Goal-Centric Traceability for Managing Non-Functional Requirements

Jane Cleland-Huang, Raffaella Settimi, Oussama BenKhadra,
Eugenia Berezhanskaya, Selvia Christina
Center for Requirements Engineering
DePaul University
243 S. Wabash
Chicago, IL 60604. USA.
1-312-362-8863

{jhuang,rsettimi}@cs.depaul.edu

ABSTRACT
This paper describes a Goal Centric approach for effectively maintaining critical system qualities such as security, performance, and usability throughout the lifetime of a software system. In Goal Centric Traceability (GCT) non-functional requirements and their interdependencies are modeled as softgoals in a Softgoal Interdependency Graph (SIG). A probabilistic network model is then used to dynamically retrieve links between classes affected by a functional change and elements within the SIG. These links enable developers to identify potentially impacted goals; to analyze the level of impact on those goals; to make informed decisions concerning the implementation of the proposed change; and finally to develop appropriate risk mitigating strategies. This paper also reports experimental results for the link retrieval and illustrates the GCT process through an example of a change applied to a road management system.

Categories and Subject Descriptors
D.2.7 [**Distribution, Maintenance, and Enhancement**]: Extensibility

General Terms
Management

Keywords
Traceability, link retrieval, impact analysis, non-functional requirements, system quality.

1. INTRODUCTION
Software products are increasingly deployed within critical applications in which failure of the system could lead to loss of life, environmental damage, or significant financial loss. There are numerous accounts of catastrophic software failures such as the Therac 25 incidents [16], Ariane 5 launch failure [17], the London Ambulance System [11], and more recently problems with the British Air Control System that introduced over $1M in overruns and subsequently led to major flight downtime at Heathrow International airport [9]. Post mortem analyses have laid at least partial blame on the incorrect implementation and management of non-functional requirements (NFRs). Whereas functional requirements describe what the system needs to do, NFRs describe constraints on the solution space, and capture a broad spectrum of properties such as reliability, portability, maintainability, usability, safety, and security [22]. These NFRs are also known as 'ilities' or quality requirements. Several studies have shown that failure to effectively manage these requirements can have devastating effects on system quality, and can result in expensive downtime or even complete failure of the system [5].

Traceability, which has been defined as "the ability to describe and follow the life of a requirement in both a forwards and backwards direction" from inception throughout the entire system's lifecycle provides useful support mechanisms for managing requirements during the ongoing change process [13]. However, NFRs are difficult to trace because they tend to have a global impact upon a software system [10,14], and because of the extensive network of interdependencies and trade-offs that exist between them [7,10]. Due to these difficulties many organizations entirely fail to trace NFRs, exposing themselves to huge risks when change is introduced or the software is redeployed in a new context.

This paper introduces a goal-centric approach to managing the impact of change upon the NFRs of a software system. Goal Centric Traceability (GCT) enables developers to understand and assess the impact of functional changes upon NFRs in order to maintain the quality of the system throughout its operational lifetime. In the GCT approach, NFRs are first modeled as goals and operationalizations within a Softgoal interdependency graph (SIG) [7,10]. During the change process, traces are then dynamically established from impacted elements of the functional design to potentially impacted elements of the SIG. The traces are generated using a probabilistic network model. They enable developers to first identify the direct impact of the change and then subsequently to evaluate its broader impact on related NFR goals.

The remainder of this paper is laid out as follows. Section 2 provides a brief introduction to Softgoal Interdependency graphs and explains their use for modeling non-functional requirements. Section 3 then introduces the GCT model and its major

ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.

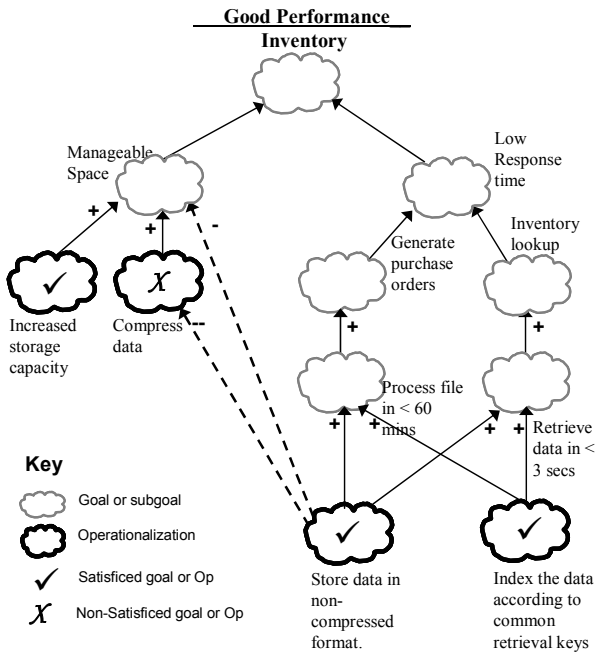


Figure 1. A softgoal interdependency graph

components. Section 4 discusses the role played by traceability in impact analysis activities, describes the probabilistic approach adopted in GCT, and reports on an experiment conducted to evaluate its effectiveness. Section 5 then discusses goal re-evaluation during a GCT change event, and section 6 provides an example in which GCT is utilized to evaluate the impact of a proposed change. Finally, section 7 concludes with an analysis of the applicability and limitations of the approach and some suggestions for future work.

2. MODELING NFRS AS SOFTGOALS

GCT utilizes SIGs to model non functional goals and their various tradeoffs [7,10]. Figure 1 provides an example of a SIG depicting the goal to achieve *good performance* for an inventory system. In a SIG, goals are decomposed into subgoals describing qualities of the system desired by one or more of its stakeholders. For example in this case *good performance* is decomposed into the subgoals of *space* and *response time*. In turn, each subgoal could be further refined into lower-level subgoals, such as the subgoal to *retrieve data in < 3 seconds.*

The term *softgoal* reflects the fact that extensive interdependencies exist between various NFRs, and it is often infeasible to entirely fulfill each and every system goal. Tradeoffs must therefore be made [2]. To understand these tradeoffs, the subgoals are decomposed into 'operationalizations', which provide candidate design solutions for achieving the goal and provide the basis for reasoning about potential tradeoffs. For example the subgoal shown in Figure 1 to *retrieve data in < 3 seconds* could be helped through *storing data in non-compressed format*, or *indexing the data*. However, the first option negatively impacts the performance related *space* subgoal and provides an example of a tradeoff between competing subgoals. Standard NFR catalogues can be used to support the task of goal decomposition [7].

An operationalization can contribute either positively or negatively to its parent goal, or sometimes have no effect on it at all. Contribution relationships are therefore measured qualitatively as: '+' helps, '++" makes, '-' hurts, '- -' breaks, or '?' unknown. SIG analysis includes determining whether input contributions from lower level operationalizations or subgoals, will "*satisfice"* the goal (ie fulfill at least its minimum requirements) [7]. Each node is then analyzed to determine if it is satisficed, based upon the satisficing of its children nodes, and on their individual contributions. The SIG's ability to relate implementation details to system goals, creates the opportunity to provide effective support for the long-term management of NFRs. The challenge is to create a feasible traceability infrastructure capable of identifying impacted regions of the SIG during a change event.

3. THE GCT MODEL

GCT is implemented through the four distinct phases of *goal modeling, impact detection, goal analysis,* and *decision making*. These phases are depicted in Figure 2.

Goal modeling primarily occurs during the elicitation, specification, and architectural design of the system. It is during this phase that non-functional goals are initially modeled as softgoals, decomposed into operationalizations, and negotiated and agreed upon between various stakeholders. The SIG is maintained throughout the life of the software system in order to provide the underlying mechanism for reasoning about the impact of change on NFRs.

During *impact detection* traceability links are established between the functional model of the system and a set of potentially impacted SIG elements. For the purposes of this project UML class diagrams were selected as the primary representation of the functional model. This decision was made for several reasons and in light of the fact that functional change is implemented at either the code or design level, depending upon the selected development environment. Many UML case tools such as Together®, by TogetherSoft™ [25] provide capabilities for managing traces and consistency between design models and code, meaning that establishing links to the design model implies traceability to code and vice versa. Furthermore the ability to understand the impact of a change on the underlying UML model, provides developers with a useful and high-level perspective of the change, and enables them to evaluate its impact prior to implementing it in the code [4]. As numerous papers have discussed the topic of functional impact analysis [3], the starting point of the GCT analysis described in this paper is the point at which a set of classes, or methods within those classes, have been identified as a target of the proposed change.

GCT implements a dynamic approach to trace retrieval which supports the extensive network of links needed between the functional and non-functional models of the system. During impact detection the retrieval algorithm returns a set of potentially impacted SIG elements. These links are then evaluated by the user and any incorrectly retrieved links are discarded. This process is very similar to issuing a query on a web-based search engine and then selecting which of the returned links are potentially valid ones and worth further exploration. The outputs

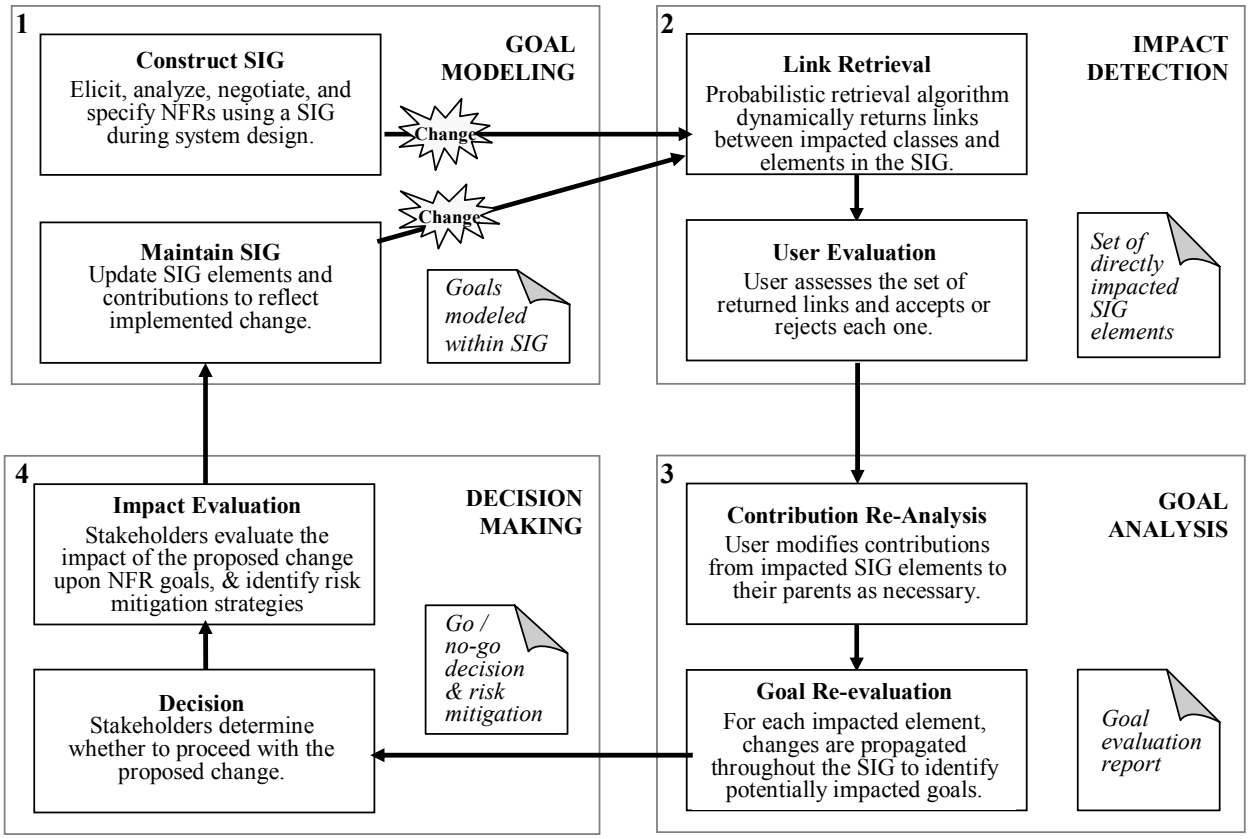


Figure 2. Goal-centric traceability

of this phase are a set of operationalizations and goals, confirmed by the user to be relevant.

During the *goal analysis* phase, the impact of the change is propagated throughout related regions of the SIG in order to evaluate its effect on system wide goals. Goal re-evaluation is a recursive activity in which each retrieved operationalization is examined to determine how the proposed change impacts the satisficing of its parent's goals. In turn, any parent goal that is no longer satisficed is also re-evaluated to determine how the change will impact its own parents. This continues until all potentially impacted goals have been re-evaluated. In order to fully understand the benefits of the change, a similar process is executed for operationalizations that strengthen their parent goals. The primary output of this phase is an impact analysis report identifying all goals that are either negatively or positively affected by the proposed change.

Finally, during the *decision making* phase, stakeholders examine the impact report and determine if the change should be implemented or not. Alternate design solutions can be proposed and explored in order to mitigate or remove potential negative impacts of the change.

4. IMPACT DETECTION

Impact detection is dependent upon the effectiveness of the traceability mechanism to establish correct links between the functional and non-functional models. In current practice, traceability links are implemented through matrices, hypertext links, graphs, or more formal methods [13]. However practice has shown that it is hard to maintain links in a constantly evolving system [2,24], and the problem is exacerbated for NFRs because of their tendency for broad ranging impact which often results in the need for an excessive number of links.

To address problems of trace maintenance, several researchers have investigated the use of dynamic schemes for runtime link generation. These include information retrieval methods [1,15,20] and heuristic approaches [23]. The advantages of dynamic methods are that they eliminate the need for long-term maintenance of traceability links and subsequent problems of link degradation. However, these approaches also suffer from a certain level of imprecision. Despite this precision problem, the extensive network of traceability links needed to support impact analysis of NFRs, indicates the impracticality of establishing and maintaining explicit links. GCT therefore utilizes a dynamic approach to link generation in which loss of precision is countered by the user's role in evaluating the set of generated links.

The problem of linking SIGs to UML artifacts was directly addressed by Cysneiros et al [10] who proposed an ontological approach in which specific keywords were embedded in both the operationalizations of the SIGs and as notations within UML diagrams. This approach has the benefit of explicitly and precisely defining relationships between two entities in which the same keyword is embedded. It does however require the formal construction of a system-wide ontology and the formality of correctly using it throughout the development process. In contrast

the more relaxed retrieval approach adopted by GCT returns less precise results but also offers several of its own advantages. First, this approach does not require the systematic construction of keywords into the SIG and UML artifacts, but allows developers to use their own words. As a result, GCT can be applied retrospectively to a project and still return meaningful results. More importantly the GCT retrieval algorithm returns a broader set of results to the developer, thereby providing a richer context for understanding the ramifications of a proposed change.

The dynamic approach adopted in GCT falls under the general category of an information retrieval (IR) method. In IR a user requests information using either keywords or a natural text based query, and a search engine retrieves a set of candidate objects from which the user selects the relevant ones [12]. In most retrieval algorithms a similarity index is calculated that measures the degree of similarity between two documents. A threshold value is then established so that any index value above the threshold indicates that the document should be retrieved.

IR algorithms are then typically evaluated using the metrics of *recall* and *precision* [12] where:

$$\text{Recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Number of relevant documents}}$$

$$\text{Precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

Maximum recall can be trivially achieved by retrieving all documents in the collection, and maximum precision by retrieving a single document that is known to be correct. For this reason, an IR system normally attempts to maximize recall and precision simultaneously.

In existing traceability-related retrieval methods, a query is typically formulated from the words in a requirement, and a search algorithm executes the query within a search space of other artifacts such as other requirements, code, test cases, or design models. Results from previous IR retrieval methods have met limited success with precision rates typically at 10-60% when the threshold level is set to achieve 90-100% recall [1,15,20,23].

4.1 Impact Detection in GCT

The GCT retrieval algorithm was implemented using a probabilistic network model. Our approach follows previous work of Wong and Yao who adopted a probabilistic inference model to support information retrieval [26,27]. We selected this method because our intention to incorporate additional factors such as hierarchical information into future retrieval algorithms is supported by the expressiveness of the approach. Wong and Yao's probabilistic retrieval model is based on an epistemological view of probability for which probabilities are regarded as degrees of belief, and may not be necessarily learned from statistical data. The model assumes that the relevance relationship between a document and a user's query cannot be determined with certainty. Both documents $(d_1,d_2,\ldots,d_n)$ and queries $(q_1,q_2,\ldots,q_m)$ are represented as propositions within a concept space U, called the *Universe of Discourse*. Singletons in the model space U are elementary concepts, so propositions are subsets of U. A probability function is defined over the model space, so that the probability $pr(d)$ of a proposition can be interpreted as the degree of coverage of U by the knowledge contained in proposition d.

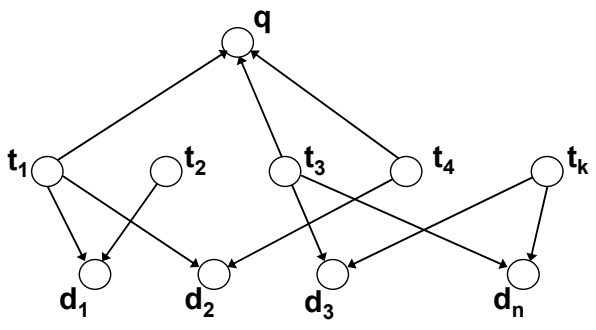


Figure 3. Probabilistic network model for information retrieval

In a preliminary phase of the information retrieval algorithm, documents and queries are reduced to sets of index terms. This greatly diminishes the computational effort, and improves the retrieval performance. In this preliminary phase, stop words which are not informative words such as conjunctions and adverbs are eliminated, and words are stemmed to their common grammatical root by eliminating prefixes and suffixes [12]. The elementary concepts in U are the index terms extracted from the collection of documents. Thus documents and queries are modeled as subsets in U.

In the probabilistic network model documents, queries, and index terms are network nodes, and each node is associated to a binary random variable. As depicted in the independence graph of Figure 3, index term nodes point to the document nodes and query nodes in which they are contained. The main model assumptions among the variables in the graph are that documents are conditionally independent on the query given the index terms, and that the index terms are pair-wise disjoint.

For each document, the probability $pr(d_j)$ is computed as $pr(d_j) = \sum_i pr(d_j \mid t_i)\, pr(t_i)$. The epistemological approach interprets $pr(d_j|t_i)$ as the user's degree of belief that the elementary concept t_i supports the knowledge in d_j. It seems natural to use the document semantics to estimate the probability values, as suggested in [27]. The conditional probability $pr(d_j|t_i)$ is assumed to be proportional to the frequency of term t_i in d_j and is estimated as follows

$$pr(d_j \mid t_i) = \frac{freq(d_j, t_i)}{\sum_k freq(d_j, t_k)}$$

The probability of each index term $pr(t_i)$ is estimated as a function of the inverse term frequency

$pr(t_i) = \eta \frac{1}{n_i}$, where η is a normalizing constant and n_i is the number of documents containing the term t_i. This expression assumes that the probability of each index term is inversely proportional to the number of documents containing the index term. Thus less frequent index terms have higher probability values, since words that appear in fewer documents are considered to be more informative in representing a concept in U. This is in

line with other information retrieval algorithms such as the vector space model algorithm that use tf-idf ranking strategies [12].

The relevance of a document d_j to a query q is evaluated as the conditional probability $pr(d_j|q)$, which can be regarded as the degree of coverage for document d_j provided by q. The conditional independence assumption in the model allows us to compute $pr(d_j|q)$ using the conditional probabilities $pr(d_j|t_i)$ and $pr(q|t_i)$ as follows:

$$pr(d_j \mid q) = \left[\sum_i pr(d_j \mid t_i) pr(q \mid t_i) pr(t_i) \right] \Big/ pr(q)$$

where $pr(q \mid t_i) = \dfrac{freq(q, t_i)}{\sum_k freq(q, t_k)}$ is computed analogously to $pr(d_j|t_i)$ and the probability of a query q is $pr(q) = \sum_i pr(q \mid t_i) pr(t_i)$

Documents d_j are ranked according to their degree of relevance to the query q, computed as $pr(d_j|q)$. A threshold value is typically established so that all documents with probability of relevance higher than the threshold will be retrieved. In this paper, the threshold values were selected by optimizing the objective function "*maximize Recall + Precision, where recall > 85%*". This is equivalent to finding the threshold value which maximizes both recall and precision while maintaining a sufficiently high recall level to effectively implement GCT. A training set can be used to select the threshold values.

4.2 Experimental evaluation

An experiment was designed to measure the effectiveness of the impact detection method. This experiment was based on artifacts from the Ice Breaker System, which is described in [19] and enhanced with requirements mined from documents obtained from the public work departments of Charlotte, Colorado [6]; Greeley, Colorado [8]; and the Region of Peel, Ontario [18]. The Ice Breaker System manages de-icing services to prevent ice formation on roads. It receives inputs from a series of weather stations and road sensors within a specified district, and uses this information to forecast freezing conditions and schedule dispersion of salt and other de-icing materials. The system maintains maps of the district in order to plan de-icing routes and to ensure complete coverage of all roads. It also manages the inventory of de-icing materials; maintains, dispatches, and tracks trucks in real time; and issues and tracks work orders. The Ice Breaker system consists of 180 functional requirements, and nine distinct SIGS representing NFRs of accuracy, availability, completeness, cost, extensibility, operational security, performance, safety, and usability. The Usability SIG from the Ice Breaker System is illustrated in Figure 4. These SIGs were composed of 82 goals and subgoals, and 98 operationalizations. The functional model of the system was constructed by a team of students and faculty at DePaul University, and was represented through nineteen use cases, and 75 classes bundled into sixteen packages. A project glossary was constructed at an early stage and developers were encouraged to utilize standard terms wherever possible.

The Ice Breaker System was selected as opposed to a larger industrial sized data set, due to the sheer enormity of the task of constructing an accurate traceability matrix against which to compare experimental retrieval results. In this system alone, the 75 classes and 180 SIG elements introduced 13,500 potential

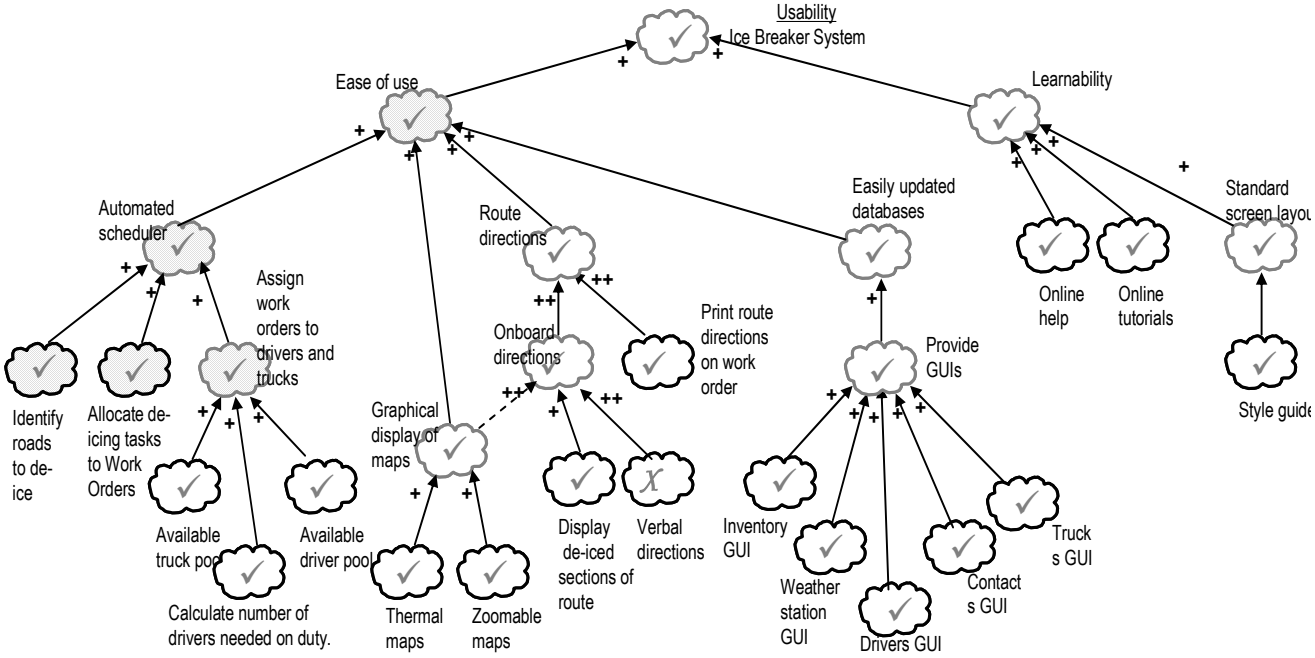


Figure 4. The Usability SIG from the Ice-Breaker system
(Shaded goals and operationalizations are incorporated into the change impact analysis of Figure 6)

Table 1. Results of link retrieval

	Training Set						*Full data set*
Threshold	0.015	**0.02**	0.025	0.030	0.04	0.05	0.02
Actual	90	**90**	90	90	90	90	619
Retrieved	226	**205**	190	180	154	135	1052
Correctly retrieved	87	**86**	80	78	76	70	540
Recall	0.9667	**0.9556**	0.8889	0.8667	0.8444	0.7778	0.8724
Precision	0.3850	**0.4195**	0.4210	0.4333	0.4935	0.5185	0.5133
Objective Function (Recall + Precision)	1.3517	**1.3751**	1.3099	1.3000	Recall < 85%	Recall < 85%	

Table 2. Results of link retrieval by SIG type

	Accuracy	*Availability*	*Completeness*	*Cost*	*Extensibility*	*Op. Security*	*Performance*	*Safety*	*Usability*
Actual	46	46	74	50	94	16	71	79	143
Recall	0.8478	0.5000	0.8514	0.9400	0.9468	0.8750	0.9014	0.8608	0.9301
Precision	0.4063	0.4107	0.5122	0.8103	0.5528	0.5000	0.4156	0.4626	0.5833

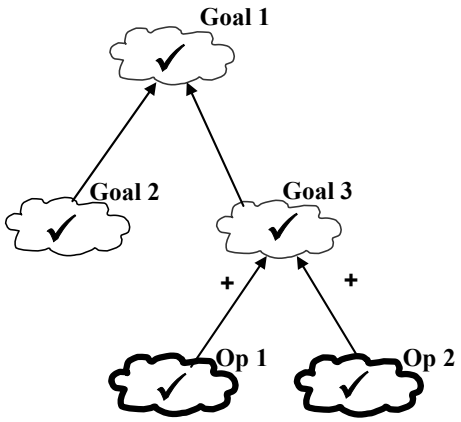
links! The traceability matrix was constructed through careful evaluation of the relationships between SIG elements and classes. Unfortunately link identification is a somewhat subjective process, because in addition to a smaller number of very clear links, and a large number of obvious non-links, there are numerous instances of possible links. These 'gray' areas occur when an operationalization or goal is somewhat related to a class. We decided to include links to these SIG elements based on the rationale that they would provide a stakeholder with a more complete context in which to perform impact analysis. It should be noted however that our tool tends to identify these "gray" classes as links, and so this decision certainly improved the measured precision of the results.

The results of the experiment were evaluated against the predefined matrix and reported in terms of precision and recall metrics. Threshold values for the retrieval algorithm were determined using a training set. In our experiment, the training set was constructed by selecting 25 classes (about one third of class documents) and 60 SIG elements, chosen by randomly selecting one third of the elements from each SIG type. The tool was run on the training set to evaluate the link retrieval performance at different threshold values. Results are displayed in Table 1. The threshold value equal to 0.02 was selected since it optimized the previously defined objective function. At this value, the tool returned roughly 95% of links with about 42% precision.

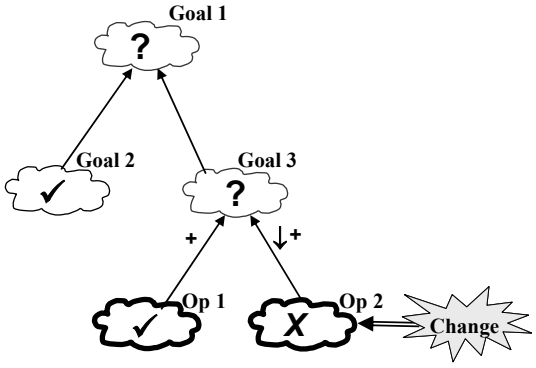
The tool was then run on the entire dataset using the threshold level of 0.02 selected from the training set. The results reported in the rightmost column of Table 1, show the tool is able to retrieve more than 87% of the actual links, with about 51% precision level. It should be noted though that for equivalent recall values, this experiment between classes and SIGS returned higher precision than for similar retrieval experiments conducted on the same dataset between other artifacts such as requirements and code or requirements and UML classes[28].

No significant differences in recall and precision metrics were observed in queries issued against operationalizations versus goals, although these results were attributed to the fact that goal-based links were desired for only a few explicitly stated goals. As reported in Table 2, all SIG types resulted in recall metrics over 85% and precision of approximately 40 – 60%. The exceptions were the *Cost* SIG which performed exceptionally well with recall of 94% and precision of 81%, and the *Availability* SIG which performed rather poorly with recall of 50% and precision of 40%. Further experiments need to be conducted to more fully understand the implications of these results for individual SIG types, and on varied datasets.

An average number of 14.03 links were returned for each query, with standard deviation equal to 9.06. Roughly 61% of queries returned fifteen or fewer links, and 90% returned twenty-five or fewer. The largest number of links returned was forty, and this occurred in only one query. These results indicate that on average a GCT impact detection query for a single class would return about 14 or 15 links that would need to be evaluated by a user to determine their correctness. Out of these links approximately 50% of them will be useful and the remaining ones will be discarded. This is an easily manageable task and compared to the alternate options of brute force analysis or maintaining an excessively complex traceability matrix, it dramatically reduces the amount of work needed to detect the impact of a functional change upon the operationalizations and goals of the SIG.



a. SIG prior to change



b. SIG showing change impact

Figure 5. Goal re-evaluation

5. GOAL ANALYSIS

Once the retrieval algorithm has returned a set of potentially impacted SIG elements, and the user has filtered these to remove non-relevant ones, the goal analysis phase can commence. During the goal analysis phase of GCT, the potential impact of the change is propagated throughout the SIG to determine its broader effect on non-functional goals. Goal evaluation follows the standard SIG methodology in which the levels of contribution from a goal's children are evaluated in order to determine if the goal can be fulfilled by the current operationalizations [7]. The goal re-evaluation process implemented in GCT is depicted through the example of Figure 5, in which a change is introduced that directly impacts Operationalization (Op) 2.

To support reasoning about the impact of change upon the SIG elements two additional symbols are introduced. The "strengthens" (↑) symbol is used to depict contributions that have been strengthened through a change in the impacted operationalization or subgoals, while the "weakens" (↓) symbol depicts contributions that have been weakened. In this example, Op2's contribution to Goal 3 is weakened and developers must therefore determine if the *helps* contribution from Op1 and the reduced help provided by Op2, is sufficient to satisfice Goal 3. Even if it is still satisficed, it is likely to be satisficed at a lower level than it previously was, and so repercussions on its higher level goals should also be evaluated.

Goal evaluation within a SIG is by nature a qualitative and subjective process. However its usefulness has been clearly demonstrated through numerous examples and case studies [7,10]. GCT extends the application of the SIG analysis to more effectively support goal re-evaluation during the ongoing change process.

6. GCT IN ACTION

In this section the GCT process is illustrated through a proposed enhancement to the Ice Breaker System intended to provide more accurate predictions of road freezing conditions. The current system predicts freezing conditions based solely on temperature readings from the road sensors and data received from local weather stations, whereas the enhanced version would incorporate elevation and exposure information for each road section into the prediction algorithms.

During functional impact analysis the sequence diagram entitled "Predict freezing conditions" was analyzed to determine how to accommodate the proposed changes. It was determined that changes should be made in the *road sensor* and *road* classes. The *road sensor* class would be extended to store elevation and exposure data for the location of the sensor, while the method contained in the *road* class for predicting freezing conditions for each road section would be modified. Using the threshold value of 0.02 identified from the training set, a query was therefore issued for each of these classes against the Ice Breaker SIG elements. Combined results from these queries resulted in a return of seventeen SIG elements shown in Table 3. Following user evaluation of the links, eight were accepted as correct, four were labeled as indirectly related, and the remaining five were out rightly rejected.

All of the directly impacted SIG elements, their potentially impacted goals, and additional elements that contributed to potentially impacted goals, were combined into an integrated SIG. This is shown in Figure 6. For example, the Operationalization "*Analyze district freezing conditions < 10 minutes"* which was identified as an impacted element, contributes a *helps* relationship to its parent goal of "*Real-time processing"*. A sibling operationalization of "*Generate weather forecast in < 10 minutes"* was also included in the SIG because both contributions needed to be considered in order to determine if the parent goal was still fulfilled. Depending upon events during the analysis process, additional elements may be included at a later time.

Affected operationalizations and goals identified in the impact detection phase are shaded gray. Impacted elements are first categorized according to whether the proposed change enhances or detracts from the original contribution to the parent goal (ie ↑ or ↓ contribution). In this example the operationalizations "*analyze district freezing conditions < 10 minutes"* and "*assign sensors to roads"* both exhibit weakened contributions to their parent goal, whereas the other impacted components either have strengthened contributions or no observed change.

The performance goal to "*analyze district freezing conditions < 10 minutes"* is re-evaluated according to normal practices. For example, if during initial system development the response time had been validated through the application of a software performance graph [21], then the graph could be updated to

Table 3. Combined results for a query against *road* and *road sensor* classes

Prob-ability	Operationalization	Directly Related Goal	SIG Type	Probability of link to "Road Sensor" class	Probability of link to "Road" class	Action
0.11176	Track truck within 50m of actual location	Real-time tracking of dispatched trucks	Performance	0.11176		Reject
0.07669	Remove road sections	Modify map	Extensibility	0.07669	0.14333	Maybe
0.05795	**Analyze freezing conditions**	**Real time processing**	**Performance**	**0.05795**	**0.10857**	**Accept**
0.05752	Add new road sections	Modify map	Extensibility	0.05752	0.10750	Maybe
0.05286	**Monitoring of road conditions**	**Road safety**	**Safety**		**0.05286**	**Accept**
0.04967	Display de-iced sections of route	Onboard directions	Usability	0.04967	0.06786	Maybe
0.04510	**Record road freezing events**	**Accurate weather predictions**	**Accuracy**	**0.04510**	**0.08036**	**Accept**
0.03964	Broadcast road conditions to contacts	Monitoring of road conditions	Safety		0.03964	Maybe
0.03725	Update truck status < 1 minute.	Real time communication through onboard computer	Performance	0.03725		Reject
0.02714	Remove weather station	Network of weather stations	Extensibility		0.02714	Reject
0.02571	**Assign roads to sensors**	**Modify road sensors**	**Extensibility**		**0.02571**	**Accept**
0.02571	**Identify roads to de-ice**	**Automated scheduler**	**Usability**		**0.02571**	**Accept**
0.02571	**Modify road sensors**	**Modifiable maps**	**Extensibility**		**0.02571**	**Accept**
0.02571	**Validation by road engineer**	**Accuracy of maps**	**Accuracy**		**0.02571**	**Accept**
0.02036	Add new weather station	Network of weather stations	Extensibility		0.02036	Reject
0.02036	**Compare actual conditions to predictions**	**Accurate weather predictions**	**Accuracy**		**0.02036**	**Accept**
0.02036	Enter weather conditions manually	Failure mode operations	Availability		0.02036	Reject

incorporate additional processing of elevation and exposure data for each road sensor in order to determine if the performance goal could still be met. In this case it was determined that the additional processing in larger districts could result in failure to meet the performance goal. This failure would have negative implications on the satisficing of *real-time processing, response time,* and *performance* goals.

The road safety goal to "*modify road sensors"* could also be negatively impacted by the impact of the proposed change upon the "*assign sensors to roads"* operationalization. This is because under the proposed enhancement, sensors could only be assigned to road sections for which additional elevation and exposure data is available. Goal analysis indicates that this problem could negatively impact the goals of *modifiable maps* and *extensibility.*

It was determined that all other impacted operationalizations and goals would contribute positively to the system goals, and would result in an improvement in road safety, more cost efficient use of de-icing materials, and improvements to the automated scheduling of de-icing tasks.

During the final stage of Decision Making, the tradeoffs were analyzed and risk mitigation strategies were considered. It was determined that the performance issue could be mitigated through the use of multiple processors and that the modifiability issue could be easily resolved through the use of default elevation and exposure values. Without the benefit of GCT analysis the performance issue may have only been discovered following deployment of the new system, and modifiability of the system could have been negatively impacted by the change. This example therefore demonstrates that GCT can provide a useful approach to discovering these potential impacts prior to deployment of the change. Once a decision has been made to implement the change all impacted operationalizations are re-evaluated to determine if any contributions to their parents have changed sufficiently in strength to merit a formal change in the SIG. Contributions are then updated in order to maintain the SIG in an accurate state to support future change events.

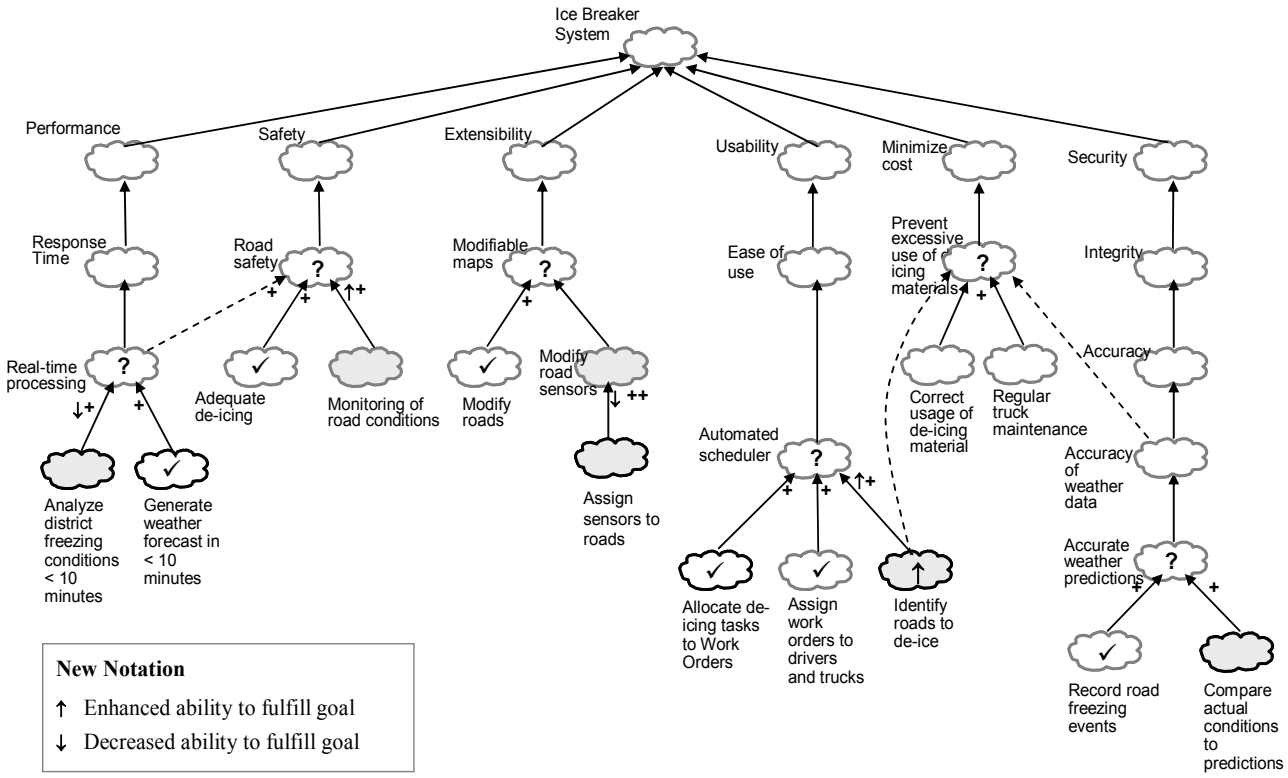


Figure 6. Integrated SIG showing potentially impacted elements of the Ice Breaker System.

7. CONCLUSIONS

Goal Centric Traceability provides developers with the means of managing the impact of functional change upon non-functional requirements. This is crucial to the long-term maintenance of critical systemic qualities such as safety, security, reliability, usability, and performance.

The experimental results reported in this paper indicate the feasibility of using a probabilistic approach to dynamically retrieve traceability links for non-functional requirements. The imprecision problems introduced through use of this method are largely mitigated through user inspection of retrieved links and through establishing a sufficiently low threshold that minimizes the number of omission errors. Although users' feedback is required to filter out unwanted links, the effort is only a small fraction of that which would be required to perform the trace manually. For example, in the query described in this paper against the *Road* and *Road sensor* classes, the user had to examine 22 candidate links. However, without the benefit of the tool, a manual impact analysis would require the user to consider 180 SIG elements for each of the two classes being traced – a total of 360 possible comparisons.

Although dynamic traceability methods do not offer a silver-bullet, they do provide a very practical effort-reducing approach to link retrieval. Furthermore, in any non-trivial system, the cost and effort required to construct and maintain a traceability matrix that would capture the extensive network of relationships between the NFRs and other downstream artifacts is often infeasible.

Nevertheless, future work will focus on improving both recall and precision metrics through incorporating additional factors, such as hierarchical information, into the retrieval process. Additional work is also needed to analyze the effectiveness of the approach for various SIG types and even for different types of operationalizations within a single SIG.

8. ACKNOWLEDGMENTS

The work described in this paper was partially funded by NSF grant CCR-0306303. We also acknowledge the contributions of DePaul students Jigar Mody, Chris DePalma, Wiktor Lukasik, and Julie Zhang in collecting and managing data, and in contributing to the development of earlier versions of the Poirot tool.

9. REFERENCES

[1] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E., "Recovering Traceability Links between Code and Documentation", *IEEE Transactions on Software Engineering*. 28,10 (Oct. 2002), 970-983.

[2] Boehm, B., and In, H., "Identifying Quality-Requirement Conflicts (Extended Abstract)," *Proc. of IEEE International Conference Requirements Engineering*, (Apr. 1996) 218.

[3] Bohner, S. and Arnold, R., *Software Change Impact Analysis*, IEEE Comp. Soc. Pub., Tutorial Series, 1996.

[4] Briand, L.C., Labiche, Y., O'Sullivan, L., "Impact Analysis and Change Management of UML Models", *IEEE Intn'l Conf. on Software Maint*, (Sept. 2003) 256-265.

[5] Brooks Jr., F.P., "No Silver Bullet: Essences and Accidents of Software Engineering," *IEEE Computer.*, 4, (Apr. 1987), 10-19.

[6] Charlotte Dept of Transportation, N.C, http://www.charmeck.org/Departments/Transportation/Home.htm

[7] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic, 2000.

[8] City of Greeley, Colorado, http://www.ci.greeley.co.us

[9] CNN News, "Air travel chaos hits Britain" http://www.cnn.com/2004/WORLD/europe/06/03/britain.flight/index.html

[10] Cysneiros, L., Sampaio de Prado Leite, J., "Nonfunctional Requirements: From Elicitation to Conceptual Models", *IEEE Transactions on Software Engineering*, 30,5, (May 2004) 328-350.

[11] Finkelstein, A., and Dowell, J., "A Comedy of Errors: The London Ambulance Service Case Study," *Proc. Eighth Intn'l Workshop Software Spec and Design*, (1996), 2-5.

[12] Frakes W.B., and Baeza-Yates, R., *Information retrieval: Data structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[13] Gotel, O., and Finkelstein, A., "An Analysis of the Requirements Traceability Problem," *1st International Conference on Requirements Eng.,* (1994), 94-101.

[14] Gross, D., and Yu, E., "From Non-Functional Requirements to Design through Patterns", *Requirements Engineering Journal*, 6,1 (2001), 18-36.

[15] Huffman Hayes, J., Dekhtyar, A., and Osborne, J., "Improving Requirements Tracing via Information Retrieval", *IEEE International Requirements Engineering Conference,* (Monterey, CA, Sept. 2003), 138-150.

[16] Leveson, L., and Turner, C. S. "An Investigation of the Therac-25 Accidents" *IEEE Computer.*, 26,7, (July 1993), 18-41.

[17] Nuseibeh, B. Ariane 5: Who Dunnit?, *IEEE Software*, 14,3 (May 1997), 15-16.

[18] Region of Peel, http://www.region.peel.on.ca

[19] Robertson, S., Robertson, J., *"Mastering the Requirements Process"*, Addison-Wesley, 1999.

[20] Settimi, R., Cleland-Huang, J., BenKhadra, O., Mody, J., Lukasik, W., and DePalma, C., "Supporting Change in Evolving Software Systems through Dynamic Traces to UML", *IEEE International Workshop on Principles of Software Evolution*, Kyoto, Japan, (Sept. 2004), 49-54.

[21] Smith, C., and Williams, L., *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley, 2002

[22] Sommerville, I., and Sawyer, P., "Viewpoints: Principles, Problems, and a Practical Approach to Requirements Eng.," *Annals of Software Eng.*, 3, N. Mead, ed., (1997), 101-130.

[23] Spanoudakis, G., "Plausible and Adaptive Requirement Traceability Structures", *14th International Conference on Software Engineering and Knowledge Engineering*, (Ischia, Italy, 2002), 135-142.

[24] Sugden, S.C., and Strens, M.R., "Strategies, Tactics, and Methods for Handling Change", *IEEE Symposium and Workshop on Eng. of Computer Based Systems*, (Fredrichshafen, Germany, Mar. 1996), 457-462.

[25] TogetherSoft™, http://www.togethersoft.com

[26] Wong, S.K.M., and Yao, Y.Y., "A probabilistic inference model for information retrieval" *Information Systems*, 16,3, (1991), 301-321.

[27] Wong, S.K.M. and Yao, Y.Y. "On Modeling Information Retrieval with Probabilistic Inference", *ACM Transactions on Information Systems*, 13,1 (1995), 38-68.

[28] X. Zou., Settimi, R., Cleland-Huang, J., Duan,C., "Empirical Study of a Probabilistic Network Approach for Retrieving Traceability Links", *Technical Report # TR05-003, School of Computer Science, Telecommunications, and Information Systems, DePaul University*, (March 2005).