Representation and Utilization of Non-Functional Requirements for Information System Design[1]

Lawrence Chung

Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract The complexity and usefulness of large information systems are determined partly by their functionality, i.e., what they do, and partly by global constraints on their accuracy, security, cost, user-friendliness, performance, and the like. Even with the growing interest in developing higher-level models and design paradigms, current technology is inadequate both *representationally* for expressing such global constraints as formal *non-functional* requirements and *methodologically* for utilizing them in generating designs. We propose both a representational and methodological framework for non-functional requirements, focusing on accuracy requirements. With the premise that accuracy is an inherent semantic attribute of information, we take a first step towards establishing a representational basis for accuracy. To guide the design process and justify design decisions, we propose a goal-oriented methodology. In the methodology, accuracy requirements are treated as (potentially conflicting) goals, for which two types of methods are presented: one for decomposing the goals in terms of affected design components, and the other for contributing, either positively or negatively, to goal satisfaction. Non-functional requirements are further investigated for their cooperation and conflicts, which enables the assessment of the quality of overall design. A detailed illustration demonstrates how the framework aids a designer's decision-making process by recommending the types of consultation needed with users in the intended application domain.

1 Introduction

Large information systems such as those for student enrolment, credit validation, or research administration deal with a large schema involving hundreds of entity types, constraints, and a large volume of transactions. In the context of the present paper, the *design* of such systems involves the description of the conceptual structure of entities and processes in terms of the building blocks offered by a data model at the conceptual level (for instance, entity-relationship models [Chen81], or semantic data models — see [Peckham88] and [Albano89] for overviews). While each design needs to be *implemented* by translation to the logical level such as relational data model [Borgida89, 90], we focus attention on those layers concerned with requirements specifications and design. *Requirements specifications* describe our conceptualization of what has to hold in the intended application domain regarding not only the *information system* but its *environment* and

[1]In R. Anderson, J. A. Bubenko, Jr., A. Sølvberg (Eds.), *Advanced Information Systems Engineering,* Proc., 3rd Int. Conf. CAiSE '91, Trondheim, Norway, May 13–15, 1991. Berlin: Springer-Verlag, 1991

their *interactions*. While *functional requirements* state the functionality of the components of the intended application domain, *non-functional requirements* (hereafter, *NFRs*) such as accuracy, security, user-friendliness, cost, performance, and the like state global constraints on how the functionality is exhibited. The complexity and usefulness of large information systems, then, are determined partly by their functionality and partly by global constraints that are initially abstracted in their requirements specification.

Traditionally, little attention has been paid to NFRs [Roman85]. Even for those NFRs such as security for which a large body of significant results has been attained, the level of the attention has been at design and implementation, with the focus primarily on systems software. Even with the growing interest in developing higher-level models and design paradigms, current technology is still inadequate both *representationally* for expressing global constraints as formal NFRs and *methodologically* for utilizing them in generating designs from functional requirements, due to:

- *Lack of representational basis:* Virtually no work seems to be available for formally representing NFRs. Presently NFRs are invariably stated informally during requirements analysis, and are often contradictory[2];

- *Lack of methodology for utilization:* In current practice, programs are designed to meet NFRs in *ad hoc* ways, while functional requirements are independently mapped into designs. The symptoms of such practice seem to include: (i) the arbitrary invention and application of procedures is non-trivial and often results in inconsistent designs; (ii) the presence of a contradiction is hard to detect, and even when detected, no guidelines are available for resolving the conflict; and (iii) justifying the overall design is extremely hard as different parts of a design cannot easily be related to one another.

We propose both a representational and methodological framework for non-functional requirements, focusing on accuracy requirements. With the premise that accuracy is an *inherent semantic* attribute of information, we take a first step towards establishing a representational basis for accuracy. To guide the process of mapping functional requirements into designs and justify design decisions, we propose a goal-oriented methodology, which is to be implemented in a *mapping assistant*. In the methodology, accuracy requirements are treated as (potentially conflicting) goals, for which two types of methods are presented: goal decomposition methods for decomposing the goals in terms of affected design components, and goal satisficing methods for contributing (either positively or negatively) to goal satisfaction. Some NFRs are further investigated for their cooperation and conflicts. The investigation results in rules for method composition in order to assess the quality of overall design. By recommending the types of consultation needed with users in the intended application domain, the framework aids the designers in making decisions and later justifying them. Once a goal is fully decomposed (recursively) into layers of subgoals, they are mapped into a target design, via a *dependency-based* methodology that provides local guidance for mapping individual elements of functional requirements. The design environment, based on the dependency-based methodology, and its implementation, are

[2]The rule of thumb seems to be that initially the user expects the intended software system to do everything perfectly and cost nothing; it is the job of the systems analyst to not only formulate realistic functional and non-functional requirements but also educate the user.

detailed in [Vassiliou90], while the overall framework, its motivation, and an extensive example of a requirements specification and its design are given in [Chung91a]. In the present paper, we focus on NFRs — their representation and utilization — and their achievement by (composition of) methods which can cooperate or conflict, and present a detailed illustration of our goal-oriented methodology.

Systematic goal-oriented approaches, in which the types of goals are similar to ours, have been used in areas other than information system designs. In the area of machine translation, [DiMarco90] takes a goal-oriented approach to automatically evaluating the translation of one natural language (e.g., English) into another (e.g., French). The evaluation criteria are stylistics such as clarity and dynamicity, which are treated as goals. According to the (stylistic) grammar associated with each goal, each target sentence is categorized into one of three types, each reflecting the bias (or orientation) of the sentence (e.g., A sentence is dynamic, neutral, or static). In the area of computer architecture design, [Agüero87] treats each requirement, a combination of functional and non-functional requirements (e.g., "maximal concurrency"), as a goal to satisfy. The plausibility or satisfiability of the decomposed subgoals (or goal if terminal node) categorizes each goal into one of four types: validated, assumed, unknown, and refuted. A goal is *validated* if there is a significant evidence for and no evidence against its satisfiability, *assumed* if there is no evidence against its satisfiability, *refuted* if there is some evidence against its satisfiability, and *unknown* when initially introduced. Then, the decomposition of a goal into subgoals, devised by human designers when needed, is guided in the sense that a goal satisfaction of one type can be achieved in terms of its subgoals of the same type. The presence or absence of (counter-) evidence should be determined by human designers through simulation, experimentation, literature search, etc. An arbitrary decomposition is also allowed when certain justification is present.

Section 2 briefly describes an example used throughout the paper and the languages for requirements specification and design. Sections 3 and 4 respectively propose a representational basis and goal-oriented utilization methodology for accuracy. After Section 5 investigates cooperation and conflicts among some NFRs, Section 6 illustrates our framework in detail in terms of examples, which are also used in Section 7 in assessing the quality of overall design through the rules of method composition. At the end, we summarize our contributions, limitations, and future directions.

2 An Example and Languages

In order to illustrate the concepts in our framework, we informally describe an example, and review the languages adopted for requirements specification, *Telos* [Mylopoulos90][3], and design, the *Taxis Design Language* (*TDL*) [Borgida90].

The requirements specification of a hypothetical research expense management system consists of functional requirements — for the environment, members of the projects should participate in various meetings held in various countries and submit their expense summaries; for the embedded system, monthly expense reports should be generated for each member, meeting, and project; and for the interaction, expense summaries in the

[3]Telos evolved from RML, a requirements modelling language based on knowledge representation ideas [Greenspan84].

environment should be submitted via electronic mail and received by some system activity — and non-functional requirements — monthly expense reports must be accurate, cost of secretaries should be minimal, etc.

We present the minimum set of language features required to understand the rest of this paper; for full power and detail, consult the references cited above. However, it is important to stress that the framework presented depends *primarily on the ontological and structural features* which determine the source and target levels of the mapping, rather than the particular languages chosen. Telos is a knowledge representation language strongly tied to "world modelling", for which the language offers the notions of entities, relationships, time, activities, etc. The language also offers associated inference mechanisms so that questions can be answered and propertied proven about a given requirements specification within a formal setting. Telos support structuring mechanisms such as aggregation (`attr`), generalization (`isA`) and classification (`instanceOf`, or `in`) to facilitate the task of building (usually large) requirements specifications. A Telos knowledge base consists of *propositions*, each a 4-tuple $<source, label, target, time>$, which are atomic units used to represent an entity or an elementary binary relationship in the application domain. For example, the fact "John's age was 17 during 1990" might be represented by the proposition <John, age, 17, 1990>. Time intervals can be related to each other in a knowledge base through temporal relations such as *before*, *during*, and *overlaps* [Allen83]. The declaration below is a portion of prescribing the component of the system under development (not the environment or the interaction), namely a class for monthly project expense reports which is a subclass of (i.e., *specializes*) a more generalized class for monthly expense reports:

```
IndividualClass &MonProjExpRpt in SystemClass, EntityClass isA &MonExpRpt with
     necessary, single  {* attributes below should be non-null and singular *}
           mon: Month          {* the month expense reports are produced    *}
           proj: Project       {*                 for a particular project  *}
           exp: Money          {* the total amount of expense               *}
           budg: Budget        {* the budget of the project                 *}
           budgetLeft: Money   {* the amount of budget left for the project *}
      ...
end &MonProjExpRpt
```

More detailed specification of the example can be found in [Chung89].

TaxisDL supports the development of a conceptual information system design by structuring the data and transactions which constitute the system according to their intended meaning rather than their implementation. TDL offers a uniform semantic data model for describing data, functions (for side-effect free computation of data), transactions (for defining atomic database operations), and scripts (for modelling long-term processes). TDL also offers the notions of entities and relationships organized along structuring mechanisms like aggregation, generalization, and classification.

3 Representation of Accuracy Requirements[4]

Informal accuracy requirements are often ambiguous and inconsistent. For a formal specification, we may start with this initial definition: "*accuracy* as a requirement for an

[4]This section is adapted from [Chung91a].

information system reflects the faithfulness of the information maintained by the system with respect to the application domain". Unfortunately, this definition is not quite formal: what do "reflects" and "faithfulness" mean, and what is "information"? Below we clarify our definitions; in particular, our definition of "information" will allow the expression of *varying granularities* of accuracy requirements for information. Based on the premise that accuracy of a piece of information depends entirely on the way that information is manipulated within the system as well as its environment, a formal model for accuracy is developed [Chung91b]. Due to space limitations, however, the focus of this paper is how to express accuracy requirements.

What does it mean for information to be accurate? Firstly, we assume that accuracy is an *inherent* (*semantic*) attribute of information chunks (hereafter, *information items*), in the same sense that weight, density or volume are fundamental (physical) attributes of material chunks. If `x` is an information item (e.g., "Brian is a research scientist" or "Bill is John's manager"), $\mathcal{A}$[`x`] specifies the degree of confidence in `x`, i.e., confidence in the proposition that `x` faithfully describes (the denotational aspect of) the application domain. Even though this might be possible for narrow domains, $\mathcal{A}$ will not be treated here as a probabilistic, fuzzy or even quantitative measure. Instead, we attempt to develop a *qualitative* model which leads to both decomposition rules for accuracy goals and design methods that enhance accuracy goals.

In the context of our mapping framework, information items maintained by the system to be built have a definite structure which can be used to clarify the nature of the accuracy measure:

- $\mathcal{A}$ [`instanceOf (e, C, t)`], measures the confidence that the (application domain) entity represented by token `e` has the property represented by class `C` during time interval (represented by) `t`;
- $\mathcal{A}$ [`attr (e, v, t)`] measures the confidence that the entity represented by `e` has attribute `attr` with value (the entity represented by) `v` during time interval `t`;
- $\mathcal{A}$ [`e`] measures the confidence in the assertion that there exists one and only one entity represented by `e` and there is no other `e'` which represents the same entity.

These definitions allow the formalization of accuracy for particular individual entities, their attributes, and properties.

$\mathcal{A}$ then is a function, $\mathcal{A} : \mathrm{I} \rightarrow \mathrm{D}$, where

$I = E \cup At \cup In$ such that E, At, and In respectively denote the set of token entities, token attributes, and instance relationships between tokens and entity classes represented in the information system, and

D is an partially ordered set with minimum element ι and maximum element α, representing respectively total lack of, and complete confidence, in the accuracy of an information item. The symbol " $\geq_a$" represents a partial order relation among the elements.

A single predicate is next introduced to simplify the representation of accuracy requirements. Let $\mathtt{X} \subset I$:

$$Accurate_d[\mathtt{X}] \stackrel{\text{def}}{=} \bigwedge_{\mathtt{x} \in \mathtt{X}} \mathcal{A}\,[\mathtt{x}] \geq_a d$$

for some threshold value d, which will generally be ignored in the rest of the discussion. *Accurate* is the basic predicate used to express accuracy requirements, such as "Insurance policies with a liability of more than \$10 000 000 must be accurate" (referring to all attributes and properties of policies):

> *Accurate*[{x, attr(x, v, t), instanceOf(x, C, t') for all attr, v, C, t, t' | x such that instanceOf(x, Policies, q), liability(x, y, q'), for some time interval q and q', and y > \$10 000 000}]

To simplify the notation, we introduce set-theoretic functions, *extension*, *attributes* and *properties*:

$$\begin{aligned}
\mathit{extension}(\mathtt{C},\mathtt{t}) &= \{\mathtt{instanceOf}(\mathtt{x},\mathtt{C},\mathtt{t}') \mid \mathtt{t}'\,\mathtt{during}\,\mathtt{t}\}\\
\mathit{attributes}(\mathtt{e},\mathtt{t}) &= \{\mathtt{attr}(\mathtt{e},\mathtt{v},\mathtt{t}') \mid \mathtt{t}'\,\mathtt{during}\,\mathtt{t}\}\\
\mathit{properties}(\mathtt{e},\mathtt{t}) &= \{\mathtt{instanceOf}(\mathtt{e},\mathtt{C},\mathtt{t}') \mid \mathtt{t}'\,\mathtt{during}\,\mathtt{t}\}
\end{aligned}$$

Another function can be obtained by using a set valued function, ^. For instance, C^attr yields the class which is the value of the definition for the attribute attr of the class C. Then,

$$\mathit{extension}(\mathtt{C\hat{}attr},\mathtt{t}) = \{\mathtt{attr}(\mathtt{e},\mathtt{v},\mathtt{t}') \mid \mathtt{instanceOf}(\mathtt{e},\mathtt{C},\mathtt{t}'')\ \mathit{and}\ \mathtt{t}'\,\mathtt{during}\,\mathtt{t}\ \mathit{and}\ \mathtt{t}''\,\mathtt{during}\,\mathtt{t}\}$$

which enables us to introduce other useful notations such as:

$$\mathit{extension}(\mathtt{C}\hat{}\,\mathit{attributes},\mathtt{t}) = \{\mathit{attributes}(\mathtt{e},\mathtt{t}) \mid \mathtt{instanceOf}(\mathtt{e},\mathtt{C},\mathtt{t}')\ \mathit{and}\ \mathtt{t}'\,\mathtt{during}\,\mathtt{t}\}$$

With these functions, the insurance policies example can be expressed as

> *Accurate*[{x, *properties* (e, t), *attributes*(e, t) | x $\in$ *extension* (Policies, AllTime) and liability(x, y, t') for some t' and y > \$10 000 000 and t' during t}]

We can apply the *Accurate* predicate to compose more complex concepts:

> *Accurate*[C] $\equiv$ *Accurate*[*extension*(C, AllTime)]
>
> *Accurate*[C^attr] $\equiv$ *Accurate*[*extension*(C^attr, AllTime)]
>
> *Accurate*[C^*attributes*] $\equiv$ *Accurate*[*extension*(C^*attributes*, AllTime)]
>
> *Accurate*[S] $\equiv$ *Accurate*[*extension*(S, AllTime)]
>
> *Accurate*[*properties* (e, t)] $\equiv$ *Accurate*[{instanceOf(e, C, t')| t' during t}]

More sophisticated notions of accuracy can be introduced by making additional assumptions about the accuracy measure, e.g., that $\mathcal{A}$ is a fuzzy function, or by introducing several accuracy predicates, such as *AbsolutelyAccurate, HighlyAccurate, MarginallyAccurate*, etc., which use different threshold intervals within D.

Note that accuracy is treated extensionally rather than intensionally here. In other words, an accuracy requirement for, say, most-prestigious- insurance policies is interpreted as a requirement on the particular policies maintained by the information system, not the description of the concept of mostPrestigiousPolicies included in the system specification[5].

[5]An intensional treatment of accuracy would attempt to measure the accuracy of generic information items such as &MonProjExpRpt, which may be accurate or inaccurate depending on its declared (generic) attributes, constraints, etc. This notion of accuracy seems appropriate when one attempts to measure the faithfulness of the *world model* (or its validation) — that is part of a functional requirement — to the application domain.

4 Goal-Oriented Methodology

In essence, non-functional requirements are treated as (potentially contradictory) *goals* that guide the refinements in the mapping process and justify particular design decisions. To link design decisions to global goals, we need *methods* for goal decomposition and others for "achieving" goals through particular design decisions. Actually, a particular design decision rarely "achieves" a non-functional requirement. More often than not, design decisions *contribute* positively or negatively towards a particular non-functional requirement and for the rest of the discussion we will speak of goal *satisficing*. The term "satisficing" was introduced by Herbert Simon [Simon69] to refer to sub-optimal solutions to well defined but computationally intractable problems, such as the travelling salesman problem. Etzioni [Etzioni89] also uses this term for a clearly formulatable problem and its heuristic solution. We use the term here in a broader sense since there is no formal definition of when a software system satisfies a set of non-functional requirements, nor an obvious measure of solution optimality. The intention is to suggest that generated software is expected to fulfill within acceptable limits, rather than absolutely, non-functional requirements.[6] The job of the mapping assistant is then (i) to "know" the general relationship, positive or negative, between goals and design decisions, (ii) to know decomposition and satisficing methods for different types of goals, (iii) to keep track of the goal expansion from the initial goals to recursive decompositions into subgoals, and (iv) to maintain a record of how goals/subgoals are affected by individual design decisions.

Treating an accuracy requirement as a goal to be satisficed leads to a class of systematic refinements whose structure forms a *goal tree*. Each node in the tree represents a goal, with the root node representing the initial accuracy goal. Moreover, each node/goal has an associated set of successor nodes/subgoals which either conjunctively or disjunctively satisfice the goal associated with the parent node. Thus goal trees are *AND* (*OR*) trees [Nilsson71] with goal satisfaction replaced by the more flexible notion of goal satisficing. Within such a framework, *goal refinement*, corresponds to the conjunctive or disjunctive decomposition of a goal into a set of successor goals. One way to obtain a refinement is by *instantiating* a goal decomposition method, leading to a conjunctive decomposition; on occasion, several *competing* methods may be available for a goal, where each method can either decompose or satisfice the goal. Such situations will lead into disjunctive goals.

Functional requirements typically have *localized* effects on target designs. In contrast, non-functional requirements have a *global* nature: satisficing one affects a multitude of design components. In addition, satisficing a goal may also be affected by designer decisions. Without guidelines, *ad hoc* actions taken to satisfice a given goal may result in an incorrect design. Then, the role of goal decomposition methods (hereafter GDMs) is to capture (ideally) all and only those affected components in the functional requirements. Additionally, GDMs offer elaboration guidelines to designers. Below we list some GDMs:

- *subclass* method: In order to prove $Accurate[\mathtt{C}\char94 attributes]$ (i.e., $Accurate[\mathtt{C}\char94\mathtt{attr}]$), where C is a class with specializations $\mathtt{C}_1, \mathtt{C}_2, ..., \mathtt{C}_n$ such that $extension[\mathtt{C},\ \mathtt{t}] = \bigcup_{i=1}^{n} extension[\mathtt{C}_i, \mathtt{t}]$, prove $\bigwedge_{i=1}^{n} Accurate[\mathtt{C}_i\char94 attributes]$ (i.e., $Accurate[\mathtt{C}_i\char94\mathtt{attr}]$).

- *derivedInfo* method: In order to prove $Accurate[\mathtt{x}]$, where x is derived from $\mathtt{y}_1, \mathtt{y}_2, ..., \mathtt{y}_n$

[6]Similarly, due to ill-defined nature of exception handling procedures in office systems, Strong [Strong89] seeks a *satisfactory* exception handling procedure instead of an *optimal* one.

through a function, f, prove $\bigwedge_{i=1}^{n}$ *Accurate*[$\mathtt{y}_i$] $\wedge$ *CorrectDerivFn*[f, x] where *CorrectDerivFn* means a correct design for f.

- *subset* method: In order to prove *Accurate*[S], where S is a set with subsets $\mathtt{S}_1$, $\mathtt{S}_2$, ..., $\mathtt{S}_n$ such that $\mathtt{S} = \bigcup_{i=1}^{n} \mathtt{S}_i$, prove $\bigwedge_{i=1}^{n}$ *Accurate*[$\mathtt{S}_i$].

- *superset* method: In order to prove *Accurate*[S'], where $\mathtt{S}' \subset \mathtt{S}$, prove a stronger proposition, *Accurate*[S].

- *attributeSelection* method: For *Accurate* [$\mathtt{e.a}_1.\mathtt{a}_2 \ldots .\mathtt{a}_n$], $n \in Integer$, prove *Accurate* [$\mathtt{e.a}_1.\mathtt{a}_2 \ldots .\mathtt{a}_i$] for all $1 \leq i \leq n$.

- *propertyTransition* method: For *Accurate*[*properties*(e, t)], prove *accurateCreation* (e) $\wedge$ *accurateTransfer* (e) $\wedge$ *accurateRemoval* (e).

Goal satisficing methods (hereafter GSMs) are methods that affect the level of our confidence in the accuracy of information items. Where do such GSMs come from? We seek methods, which (ideally) have empirically been proven effective, by observing and clarifying industry practices (e.g., [Martin73] offers a glossary of methods) and then exploring new methods by examining their combinations. Depending on the direction of shift of confidence they induce, GSMs are categorized into *positive* or *negative* (*antagonistic*) GSMs. Positive GSMs (hereafter +GSMs) either detect the inaccuracy and prevent it from permeating the system (like a filter) or upgrade the components involved in information transmissions (like a purifier before the filter). Negative GSMs (hereafter –GSMs), on the other hand, adversely affect +GSMs with the tendency to oppose +GSMs, although they do not make complete blockage. Firstly, we present some +GSMs:

- *confirmation*: the informant double-checks the information item previously submitted.

- *verification*: the *verifier* makes dual entry of the information item resident in some recording medium into the system (e.g., IBM key-entry operation). For a verification to be applicable, the verifier should be available and reliable (*Available&Reliable*), and the system should enquire (*enquire*) and receive (*receive*) the information item (i) from the verifier, say v:

$$verification[\mathtt{i}] \;\leftarrow\; Available\&Reliable[\mathtt{v}] \wedge enquire(\mathtt{v}, \mathtt{i}) \wedge receive(\mathtt{v}, \mathtt{i})$$

  Although not shown here, similar requirements apply to other methods;

- *authorization*: the *authorizer* grants the receiver of certain information items to transmit it to the system (e.g., certain bank authorization processes);

- *certification*: the *certifier*, regarded as more reliable than the sender, assumes certain responsibilities for the future. Certain justifications are associated with the information item, which may be continually reviewed by the certifier (e.g., a letter of credit issued by a bank).

- *validation*: the *validator*, performs comprehensive checking with the application domain to ensure that the information item meets certain predetermined standards (e.g., direct contact with the information source, experimentation, consultation with experts, inspecting transmissions to unveil the presence of some faulty substance);

- *support-attachment*: To promote or defend the information item, certain evidential information is accepted either directly by the system or indirectly through some external agent (e.g., ascertaining a student's address at the registration office by comparison with the student's driver licence, or identification process with a credit card and driver's license for accepting personal cheque);

- *consistency-checking*[7]: To prevent frequently-occurring errors, the system enforces certain integrity constraints (e.g., check-sums incorporated into ISBNs); For consistency-checking to be effective for an information item (`i`), its formulae and parameters should respectively be correct (*CorrectFunction*) and accurate (*AccurateParameters*):

$$consistency\!-\!checking(\mathtt{i}) \;\leftarrow\; CorrectFunction[f, \mathtt{i}] \wedge AccurateParameters[\mathtt{i}]$$

- *cross-examination*: a list of referential information items such as past history or statistics is displayed for assisting an external *examiner* to investigate the exceptionality of the information item;

- *auditing*: An *accuracy auditor* goes through suspicious information items[8];

- *exception handling*: A mutual cooperation of the system and external agents traces the sources of inaccuracies and recovers from inaccuracies, while taking preventive measures against any further propagation or recurrence of similar inaccuracies; and

- *better information flow*: Either during the design process or at run time, enhancement is made for more reliable information transmissions in terms of senders, receivers, communication channels, etc.

In economic theory, an action is said to have an *opportunity cost* if the utilization of a resource such as time or money for the action prevents performing another action [Samuelson76]. The concept of opportunity cost applies to some +GSMs: the use of one +GSM may prohibit that of other +GSMs (i.e., mutually *opportunistic GSMs*). For instance, if the project consultant certifies expenses, he or she may not be able to verify meeting participants due to his limited availability — either the consultant's total amount of time available is not sufficient for both or the two GSMs need to be employed concurrently. Current example illustrates that conflicts may be introduced even among +GSMs of the same type of NFRs when different types of NFRs come into consideration (cost consideration in the example).

Next, we present a couple of –GSMs:

- *frequent-modification:* Frequent updates, insertions and deletions induce inaccuracy of information items.

- *waived-justification:* Modification requests are allowed without any needed justification.

More examples and detailed explanation follow in the next section.

[7]Melli, *in private communication 1990*, reports that employing an user-interface that does a variety of validating and constraints checking alone reduced error listing for a particular application from forty five pages to three quarters of a page.

[8][Svanks81] reports that internal auditing techniques enabled the discovery of several welfare "claimants" who were in fact deceased.

5 Correlating Non-Functional Requirements

How do negative GSMs for one type of NFR (say, accuracy) become positive for another type of NFR (say, security), and *vice versa*? In this section, we show how one can adopt the approach taken for accuracy to dealing with other NFRs including security, cost, and user-friendliness. After considering them individually, we discuss how NFRs are related to each other.

Security By *security*, we mean the protection of information items: disallowing unauthorized access, while allowing authorized access. Traditional security concern on databases was centered around primarily computer systems — only those persons in direct contact with the information system, primitive low-level database operations, and data within the information system. For efficient and effective implementation of security policies, data structures (such as partial orders) and algorithms were introduced to better organize the accessors in relation to their permitted database operations and data [Denning79] [Thomson88]. With the emergence of object-oriented paradigms, richer models and their enforcement techniques [Rabitti88] [Lunt89] have started to appear. The type of persons, operations, and data is no longer homogeneous but finer-grained in terms of such abstraction mechanisms as classification, generalization, and aggregation that capture more semantics of the application domain. Message passing is prominent, thus diversifying the primitive database operations.

Our treatment of security is based on the premise that secure information items demand not only the *internal security* (protecting the information item maintained by the information system) but also the *external security* (protecting physical resources such as recording devices and communication channels related to the information items in concern). The current practice of separately dealing with internal and external security measures makes it hard to relate internal and external security actions, and is less effective. In other words, security requirements should *no longer* be specified only in terms of computer systems and those agents directly in communication with the system.

Adopting concepts from [Hartson76], we characterize the protection required for an information item in terms of an *authorizer*, *information item*, *agents*, and a *condition*.[9] An authorizer, an agent, specifies what agents are allowed to access what information items under what condition. Depending on the nature of conditions, an information access may be *alwaysDisallowed* (the condition is always false and the access is always prohibited), *conditionallyAllowed* (the access is permitted if the condition predicate evaluates to true at the time of access), or *alwaysAllowed* (the condition is always true and the access is always permitted). Authorizers will be omitted in the rest of the paper, as they are not needed in the discussion. As with accuracy, a function $\mathcal{S}$ may be introduced which indicates the degree of confidence in the protection of an information item, `i`, protected against disallowed agents where `i` may be `instanceOf(e, C, t)`, `attr(e, v, t)`, etc., such that

$$Secure_{d'}[\mathtt{i}, \mathtt{agent}, \mathtt{condition}] \stackrel{\text{def}}{=} \mathcal{S}[\mathtt{i}, \mathtt{agent}, \mathtt{condition}] \geq_s d'$$

for some threshold value *d'*. For instance, the requirement for always disallowing junior

[9] For a more comprehensive security enforcement, the notion of information flow network has to come into the picture where the protection of information items would be expressed in terms of allowing unit transmissions between two agents. See [Chung89] for more on this.

clerks access to information items in the reimbursement cheque can be stated as:

$$Secure[\texttt{Reimbursement\^{}att}, \texttt{junior-clerk}, \text{alwaysDisallowed}]$$

As with accuracy, goal decomposition methods (GDMs) need to be introduced in order to guide refinements during the mapping process. They include:

- *subclass* method: For *Secure*[`C`, `agent`, `condition`], prove *Secure*[`C`,`agent`, `condition`] $\wedge$ *Secure* [$\texttt{C}_k$, `agent`, `condition`] where $k \in Integer$ and $\texttt{C}_k$ isA `C`;
- *derivedInfo* method: For *Secure* [`i`, `agent`, `condition`] where `i` is derived from $\texttt{i}_1 \ldots \texttt{i}_n$, $n \in Integer$, prove *Secure* [`i`, `agent`, `condition`] $\wedge$ *Secure* [$\texttt{i}_k$, `agent`, `condition`] for $1 \leq k \leq n$.
- *composite* method: For *Secure* [`i`, `agent`, `condition`], prove *SystemSecure* [`i`, `agent`, `condition`] (internal security) $\wedge$ *PhysicallySecure* [`i`, `agent`, `condition`] (external security).

For instance, *Secure* [`Reimbursement^att`, `junior-clerk`, alwaysDisallowed] can, via the *derivedInfo* method, be decomposed into (assuming a reimbursement cheque is directly obtained from its corresponding expense):

$$\begin{aligned} &Secure[\texttt{Reimbursement\^{}att}, \texttt{junior-clerk}, alwaysDisallowed] \leftarrow \\ &\qquad Secure[\texttt{Expense\^{}att}, \texttt{junior-clerk}, alwaysDisallowed] \end{aligned}$$

Security GSMs include *authorized-identification* (the identification requires the presence of external agent to avoid false identification), *periodic-reidentification* (for lengthy sessions, the identity of the agent using the system is periodically re-examined), *security-auditing* (a security auditor carries out a comprehensive examination of information accesses to detect security breaches), *soft-alarm* (access to a sensitive information item is reported to a higher authority via computer network), *limiting-access-time* (to reduce the potential for theft, access time is limited), *external-security-measure* (setting up physical barriers and objects such as alarms and locks), *disguise* (encrypting items, or intentionally inserting inaccuracies into information items), etc.

Cost Although a full treatment of cost would require a comprehensive study of related fields such as cost accounting, linear analysis, etc., at least three components seem to be involved in dealing with cost requirement: *operation*, *equipment*, and *system* cost. Operation cost involves the cost related to employees such as training cost, salaries, expenses, etc. Operation cost may be further refined according to either job positions such as secretarial, junior, senior, and managerial or job categories such as temporary or permanent. Equipment cost stands for the cost of articles or physical resources such as desk, phone, etc. System cost stands for the cost related to the usage of the information system such as hardware cost for disk, main memory, etc., or software cost for maintaining old expenses, allowing frequent queries, generating reports, etc. Assuming that other components are insignificant, formalization of cost may start with:

$$cost = operation\ cost + equipment\ cost + system\ cost.$$

The effort to minimize one cost component may adversely affect the overall cost by increasing other cost component. For instance, using cheap communication tools for minimizing equipment cost increases the likelihood of errors and subsequent retransmissions which increase the operation cost. Conversely, increasing one cost component may significantly decrease other cost component. For instance, although the system's maintaining old expenses increases the system cost, it may significantly decrease the operational cost involved in retrieving old expenses whose voluminous nature may require large amount of time and effort by human agents. Besides the component, cost needs to be also considered along the temporal avenue: short-term and long-term. An attempt to minimize the short-term cost (e.g., by hiring an employee without adequate training, or inexperienced temporary clerks) may result in later use of heavy auditing to recover from inaccuracies and consequently increase the long-term cost. Thus, the effort to minimize short-term cost may adversely affect long-term cost, and *vice versa*. As more breakdown is performed for the cost components, more realistic forecast of cost behavior would be achieved.

For stating a cost requirement, the most general requirement may be stated as *MinimalCost* [`entity`][10] where `entity` denotes some general concept (*not* only information item) such as company, department, or even some collective operation (e.g., `Project`). For GDMs, we would use methods similar to accuracy GDMs such as *derived* method, *subset* method, etc. For instance, a decomposition of *MinimalCost*[`e`] by the derived method would be:

$$MinimalCost[\mathtt{e}] \leftarrow MinimalOperationCost[\mathtt{e}] \wedge MinimalOperationCost[\mathtt{e}] \wedge MinimalEquipmentCost[\mathtt{e}]$$

In turn, each subgoal may be further decomposed. Also, more specialized cost requirements may be stated and decomposed. For instance, *MinimalOperationCost*[`secretary`] may be stated and, by the subset method, further decomposed into:

$$\begin{aligned} MinimalOperationCost[\mathtt{secretary}] \quad \leftarrow \quad & MinimalOperationCost[\mathtt{member{-}secretary}] \ \wedge \\ & MinimalOperationCost[\mathtt{cmo{-}secretary}] \end{aligned}$$

where secretaries are assumed to work either for members (`member-secretary`) or for the central management office (`cmo-secretary`).

Cost GSMs include *limited-training* (to reduce operating cost, an insufficient training is offered to selected employees), *limited-employment* (to reduce operating cost, insufficient number of employees are hired), *casual-modification* (to reduce the short-term cost, the number of employees servicing customers is decreased at the expense of increased system cost, and direct access to the system is allowed to non-employees), *sole-channel-communication* (to reduce equipment cost, only one communication line is available), *limiting-equipment-usage* (to reduce equipment cost, the use of papers, phones, etc., is limited), *infrequent-maintenance* (to reduce system cost, information system maintenance is performed infrequently), etc.

User-friendliness Although there is no general agreement about what constitutes user-friendliness, there seems to be some criteria for user-friendliness that a large number of people tend to agree with. Such criteria, among other things, usually include:[11] *promptness* — no undue delay in accepting information items and responding to requests (which

[10]From management perspective, [Lee90] considers the goal of minimizing some system cost, expressed as *MinimizeCost*.

[11]For instance, [Nielsen90] describes "nine usability heuristics" for user interface.

would be related to system load and performance that may necessitate the use of multiple processors, faster disk drivers with higher capacity, etc.), *tolerance* against ill-behavior — no hang-ups against errors, delays, unexpected behavior, etc., *guidance* (or *responsiveness*) — providing guidance for correcting errors, generating reminders to cope with human forgetfulness, etc., *coherence* — exhibiting a sequence of system responses closely related to one another with respect to some principle or criteria[12], etc. User-friendliness of a system seems to depend strongly upon the type of the users: one type of human computer interaction may be favorable to one group of people and yet unfavorable to another. Nonetheless, promptness, tolerance, guidance, coherence, etc. are expected to play dominant role for user-friendliness. Although the exact nature of such concepts awaits further research, deriving some GSMs is not defeasible.

User-friendliness GSMs include *prompt-modification* (for promptness, modification request is immediately met without any external assistance), *prompt-identification* (for promptness, the user identification procedure to access the system is brief), *frequent-modification* (for tolerance, frequent corrections are allowed to cope with human errors and carelessness), *waived-justification* (for tolerance, the absence of proper justifications is excused), *generating-reminders* (for guidance against human forgetfulness), *constructing-user-model* (for coherence, a user-model is constructed to facilitate human-computer interaction).

NFRs are correlated to one another via their GSMs either *positively* supplying one another's lack or *negatively* inducing conflicts. In the absence of their formalizations or any concrete work, the discussion below is intended to serve as illustrations for *approximate* relationships among the NFRs above.

Accuracy & Security: Enhancing accuracy either contributes cooperatively to detecting (security-auditing) security violations or acts incompatibly, in case of intrusion, by inducing the right information items to be revealed (i.e., against the disguise method). Enhancing security, also, either contributes cooperatively to reducing the chance of false information items' permeating the system or acts incompatibly by limiting the number of available agents, the type of information items (against cross-examination), or the time (limiting-access-time) needed for filtering accurate information items.

Accuracy & Cost: Accuracy GSMs act incompatibly with cost as they require more time and higher reliability of employees, the system, and equipment. However, they contribute to cost goals by reducing the chance of potentially prohibitive cost for later recovery from inaccuracy. Cost GSMs, in general, act negatively for accuracy as they limit the use of employees, the system, and equipment.

Accuracy & User-friendliness: Accuracy GSMs would be seen favorably either by people such as accuracy auditor and managers, who are responsible for maintaining accurate information items or the owners of the information items. However, accuracy GSMs either involving external agents or requiring justifications may be seen by other parties as cumbersome and the sign of user distrust, thus acting against user-friendliness. Since a coherent, responsive, and tolerating human-computer interaction is believed to make agents less error-prone (assuming errors tolerated will be corrected by the users), user-friendliness GSMs contribute to accuracy goals. On the other hand, they may also act negatively for accuracy goals, as system tolerance may induce human carelessness

[12] Coherence may be enhanced by constructing user-models.

NFR / GSM	*Accuracy*	*Security*	*Cost*	*User-friendliness*
for Accuracy:				
confirmation	+		−	
validation	+		−	
certification	+		−	
verification	+		−	
cross-examination	+		−	
better-information-flow	+		−	
for Security:				
periodic-reidentification	+	+	−	
security-auditing	+	+	−	
soft-alarm	+	+	−	
limiting-access-time		+		
external-security-measure		+	−	
disguise	−	+	−	
for Cost:				
limited-training	−	−	+	−
limited-employment	−	−	+	−
casual-modification	−	−	+	
sole-channel-communication	−	−	+	−
limiting-equipment-usage	−	−	+	−
infrequent-maintenance	−	−	+	−
for User-friendliness:				
prompt-modification	−		−	+
prompt-identification		−	−	+
frequent-modification	−		−	+
waived-justification	−		−	+
generating-reminders	+	−	−	+
constructing-user-model	+		−	+

Table 1. Correlation of NFRs with GSMs.

and consequently errors or omissions of justifications which may not be corrected.

Correlations between security and cost, security and user-friendliness, and cost and user-friendliness are described in [Chung91b]. Note that the discussion above dealt primarily with direct relationships between two NFRs rather than indirect ones (E.g., user-friendliness may contribute to cost goals by inducing more accurate information items which reduce the chance of later recovery attempts). However, the discussion has laid the ground to explore various indirect relationships. Table 1 reflects our discussion above. The value of each entry should be understood in accordance with the discussion.

6 Illustration

In developing a goal tree for an NFR, decomposition and satisficing methods aid the designer in respectively capturing those affected design components and satisficing the

Figure 1: The Goal Tree with GDMs for Accurate Monthly Expense Reports.

decomposed goal. In applying a satisficing method, the designer needs to consider not only the goal expressed but other NFRs that may induce conflicts. In order to foresee and resolve potential conflicts, our mapping assistant offers guidelines in terms of the types of consultation needed with the users in the application domain.

Consider again the example of research expense management system of Section 2 and assume that $Accurate$[&`MonExpRpt`^$attributes$] is the root node of the goal tree representing an accuracy requirement, "all the attributes of &MonExpRpt should be accurate" (In what follows, *attributes* is abbreviated by `att`). The root goal can then be refined using the *subclass* method mentioned in Section 4:

$$Accurate[\&\texttt{MonExpRpt}^\wedge\texttt{att}] \leftarrow Accurate[\texttt{MonProjExpRpt}^\wedge\texttt{att}] \wedge Accurate[\texttt{MonMtgExpRpt}^\wedge\texttt{att}] \wedge Accurate[\texttt{MonMbrExpRpt}^\wedge\texttt{att}]$$

The result of this decomposition is that $Accurate$[&`MonExpRpt^att`] has been decomposed into three subgoals of $Accurate$[`MonProjExpRpt^att`], $Accurate$[`MonMtgExpRpt^att`] and $Accurate$[`MonMbrExpRpt^att`]. Now each of these subgoals needs to be satisficed. For instance, satisficing the subgoal of $Accurate$[`MonProjExpRpt^att`] requires that all attributes of `MonProjExpRpt` be accurate. At this point, the designer indicates that each instance `monProjExpRpt in` the class `MonProjExpRpt` (from here on, assume that the argument *Accurate* is universally quantified) has five attributes: month (`mon`), project (`proj`), total monthly expense (`exp`), project budget (`budg`) and remaining project budget (`budgetLeft`). Then, the goal of $Accurate$[`monProjExpRpt^att`] is decomposed into:

$$Accurate[\texttt{monProjExpRpt}^\wedge\texttt{att}] \leftarrow Accurate[\texttt{monProjExpRpt.mon}] \wedge Accurate[\texttt{monProjExpRpt.proj}] \wedge Accurate[\texttt{monProjExpRpt.exp}] \wedge Accurate[\texttt{monProjExpRpt.budg}] \wedge Accurate[\texttt{monProjExpRpt.budgetLeft}]$$

Note that the designer, at this point, may choose either to map `monProjExpRpt` into TDL classes through a dependency-based mapping or to continue to make further refinements in order to satisfice the initially posted goal.

While trying to satisfice the subgoal of $Accurate$[`monProjExpRpt.exp`], the designer may choose to indicate that `monProjExpRpt.exp` is *derived* information,

$$Accurate[\texttt{monProjExpRpt.exp}] \leftarrow CorrectDerivFn[\texttt{f}, \texttt{monProjExpRpt.exp}] \wedge AccurateParameters[\texttt{f}]$$

where the formula for the derivation function, `f`, is

$$\texttt{monProjExpRpt.exp} \leftarrow \texttt{Sum}(\{\texttt{exp in Expense} \mid \texttt{exp.date during monProjExpRpt.mon and exp.proj} = \texttt{monProjExpRpt.proj}\}.\texttt{ECUamount})$$

It is then possible to instantiate the *derivedInfo* decomposition method:

$$AccurateParameters[\texttt{f}] \leftarrow Accurate[\{\texttt{exp} \ldots \mid \ldots\}.\texttt{ECUamount}] \wedge Accurate[\{\texttt{exp in Expense} \mid \ldots\}] \wedge Accurate[\texttt{exp.date}] \wedge Accurate[\texttt{exp.proj}]$$

The designer may now choose either to map (via *CorrectDesign*) the function, `f`, of *CorrectDerivFn* into a function, transaction or script, according to the *dependency-based mapping methodology* described in Section 1, or to continue to make further goal refinements dealing with activation conditions, termination conditions, etc. Figure 1 shows the results of a dependency-based mapping (mentioned in Section 1) for the derivation function, `f`. Firstly, `f` is mapped into a TDL function which is called by the newly introduced transaction `addMonProjExp` that computes `monProjExpRpt.exp` (as well as the `monProjExpRpt.budgetLeft`) for a `monProjExpRpt` and inserts it (with appropriate `mon`, `proj`, and `budg` values) into `MonProjExpRpt`. Note that, although not shown here, `addMonProjExp` may have pre- and/or postconditions thereby improving the accuracy attribute of the derived information. Secondly, since monthly project expense summaries need to be produced at the end of each month, a script needs to be introduced which includes a single transition to be activated at the end of each month.

Two competing alternatives are foreseen by the designer for `exp.date`: the date the expense incurred may come from either the expense voucher's (by requiring the members to send their vouchers to the central management office), `exp.voucher.date`, or the meeting's in the expense summary (by requiring the secretary to directly include the meeting), `exp.mtg.when`:

$$\mathit{Accurate}[\texttt{exp.date}] \;\leftarrow\; \mathit{Accurate}[\texttt{exp.voucher.date}] \vee \mathit{Accurate}[\texttt{exp.mtg.when}]$$

The designer decides to firstly explore the first alternative, and applies the *attribute-Selection* method:

$$\mathit{Accurate}[\texttt{exp.voucher.date}] \;\leftarrow\; \mathit{Accurate}[\texttt{exp.voucher}] \wedge \mathit{Accurate}[\texttt{voucher.date}]$$

Now focus on the refinement of *Accurate*`[exp.voucher]`. Every information item in the information system is either received from an external agent or derived from what already resides within the system[13]. The designer indicates that no further decomposition is needed for `exp.voucher`. Instead, it should be received. Then, all the received information items for `exp.voucher`, *Rec* (`exp.voucher`), have to be accurate. Further, accuracy of received information items depend on accurate initial creation and correctness, *CorrectTransmission*, of subsequent transmissions from the creator to the receiver (now the information system). Unfortunately, ensuring that the creation and transmission of an information item is accurate is in many cases costly and impractical. Accordingly, the designer may choose to resign himself to using several satisficing methods for *Accurate*[*Rec* (`exp.voucher`)]. Clearly, using more methods leads to increased confidence that the information maintained by the system is indeed accurate. That is,

$$\mathit{Accurate}[\texttt{exp.voucher}] \;\leftarrow\; \mathit{confirmation}(\texttt{exp.voucher}) \vee \cdots \vee \mathit{auditing}(\texttt{exp.voucher})$$

The designer indicates that the validation method is applicable. Now suppose, as in Figure 2, that a security requirement was considered earlier: reimbursements should not be revealed to junior clerks, i.e., *Secure* [`Reimbursement^att`, `junior-clerk`, `alwaysDisallowed`]. Reimbursements are based upon expenses. According to the previous section, satisficing a security requirement implies satisficing both the information system security (internal security) and the physical security (external security):

$$\begin{aligned}\mathit{Secure}[\texttt{Expense\^{}att}, \texttt{junior-clerk}, \texttt{alwaysDisallowed}] \;\leftarrow\; & \\ & \mathit{PhysicallySecure}\,[\texttt{Voucher\^{}att}, \texttt{junior-clerk}, \texttt{alwaysDisallowed}] \;\wedge \\ & \mathit{SystemSecure}[\texttt{Voucher\^{}att}, \texttt{junior-clerk}, \texttt{alwaysDisallowed}]\end{aligned}$$

[13]The system clock may be viewed as an external agent.

Figure 2: The Failure of Satisficing Accurate Expense Voucher.

However, in using the validation method, the designer finds out that a junior-clerk should be used as a validator:

$$\begin{aligned} \mathit{validation}[\texttt{exp.voucher}] &\leftarrow \mathit{Available\&Reliable}[\texttt{validator}] \wedge \\ &\quad\ \mathit{enquire}(\texttt{validator}, \texttt{exp.voucher}) \wedge \mathit{receive}(\ldots) \\ \mathit{Available\&Reliable}[\texttt{validator}] &\leftarrow \mathit{Available}[\texttt{validator}] \wedge \mathit{Reliable}[\texttt{validator}] \\ \mathit{Available}[\texttt{validator}] &\leftarrow \mathit{Available}[\texttt{junior-clerk}] \end{aligned}$$

Since the security goal prohibits the access of any junior clerk to expense voucher, a junior cannot be used as a validator. However, since junior-clerks are the only candidate, no validator would be available, and hence no validation:

$$\neg\, \mathit{Available}[\texttt{junior-clerk}] \ \rightarrow \cdots \rightarrow\ \neg\, \mathit{validation}[\texttt{exp.voucher}]$$

Thus, the designer has encountered a conflict. He or she can either accept `exp.voucher` without any positive method employed or backtrack to where some alternative can be sought after. Note that in order for *Accurate* [`exp.voucher.date`], both *Accurate* [`exp.voucher`] and *Accurate* [`voucher.date`] have to be satisficed. Since *Accurate* [`exp.voucher`] cannot be satisficed, regardless of whether *Accurate* [`voucher.date`] can be satisficed, *Accurate* [`exp.voucher`] cannot be satisficed with the first selection. Thus, any attempt to further decompose *Accurate* [`exp.voucher`] would be futile.

Since the selection of `exp.voucher.date` has turned out to be inappropriate for *Accurate* [`exp.date`], the designer next considers an alternative decomposition for *Accurate* [`exp.mtg.when`] to which *attributeSelection* method is applied:

$$\mathit{Accurate}[\texttt{exp.mtg.when}] \ \leftarrow\ \mathit{Accurate}[\texttt{exp.mtg}] \wedge \mathit{Accurate}[\texttt{mtg.when}]$$

Figure 3: Alternative Selection for Expense Date.

For *Accurate* [exp.mtg], as shown in Figure 3, the designer has firstly chosen the confirmation method (intermediate steps are omitted) without encountering any conflict:

$$Accurate[\texttt{exp.mtg}] \;\leftarrow \cdots \leftarrow\; confirmation[\texttt{exp.mtg}]$$

While trying to employ another method, a verification method, for satisficing *Accurate* [exp.mtg], the designer encounters a conflict due to an earlier effort to satisfice a cost requirement. As seen in Figure 4, the requirement was to minimize the operation cost of secretaries, which was decomposed into:

$$\begin{aligned} MinimalOperationCost[\texttt{secretary}] \;\leftarrow\; & MinimalOperationCost[\texttt{member-secretary}] \;\wedge \\ & MinimalOperationCost[\texttt{cmo-secretary}] \end{aligned}$$

For minimizing the operation cost for the secretaries of members, it was decided that no training would be offered:

$$MinimalOperationCost[\texttt{member-secretary}] \;\leftarrow\; no{-}training(\texttt{member-secretary})$$

However, without proper training, member secretaries are deemed unreliable (i.e., ¬ *Reliable* [member-secretary]). Thus, due to the lack of reliable member secretaries, the verification method cannot be used for satisficing *Accurate* [exp.mtg]. That is,

$$\neg\; Reliable[\texttt{member-secretary}] \;\rightarrow \cdots \rightarrow\; \neg\; verification[\texttt{exp.mtg}]$$

Nonetheless, a third method, a certification method, introduces no further conflict (i.e., *succeeds*) where the manager of the accounting department in the central management office (cmo-acct-mgr) is assigned the role of the certifier. Thus, as seen in Figure 5, two methods are used for *Accurate* [exp.mtg]:

$$Accurate[\texttt{exp.mtg}] \;\leftarrow \cdots \leftarrow\; confirmation[\texttt{exp.mtg}] \;\vee\; certification[\texttt{exp.mtg}]$$

Figure 4: Temporarily Successful Methods for Accurate Meeting of Expense.

Figure 5: Opportunistic Method for Accurate Time of Meeting.

Note that the two methods above are *temporarily successful*: although no conflict has been encountered for their fully decomposed subgoals, some later attempt to satisfice other goals may necessitate retracting any of the two methods. In fact, the next example illustrates how the certification method gets retracted. Since the first subgoal of satisficing *Accurate* [`exp.mtg.when`] is successful, the designer now moves on to the second subgoal of *Accurate* [`mtg.when`]. The designer tries a certification method by assigning `cmo-acct-mgr` to the role of the certifier:

$$Accurate[\texttt{mtg.when}] \leftarrow \cdots \leftarrow certification[\texttt{mtg.when}] \leftarrow \cdots \leftarrow Available[\texttt{cmo-acct-mgr}]$$

However, the designer realizes that `cmo-acct-mgr` cannot do the certification of both `exp.mtg` and `mtg.when` due to his or her limited time. In other words, the concept of *opportunity cost* brings a conflict. In resolving the conflict, the designer gives a higher priority to *Accurate* [`mtg.when`] than *Accurate* [`exp.mtg`]. Thus, the use of `cmo-acct-mgr` as the certifier for `exp.mtg` is denied, and, hence, the certification method is no longer applicable for satisficing *Accurate* [`exp.mtg`].

The designer now tries a consistency-checking method for satisficing *Accurate* [`mtg.when`]:

$$\begin{aligned}
Accurate[\texttt{mtg.when}] &\leftarrow consistency{-}checking(\texttt{mtg.when}) \\
consistency{-}checking(\texttt{mtg.when}) &\leftarrow CorrectFunction[\texttt{f}, \texttt{mtg.when}] \wedge AccurateParameters[\texttt{f}] \\
AccurateParameters[\texttt{f}] &\leftarrow Accurate[\texttt{today}] \wedge Accurate[\texttt{mtg.proj.terminationDay}]
\end{aligned}$$

where the consistency checking function, `f`, is given as:

$$(\texttt{today before mtg.when}) \quad \textit{and} \quad (\texttt{mtg.when before mtg.proj.terminationDay})$$

Note that the consistency checking method is *partially successful*: although no conflict has been encountered for the subgoals, the subgoals (e.g., *Accurate* [`today`]) are not yet fully decomposed; in further decomposition, some conflict may arise.

The success (or lack thereof) of goal satisficing methods relies on the cooperation between the system and agents in the environment. Overall responsibility for accuracy is vested in the *accuracy auditor,* who contributes to the *procedure manual*[14], which is initially drafted during the design process. The manual indicates policies that the agents in the environment should obey in order to satisfice the subgoals selected. It affects the organizational structure and the interaction between the system and agents during usage of the system. For instance, if a *verification* method is selected, the procedure manual indicates that a member must transfer his expense information to the information system and to the central management office which will enter the same information into the information system. The effect of this procedure is to open communication channels between the member, the system, and the office. Especially when *few* methods are selected, the manual indicates the duties of agents in satisficing the non-selected ones.

7 Method Composition

In a goal tree expansion, the application of goal decomposition methods (GDMs) to a goal results in producing either conjunctive or disjunctive subgoals. For instance, in Fig-

[14] In acquiring formal requirements from possibly inconsistent or ambiguous informal descriptions, [Reubenstein90] recognizes the need for generating documents, which in spirit is similar to our procedure manual.

ure 1, the subclass method decomposed *Accurate* [`MonExpRpt^att`] into the conjunctive subgoals of *Accurate* [`MonMtgExpRpt^att`], *Accurate* [`MonMbrExpRpt^att`], and *Accurate* [`MonProjExpRpt^att`]), while the derivedInfo method decomposed *Accurate* [`exp.date`] into disjunctive subgoals of *Accurate* [`exp.voucher.date`] and *Accurate* [`exp.mtg.when`]. Also, many GSMs may be applied to a goal. Then, after some conflict-inducing GSMs are discarded, other GSMs remain under consideration as successful. For instance, after validation and certification methods are discarded, the successful confirmation method stays for *Accurate* [`exp.mtg`]. If only one method remains, its type would solely determine whether the goal is either positively or negatively satisficed. However, rarely would only one method remain. Instead, more than two methods, which are not mutually exclusive (thus not mutually contradictory) may be applied to a goal. For instance, in Figure 5, certification, consistency-checking, waived-justification, and frequent-modification methods were collectively applied to *Accurate* [`mtg.when`].

Can we determine, either individually in isolation or relative to alternative goal trees, *the strength or weakness of each goal tree* (or less meaningfully, each design as a whole)*?* Below we present a scheme that answers the question. With respect to a stated accuracy requirement, the initial (root) goal, without any GSM applied, starts as neutral, may be decomposed into subgoals, and changes its status as more GSMs are added. When the goal and all of its subgoals are (recursively) fully decomposed, a variety of GSMs would have been applied to the fully decomposed leaves. Our scheme consists of two steps, the first for combining GSMs and the second for subgoals of GDMs. When repeatedly applied to the nodes from the bottom leaves, going upwards, until the stated root goal is reached, our scheme determines the bias of the goal tree for or against the root goal:

Step 1 *Composition of GSMs*: According to the table[15] below, repeatedly combine GSMs, two at a time. The consequence is that a goal is *positively-biased* (+) if only +GSMs are applied, *negatively-biased* (–) if only –GSMs are applied, *neutral* (0) if no GSM is applied, or *unknown* (?) if at least one + and one – GSM are applied.

Step 2 *Composition of subgoals of GDM*: Determine the bias for each goal by taking the minimum and maximum element respectively of conjunctive and disjunctive subgoals where the minimum and maximum element is based on the order: $+ \geq_a 0 \geq_a -$.[16]

$subgoal_1$ / $subgoal_2$	–	0	+	?
–	–	–	?	?
0	–	0	+	?
+	?	+	+	?
?	?	?	?	?

Table 2. The Cumulative Effect of Two Subgoals.

[15]It is interesting to note that our table, originally motivated by the additive table in qualitative reasoning [AI84] is symbolically comparable to the error algebra for integers proposed in [Wetherell82] for dealing with the propagatory effect of negative, positive, and unknown underflow and overflow.

[16]When one of the subgoals is ?, the rule of conjunction is: $+ \wedge ? \rightarrow ?$; $0 \wedge ? \rightarrow ?$; and $- \wedge ? \rightarrow -$. The rule of disjunction is: $+ \vee ? \rightarrow +$; $0 \vee ? \rightarrow 0 \vee ?$; and $- \vee ? \rightarrow ?$. However, the composition of disjunctive subgoals is not very meaningful, as a final design can take only one of the disjuncts.

For instance, when certification (+), consistency-checking (+), waived-justification (–), and frequent-modification (–) methods are all applied to *Accurate* [`mtg.when`], as in Figure 5, *Accurate* [`mtg.when`], by the first step, becomes an unknown goal (assuming consistency-checking method does not induce any conflict). Thus, at least this example illustrates that the composition rules in the table correctly captures our intuition for combining GSMs: certification or consistency-checking may or may not significantly improve the accuracy of received `mtg.when`, when the received `mtg.when` is associated with waived-justification (even when the time is too close or far ahead) or frequent-modification (the quality of the secretary, channel, or medium involved in the transmission may be poor). Then, *Accurate* [`exp.mtg.when`] would also become an unknown goal as, by the second step, the composition of *Accurate* [`mtg.when`] (unknown) and *Accurate* [`exp.mtg`] (positively-biased) results in unknown. The process can be repeated until the root goal of *Accurate* [`MonExpRpt^att`] is reached.

The above rules basically categorize each and every goal (or subgoal) into four classes of positively-biased, neutral, negatively-biased, and *unknown* goals, and ultimately partition all design spaces into the four classes (although their utility may be uncertain). The rules in the table are concise at the expense of non-discernability within categories. In other words, the relative goodness of any two goals in the same class is not directly derivable from the rules *per se*. However, within each class of design, a partial order exists (e.g., a goal with two +GSMs and one –GSM may be better than one with one +GSM and one –GSM, although both belong to unknown class). [Chung91b] offers rules for determining the relative strength or weakness of some type of class members.

Note that the rules above, plus the correlation table in Section 5, would be the basis for predicting the effect of using a GSM during a design process. In the presence of many goals and their GDMs and GSMs, the rules, although somewhat crude, are often expected to be highly beneficial.

8 Conclusion

This paper proposes a novel framework for representing non-functional requirements (NFRs) and utilizing them in generating designs for large information systems. Accuracy has been taken as an example of an NFR. With the premise that accuracy is an inherent semantic attribute of information, this paper takes a first step towards establishing a representational basis for accuracy: (i) The notion of accuracy is relegated to a qualitative function whose evaluation depends on the environmental situation, which in turn determines the feasibility of using accuracy-related satisficing methods. This provides evidence for the thesis that *consultation* among agents in the environment and designers is needed to produce more accurate designs; (ii) By allowing the user to be indefinite about an NFR measure (here, a partial order for the accuracy measures) thus reflecting the difficulty involved in providing a definitive measure for an NFR, our scheme is flexible enough to be *applicable to several NFR types*; (iii) Close association of the accuracy predicate with a small number of basic modelling primitives enables the *full utilization* of the power of our requirements specification (or knowledge representation) language, hence *facilitating* the requirements-capturing process.

Treating accuracy requirements as potentially conflicting goals, our goal-oriented methodology offers: *methods for decomposing goals* in terms of affected design components, and *methods for satisficing goals* which contribute (positively or negatively) to goal satisfaction. Advantages of our methodology are: (i) The two types of methods guide the process of mapping functional requirements into designs *formally and systematically*, thus avoiding an inconsistent design; (ii) The table of correlations for some NFRs in terms of their satisficing methods acts as *the basis for detecting and resolving conflicts* during the design process; (iii) Then, the rules of composition in conjunction with the correlation table enables *predicting the effect of each design decision* and *assessing the quality of overall design*; (iv) By making recommendations for the designer in terms of the types of consultation needed with users in the intended application domain, the framework *aids* the designers in making decisions in each application of the methods and *allows the users to express the feasibility* of the decisions in the mapping process; (v) *Justifying the overall design* is straightforward since the relationships among different parts of a design can be easily compiled in terms of the two types of methods, and the responsibility (possibly recorded in the procedure manual) can be traced to those users who participated in the mapping process.

Long-term extensions to our work would include adding facilities to: (i) allow the users to express, when available and appropriate, approximate accuracy measurements (e.g., statistics-based measures), and (ii) fully distinguish two designs within the same class (i.e., positively- or negatively- biased, or unknown) with respect to their performance for accuracy. For the former, a much deeper theory would have to be developed to remedy the deficiencies caused by our partial and approximate formalization of accuracy. For the latter, a scheme is needed to marry quantitative and qualitative representations and their reasoning.

Future plans include the provision of a control structure for the mapping assistant, which would involve investigating schemes for relating decisions, positive or negative, made for different types of goals, and efficient backtracking when conflicts arise. Finally, we need a prototype implementation to refine our framework, serve as an experimentational instrument for discoverying improvement, and act as a nucleus on which to grow other components of information system design.

Acknowledgements

Foremost, sincere thanks are due to Professor John Mylopoulos for his initial recognition of the importance of the current work, encouragement, and guidance. Sol Greenspan maintained an ongoing interest and provided criticism, moral support, and insights. Gratitude is also due to Brian Nixon, Eric Yu, and members of the Taxis group at the University of Toronto for their input and interest in this work, where research on the design of information systems has been going on for almost fifteen years. Special thanks are extended to members participating in the DAIDA project at the University of Crete including Panagiotis Katalagarianos, Manolis Marakakis, Michalis Mertikas, and Yannis Vassiliou whose earlier collaboration provided impetus for the current work. Partial support from the Institute of Robotics and Intelligent Systems (IRIS) funded by the Networks of Excellence program of the Government of Canada is gratefully acknowledged.

Bibliography

[Agüero87] Ulises Agüero and Subrata Dasgupta, A Plausibility-Driven Approach to Computer Architecture Design, *CACM*, Pp. 922-932, Vol. 30, No. 11, Nov. 1987.

[AI84] *Artificial Intelligence*, An International Journal, Vol. 24, No. 1-3, Dec., 1984.

[Albano89] Antonio Albano, Conceptual Languages: A Comparison of ADAPLEX, Galileo and Taxis. In Joachim W. Schmidt and Constantino Thanos (Editors), *Foundations of Knowledge Base Management.* Berlin: Springer-Verlag, 1989, pp. 395–408.

[Allen83] James F. Allen, Maintaining Knowledge about Temporal Intervals, *CACM*, Vol. 26, No. 11, pp. 832–843, 1983.

[Borgida89] A. Borgida, M. Jarke, J. Mylopoulos, J. W. Schmidt and Y. Vassiliou, The Software Development Environment as a Knowledge Base Management System, J. W. Schmidt and C. Thanos (eds.), *Foundations of Knowledge Base Management.* Berlin: Springer-Verlag, 1989, pp. 411–442.

[Borgida90] A. Borgida, J. Mylopoulos, J. W. Schmidt and I. Wetzel, Support for Data-Intensive Applications: Conceptual Design and Software Development, R. Hull, R. Morrison, and D. Stemple (eds.), *Proc. of the 2nd International Workshop on Database Programming Languages,* Morgan Kaufmann Publishers, 1990, pp. 258–280.

[Chen76] Peter Pin-Shan Chen, The Entity-Relationship Model — Toward a Unified View of Data. *ACM TODS,* Vol. 1, No. 1, March 1976, pp. 9–36.

[Chung89] L. Chung, P. Katalagarianos, M. Marakakis, M. Mertikas, J. Mylopoulos, and Y. Vassiliou, *From Information System Requirements to Designs: A Mapping Framework*, FORTH/CSI/TR/1989/020, Technical Report Series, Institute of Computer Science – FORTH, Heraklion, Nov. 1989. Also Technical Note CSRI-53, Dept. of Computer Science, Univ. of Toronto, Nov. 1989.

[Chung91a] L. Chung, P. Katalagarianos, M. Marakakis, M. Mertikas, J. Mylopoulos, and Y. Vassiliou, From Information System Requirements to Designs: A Mapping Framework, *Information Systems*, Vol. 16, No. 3. Also Technical Report CSRI–245, Univ. of Toronto, Sept. 1990.

[Chung91b] L. Chung, Goal-Oriented Processing of Requirements Models into Information System Designs, Ph.D. thesis, Dept. of Computer Science, Univ. of Toronto, forthcoming, 1991.

[Denning79] D. E. Denning and P. J. Denning, Data Security, *ACM Computing Surveys*, Vol. 11, No. 3, Sept., Pp. 227–249, 1979.

[DiMarco90] Chrysanne DiMarco, *Computational Stylistics for Natural Language Translation*, Ph. D. Thesis, Dept. of Computer Science, Univ. of Toronto, 1990.

[Etzioni89] Oren Etzioni, Tractable Decision-Analytic control, *Proc. 1st International Conf. on Principles of Knowledge Representation and Reasoning*, Pp. 114 –125, Toronto, Ontario, Canada, May 15 – 18, 1989.

[Greenspan84] S. Greenspan, *Requirements Modelling: The Use of Knowledge Representation Techniques for Requirements Specification,* Ph. D. thesis, Dept. of Computer Science, Univ. of Toronto, 1984.

[Hartson76] H. Rex Hartson and David K. Hsiao, Full Protection Specification in the Semantic Model for Database Protection Languages, *Proc. ACM Annual Conf.*, Oct., 1976.

[Lee90] Jintae Lee, SIBYL: A Qualitative Decision Management System, P. H. Winston and S. A. Shellard (eds.), *Artificial Intelligence at MIT: Expanding Frontiers*, Volume 1, Pp. 105 – 133, The MIT Press, Cambridge, Mass., 1990.

[Lunt89] Teresa F. Lunt and Jonathan K. Millen, *Secure Knowledge-Based Systems*, Technical Report SRI–CSL–90–04, SRI International, Aug. 1989.

[Martin73] James Martin, *Security, Accuracy, and Privacy in Computer Systems.* Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

[Mylopoulos90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, Telos: A Language for Representing Knowledge about Information Systems, To appear in *ACM Trans. Information Systems*, 1990.

[Peckham88] Joan Peckham and Fred Maryanski, Semantic Data Models. *Computing Surveys,* Vol. 20, No. 3, Sept. 1988, pp. 153–189.

[Nielsen90] J. Nielson and R. Molich, Heuristic Evaluation of User Interfaces, *Proc. of CHI '90*, Pp. 249 – 256, April, 1990.

[Nilsson71] Nils Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York, McGraw-Hill, 1971.

[Rabitti88] F. Rabitti, D. Woeld, and W. Kim, A Model of Authorization for Object-Oriented and Semantic Databases, *Proc. EDBT*, 1988.

[Reubenstein90] Howard Reubenstein, *Automated Acquisition of Evolving Informal Descriptions*, Ph. D. Thesis, Also Tech. Report 1205, MIT Artificial Intelligence Lab., 1990.

[Roman85] Gruia-Catalin Roman, A Taxonomy of Current Issues in Requirements Engineering, *IEEE Computer,* pp. 14-21, Apr., 1985.

[Samuelson76] P. A. Samuelson, *Economics*, McGraw-Hill, 10th ed., 1976.

[Simon81] H. A. Simon, *The Sciences of the Artificial*, Second ed. MIT Press, 1981.

[Strong89] D. M. Strong and S. M. Miller, *Exception Handling and Quality Control in Office Operations,* Working paper No. 89-16, Boston Univ., School of Management, 1989.

[Svanks81] M. I. Svanks, *Integrity Analysis: A Metholodology for EDT AUDIT and Data Quality Control*, Ph. D. Thesis, Univ. of Toronto, 1981.

[Thomson88] B. Thomson, E. S. Lee, P. I. P. Boulton, M. Stumm, and D. M. Lewis, *A Trusted Network Architecture*, Computer Systems Research Institute, Univ. of Toronto, Oct., 1988.

[Vassiliou90] Y. Vassiliou, M. Marakakis, P. Katalagarianos, L. Chung, M. Mertikas, and J. Mylopoulos, IRIS — A Mapping Assistant for Generating Designs from Requirements, *Proc. The 2nd Nordic Conference on Advanced Information Systems Engineering, CAiSE '90*, Stockholm, May 1990, pp. 307-338.

[Wetherell82] Error Data Values in the Data-Flow Language VAL, *ACM TOPLAS*, Vol. 4, No. 2, Pp. 226 – 238, April 1982.