Automated Extraction and Visualization of Quality Concerns from Requirements Specifications

Mona Rahimi
School of Computing
DePaul University
Chicago, IL, 60604, USA
m.rahimi@acm.org

Mehdi Mirakhorli
School of Computing
DePaul University
Chicago, IL, 60604, USA
mehdi@cs.depaul.edu

Jane Cleland-Huang
School of Computing
DePaul University
Chicago, IL, 60604, USA
jhuang@cs.depaul.edu

***Abstract*—Software requirements specifications often focus on functionality and fail to adequately capture quality concerns such as security, performance, and usability. In many projects, quality-related requirements are either entirely lacking from the specification or intermingled with functional concerns. This makes it difficult for stakeholders to fully understand the quality concerns of the system and to evaluate their scope of impact. In this paper we present a data mining approach for automating the extraction and subsequent modeling of quality concerns from requirements, feature requests, and online forums. We extend our prior work in mining quality concerns from textual documents and apply a sequence of machine learning steps to detect quality-related requirements, generate goal graphs contextualized by project-level information, and ultimately to visualize the results. We illustrate and evaluate our approach against two industrial health-care related systems.**

***Index Terms*—requirements, goal Model, quality concerns, visualization**

I. Introduction

Software requirements come in many different shapes and sizes ranging from user stories to more traditional formats centered around "shall" statements. Their purpose is to clearly define the required functionality of the system and also to specify any quality constraints. However, in practice, requirements specifications tend to focus on the functional needs of stakeholders and often fail to describe their underlying quality concerns [2], [6], [8], [12]. Such concerns cover a broad range of qualities such as security, performance, availability, extensibility, and portability [8], [30] and play a critical role in the architectural design of the system.

When quality related requirements do exist, they are often incomplete and intermingled with functional requirements. The problem seems to arise from the underlying misconception that developers and customers have a shared, often unspoken, vision of the quality concerns of the system. However, this is often not the case and as a result, the delivered system may underperform against users expectations [2].

Having the ability to fully understand the quality concerns of a system and to identify missing or incomplete quality related requirements is especially important in domains such as the healthcare industry, governed by privacy concerns [4], [22], or in systems where accessibility, safety, or security concerns are regulated [29].

In this paper we present a novel approach for extracting and visualizing quality concerns from an existing set of requirements. Our approach, which is illustrated in Fig. 1, builds upon our prior work in using data mining techniques to detect non-functional requirements [9], and then introduces a novel approach for contextualizing the concerns according to specific domain topics. Instead of simply returning a flat list of quality concerns, we compose them into a meaningful hierarchy of topics. Our process includes three phases of modeling and detecting quality concerns, discovering impacted areas of functionality, and discovering and generating a meaningful hierarchy.

Visualizing concerns in such a hierarchy provides potential support for several important software engineering activities including: (1) understanding quality related concerns for the system in order to support compliance verification, (2) identifying underspecified concerns, (3) driving architectural design and assessment, (4) communicating with customers, and (5) supporting change impact analysis.

Throughout this paper we illustrate and evaluate our approach against two requirements specifications taken from the healthcare domain. The first, dataset is developed by the Certification Commission for Healthcare Information Technology (CCHIT) [5] and includes 235 requirements for managing electronic healthcare records. The second dataset represents 1,736 requirements extracted from documentation for WorldVista, the USA Veterans Health Care system [16]. We provide examples of requirements from each of these datasets in Table I along with the concerns they address.

The remainder of the paper is laid out as follows. Section II introduces the notion of a Softgoal Interdependency Graph (SIG) and discusses the specific challenges for automating its construction. Section III describes our existing approach for retrieving quality-related requirements from a specification. Section IV describes our solution for discovering topics, constructing a hierarchy of contextualized requirements, and generating the SIG. Sections V and VI then report on a user evaluation of the generated solution, and discuss several different usage scenarios. Finally, Section VII describes related work, Section VIII discusses threats to validity and Section IX summarizes our findings and proposes ideas for future work.

978-1-4799-3033-3/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

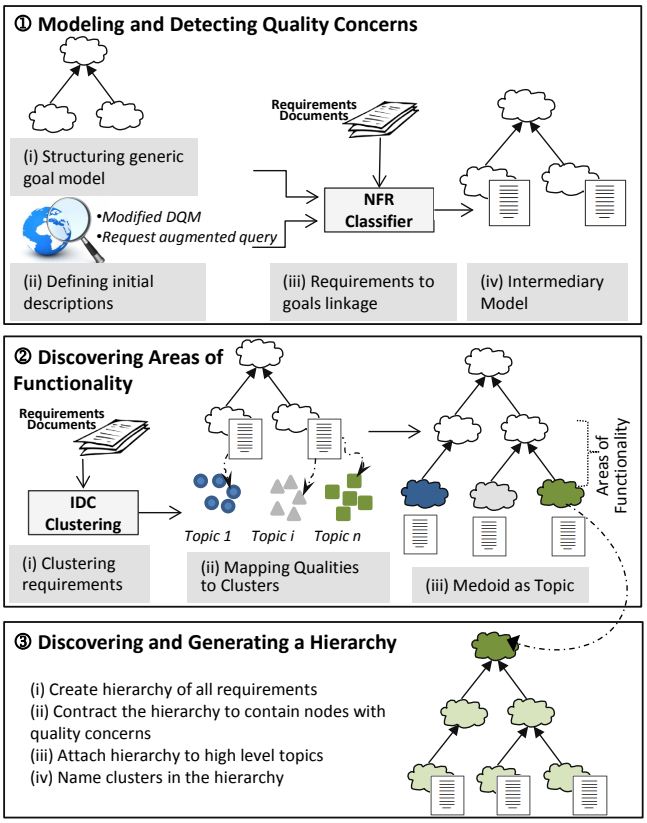

Fig. 1. Overview of the SIG construction and visualization process

II. Goal Models

Our approach is designed to automatically generate a basic goal model from the requirements specification. We adopt the notation of Softgoal Interdependency Graphs (SIG), which are designed to model quality concerns and their various tradeoffs [8], [12]. While there are several approaches for modeling goals, including KAOS [36] and i* [37], we selected the SIG approach because it matched our objectives of visualizing a simple hierarchy of quality goals.

A. Softgoal Interdependency Graph

A SIG provides the graphical notation needed to depict the quality concerns of a system. Each concern is modeled as a softgoal, reflecting the fact that extensive interdependencies exist between various quality concerns, and furthermore, that trade-offs must be made in order to satisfice (i.e. sufficiently satisfy) each of the goals.

Fig. 2 provides an example of a security-related integrity goal for a customer-serving application. In this example, the *Integrity* softgoal is decomposed into *completeness* and *accuracy* subgoals. *Accuracy* is refined into *billing* and *patient information*, and *billing* is further refined into *accounts receivable* and *accurate invoices*. Finally, *nightly batch reconciliation* and *transaction controls* are proposed as design solutions (referred to as operationalizations) for achieving accurate invoices. The SIG also depicts contribution structures

TABLE I
Sample requirements from WorldVista and CHHIT showing their quality type

Project	Sample Requirement	Type
WV	Generates progress notes. Displays all information at the time of an ART event on the Progress Notes API and allows editing of the note prior to sign off.	F
WV	VistA Esig applications are required to authorize and authenticate their users.	A
WV	The system shall audit sensitive records, enabling a site to follow up on a record accessed by a remote location.	AU
CCHIT	The system shall provide the ability to capture and store discrete data regarding symptoms, signs and clinical history, from a clinical encounter and to associate that data with codes from standardized nomenclatures.	F
CCHIT	The system shall enforce the most restrictive set of rights/privileges or accesses needed by users groups (e.g. System Administration, Clerical, Nurse, Doctor, etc.), or processes acting on behalf of users, for the performance of specified tasks.	AC
CCHIT	The system upon detection of inactivity of an interactive session shall prevent further viewing and access to the system by that session by terminating the session, or by initiating a session lock that remains in effect until the user reestablishes access using appropriate identification and authentication procedures. The inactivity timeout shall be configurable.	AL
F=Functional, A=Authentication, I=Integrity Controls AUD=Audit, AC=Access, AL=Automated Logoff		

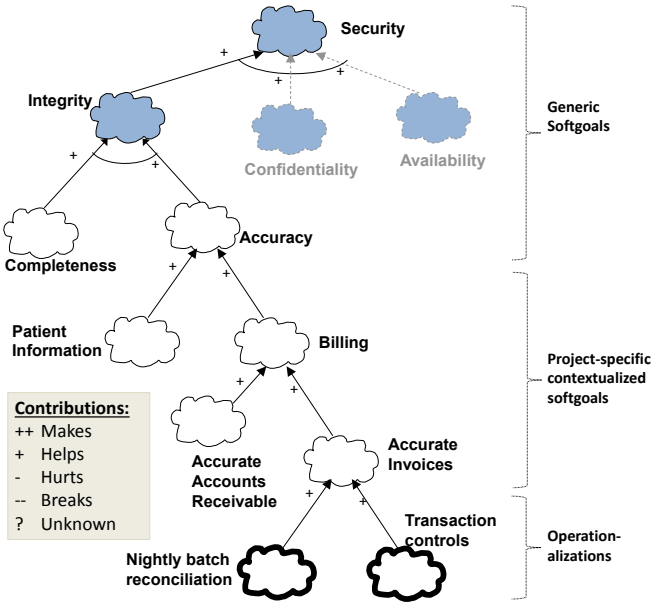

Fig. 2. An Example of a Softgoal Interdependency Graph (SIG) showing generic softgoals, contextualized softgoals, and candidate operationalizations

which capture the extent to which a child goal contributes to the satisficing of its parent. Each contribution is labeled as ++ makes, + helps, - hurts, and –breaks. A SIG would typically also include several candidate operationalizations and a variety of contribution structures. However, the reverse engineering

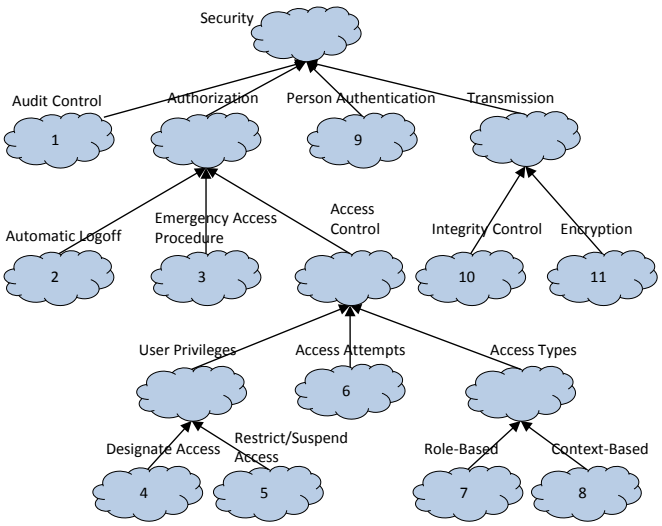

Fig. 3. Security concerns specified as a SIG scoping model define the SIG to be generated. Leaf nodes are numbered.

process presented in this paper seeks to reconstruct a current snapshot of the goals, quality concerns, and requirements, and therefore is not designed to reconstruct alternate design decisions or to model negative contributions.

B. Visualization Challenge

Generating a SIG automatically from a set of requirements is clearly a non-trivial multi-step task. As depicted in Fig. 1, the first phase involves the modeling of quality concerns and the classification and retrieval of related requirements from the specification. In phase 2, topics are identified and attached as subgoals to their relevant quality concerns. Finally, in phase 3, requirements associated with each topic are organized into a lower level hierarchy, thereby generating a candidate SIG. In the following sections we discuss each of these steps in greater detail.

C. Scope the Goal Graph

SIGS can be used to represent a wide variety of quality concerns, therefore it is necessary to clearly specify which quality concerns are to be included in the generated graph. This is accomplished through creating a *SIG Scoping Model* which defines the uppermost nodes of the SIG. For purposes of our case study we focus on security-related goals extracted from the US 1996 Health Insurance Portability and Accountability Act (HIPAA) technical safeguards. These goals, which are depicted in the SIG Scoping Model shown in Fig. 3, focus primarily on access controls including user authentication, activity audits, time-out controls, and the encryption and decryption of critical data.

III. Identify Quality-Related Requirements

In the next step of the process, the requirements specification is searched for any requirement that is related to any of the quality concerns defined in the SIG Scoping Model. This is accomplished through manually creating an initial search query for each leaf node. For example, as depicted in Table II, the leaf node *audit control* representing goal $G1$ is augmented to read *audit control security*. To further improve the quality of the search query, we enhance it using query augmentation [23]. This provides two primary benefits. First it automatically identifies a more extensive set of search terms and secondly it weights each of those terms according to their importance for the search. We utilize a tool that we had previously developed to augment trace queries [23]. This tool (1) utilizes the initial search query and a selection of standard search engines (Google, Yahoo, Bing etc) to retrieve a set of relevant documents, (ii) filters the results to remove documents which are difficult to parse (e.g, image-only files), (iii) splits each document into chunks of overlapping text, (iv) uses the basic vector space model cosine similarity approach to identify the chunk most relevant to the search query, (v) identifies and extracts domain specific terms and phrases which serve as candidate indicator terms, (vi) computes relevance metrics for each term (or phrase), and finally (vii) uses the metrics to identify the most relevant terms to serve as quality-type indicators.

TABLE II
AUGMENTED GOAL DESCRIPTIONS USED TO TRAIN THE REQUIREMENTS CLASSIFIER AND THE FINAL QUERY TERMS

	Augmented Goal	**Terms (weights not shown)**
1	audit control security	secur , audit , system , secur audit , procedur , practic , govern , risk , center
2	automatic log off authorization	author , authent , payment , bank , comment , post , comput , login , click
3	emergency access procedure authorization	emerg, procedur, author, health, care, plan, provid, access procedur, secur
4	designate access user privilege	privileg, administr, account, permiss, role, databas, owner, tabl, click
5	restrict suspend access user privilege	privileg , account , restrict , administr , databas , right , agent , logonid
6	access attempt access control	attempt, access control, access attempt, connect, network, polici
7	role based access type	role, access control, secur, rbac, permiss, polici, type, resourc, server
8	context based access type	context , access control , secur , role , author , cbac , polici , cisco , decis
9	person authentication security	person, authent, secur, password, biometr, ident, method, person authent
10	integrity control transmission	secur , network , transmiss secur , integr control , transmiss integr , entiti , system , authent
11	encryption transmission	encrypt, data transmiss, secur, kei, transmiss encrypt, peer, connect, mac

Table II shows the top-ranking indicator terms which were discovered using query augmentation for each goal. Terms are displayed in their root forms. Applying this technique to each leaf node produces a modified description of each associated goal, represented as a set of weighted terms.

The weighted indicator terms for each goal g are then used to evaluate the likelihood ($Pr_g(r)$) that a given requirement r is associated with the goal. Let I_g be the set of indicator terms for goal g identified during the query augmentation phase. The classification score that requirement r is associated with goal

g is then defined as follows:

$$Pr_g(r) = \frac{\sum_{t \in r \cap I_g} Pr_g(t)}{\sum_{t \in I_q} Pr_g(t)} \quad (1)$$

where the numerator is computed as the sum of the term weights of all type g indicator terms that are contained in r, and the denominator is the sum of the term weights for all type g indicator terms. The probabilistic classifier for a given type g will assign a higher score $Pr_g(r)$ to requirement r that contains several strong indicator terms for g.

This classification step attaches a list of relevant requirements to each quality-related leaf node in the SIG Scoping Model.

IV. Create Hierarchy of Topics

Each set of requirements is then organized into a meaningful hierarchy of topics. This kind of topical breakdown is common in practice. For example, the *accuracy* goal in Fig. 2 is decomposed into topical subgoals related to *patient information* and *billing* – implying that there are accuracy concerns in these two functional areas of the system.

We adopt two separate clustering techniques. The first identifies dominant topics of relevance to a specific quality concern. It uses the Incremental Diffusive Clustering (IDC) technique which has been shown to produce highly cohesive clusters of requirements [18], [24]. However, while IDC produces high quality clusters, it does not place every requirement into a cluster. Therefore we implement a second lower-level clustering which creates a sub-hierarchy of topics underneath each of the IDC-identified topics. Each clustering technique requires its own naming process, and additional steps are needed to generate the complete SIG. Given the multi-step process, we follow a workflow of data-mining techniques.

A. Step 1: Identify primary topics

Dominant topics are identified through the use of IDC. This is an incremental process, in which a simple clustering technique, such as SPK-Means is used to cluster all of the requirements into k clusters. The number k can be determined either through trial-and-error (i.e. by observing the output to determine if the clusters are sufficiently cohesive and at an appropriate level of granularity), or as in our case, through applying an information-entropy based metric such as Can's Metric [24]. Following each round of SPK-Means clustering, the quality of each cluster is evaluated according to its size and cohesion – with the goal being to identify highly-cohesive, non-trivially sized clusters. The 'best' cluster is then selected, and its key terms are identified and removed from all requirements in the dataset. The IDC clustering process is then repeated against the reduced requirements. The process is repeated and one new topic (i.e. one cluster) is output following each iteration. A more complete description of IDC is provided in our prior work [24].

As the IDC clusters need to be displayed to the user, it is important to provide meaningful names for each topic. The common practice is either to substitute the most prominent terms (i.e. a bag of words) for the name, or to select a recurring phrase from the clustered requirements. Our approach builds upon the second option and involves first identifying the cluster's medoid, defined as the term or phrase which is most representative of all requirements in the cluster, and then removing quality-related terms from the medoid of each cluster. These are the terms from the initial search query such as the terms *encrypt*, data *transmiss*, *secur*, *kei*, taken from the *encryption* goal etc. Stop words, i.e. very common words such as *this* and *or* are also removed. The highest scoring remaining term is then used to seed the topic name. Each seed term is then traced back into the requirements of the cluster and its prior- and post terms are attached to the *medoid* terms. In the case of having multiple compound terms, the term with the highest frequency is selected. The final compound term selected for each of the encryption topics is shown in the fourth column of Table III. For example, topic # 25 is named *enhance security*.

B. Step 2: Attach Topics to SoftGoals

Topics are attached to a softgoal if the IDC generated cluster contains two or more requirements related to that softgoal. For example, in Table III, the topic related to *client-server* has four requirements classified as *encryption*-related, and is therefore attached to the encryption goal. In contrast, the *printed form* topic contains only a single requirement and therefore is not attached. We refer to this requirement as a singleton. Unattached requirements include all singletons, plus any quality-related requirements not placed into a cluster by IDC. We explored two different approaches for placement of these requirements into the hierarchy, and found that the CHITT dataset performed best with the first approach, and World Vista with the second. While our goal is to identify an approach which works consistently across all datasets, for now we report both techniques.

In the first approach, all unattached requirements were initially assigned to each of the two identified topics and the assignment decision was deferred for later in the process. In the second approach, Latent Dirichlet Allocation (LDA) was used to place each requirement into the topic with the most contextually similar requirements. LDA is a generative probabilistic model that discovers short descriptions (topics) for the members of a large collection [3]. It estimates the similarity between each requirement and the discovered topics and was therefore used to identify the best sub-goal placement for each of the unattached requirements. We implemented it using the Stanford Topic Modeling Toolbox with default parameters [28].

C. Step 3: Build a Sub-Hierarchy

At this stage in the workflow, we have high level topics assigned to each quality concern.

To create the lower part of the SIG, we used Agglomerate Hierarchical Clustering (AHC) because it explicitly constructs a hierarchy of topics. It works on the term vector representation of the requirements previously created and utilized

TABLE III
CLUSTERS IDENTIFIED FOR THE *encryption* GOAL IN WORLD VISTA. THE FIRST COLUMN DEPICTS THE MEDOID OF EACH CLUSTER. IN COLUMN 2 THESE ARE FILTERED TO REMOVE STOPWORDS AND SEARCH TERMS, AND IN COLUMN 3 THE HIGHEST SCORING REMAINING TERM IS EXPANDED (WHERE POSSIBLE) INTO A MORE DESCRIPTIVE PHRASE

ID	Topics extracted from IDC	Filtered	Compound Term	Count
16	client, server, broker, rpc, vist, connect'	client	client/server	4
27	entri, clerk, data'	entri	clinician data	2
13	'form, encount'	form	printed form	1
21	'site'	site	site-specific	1
22	'updat, demograph, patient'	updat	updated demographic	1
25	'secur, enhanc'	enhanc	enhance security	1
28	'mail, group'	mail	MailMan	1
35	'ad, ami, hoc'	ad	AD Hoc	1

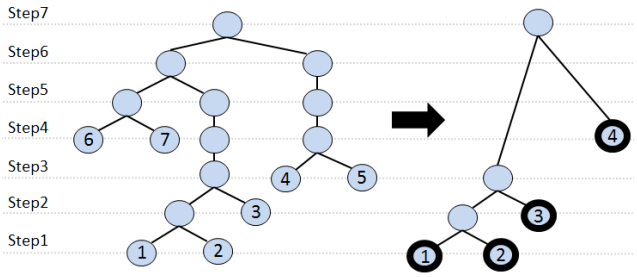

Fig. 4. Result of Agglomerate Hierarchical Clustering. Topic related requirements are found in clusters 1, 2, 3 and 4

for classification purposes. AHC is a bottom-up hierarchical approach that starts by placing each requirement into an individual cluster, and then in each subsequent iteration, merges the two clusters which are most similar using a standard similarity metric and a linkage method. We adopted Squared Euclidean Distance and Ward's method to link the most similar clusters. In Ward's method the two most similar clusters are those for which the merger result has the minimum increase in the error sum of squares [15].

Hierarchical clustering techniques produce dendograms in which clusters are connected at different levels of the hierarchy. Our goal is to create a partition of disjoint clusters to visualize a meaningful hierarchy of contexts. To determine the cutting point we selected the point at which the gap, measured by similarity scores, between two successive combinations of subtrees is the largest. In effect, splitting clusters any further would decrease their quality [7].

While it might intuitively make sense to apply AHC to only those requirements associated with each higher-level topic this is impractical when only a few requirements are involved. We therefore applied AHC to all of the requirements in the specification creating a complete hierarchy of topics. The nodes in the hierarchy which contain relevant requirements are then extracted from the general hierarchy and rejoined to form a proper subtree. This is illustrated in Fig. 4. The left hand side shows the complete hierarchy of WorldVista requirements, while the right hand side of the same figure

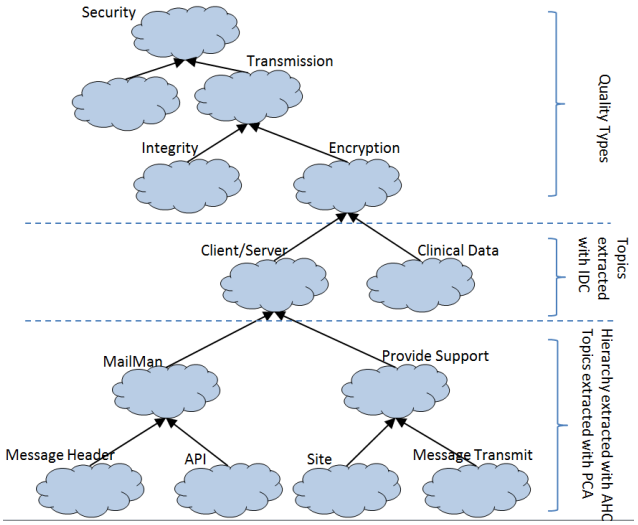

Fig. 5. Steps of Contextualizing Quality Concerns for Encryption subgoal of Transmission Security

represents the sub-hierarchy for a security concern subgoal *Transmission Integrity*. Its requirements are located in clusters 1, 2, 3 and 4. We prune any nodes which neither contain a requirement relevant to the topic, nor are located on a direct path from such a node to the root. Finally, any linear sequence of nodes without siblings are merged into a single node. In this manner the subset of relevant nodes are extracted from the complete hierarchy.

Once again, we need to label each of the newly identified nodes in a meaningful way. Unfortunately, the medoid approach does not work well on the smaller number of requirements assigned to each subnode, and therefore we adopted a different approach based on the use of Principal Component Analysis (PCA). PCA is a feature extraction method which transforms a set of features to a smaller set of attributes called Principal Components (PC). We applied PCA against the vector of terms representing each requirement in an individual cluster with the goal of extracting the underlying structure of the data [1]. While PCs are discovered incrementally, the first PC captures as much of the variability in the data as possible. Each succeeding component captures as much of the remaining variability as possible. Mathematically, this transformation is defined by a set of d-dimensional vectors of weights or *loadings* $w_{(i)} = (w_1, \ldots, w_p)_{(i)}$ that map each original feature $F_{(j)}$ of F to a new vector of principal component scores $PC_{(j)} = (PC_1, \ldots, PC_p)_{(j)}$. Below is the general formula for the first PC:

$$PC_1 = w_{11}(F_1) + w_{12}(F_2) + \cdots + w_{1p}(F_p) \qquad (2)$$

The loadings represent how much the original feature has contributed in forming the related PC. The larger the loading, the greater the contribution of the feature in forming the associated PC. For example, w_{11} in equation 2 represents the contribution of the original term F_1 in forming PC_1.

We applied PCA to the set of requirements in the cluster, identified the first PC, and then selected the terms with the

TABLE IV
RECALL AND PRECISION ACHIEVED FOR REQUIREMENTS CATEGORIZATION

Goal	Name	Count	DQM			Augmented			Basic		
			Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
G1	audit control	26	0.75	0.58	0.65	0.72	0.50	0.59	0.25	1.00	0.40
G2	automatic log off	2	0.50	0.50	0.50	0.77	1.00	0.87	0.72	0.50	0.59
G3	emergency access	6	1.00	0.83	0.91	1.00	0.50	0.67	0.00	0.00	0.00
G4	designate access	1	1.00	1.00	1.00	0.50	1.00	0.67	1.00	1.00	1.00
G5	suspend access	9	0.71	0.56	0.63	0.20	0.25	0.22	0.50	0.50	0.50
G6	access attempt	1	1.00	1.00	1.00	0.63	1.00	0.77	0.83	1.00	0.91
G7	role based access	7	1.00	1.00	1.00	0.71	0.57	0.63	0.50	0.14	0.22
G8	context based access	2	0.50	1.00	0.67	0.11	0.50	0.18	0.00	0.00	0.00
G9	integrity control	7	0.57	0.71	0.63	0.50	0.28	0.36	0.50	0.28	0.36
G10	encryption transmission	4	0.25	0.25	0.25	0.33	0.50	0.40	0.25	0.50	0.33

largest loading. For instance, the PC vector below represents the first PC for the context *Message Header* located in Fig. 5.

$PC_1 = 0.598header + 0.276includes + 0.187generate...$

As shown above, the term associated with the largest loading is *header*. In order to retain the meaning of compound terms, we search for the selected term in the requirements description and return the phrase in which the term most frequently appears.

Furthermore, in our first approach for placement of unassigned requirements, we place a requirement into a topic if the terms in that requirement contribute to the first PC.

D. Step 4: Put it all together

Finally, the SIG is constructed by attaching topics to the lowest level of soft goals, and then by attaching the hierarchy of relevant requirements nodes under each of the topics. The SIG structure is generated as an XML file which can then be read into a SIG graphing tool. We have developed a prototype tool for this purpose.

V. EVALUATION

For evaluation purposes we conducted two different experiments. The first comparatively evaluated three classification techniques for retrieving requirements related to each of the specified soft goals, while the second evaluated the quality of the generated SIG itself.

A. Connecting Softgoals to Requirements

For purposes of the first experiment we constructed an 'answer set' for the CCHIT data. This answer set consisted of 235 requirements categorized according to quality concerns specified in Fig. 3.

1) Experimental Design: Three different query types were used to classify requirements. The *basic* approach used the initial terms describing each goal. The *augmented* approach used the web-based search technique to augment the query. We also compared a third more interactive approach in which the user built the query iteratively as he/she reviewed the results. This approach is aligned with a technique referred to as Direct Query Manipulation (DQM) which has been used effectively for trace retrieval purposes [33]. While this third approach requires human interaction, it is highly appropriate in a context for which it makes sense for the user to interact with, and refine the search results.

2) Results: Results from this experiment are reported using standard metrics of *recall* (i.e. the fraction of correct requirements which were retrieved), *precision* (i.e. the fraction of retrieved requirements which are correct), and F-Measure which computes the harmonic mean of recall and precision. These results show that DQM performed the best with a mean F-Measure of 0.72, compared to 0.54 for the augmented search query, and 0.43 for the basic approach. Furthermore, DQM returned the best results in 8 out of 10 cases (tying with basic in one case). Comparing the two fully automated approaches, Augmented outperformed Basic in six cases, and tied in one case. These results suggest that the initial search could be performed using the augmented search query provided by the user; however, if the user is not satisfied with the results she could use DQM to manually modify the augmented query.

B. Evaluating the Generated SIGs

The second experiment evaluated the quality of the SIGs generated for the CCHIT and WorldVista requirements. We first performed a preliminary analysis to determine the feasibility of manually creating SIGs that could serve as answer sets for evaluation purposes. However, it became obvious that there were many different viable ways to decompose the quality concerns and topics, and that no single *ground truth* model existed. This observation has also been made for constructing and evaluating graphs of feature models [13]. Therefore, the feasibility of evaluating our approach against a ground truth model was rejected. As a result, we conducted a qualitative evaluation.

C. Qualitative Evaluation

To evaluate the quality of the generated SIG we asked eight software engineers from the Professional MS Software Engineering program at DePaul, to assess the quality of the models. A SIG was generated for each of the datasets using the SIG Scoping Model depicted in Fig. 3. Given the size of each dataset, the number of nodes in the generated SIG, and the number of quality-related requirements attached to each node, we used only a sample of requirements and generated nodes for the evaluation.

TABLE V
SAMPLE OF THE QUALITATIVE EVALUATION

Goal Hierarchy	Requirement Specification	Evaluation
Audit Control>Patients Record>Audit Record>	The system shall require documentation of the audit support functionality in the vendor provided user guides and other support documentation, including how to identify and retrospectively reconstruct all data elements in the audit log including date, time.	All match
Authorization>Access Control>Restrict, Suspend Access>Standard base>Designated Patient's Chart>	The system, prior to a user login, shall display a (configurable) notice warning (e.g. The system should only be accessed by authorized users).	Only related to the sub-goal
Authorization>Automatic Logoff>HIR>Medical Record>	The system shall support medical record entries that are dated, the author identified and when necessary according to law or regulation or hospital policy, authenticated, either by written signature, electronic signature or computer key.	Only related to the project-specific context

TABLE VI
RESULTS OF QUALITATIVE EVALUATION

	CCHIT	Unrelated	WorldVista	Unrelated
Not Related at all	0.11	0.68	0.08	0.37
Only related to the sub-goal	0.27	0.12	0.10	0.12
Only related to the project-specific context	0.16	0.04	0.05	0.06
All match	0.46	0.16	0.76	0.44

However, in order to provide coverage of all goal types, requirements assigned to nodes were selected using *random stratified sampling* in which samples were randomly taken from each node in a number proportional to the size of the group and independent from each other. Furthermore, in order to test for participant bias, to minimize the impact of the lack of domain knowledge, and to proactively encourage users to intelligently evaluate the results, we included a set of randomly selected unrelated requirements into the SIG. The participants were informed that some percentage (unspecified) of the requirements had been assigned randomly to nodes.

Prior to the study, one of the researchers gave a one hour presentation about SIGs and quality related requirements, in order to familiarize participants with the approach. Participants were then randomly assigned to either the CCHIT or World-Vista group, such that four people evaluated each goal model.

Each participant was given a copy of one of the goal models and its associated sample requirements and was asked to rate the correctness of the requirements placement in the model, and the relevant hierarchy based on the following criteria:

1) *Not Related at all*, The Requirement is misclassified. It neither belongs to the security sub-goal nor is relevant to the context. (Score = 1).
2) *Related to the sub-goal* The requirement is correctly classified under a security subgoal but does not relate to the project-specific context. (Score = 2).
3) *Related to the project-specific context*, The requirement is correctly assigned to a topic in the project-specific context, but does not relate to the security sub-goal. (Score = 3).
4) *Complete match*, The requirement is correctly placed under an appropriate sub-goal and is also correctly related to the project-specific context. (Score = 4).

Table V presents three detailed examples of extracted sub-goals. The first example shows a requirement which is correctly classified as *Audit Control* and also correctly contextualized around the domain concept of *Patient Record*. The second example shows a requirement which is correctly contextualized within its system function but is misclassified under a wrong security sub-goal. Lastly, the third example shows a correct classification under the *Restrict, Suspend Sub-Goal* but which is not presented within a meaningful project-level context.

D. Analysis of Results

An analysis of the results showed that our approach performed better for the World Vista dataset than for CCHIT. In the case of WorldVista, 76% of the generated nodes were graded as correctly placed, but 44% of the unrelated nodes were also graded this way. In the case of CHITT the ratio was 46% vs. 16%. Furthermore, in CCHIT an additional 43% of requirements were placed only partially correctly, with 16% for WorldVista.

One explanation for these less than stellar results is the fact that our SIG Scoping Model includes closely related quality concerns, making it difficult to decide which node a given requirement is most closely related to. For example, a requirement describing role-based access control to restrict access to a specific functional area could equally well fit under nodes 4 or 7 in the SIG Scoping Model depicted in Fig. 3. This suggests that future attempts to generate SIGS should focus on coarser grained categories of concerns.

In conclusion, while our approach performs better than the random baseline in both cases, the generated SIG is far from perfect and not yet at a sufficient quality for practical use. Nevertheless, we present our results as an initial baseline for further work in the area.

VI. Applications

Earlier in the paper we made several claims concerning the benefits that SIG visualization, if achieved successfully, could provide for various Software Engineering activities. In this section we briefly support these claims using illustrations from the SIG generated for the WorldVista project.

A. Understanding quality related concerns

Understanding the specified non-functional requirements of the system is critical for downstream design purposes and also for compliance verification. We illustrate this in Fig. 6 with a visualization that depicts the extent to which each soft goal in the upper part of the SIG is covered by requirements. In this visualization we annotate each goal node with the count of associated requirements, and use a shading scheme which depicts nodes with more requirements in darker shades and those with fewer requirements in lighter shades. Nodes with no identified coverage are shown with dashed lines. Tool support could allow an analyst to explore the lower levels of the SIG to see the expanded hierarchy of these requirements. Such views enable an analyst to answer questions such as "to what extent is quality concern Q specified in the requirements?" or "Which parts of the system are affected by quality Q?" In many projects, for which requirements are listed only as flat files, this kind of perspective is only feasible if the goal view is generated automatically.

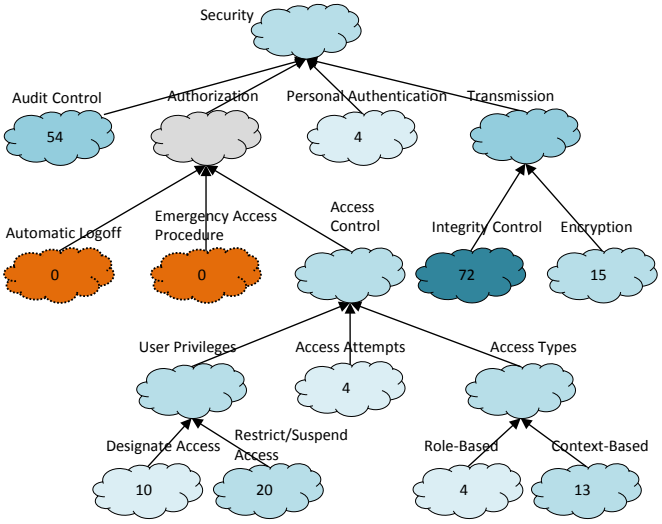

Fig. 6. A Visualization of the Overall Security Coverage for World Vista

B. Identifying Underspecified Concerns

In Figure 6 two of the concerns, namely *automatic logoff* and *emergency access procedures* are clearly underspecified with no associated requirements. An analyst could explore this more closely. First, he might modify the initial search terms for the goal, either manually, or through requesting a query augmentation, and then reexecute the search. If no additional requirements are retrieved, then the analyst would need to investigate whether the goal is relevant or not, and if not, whether missing requirements should be specified. In the case of World Vista, both of the underspecified goals are required by HIPAA and therefore likely should be specified as requirements.

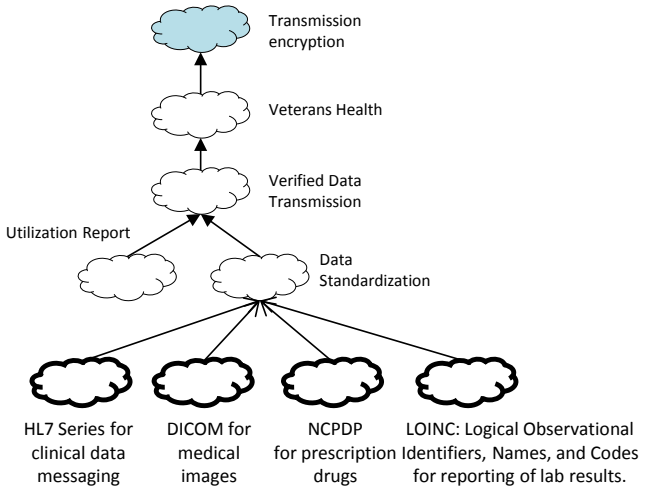

Fig. 7. Attaching candidate design decisions to the goal graph. These candidate operationalizations are shown with bold borders at the leaf nodes of the SIG.

C. Driving design and assessment

Goal graphs explicitly model softgoals and therefore provide the opportunity for proposing and evaluating various design decisions. For illustrative purposes consider the goal to achieve *verified data transmission*. This goal contributes to achieving veterans health and privacy through transmission encryption. The generated goal graph provides a visual environment in which to reason about design solutions. Four candidate solutions are shown for consideration in the SIG. These propose various standards for transmitting general health information, medical images, prescription drug information, and laboratory results. These candidate operationalizations can be manually added to the automatically generated goal graph as depicted in Figure 7 and used to reason about the contribution of various options.

D. Supporting Change Impact Analysis

It is well documented that long-term changes to code often result in inadvertent degradation of quality concerns [20]. Reverse engineering the SIG from requirements and mapping quality concerns to various functional topics in the system introduces the potential for providing information about quality concerns during change impact analysis. For example, we can extract information from the generated SIG such that *blood bank* has role-based access requirements or that *clinician entries* must be encrypted before transmission. When new requirements are introduced to a part of the system, the SIG provides the means of identifying relevant quality concerns.

E. Communicating with Customers

Much software development effort occurs in existing systems. Customers request new features and/or modifications

to existing ones. By reverse engineering a SIG we are able to discuss new features in light of existing quality concerns, thereby increasing the likelihood of delivering a satisfactory product.

VII. Related Work

There have been several attempts to develop visualization techniques for quality concerns. These techniques range from simple tabular and relational formats to hierarchical, quantitative, and metaphorical approaches [10]. The majority of these approaches focus on visual notations for capturing concerns, and for presenting relationships between different aspects. However, there is limited work on automated extraction and visualization of software requirements, especially related to non-functional qualities.

Goal-oriented approaches capture and visualize non-functional concerns [19]. Such techniques include the soft-goal interdependency model [8] , the i* framework [37], and KAOS [36] as well as more general approaches based upon visualizing quality concerns across UML diagrams [35], [11].

However, such approaches tend to focus on capturing requirements knowledge without providing automated support. Some exceptions are a visual modeling tool developed by Farid and Mitropoulos [21], which provided a semi-automated approach for capturing, visualizing, and reasoning about quality concerns in agile projects. They used a modified form of the SIG to assist agile developers during the requirements gathering and analysis phase.

ConcernMapper [31] and AoGRL [26] tools provide visualization support for aspect oriented software development. These tools attempt to depict the way in which cross-cutting concerns, such as security, logging, etc., are scattered across functional parts of a software system. However such approaches are not fully automated and the visualizations fail to provide detailed information about interactions between concerns and functional features. EAMiner, [32] utilizes natural language processing techniques, to mine various types of cross cutting concerns from requirements documents. The output is a flat modeling of the early aspects found in specification documents, and is therefore akin to the initial part of our process in which we retrieve a list of requirements associated with a specific goal.

Several researchers have utilized information retrieval techniques to extract quality concerns from software documents [34], [25], [9]; however, such approaches only focus on detecting and retrieving quality concerns and, like EA-Miner, return the requirements in the form of a flat list. In contrast our approach attempts to create a hierarchy of goals driven the the functionality of the system in which the goals are applied.

Finally, researchers [17], [27] have also utilized clustering algorithms to discover key functional features in software documents and then to visualize the requirements and their dependencies. These works have mainly focused on functional requirements and have not explored the visualization of quality concerns. Similarly Delfosse et al., [14] automated the visualization of a feature model for a software domain by first discovering the domain features through clustering hundreds of partial product descriptions mined from SoftPedia and then using association rule mining and inferencing to generate the goal tree. The unique characteristic of their approach is the fact that the feature model was generated from thousands of different systems, enabling the use of techniques which are not applicable in a single project. Furthermore, their approach focused on functionality and generated a feature model used traditionally for product-line development, whereas the approach reported in this paper focuses on quality concerns.

VIII. Threats to Validity

Threats to validity are classified as internal, construct and external validity.

Internal validity reflects the extent to which the bias or error has been reduced in the method so a causal relationship can be concluded from the study. In our work, in the evaluation process of the goal hierarchy, it was difficult to totally separate out the task of evaluating the replacement of requirements, with the quality of the associated topics generated for them. For instance, if we asked a user to evaluate whether the hierarchy of context $C1, C2, \ldots$ was meaningful for requirement R, then we might bias the participant with both the essence of the topic for the related requirement(i.e., what the requirement truly represents) as well as the selected terms for the topic. In order to mitigate the bias, we embedded unrelated requirements into several of the nodes.

Construct validity discusses the quality of choices for the dependent and independent variables and evaluates the extent to which the construct that was intended to be measured was actually measured. In qualitative evaluation, although the participants have been selected from experts in software engineering they were familiar with, yet not experts in, the health care domain. On the other hand, the provided requirements were self-explanatory even to a non-expert. We minimized this error by providing a one hour introduction to SIGs showing some of the concepts from the domain. In future work we will repeat the evaluation using experts from the domain.

External validity is the generalizability of the approach. Although, we applied our method to two different large health care related systems exhibiting different characteristics of size and quality concerns, we will still need to extend our evaluation into other domains, and indeed into more projects within the current domain before we can claim generalizability.

IX. Conclusion

In this paper we have proposed an approach for automatically extracting a goal model from a requirements specification. The specific quality concerns of interest are defined manually by the analyst. We then implemented a sequence of machine learning and data mining methods to automatically detect quality concerns and to organize them into a meaningful hierarchy of topics.

Our evaluation demonstrated the viability of identifying and attaching requirements to quality concerns. However, we achieved mixed results in generating a meaningful hierarchy

of well-named goals and sub-goals. Further work is clearly needed to fine-tune our techniques so that each node of the graph is appropriately named, and that all functional areas of the system impacted by a specific quality concern are faithfully depicted in a well-structured hierarchy. We therefore plan to explore alternate goal decompositions and naming schemes in order to further increase the understandability and completeness of the generated SIG. Finally, we need to explore SIG generation across domains other than the healthcare industry, and explore a different and broader set of topics and quality concerns. We consider the solutions presented in this paper to be a baseline for future approaches.

Finally, we plan to extend our initial prototyping tool in order to create a more interactive environment in which the user can evaluate quality concerns for a project.

ACKNOWLEDGMENTS

This work was partially funded under National Science Foundation grant CCF-1218303.

REFERENCES

[1] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[2] D. Ameller, C. P. Ayala, J. Cabot, and X. Franch. Non-functional requirements in architectural decision making. *IEEE Software*, 30(2):61–67, 2013.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] T. D. Breaux and A. Rao. Formal analysis of privacy requirements specifications for multi-tier applications. In *RE*, pages 14–20, 2013.

[5] Certification Commission for Healthcare Information Technology. Guidelines for electronic healthcare records. http://cchit.org.

[6] L. Chen, M. A. Babar, and B. Nuseibeh. Characterizing architecturally significant requirements. *IEEE Software*, 30(2):38–45, 2013.

[7] H. S. Christopher D. Manning, Prabhakar Raghavan. Introduction to information retrieval. 1, 2008.

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

[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *RE*, pages 36–45, 2006.

[10] J. Cooper, S.-W. Lee, R. Gandhi, and O. Gotel. Requirements engineering visualization: A survey on the state-of-the-art. In *Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on*, pages 46–55, Sept 2009.

[11] L. Cysneiros and J. Sampaio do Prado Leite. Nonfunctional requirements: from elicitation to conceptual models. *Software Engineering, IEEE Transactions on*, 30(5):328–350, May 2004.

[12] L. M. Cysneiros and J. C. S. do Prado Leite. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Trans. Software Eng.*, 30(5):328–350, 2004.

[13] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *ESEC/SIGSOFT FSE*, pages 290–300, 2013.

[14] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 290–300, New York, NY, USA, 2013. ACM.

[15] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[16] Department of Veterans Affairs Office of Enterprise Development. vistahealthvet monograph.

[17] C. Duan and J. Cleland-Huang. Clustering support for automated tracing. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 244–253, New York, NY, USA, 2007. ACM.

[18] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *ICSE*, pages 181–190, 2011.

[19] N. Ernst, Y. Yu, and J. Mylopoulos. Visualizing non-functional requirements. In *Requirements Engineering Visualization, 2006. REV '06. First International Workshop on*, pages 2–2, Sept 2006.

[20] D. Falessi, L. C. Briand, G. Cantone, R. Capilla, and P. Kruchten. The value of design rationale information. *ACM Trans. Softw. Eng. Methodol.*, 22(3):21:1–21:32, July 2013.

[21] W. Farid and F. Mitropoulos. Normatic: A visual tool for modeling non-functional requirements in agile processes. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–8, March 2012.

[22] A. Finnegan, F. McCaffery, and G. Coleman. Development of a process assessment model for assessing security of it networks incorporating medical devices against iso/iec 15026-4. In *HEALTHINF*, pages 250–255, 2013.

[23] M. Gibiec, A. Czauderna, and J. Cleland-Huang. Towards mining replacement queries for hard-to-retrieve traces. In *ASE*, pages 245–254, 2010.

[24] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Trans. Software Eng.*, 39(12):1736–1752, 2013.

[25] I. Hussain, L. Kosseim, and O. Ormandjieva. Using linguistic knowledge to classify non-functional requirements in srs documents. In *Proceedings of the 13th International Conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems*, NLDB '08, pages 287–298, Berlin, Heidelberg, 2008. Springer-Verlag.

[26] G. Mussbacher, D. Amyot, J. Araujo, A. Moreira, and M. Weiss. Visualizing aspect-oriented goal models with aogrl. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 1–1, Oct 2007.

[27] N. Niu and S. Easterbrook. On-demand cluster analysis for product line functional requirements. In *Proceedings of the 2008 12th International Software Product Line Conference*, SPLC '08, pages 87–96, Washington, DC, USA, 2008. IEEE Computer Society.

[28] D. Ramage. The stanford topic modeling toolbox, Sept. 2009. http://nlp.stanford.edu.

[29] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang. Mind the gap: Assessing the conformance of software traceability to relevant guidelines. In *36th International Conference on Software Engineering (ICSE)*, 2014.

[30] S. Robertson and J. Robertson. *Mastering the Requirements Process*. Addison-Wesley, 2013.

[31] M. Robillard and G. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 406–416, May 2002.

[32] A. Sampaio and A. Rashid. Mining early aspects from requirements with ea-miner. In *Companion of the 30th International Conference on Software Engineering*, ICSE Companion '08, pages 911–912, New York, NY, USA, 2008. ACM.

[33] Y. Shin and J. Cleland-Huang. A comparative evaluation of two user feedback techniques for requirements trace retrieval. In *SAC*, pages 1069–1074, 2012.

[34] J. Slankas and L. Williams. Automated extraction of non-functional requirements in available documentation. In *Natural Language Analysis in Software Engineering (NaturaLiSE), 2013 1st International Workshop on*, pages 9–16, May 2013.

[35] S. Supakkul and L. Chung. Visualizing non-functional requirements patterns. In *Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on*, pages 25–34, Sept 2010.

[36] A. van Lamsweerde. Goal-oriented requirements enginering: A roundtrip from research to practice. In *RE*, pages 4–7, 2004.

[37] E. S. K. Yu. Social modeling and i*. In *Conceptual Modeling: Foundations and Applications*, pages 99–121, 2009.