consider design tradeoffs, justify decisions and evaluate their impact on NFRs, while giving the developer control of the development process. To deal with performance requirements, we draw on the Performance Requirements Framework [33, 34, 35], a specialization of the NFR Framework.

The factors which must be considered during development may all change during a system's lifetime. This greatly increases the complexity of development, and further motivates the need for a systematic approach. By using a knowledge-based approach, and by drawing on the NFR Framework's facilities for dealing with change [7], we can systematically deal with change.

There are two possible combinations of techniques for CBR and NFRs: (1) using CBR to support reasoning about and reuse of NFRs, and (2) using NFRs to systematically build quality into a CBR system. This paper addresses the latter issue, using the QualityCBR approach. In particular, we describe the process of using QualityCBR and providing catalogues that deal with alternative implementations. QualityCBR draws on a flexible knowledge representation language for information systems — Telos [30], relevance assessment [20], similarity-based retrieval algorithm [22], and the NFR Framework's goal-oriented, qualitative approach [31]. In addition, QualityCBR uses knowledge discovery algorithms [1, 24] and data model implementation experience [36]. QualityCBR is applied to a complex medical DSS for IVF practitioners $\mathcal{TA}$**3** [24]. During the development of the system we considered some NFRs, albeit in an *ad hoc* manner. We show how a developer could use QualityCBR to systematically build a CBR system for IVF by addressing NFRs such as performance – "Select relevant cases quickly", confidentiality – "Store patient records securely", and informativeness – "Display results informatively". We also consider the impact of some changes.

2 The QualityCBR Approach

This section presents the elements of the QualityCBR approach, for addressing non-functional requirements of case-based reasoning systems. Traditionally, CBR system were developed for a specific application. The presented work aims at defining a generic framework that is adaptable for different domains, while ensuring that both functional and non-functional requirements are systematically met.

2.1 Case-Based Reasoning

This section describes principles of case-based reasoning (CBR) and a particular prototype $\mathcal{TA}$**3**. Our aim is a flexible system that can be applied to various domains, without sacrificing system performance. We consider system performance as a quality of solution and its timeliness.

A case-based reasoning approach [25] relies on the idea that similar problems have similar solutions. Facing a new problem, a CBR system retrieves similar cases stored in a case base and adapts them to fit the problem at hand. Informally, a case comprises an input (the problem), an output (the solution) and feedback (an evaluation of the solution). CBR involves the process of: (1) *Accepting* a new problem description; (2) *Retrieving* relevant cases from a case base (past problems with similar input); (3) *Adapting* retrieved cases to fit the input problem and finding a solution to it; and (4) *Evaluating* the solution (producing feedback for the case).

Considering the above CBR cycle, one can say that the more similar the cases are, the less adaptation is necessary, and consequently, the proposed solution may be more correct. Then, an important task is how to measure case relevance (similarity or closeness) to guarantee retrieving only highly relevant cases, i.e., cases that are similar according to specified criteria, and thus can be useful in solving the input problem in a particular context. Thus, we need a variable-context similarity assessment. In many processes, it is better to retrieve fewer cases, or none, than to retrieve less useful cases that would result in a poor solution. But similarity of cases is only one measure of system quality. It is also important that the solution be provided quickly. It should be noted that the tradeoff between closeness and timeliness of a solution depends on requirements of a particular application [19]. For these reasons we use a variable-context similarity assessment and case base clustering as described next.

$\mathcal{TA}$**3** is a CBR system, which uses a variable-context similarity-based retrieval algorithm [22] and a flexible representation language. Knowledge must be represented in a form appropriate for the intended user, and the representation should be rich enough to support complex, yet efficient processing [23]. Cases are represented as a collection of attribute-value pairs. Individual attributes are grouped into one or more categories [22]. Categories bring additional structure to a case representation. This reduces the impact of irrelevant attributes on system performance by selectively using individual categories during matching. As a result, we get a more flexible reasoning system [19], a more comprehensible presentation of complex information [20], improved solution quality [24], and improved scalability [23].

During the CBR process, we want to handle partial as well as exact matches. We have a *partial matching* when attribute values of one case match only a subset of values of another case. In order to retrieve and control both exact and partial matching, a view of a case, called a *context,* is defined. Thus, a case to be interpreted in a given context. By controlling what constitutes a partial match, context specifies important attributes and how “close” an attribute value must be. We say that a case satisfies (or matches) a particular context, if for each attribute specified in the context, the value of that attribute in the case satisfies the constraint [22]. Thus, the matching process can be described as a constraint satisfaction problem [40]. The quality of the matching process is measured by the closeness of retrieved cases [22], timeliness of the answer [23], and adaptability of the suggested solution [26].

Ortega has shown that partial *m-of-n* matches improve performance if m is reasonably selected [37]. Our approach of representing cases as sets of Telos-style categories [30], each comprising a set of tuples, allows for multiple levels of *m-of-n* matching. Thus, important attributes may require *n-of-n* matches for a given category, and less important attributes may allow for *k-of-n* matches ($k < n$). The problem is to find these attribute groupings, i.e., a context that specifies which attributes are needed for accurate prediction, and what range or similarity should be allowed for attribute values.

This knowledge can be automatically discovered [24] and can be used for case base clustering by: (1) appropriately grouping attributes into categories (clustering of attributes); (2) discovering what values are "close" for particular attributes (clustering of attribute values); and (3) structuring the case base into clusters of relevant cases (clustering of cases).

2.2 Handling Non-Functional Requirements

The NFR Framework [4, 31] helps a developer represent and use key concepts about NFRs (e.g., security and performance), the particular domain (e.g., IVF), and development expertise, (e.g., CBR, databases and system development).

Being influenced by work in DSSs [28], the NFR Framework maintains a concise and structured *development graph* whose *components* record the developer's goals, decisions and design rationale. The developer states a set of NFRs for the system, which are represented as *goals* that are considered throughout the system development process. In trying to meet the requirements, developers are helped in choosing among design alternatives, which are organized in a *catalogue* of *methods*. Partial positive or negative relationships among goals are recorded as qualitative *link types*. Knowledge of design tradeoffs is arranged in a catalogue of *correlation rules*. After decisions are made, the NFR Framework uses its *evaluation algorithm* to help the developer determine if overall goals have been met. Section 3.2 presents the components of the NFR Framework in more detail, and illustrates their use.

The NFR Framework has been previously applied to information systems in several domains, in both the public and private sectors (e.g., health insurance, banking and government systems) [5, 7]. Its approach can be specialized to deal with a number of NFRs, such as performance [33, 34, 35], accuracy [4] and security [3]. For performance, for example, we represented principles for building good response time into systems [39] and arranged information system implementation knowledge using a layering approach [17] based on data model features, to reduce the number of issues considered at a time.

The "NFR Assistant" prototype tool [4], provides support to a developer using the NFR Framework, by providing catalogues of concepts and methods, aiding the construction and evaluation of development graphs, and keeping track of correlations. The tool draws on the ConceptBase system [18] which uses the Telos [15, 30] knowledge representation language. A specialization of the tool, the Performance Requirements Assistant [34, 35], offers catalogues of concepts and techniques for treating performance requirements, using other Telos-based knowledge base management tools,[1] but offers only a subset of the functionality of the NFR Assistant.

[1] M. Stanley's *Telos sh* and B. Kramer's *RepBrowser*, at the University of Toronto.

2.3 Cataloguing CBR and NFR Knowledge

The QualityCBR approach organizes knowledge about issues and techniques for CBR and NFRs. These knowledge bases, represented in Telos, serve as a basis for recording experts' knowledge and are used during system development, They help a user to satisfy NFRs (such as performance and confidentiality), effectively use CBR techniques (e.g., knowledge representation, retrieval), and consider particular characteristics of the system under development (e.g., workload, confidentiality concerns).

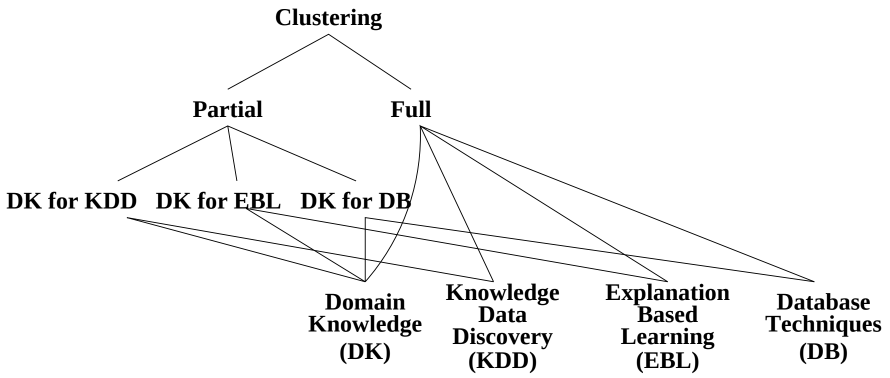

Fig. 1. A Catalogue of Clustering Techniques for CBR.

Some steps, typically done early in using the approach, involve defining and organizing a variety of types of knowledge applicable to the system under development. This produces a number of *catalogues* of concepts:

- Concepts about a particular class of *reasoning systems* (e.g., CBR), such as components of the CBR system, problem decomposition techniques and implementation alternatives. Figure 1 shows a sample catalogue for implementation alternatives for clustering techniques. Specialized catalogues draw on combinations of aspects, e.g., domain knowledge for knowledge data discovery.
- Concepts about particular *NFRs* (e.g., performance and security). For example, a terminology of performance concepts is made available, along with a catalogue which shows the impact of implementation techniques on time and space goals [34].
- Concepts about the particular *application domain*, e.g., IVF: descriptions of processes (e.g., a cycle of patient treatment) and workload (e.g., number of patients).
- Generic concepts associated with the NFR Framework, e.g., definitions of the components of development graphs which record developers' decisions.

3 Illustrating the QualityCBR Approach

This section shows the use of QualityCBR's components and cataloguing to support several NFRs for the IVF domain. Section 3.1 presents the domain of our study of the IVF system. We consider *top level non-functional requirements* for an IVF system, which could be stated by a developer, involving: *performance* – patient records must be retrieved and analyzed quickly (Section 3.2), and *confidentiality* – records must be stored securely (Section 3.3). In addition, the system should be *robust* and *user-friendly* (Section 3.4).

3.1 Functional Requirements in the IVF Domain

In vitro fertilization (IVF) is an example of a complex medical domain, where DSS can be used to suggest the hormonal treatment and to support research [24]. Individual patients respond to the treatment differently. A patient's response and the pregnancy rate depends on many attributes. While experienced doctors can use their knowledge to suggest a treatment for a patient, it is difficult for them to perceive trends and make informed decisions to optimize success rates for each individual infertile couple. This is especially a concern when knowledge about influencing factors changes.

Prediction of the likelihood of pregnancy involves suggestion of a treatment. This is performed in two stages. First, given initial information about the patient (diagnosis, previous treatment history, etc.) the task is to find similar patients from the case base and make a suggestion of how to treat the current patient to increase the probability of successful pregnancy. This includes finding all relevant cases, and considering retrieved cases with pregnancy as successful examples and retrieved cases without pregnancy as negative cases. An *adaptation* process uses this information to suggest values for remaining attributes in the current case, namely how long the patient should be stimulated and what amount of the hormones should be used. Second, after the initial treatment is completed, additional attributes are available (patient's responsiveness to the hormonal stimulation). The task is then to predict the outcome of the whole treatment, i.e., to predict likelihood values for pregnancy and for unsuccessful cases. The prediction task can also be considered as an optimization problem: for a given patient minimize the amount of hormonal therapy required, without compromising the outcome.

Knowledge discovery is used to find regularities in the case base by using knowledge-mining techniques, as well as to suggest missing data. Here, physicians have no particular case in mind, however, they may consider the whole knowledge base or only certain cases. Knowledge mining in $\mathcal{TA}\mathbf{3}$ involves finding a context in which a particular group of cases is considered similar. The user has the ability to specify a threshold, which controls the quality and the quantity of discovered information [24].

Considering that each patient is described by about a hundred attributes [24], that there are about 600 patients per year per clinic and that there are about 300 IVF clinics in North America [29], the problem is not simple. Moreover, IVF information is more sensitive than general medical information and the complex IVF process involves various professionals, which need to access part or whole information about the patient. IVF has relevance to both the public and private sectors. In the Province of Ontario, Canada, for example, publicly-funded health insurance covers the cost of IVF for certain forms of infertility, e.g., tubal blockage, while others are not covered, and are handled by private clinics.

3.2 Dealing with Performance Requirements

We now show how performance requirements for the IVF domain are handled using QualityCBR. We also describe components of the NFR Framework used in QualityCBR.

System performance is an important factor for complex applications. Good performance includes fast response time and low space requirements. For the IVF system, a developer might state that one important goal is to have fast response time when accessing patient records, for reasoning as well as case updating. This requirement is represented as a *goal*: `Time[Patient Records and Reasoning]`, as shown in Figure 2. `Time` is the *sort* of the goal (i.e., the particular NFR concept, addressed by the goal) and `[Patient Records and Reasoning]` is the *parameter* (i.e., the subject) of the goal. (The entries within circles will be discussed below.) Another main goal is to have fast response time for reasoning operations done by researchers, represented by `Time[Research Reasoning]`.

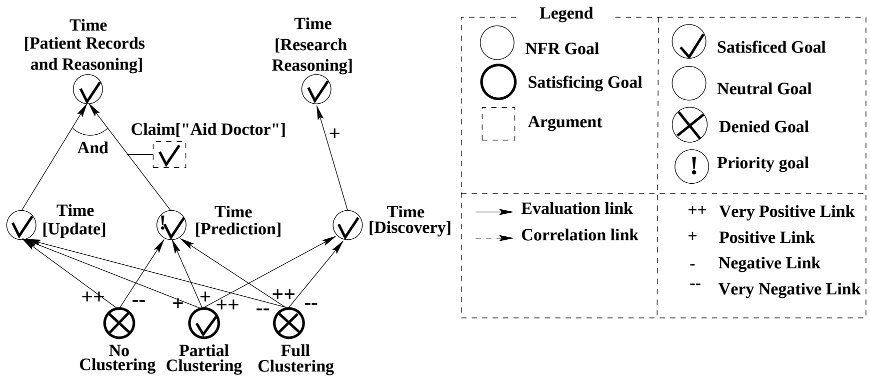

Fig. 2. Dealing with Performance Requirements for Reasoning.

Using *methods* and catalogues of knowledge (for performance, CBR, IVF, etc.), goals can be *refined* into more specialized goals. Here, the developer used knowledge of the IVF domain to refine the time goal for patient information into two goals, one for good response time for updating patient records and the other for good response time for the retrieval and decision making process. These two *offspring* goals are connected by an *And* link to the *parent* goal. This means that if both the goal for fast updates and the goal for fast prediction are accomplished then we can say that the parent goal of fast access to patient records will in some sense be accomplished. The NFR Framework takes a qualitative, "satisficing" approach, in which goals are more-or-less met, although they may not be satisfied in an absolute sense [38].

Similarly, the goal of good response time for research reasoning can be refined into a goal of fast response for the "discovery" process which searches patient records for patterns. Here, the parent has one offspring, connected by a positive ("+") link, which indicates that accomplishing the offspring will contribute positively towards accomplishing the parent goal. Other types of relationships can be shown by other *link types* (see Figure 2).

In building quality into a system, it is important to identify priorities.

For the case of building performance in, we should identify time-critical operations as well as those which dominate the workload [39]. Here, we identify the prediction operation as being time-critical (indicated by "!"), and provide a reason using *domain knowledge:* it is important to aid the doctor by quickly suggesting a treatment. This is an example of recording *design rationale* [28] – the reasons for decisions – using the NFR Framework's *arguments.* As part of the development graph, arguments are available when making further decisions and changes.

It is important to note that the *developers* use their expertise to determine what to refine, how to refine it, to what extent to refine it, as well as when to refine it. The NFR Framework and its associated tool help the developer, do some consistency checking, and keep track of decisions, but it is the *developer* who is in control of development process.

Implementation Alternatives.

In moving towards a target system, one must consider implementation alternatives for case base *clustering,* which appropriately groups attributes, their values, and relevant cases together. The main concern for clustering is with the storage of patient records, which besides general patient information (name, address, etc.) consist of attribute-value pairs describing the diagnosis of infertility, previous and current treatments, the result, etc. Effective storage of this information facilitates the various CBR operations, because individual pieces of information have different importance and different effects on the treatment and on the overall outcome. Currently, the information is recorded in a paper-based form with general patient information being sent to a central hospital computer. A computerized IVF case base is populated in a batch process.

Many of the implementation alternatives (shown as dark circles in Figure 2) will be drawn from appropriate catalogues. Implementation alternatives for the following clustering operations must be considered:

- *Storage and update.* In the IVF application, data entry and updates have the form of filling in blanks, either selecting a value from a pick-lists or typing it. Considering the amount of data in one clinic, storage and update are not major problems. However, taking into account possible extensions, e.g., linking several IVF clinics in a network to share their case bases, it is useful to note this requirement.

- *Prediction.* A doctor uses the system to suggest a hormonal therapy for the current patient (see Section 3.1). It is important that the accuracy of predicted information is within reasonable bounds and a solution is provided swiftly. There is a relationship between accuracy, time and space: the more cases are stored, the more accurate solutions can be provided, but the longer it takes to find cases relevant to a given problem.
- *Knowledge discovery.* Treatment protocols can be improved by using knowledge discovery [24]. Discovered knowledge is used to organize attributes into categories, and cases into clusters (equivalence classes).

The above considerations affect implementation alternatives ("satisficing goals") for case base clustering: (1) the system may not use any clustering; (2) it may use full clustering; or (3) an hybrid, a partial clustering scheme can be deployed; further variations of clustering from the methods catalogue can be considered (see Figure 1). Without clustering, updates are faster, as data need not be reorganized; however, prediction is slower as there is no clustering to aid the retrieval process. Thus, at the bottom left of Figure 2, `No Clustering` is shown to have a positive impact on update time, and a negative impact on prediction time. Full clustering is done by knowledge discovery: it speeds up prediction, but hinders update time. No and full clustering each slow down at least one of the three operations. The developer can formulate alternatives which reduce or avoid this problem. Partial clustering may start with cases clustered using domain knowledge, but may subdivide certain clusters into more detailed groups. Its main advantage is that it speeds up all three operations, instead of slowing any of them. However, no clustering is better ("++") than partial for update, and full clustering is better than partial for retrieval. Thus, partial clustering offers intermediate performance for some operations, but avoids bad performance for all of them. As a result, partial clustering is selected ("$\surd$") over the unchosen ("$\times$") alternatives. Note, that an IVF facility that does not support research may give low priority to performance for knowledge discovery. Since the hormonal therapy suggestion would have high priority, full clustering would be selected.

Evaluating Goal Accomplishment.

After decisions are made, the developer can determine their overall impact on the system's requirements. The developer is aided by the NFR Framework's semi-automatic *evaluation algorithm,* which examines the development graph, generally bottom-up. It starts with implementation decisions to accept ("$\surd$") or reject ("$\times$") alternatives (shown in dark circles at the bottom of Figure 2),

Results then propagate upward along evaluation links. Evaluation assigns values (e.g., "$\surd$" or "$\times$") to parent goals based on the values of offspring goals, and and the relationships (link types, e.g., "+" or "-") between offspring and parent goals. For example, with a "+" link type, meeting ("$\surd$") the offspring (e.g., Partial Clustering) helps meet ("$\surd$") the parent; however, if the offspring is denied ("×", not achieved), the parent will be denied ("×"). The "-" link type can be thought of as giving the parent the "opposite" of the offspring's label. Values from all applicable offspring propagate to a parent. Here, partial clustering helps quickly accomplish updating, presentation and discovery. During the process, the developer may step in, to determine propagated values. For example, if a parent goal received positive and negative values from different offsprings, the developer is able to resolve the conflict using domain knowledge.

It should be noted that not all goals can always be met, but performance can be enhanced if the priorities are accomplished [39]. As presented in Figure 2, the critical goal for prediction has been met. Since the update time goal was also met, the top goal for records and reasoning was met. As the discovery goal was met, the top goal for research reasoning was also met.

Dealing with Changes in Priorities.

Let's consider four imaginary IVF clinics with different priorities: (1) fast update of records, (2) fast prediction, (3) both fast prediction and fast update are important, and (4) fast case base analysis (discovery). Depending on the priorities, we may adjust the solution of Figure 2 by choosing a different alternative. As a result, the first clinic would not use clustering, the second would use full clustering, and the third and fourth clinics would achieve their requirements by deploying partial (hybrid) clustering. This is an example of reusing an existing development graph, which uses the NFR Framework's components to capture decisions and rationale, as a resource for rapid analysis of the impact of change upon the achievement of NFRs [7]. In addition, we have used domain knowledge, priorities and performance catalogues to produce customized solutions which meet needs of a particular organization.

3.3 Security Requirements

Security is an important factor, especially in medicine, and IVF is a particularly sensitive application. Security includes such concepts as integrity, confidentiality and availability [4], whose combination is used in a generic methodology for medical data security [14].

For the IVF clinic, we identified two primary goals (top part of Figure 3): (1) The physical integrity of gametes of the patient is extremely crucial (indicated by "!!"). (2) The confidentiality of patient data should be maintained. A third goal is to maintain the professional integrity (reputation) of the doctor (researcher).

Physical Integrity of Patient Material.

The crucial concern is that a patient's gametes must not be mistaken for someone else's. Thus, accurate identification of gametes strongly contributes to physical integrity. This can be accomplished either by using patient's name or an identifying number. Using only a number might contribute to *confidentiality*; for example, the lab technician, who deals with gametes, but not directly

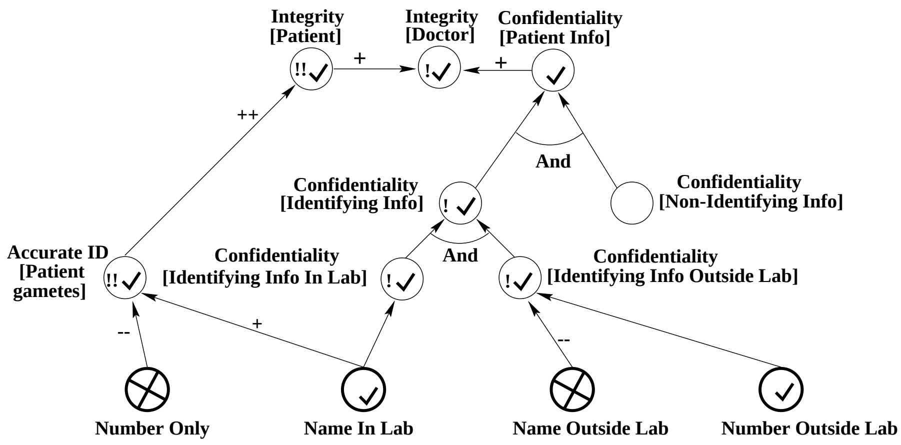

Fig. 3. Dealing with Security Requirements for an IVF Clinic.

with patients, could in principle use only numbered dishes without knowing patient's name. However, this could increase the chance of confusing gametes of two patients, which must be avoided. Instead, the lab labels dishes with gametes using the patient's name, which is only made available to authorized personnel, including the technician. The analysis is shown in the lower left of Figure 3.

It's interesting to note the interaction between the goals of physical integrity of gametes, and the confidentiality of patient information, and the resolution of the conflict to benefit the higher-priority goal. In addition, to help meet *both* goals, the lab has a system of *physical* security. While this is not shown in the figure, it is important to note that measures taken outside the computer system can have an impact on the NFRs being considered.

Confidentiality of Patient Information.

The IVF clinic records some basic identifying information about a patient (name, age, hospital number, etc.), a record of the patient's visits during a treatment cycle, treatments made, and observations. In addition, the central hospital computer maintains accounting records, which do not have the details of IVF treatment. Patient information is used for both tracking individual patients for treatment, and for reviewing groups of patients for research purposes. This dual usage complicates confidentiality considerations. Furthermore, researchers sometimes need to obtain further information about particular patients, hence the statistical research information must contain some patient identifiers. Clearly, access to medical data should be restricted to authorized persons, in relation to their status [13]. In the case of the IVF clinic, the mere fact that someone is an IVF patient is considered quite a personal matter, hence confidential [10].

The issue of security of statistical data is a complex one. According to [32]: "confusion still surrounds the question of whether privacy can be fundamentally violated by statistics drawn from personal records". However, it was also shown that statistical information could provide detailed information about individuals [9]. The more information pieces are tied together the more identifiable the individual is.

Confidentiality of patient information must handle two goals: (1) information that identifies a patient and (2) information that does not (see Figure 3). In IVF domain, data can be used both for clinical treatment and for research. Thus, the goal of confidentiality of identifying information can be refined by the developer to handle these situations. As discussed earlier, the patient's name will be used within the lab, to meet the overriding goal of integrity of gametes, which (along with the goal of confidentiality of records) will be aided by physical security measures. To reduce the risk of names being divulged to third parties, the patient's name should not be used outside the lab. Instead, an identification number (hospital number, sample number or user generated number [16]) is used.

Evaluating the Overall Impact of Decisions.

Using a name within the lab helps accurately identify gametes, and maintain its physical integrity. The selective use of name and number provides confidentiality of identifying information, both inside and outside the lab. Meeting this critical goal contributes to the overall confidentiality of patient information. In turn, meeting both that confidentiality goal and the goal for physical integrity of gametes contributes positively to maintaining the professional integrity of the doctor (researcher). While we did not initially identify professional integrity as a main goal, it is very interesting to see that the result of our analysis using the NFR Framework was in harmony with the observation that the integrity of researchers is paramount [2].

3.4 Other NFRs

Additional NFRs for the presented system include: (1) *Robustness:* the ability to gracefully handle a variety of situations; (2) *User friendliness:* providing the right degree of assistance to users; and (3) *Informativeness:* providing the right amount of information, appropriately arranged.

Robustness concerns for the CBR system include: (1) reducing the effect of irrelevant attributes on CBR so that the prediction accuracy does not degrade with an increased number of irrelevant attributes and presenting only attributes relevant to the task; (2) fault tolerance during data entry and reasoning. Thus, the goal for robustness of the system is refined into goals for data integrity, robustness of reasoning and robustness of presentation (Figure 4, top left).

Data integrity is important [14]. As suggested in [13], verification and validation of data completeness and accuracy is an additional measure ensuring data integrity. Thus, especially in the *early* stages of system development, *all attributes* available should be used. This allows for correlating the attributes, which can lead to identifying data integrity violations. However, if all attributes are also used in later stages, this would lead to problems with reasoning and

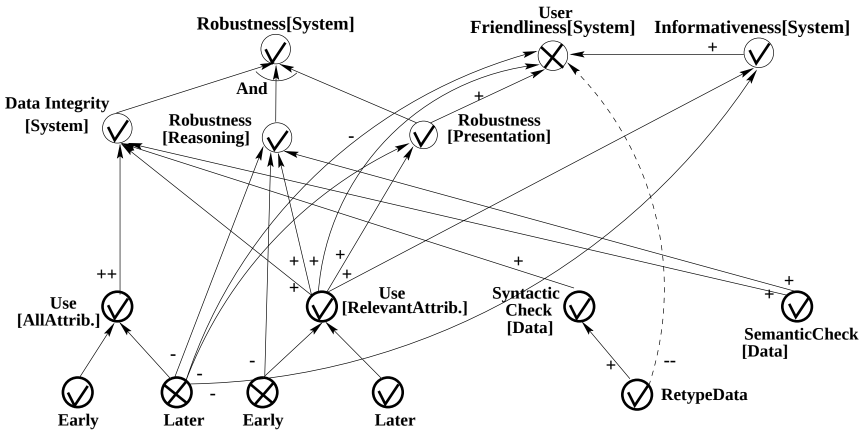

Fig. 4. Dealing with Several NFRs for the System.

presentation. Thus, only *relevant attributes* should be used in *later* phases. As described in Section 2.1, knowledge-discovery techniques can be used to locate features relevant to the problem solving task [24]. Using only relevant features improves flexibility [20], accuracy [22], and efficiency [23]. The effect of this selective use of attributes contributes positively to the top goals of robustness and informativeness, both of which are accomplished, but user friendliness is not accomplished for the reasons described below.

Generic relationships between NFRs and satisficing goals can be catalogued in advance as "correlation rules." These relationships can then be detected by the NFR assistant system, even if the developer has not explicitly linked the goals. Here, to syntactically verify data, the developer has the operator type it twice, which is helpful for data integrity. However, the NFR Assistant system detects that this is bad for user friendliness (the "correlation link," shown as a dashed line, is negative), which results in the user friendliness goal not being met. Correlation links (dashed lines) propagate values in the same way that evaluation links to.

Selecting Different Implementation Alternatives.

Recognizing that system friendliness is important for users, the developer may consider ways of achieving this goal, such as implementation alternatives presented in Figure 5 (an extension of the lower left part in Figure 4). These include another user-oriented method – menu-driven input, and as system-provided checking – a dictionary of used terms, and using n-grams, which supports automatic recognition of misspelled words. In the example, n-grams are selected, so that syntactic checking remains accomplished, albeit by a different method, which contributes *positively* to user friendliness. In addition, the chosen methods for displaying all attributes early and relevant attributes later remain unchanged from Figure 4, hence continue to contribute positively to user friendliness, which is now accomplished.

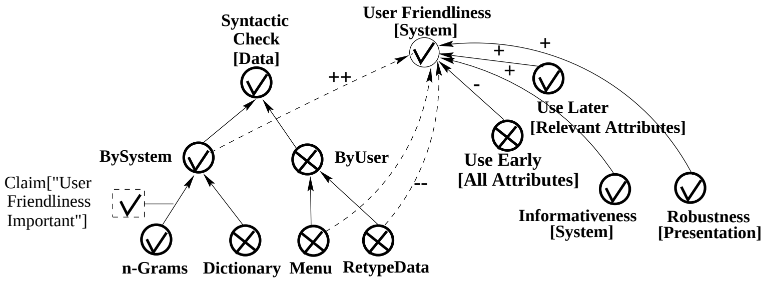

Fig. 5. A Re-examination of Methods for Syntactic Checking.

This is another example of dealing with change – namely, a change in implementation alternatives. The net result is that the developer's expertise was used to accomplish the remaining top goal of user friendliness, while maintaining robustness and informativeness. This was done by reusing the information already captured in Figure 4, which dealt with several NFRs.

4 Conclusions

We are concerned with quality issues in decision support for complex information systems. We have presented an approach, called QualityCBR, for dealing with non-functional requirements for case-based reasoning systems. This integrates the NFR Framework's systematic process for building quality with the $\mathcal{TA}$**3** CBR system, intended for decision support. In developing QualityCBR, catalogues have been organized to represent diverse knowledge concerning CBR, NFRs, IVF, and development techniques. By drawing on several axes (e.g., CBR and performance), we can focus on small groups of specific methods. This approach is similar to the organization of information system performance requirements issues [34]. We feel that the use of such catalogues is helpful in dealing with NFRs in medical computing and other complex domains, public and private.

To demonstrate how a developer can use QualityCBR to deal with conflicting and changing requirements, we illustrated its use in a medical domain. A variety of NFRs (e.g., performance, security, informativeness), and tradeoffs between individual requirements have been considered.

We also found that the NFR Framework's development graphs and change facilities [7] made the process of dealing with change easier. In this paper we have considered changes in priorities of NFRs and in implementation techniques. This is consistent with results of using the NFR Framework to deal with changes in requirements for a commercial system.

$\mathcal{TA}$**3**'s performance evaluation has been conducted on several domains: prediction and knowledge mining in medicine [24], [24], control task in robotic domains [21], character recognition [22], iterative browsing and intelligent retrieval [20]. Each domain has different characteristics; this helps evaluation of different aspects of the system. We have evaluated both the competence [24] and scalability [23] of the system.

It would be interesting to see if QualityCBR could be used to use other goal-oriented approaches to requirements engineering, e.g., [8, 11, 12]. This would draw on several facilities, such as representation of goals, priorities, and positive and negative links.

We would like to conduct fuller studies of applying $\mathcal{TA}$**3** to a variety of areas, both public and private, such as medicine, engineering and commerce, which require a variety of NFRs. Notably, we plan to explore the capability of using QualityCBR during building engineering applications, such as robotics [21], where real time response is critical. For example, the use of an "any time system" (which must produce a valid answer at any time) entails flexible and adaptive procedures to meet accuracy and safety requirements [19]. These steps will help us to better asses the generality of the approach and proposed combined tools to evaluate its costs and benefits. Studies should use a methodology, such as [27] which allows us to have the kind of confidence in the results that one would have in using the scientific method.

An important direction for future work is to apply CBR to the NFR Framework and its associated tool. For example, sets of development graphs for a variety of systems could be examined and analyzed to find patterns (templates) of sequences of method applications. This could be aided by facilities for critiquing and rationalizing specifications [11]. Such templates could then be used as larger building blocks when using the NFR Framework to develop a variety of systems. Thus, CBR would provide underlying technology for a reuse assistant for the NFR Framework.

We trust that building quality into CBR, and using CBR in tools for dealing with NFRs, will aid the development of complex information systems for a variety of public and private domains.[2]

References

1. R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Eng. Learning and Discovery*

[2] *Acknowledgments.* The authors were with the Dept. of Computer Science, University of Toronto, when this paper was initially prepared. This research was supported by the Information Technology Research Centre of Ontario, Canadian Consortium for Software Engineering Research and the IBM Centre for Advanced Studies. We thank Robert F. Casper and Andrea Jurisicova for much helpful information on IVF procedures. Over the years we have benefited from the insight and direction of Professors John Mylopoulos and Janice Glasgow. This paper benefits from earlier NFR work with Lawrence Chung and Eric Yu. Our families have been a constant source of support.

in Knowledge-Based Databases, 5(6):914–925, 1993.

2. R. Behi and M. Nolan. Ethical issues in research. *British Journal of Nursing*, 4(12):712–716, 1995.
3. L. Chung. Dealing with security requirements during the development of information systems. In *Proc. 5^{th} Int. Conf. on Advanced Information Systems Engineering*, pages 234–251, Paris, France, 1993. Springer-Verlag.
4. L. Chung. *Representing and Using Non-Functional Requirements: A Process-Oriented Approach.* PhD thesis, Dept. of Computer Science, Univ. of Toronto, 1993.
5. L. Chung and B. A. Nixon. Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach. In *Proc. 17^{th} Int. Conf. on Software Eng.*, pages 25–37, Seattle, WA, 1995.
6. L. Chung, B. A. Nixon, J. Mylopoulos, and E. Yu. *Non-Functional Requirements in Software Engineering.* In preparation, 1998.
7. L. Chung, B. A. Nixon, and E. Yu. Dealing with Change: An Approach using Non-Functional Requirements. *Requirements Engineering*, 1(4):238–260, 1996. An earlier version, Using Non-Functional Requirements to Systematically Support Change, appeared in Proc. 2^{nd} IEEE Int. Symp. on Requirements Eng., York, U.K., 1995, pp. 132-139.
8. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.
9. D. E. Denning, P. J. Denning, and M. D. Schwartz. The tracker: A threat to statistical database security. *ACM TODS*, 4:76–96, 1979.
10. C.L. Early and L. C. Strong. Certificates of confidentiality: A valuable tool for protecting genetic data. *American Journal of Human Genetics*, 57(3):727–731, 1995.
11. S. Fickas and P. Nagarajan. Being suspicious: Critiquing problem specifications. In *Proc. AAAI–88*, pages 19–24, Saint Paul, MN, 1988.
12. S. F. Fickas. Automating the transformational development of software. *IEEE Trans. on Software Eng.*, SE–11(11):1268–1277, 1985.
13. F.H. France and P.N. Gaunt. The need for security – A clinical view. *International Journal of Bio-Medical Computing*, 35(Suppl. 1):189–194, 1994.
14. S. Furnell, P. Gaunt, G. Pangalos, P. Sanders, and M. Warren. A generic methodology for health care data security. *Medical Informatics*, 19(3):229–245, 1994.
15. S. Greenspan, J. Mylopoulos, and A. Borgida. On Formal Requirements Modeling Languages: RML Revisited. In *Proceedings, 16th International Conference on Software Engineering*, pages 135–147, Sorrento, Italy, 1994.
16. F. Honig. When you can't ask their name: Linking anonymous respondents with the Hogben number. *Australian Journal of Public Health*, 19(1):94–96, 1995.
17. W. F. Hyslop. *Performance Prediction of Relational Database Management Systems.* PhD thesis, Dept. of Computer Science, Univ. of Toronto, 1991.
18. M. Jarke. *ConceptBase V3.1 User Manual.* Univ. of Passau, 1992.
19. I. Jurisica. Supporting flexibility. A case-based reasoning approach. In *The AAAI Fall Symposium. Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities*, Cambridge, MA, 1996.
20. I. Jurisica. Similarity-based retrieval for diverse Bookshelf software repository users. In *IBM CASCON Conference*, pages 224–235, Toronto, Canada, 1997.
21. I. Jurisica and J. Glasgow. A case-based reasoning approach to learning control. In *5^{th} International Conference on Data and Knowledge Systems for Manufacturing and Engineering, DKSME-96*, Phoenix, AZ, 1996.

22. I. Jurisica and J. Glasgow. Case-based classification using similarity-based retrieval. *International Journal of Artificial Intelligence Tools. Special Issue of IEEE ICTAI-96 Best Papers*, 6(4):511–536, 1997.
23. I. Jurisica and J. Glasgow. An efficient approach to iterative browsing and retrieval for case-based reasoning. In Angel Pasqual del Pobil, Jose Mira, and Moonis Ali, editors, *Lecture Notes in Computer Science, IEA/AIE*98.* Springer-Verlag, 1998.
24. I. Jurisica, J. Mylopoulos, J. Glasgow, H. Shapiro, and R. F. Casper. Case-based reasoning in IVF: Prediction and knowledge mining. *Artificial Intelligence in Medicine*, 12(1):1–24, 1998.
25. D. Leake, editor. *Case-Based Reasoning: Experiences, lessons and future directions.* AAAI Press, 1996.
26. D. Leake, A. Kinley, and D. Wilson. Case-based similarity assessment: Estimating adaptability from experience. In *Proc. of the AAAI-97*, 1997.
27. A. S. Lee. A scientific methodology for MIS case studies. *MIS Quarterly*, pages 30–50, 1991.
28. J. Lee. Extending the Potts and Bruns Model for Recording Design Rationale. In *Proc., 13th Int. Conf. on Software Eng.*, pages 114–125, Austin, Texas, 1991.
29. P. M. McShane. Customized comparative clinical results assessment of your IVF program. *IVF America*, 1993.
30. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, 1990.
31. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18:483–497, 1992.
32. H.B. Newcombe. When privacy threatens public health. *Canadian Journal of Public Health. Revue Canadienne de Santé Publique.*, 83(3):188–192, 1995.
33. B. A. Nixon. Dealing with performance requirements during the development of information systems. In *Proc. IEEE International Symposium on Requirements Engineering*, pages 42–49, San Diego, CA, 1994.
34. B. A. Nixon. Representing and using performance requirements during the development of information systems. In *Proc. 4^{th} Int. Conf. on Extending Database Technology*, pages 187–200, Cambridge, U.K., 1994.
35. B. A. Nixon. *Performance Requirements for Information Systems.* PhD thesis, Dept. of Computer Science, Univ. of Toronto, 1997.
36. B. A. Nixon, K. L. Chung, D. Lauzon, A. Borgida, J. Mylopoulos, and M. Stanley. Design of a compiler for a semantic data model. In J. W. Schmidt and C. Thanos, editors, *Foundations of Knowledge Base Management*, pages 293–343. Springer-Verlag, 1989.
37. J. Ortega. On the informativeness of the DNA promoter sequences domain theory. *Journal of Artificial Intelligence Research*, 2:361–367, 1995. Research Note.
38. H. A. Simon. *The Sciences of the Artificial, 2^{nd} Edition.* MIT Press, Cambridge, MA, 1981.
39. C. U. Smith. *Performance Engineering of Software Systems.* Addison-Wesley, Reading, MA, 1990.
40. P. R. Thagard, K. J. Holyoak, G. Nelson, and D. Gotchfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310, 1990.

This article was processed using the LaTeX macro package with LLNCS style