2010 18th IEEE International Requirements Engineering Conference

An NFR Pattern Approach to Dealing with NFRs

Sam Supakkul, Tom Hill, and Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
ssupakkul@ieee.org,
tom.hill.fellow@gmail.com,
chung@utdallas.edu

Thein Than Tun
Department of Computing
The Open University
Milton Keynes, UK
t.t.tun@open.ac.uk

Julio Cesar Sampaio do Prado Leite
Departamento de Informática
Pontifícia Universidade Católica
do Rio de Janeiro, Brazil
www.inf.puc-rio.br/~julio

***Abstract*—Non-functional requirements (NFRs), such as security and cost, are generally subjective and oftentimes synergistic or conflicting with each other. Properly dealing with such NFRs requires a large body of knowledge – goals to be achieved, problems or obstacles to be avoided, alternative solutions to mitigate the problems, and the best compromising alternative solution to be selected. However, few patterns exist for dealing with these kinds of knowledge of NFRs. In this paper, we present four kinds of NFR patterns for capturing and reusing knowledge of NFRs – *objective pattern*, *problem pattern*, *alternatives pattern* and *selection pattern*. NFR patterns may be visually represented, and organized by rules of *specialization* to create more specific patterns, of *composition* to build larger patterns, and of *instantiation* to create new patterns using existing patterns as templates. We have applied the NFR pattern approach to the TJX incident, one of the largest credit card theft in history, as a realistic case study.**

***Keywords*-non-functional requirements, requirements patterns, goal-oriented**

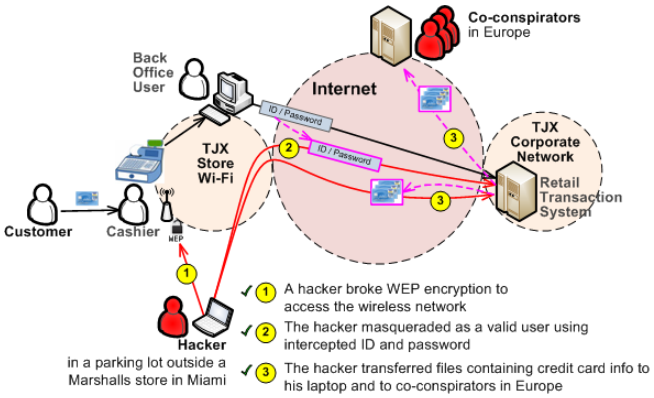

Figure 1. The TJX Incident

I. Introduction

Non-functional requirements (NFRs), such as security, cost and trustworthiness, are oftentimes subjective and interacting with each other in a synergistic or conflicting manner. Properly dealing with NFRs requires a large body of knowledge, including the knowledge of goals to be achieved, problems or obstacles to be avoided, alternative means to achieve the goals, solutions to mitigate the problems, and the best alternative solutions to be selected, as well as the knowledge of alternative mappings of selected means or solutions to functional requirements and specifications.

Although pattern-based approaches have been popular for capturing software knowledge, they have not been adequately extended for the large body of knowledge of NFRs. Instead, the knowledge of NFRs by and large has been dealt with implicitly and informally, or only in a limited way.

In this paper, we present four kinds of NFR patterns for capturing and reusing knowledge of NFRs – *objective pattern*, *problem pattern*, *alternatives pattern* and *selection pattern*. Objective patterns are used to capture the definition of NFRs in terms of specific (soft)goals to be achieved. Knowledge of (soft)problems or obstacles to achieving the goals can be captured by problem patterns. Alternatives patterns are used to capture different means, solutions, and requirements mappings, in consideration of the side-effects, so that the best alternative may be selected using either a quantitative or qualitative selection pattern. NFR patterns may be visually represented, but internally encoded using refinement rules to aid model re-generation during reuse in a tool-assisted environment. NFR patterns may be specialized to create more specific patterns, composed to build larger patterns, or instantiated to create occurrence patterns using existing patterns as templates.

We have applied the NFR Pattern approach to the TJX incident, one of the largest credit card theft in history, as a realistic case study. We feel that this study has shown that this approach could help capture and reuse knowledge of NFRs in a more structured and systematic way.

The rest of the paper is organized as follows. Section II briefly describes the TJX incident. The NFR Pattern approach is presented in sections III and IV. Section V presents an empirical study based on the TJX incident. Related works are discussed in Sec. VI. At the end, we summarize the paper and point out some future work.

1090-705X/10 $26.00 © 2010 IEEE
DOI 10.1109/RE.2010.31

II. The TJX Incident

TJX, the holding company of many large retail chains, including Marshalls and TJ Maxx, suffered one of the largest credit card thefts in history that was initially estimated to have affected 45 million credit cards, compromising the identity information of 451,000 customers, involving fraudulent transactions of $20 million, and to cost TJX $1 billion over 5 years, excluding lawsuits [1]. Although the company implemented several security measures, such as encryption and user authentication, it lacked the knowledge to prevent the security breach.

We conducted a case study based on the TJX incident to study how the NFR Pattern approach presented in this paper may be used to capture and reuse knowledge from the case and the literature in a similar situation or project.

Figure 1 illustrates a simplified likely attack scenario derived from the publicly available sources [1][2][3] where an attacker first broke into a local store's wireless network. He then intercepted ID's and passwords that the store personnel used to login to a corporate server. Using the stolen ID's and passwords, the attacker masqueraded as a valid user to access the server to download and transfer to his conspirators the files containing millions of credit card and customer identity records. The unauthorized server access attack in this scenario is used as a running example in this paper.

III. NFR Patterns

NFRs, such as security and trustworthiness, are generally treated as (soft) goals or objectives to be achieved [4][5]. For example, a US law on information security defines security as confidentiality, integrity, and availability objectives to be achieved by protecting assets from problems such as unauthorized access, use, and disclosure [6]. Because of the oftentimes conflicting nature of NFRs, their achievement also requires exploring alternatives and making acceptable trade-offs. For example, ID/password provides acceptable security but is highly cost-effective and practical for e-commerce applications. On the other hand, although biometric authentication is generally expensive and difficult to maintain, it is often employed in sensitive or military applications.

Based on this general treatment of NFRs, we define objective pattern, problem pattern, alternatives pattern, and selection pattern to help capture the needed knowledge.

Patterns can be recorded in many forms. While literature-based patterns are useful, but their reuse would require manual re-creation of models during requirements engineering, which can be time consuming and error prone. NFR patterns presented in in paper also capture model refinement rules that can be applied by a supporting tool to automatically refine a target model.

The refinement rules capture model refinements that are manually performed during a typical model-based requirements engineering session, where an engineer refines a model by incrementally adding a new model element and linking it to an existing one. For example, to refine an organizational resource [7], such as credit card, the engineer may identify security as an important softgoal for the resource (refinement 1). She may then refine security as confidentiality, integrity, and availability softgoals (refinement 2). She may also identify trustworthiness and cost as important security related softgoals to record the stakeholders' needs that the security achievement should also be cost-effective and trustworthy (refinement 3 and 4).

Such knowledge of credit card security may be captured as a pattern as shown in Fig. 2. In this pattern, *Result* captures the available knowledge about credit card security; *Initial* captures a key concept that provides the context for reuse; and *Refinement rules* capture individual model refinements and their application orders. R1 refinement rule in this example is a root rule that has no dependency. It is to be first applied to refine Credit Card. R2, R3, and R4 can then be applied independently, as they only depend on R1, to refine the result produced by R1.

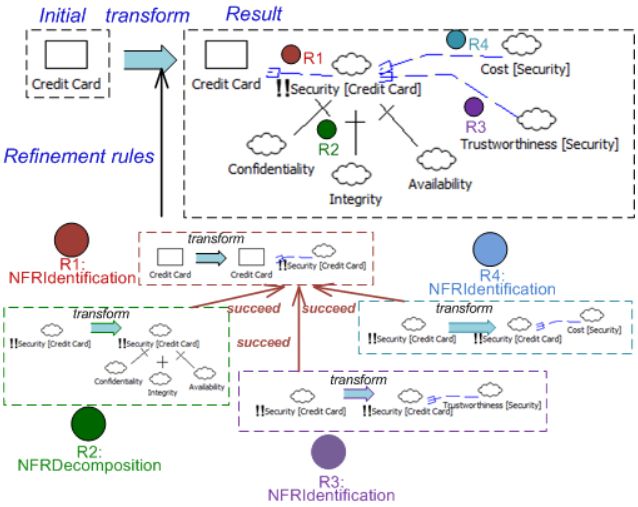

Figure 2. An NFR pattern and its application

In addition to the captured knowledge, each NFR pattern also consists of a unique name, and optional supplemental applicability and credential information, as shown in Fig. 3. The applicability information helps determine the appropriateness of the pattern while the credential information provides social-based confidence when choosing patterns.

The applicability information is adapted from the quality-based software reuse approach [8] that could help answer the 5W2H questions: who, what, why, when, where, how, and how much. Many uses of the applicability information are possible to answer the 5W2H questions. For example, *who* attribute can be used to help identify the target application domain of the pattern (e.g. "retail" in this example); *what*, contextual information of a given "who" that refers to

"topic" in the NFR Framework [8] (e.g. "information"); *why*, the rationale for knowledge that refers to "type" in the NFR Framework [8] (e.g. "security"); *when*, the time to apply the pattern, which can refer to the software artifact to which this pattern is applicable (e.g. "world", "requirements", "specifications", "program", or "machine" artifacts as defined by the reference model for requirements and specification [9]); *where*, a specific type of artifact in the context of "when" (e.g. "architecture" or "design" for "program" when the pattern captures knowledge developers' NFRs [10], such as maintainability); *how*, how the pattern may be applied (e.g. "automated", "semi-automated", "manual") to indicate whether the pattern consists of complete refinement rules for fully automated application, partial refinement rules for semi-automated application, or no refinement rule for manual application; and *how much*, the implication of the knowledge (e.g. "regulation" for knowledge with legal implications, "standard" for knowledge related to industrial or governmental standards, or "policy" knowledge pertaining to organizational policies).

The credential information is adapted from design patterns [11] and the reputation systems [12] to provide social-based confidence on the pattern that may be judged by the credibility of the authors, sources of knowledge, endorsements or opinions. or known uses of the knowledge or pattern.

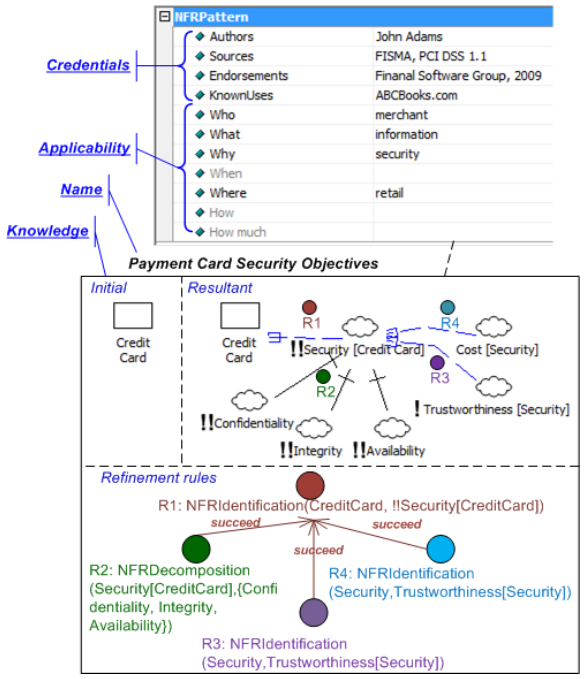

Figure 3. Content of an NFR pattern

A. Objective Pattern

An objective pattern can be used to identify important NFRs for a context or capture a specific definition of an NFR as different stakeholders may have different interpretations for an NFR. For example, the objective pattern in Fig. 3 identifies security as an important NFR for credit card, as well as captures a different of security in terms of confidentiality, integrity, and availability as defined by a US law [6]. A different objective pattern may capture a definition of security from the view of Payment Card Industry (PCI) that is only concerned with the confidentiality aspect in its Data Security Standard [13].

The refinement rules defined for objective patterns include: (i) *NFRIdentification*, a rule that identifies an NFR softgoal in the context of a given NFR "topic" [4] ; and (ii) *NFRDecomposition*, a rule that refines an NFR softgoal with an AND/OR decomposition to more specific sub-goals.

Using Fig. 3 as an example, *R1:NFRIdentification* rule is a root rule that associates Security softgoal with Credit Card "topic", while *R2:NFRDecomposition*, *R3:NFRIdentification*, and *R4:NFRIdentification* rules respectively refine Security and associate trustworthiness and cost with Security softgoals. Criticality of NFRs are represented by "!!" for "very critical", "!" for "critical", and no marking for "neutrally critical" [4].

B. Problem Pattern

A problem pattern can be used to capture knowledge of soft-problems (problems without clear-cut resolution criteria [14]) or obstacles to achieving an NFR softgoal or one of its operationalizations. Using the Resultant section in Fig. 4 as an example, *Unauthorized access [Server]* undesirable situation is a soft-problem that can break (–) *Confidentiality [Credit Card]* softgoal in the TJX case is. It is caused by an *Masquerading user login* attack or operational threat, which is composed of *Intercepting user IDs and passwords* and *Login using stolen ID/password* threats, where the former threat was made possible by *Transmission of ID/password in clear text* vulnerability.

The refinement rules defined for problem patterns include: (i) *ProblemIdentification*, a rule that associates a soft-problem to a context; (ii) *NFRObstacle*, a rule that identifies an undesirable situation that hurts or breaks an NFR softgoal; (iii) *OpSoftgoalObstacle*, a rule that identifies an undesirable situation that hurts or breaks an operationalizing softgoal; (iv) *UndesirableSitDecomp*, a rule that refines an undesirable situation with an AND/OR decomposition; (v) *UndesirableSitCausalAttribution*, a rule that identifies an operational threat that helps or makes an undesirable situation; (vi) *ThreatDecomp*, a rule that refines an operational threat with an AND/OR decomposition; (vii) *ThreatCausalAttribution*, a rule that identifies a vulnerability that helps or makes an operational threat; and (viii) *VulnerabilityDecomp*, a rule that refines a vulnerability with an AND/OR decomposition.

In Fig. 4, *R1:NFRObstacle* is a root rule that is first applied to identify *Unauthorized access [Server]* as a soft-problem that breaks(–) *Confidentiality [Credit Card]*

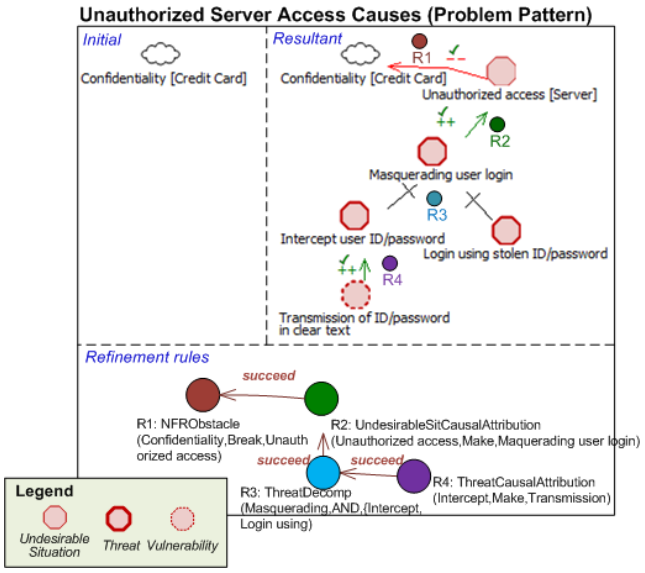

Figure 4. A problem pattern

softgoal. *R2:UndesirableSitCausalAttribution* rule is then applied to identify *Masquerade user login* as an operational threat that makes or causes *Unauthorized access* undesirable situation. *R3:ThreatDecomp* is then applied to AND-decompose the threat to *Intercept user ID/password* and *Login using stolen ID/password* threats. Finally, *R4:ThreatCausalAttribution* is then applied to identify *Transmission ID/password in clear text* as a vulnerability that makes or leads to *Intercept user ID/password* operational threat.

C. Alternatives Pattern

Knowledge of alternative means for achieving a softgoal can be captured in an alternative means pattern, while knowledge of alternative solutions for mitigating a soft-problem can be captured in an alternative solution pattern. Similarly, knowledge of alternative mappings of means or solution to requirement models can be captured in alternative requirements pattern. Each alternative may also be captured along with one or more side-effects, or NFR softgoals that are impacted positively or negatively by the alternative.

The refinement rules for alternative means patterns include: (i) *NFRSatisficing*, a rule that identifies an operationalizing softgoal as a means to help or make an NFR softgoal; (ii) *OperationalizationSatisficing*, a rule that identifies an operationalizing softgoal as a means to help or make another operationalizing softgoal; (iii) *OperationalizationDecomp*, a rule that refines an operationalizing softgoal with an AND/OR decomposition; and (iv) *SideEffect*, a rule that identifies an NFR softgoal that is affected by an alternative.

The refinement rules defined for alternative solutions patterns include: (i) *RiskTransfer*, a rule that identifies an operationalizing softgoal as a solutions to hurt or break a vulnerability soft-problem; (ii) *ThreatPrevention*, a rule that identifies an operationalizing softgoal as a solution to hurt or break an operational threat; (iii) *ThreatContainment*, a rule that identifies an operationalizing softgoal that hurts or breaks contribution between an operational threat and an undesirable situation; (iv) *ImpactAlleviation*, a rule that identifies an operationalizing softgoal that hurts or breaks an undesirable situation; and (v) *SideEffect*, a rule that identifies an NFR softgoal that is affected by an alternative.

Using the alternative solutions pattern in Fig. 5 as an example, *Encrypt ID/password* and *Biometric authentication* are alternative solutions that break(–) *Transmission of ID/password in the clear text* vulnerability soft-problem. The solutions are captured by *R1:RiskTransfer* and *R4:RiskTransfer* rules. The *Encrypt ID/password* alternative has two help(+) contributions towards cost and trustworthiness softgoal, while the biometric alternative has an hurt(-) contribution towards cost and a make(++) contribution towards trustworthiness. The side-effects are recorded by *R2:SideEffect* and *R3:SideEffect* rules for the *Encrypt ID/password* solution, and by *R5:SideEffect* and *R6:SideEffec*t rules for the *Biometric authentication* solution respectively.

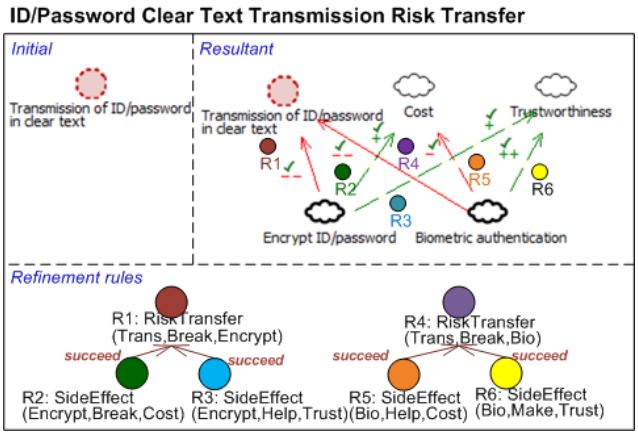

Figure 5. An alternative solutions pattern

Similarly, Figure 6 depicts an example of alternative requirements patterns that identifies an authentication problem frame [15] and a use case model as alternative requirement models that realize, as denoted by make(++) contributions, the *Token + password* authentication operationalizing softgoal, with maintainability as side-effects.

The refinement rules defined for alternative requirements pattern include: (i) ProblemFrameMapping, a rule that identifies a problem frame to realize an operationalizing softgoal; (ii) *UseCaseMapping*, a rule that identifies a use case model to realize an operationalizing softgoal; and (iii) *SideEffect*, a rule that identifies an NFR softgoal that is affected by a requirements model.

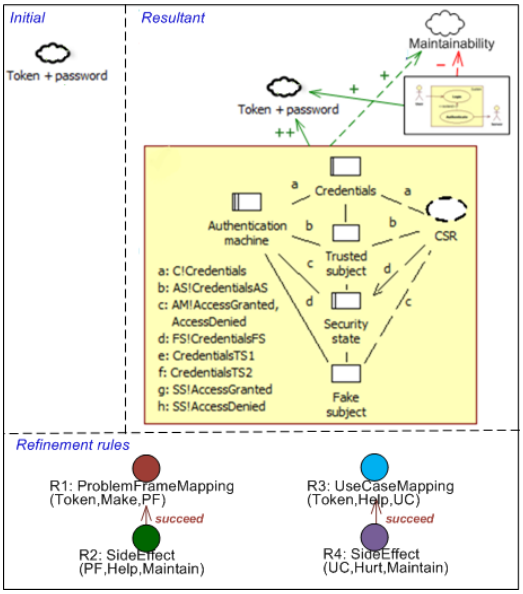

Figure 6. An alternative requirements pattern

D. Selection Pattern

When faced with alternatives for a softgoal or soft-problem, along with their side-effects, an engineer must choose an alternative that not only satisfices (a term referring to sufficient satisfaction or good enough [4]) the softgoal or sufficiently denies the soft-problem, but also maximizes the positive while minimizing the negative side-effects, a task that is difficult due to large decision space [16].

To cope with the complexity, two general schemes – quantitative and qualitative selections – have been used to make the selection more systematic. Quantitatively, the most used method has been the linear weighting approach [17], also known as Multi-Attribute Utility Theory [18], where a fitness function calculates a cumulative score for each alternative based on the weight of the criteria and the degree of how well (or bad) each alternative is towards each criterion. The alternative that receives the highest score is considered the most desirable, therefore selected.

However, assigning weights prior to the selection is oftentimes performed subjectively as is difficult to be precise with [19]. The subjectiveness may become a concern when the resulting fitness values are too close (e.g. 0.71 vs. 0.75) as one alternative may not be clearly better than the other, a problem known as the scaling problem [20]. For some situations, a qualitative selection scheme, such as a rank-based scheme, may be more desirable [19][20].

In this approach, we capture two application-independent selection schemes as patterns: weight-based quantitative selection pattern and rank-based qualitative selection pattern.

1) Weight-based Quantitative Selection Pattern: Weight-based selection schemes have been used for one-level alternatives selection in goal-oriented models [21][22]. The weight-based selection in this paper generalizes and extends the scheme to support multi-level alternatives selection, and the notions of problem mitigation and side-effect.

Using Fig. 7 as an example, two alternatives, *Encrypt ID/password* and *Biometric authentication*, are alternatives for mitigating *Transmission of ID/password in clear text* vulnerability, where the former has two positive side-effects towards *Cost* and *Trustworthiness* softgoals, while the latter is, in turn, a goal to be operationalized by two more specific alternatives, *Fingerprint* and *Retina* authentications with different side-effects towards *Cost* and *Trustworthiness*.

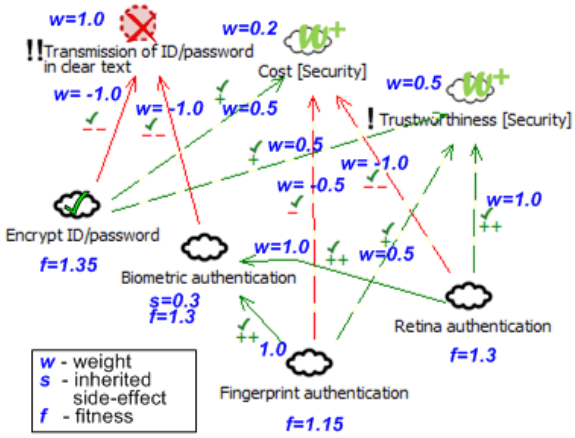

Figure 7. A weight-based selection example

Let us suppose that the stakeholders agree with the following weight assignments: 1.0, 0.5, 0.2 to each of "very critical (!!)", "critical (!)", and neutrally critical N respectively, where N is a softgoal, soft-problem, or side-effect NFR; 1.0, 0.5, -1.0, -0.5 to each of Make(++), Help(+), Break(–), and Hurt(-) contributions respectively, as shown in Fig. 7. A different group of stakeholders may prefer different weights based on their experience or preference.

Selecting among hierarchically structured alternatives is a multi-step bottom-up process. In this example, Fingerprint and Retina authentications are first evaluated and one is selected to achieve the abstract Biometric authentication goal. Then, Encrypt ID/password or Biometric authentication are evaluated where the latter inherits the site-effect related utility value from that of the lower level selection.

In this example, Retina authentication is first selected for its higher fitness value of 1.3, which is obtained from:

$$\begin{aligned}&\text{fitness(Retina authen.)}\\&\quad=\text{weight(Bio.)}\times\text{weight(Make(Retina,Bio))}+\\&\qquad\text{weight(Cost)}\times\text{weight(Break(Retina,Cost))}+\\&\qquad\text{weight(Trust.)}\times\text{weight(Make(Retina,Trust.))}\\&\quad=1.0\times1.0+0.2\times(-1.0)+0.5\times1.0\\&\quad=1.0+(-0.2)+0.5=1.3\end{aligned}$$

Next, *Encrypt ID/password* is selected for its higher fitness value of 1.35, over *Biometric authentication*, which has a fitness value of 1.3 as obtained from:

fitness(Biometric authen.)
= weight(Trans.) × weight(Break(Bio.,Trans.)
+ inherited-sideeffect(Bio.)
= -1.0×(-1.0) + 0.3 = 1.3

The inherited side-effect utility value is the cumulative utility values concerning only the side-effects. For the Retina authentication, the side-effect value of 0.3 is calculated from -0.2 + 0.5.

To capture the weight-based selection in a pattern, the applicable refinement rules include: (i) *CriticalityWeightAssignement*, a rule that assigns a weight to all propositions by their criticality (e.g. "!!" for very critical and "!" for critical); (ii) *ContributionWeightAssignement*, a rule that assigns a weight to all contributions by their contribution type (e.g. hurt(-) or help(+)); and (iii) *WeightBasedSelection*, a rule that selects an alternative with the highest fitness value. Different weight-based selection patterns may capture different weight assignments as preferred by different groups of stakeholders.

Once an alternative is selected, a goal-oriented analysis (e.g. the label propagation procedure [4]) is performed to determine whether the high-level softgoals are satisficed.

2) Rank-based Qualitative Selection Pattern: Qualitative selection schemes, such as pair-wise selection, have been used for selecting alternatives in goal-oriented models [23], which are useful when specific knowledge and experience about the alternatives is available.

The rank-based selection pattern in our approach, on the other hand, is application-independent that systematically selects an alternative based on a ranking scheme agree that enumerates relative preferences of all contribution types against different criteria's criticality.

For example, let us suppose that the stakeholders agree upon a ranking scheme shown in Fig. 8 that prefers a Make(++) contribution towards the goal, as denoted by "g++", to a "g+" or an Help(+) contribution towards the goal. In turn, a "g+" contribution is preferred to a "!!++" or a Make(++) contribution towards a "very critical" side-effect goal, as the stakeholders may consider achieving the goal is more important than side-effect factors

For some stakeholders, it may not be agreeable whether a Help(+) contribution towards a very critical side-effect goal (!!+) is more preferable than a Make(++) contribution towards a critical side-effect goal (!++). For such cases, multiple contributions may be ranked equally. For example, both "!!+" and "!++" contributions have ranking value of 4 in this example. For some other stakeholders, a forced ranking scheme without duplicate ranking values may be more desirable.

Consider using the ranking scheme for choosing among the alternatives in Fig. 7. Similar to the quantitative selection pattern, the qualitative selection pattern also supports bottom-up multi-level alternatives selection. In each level, the alternatives are evaluated based on their cumulative ranking values. Then, the selection is used for the higher level selection using the inherited ranking value from the lower level selection.

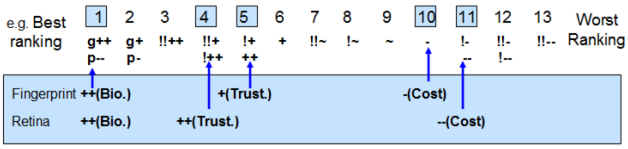

Figure 8. A ranking scheme for qualitative selection

As shown in Fig. 8, both Fingerprint and Retina authentication alternatives have the same cumulative ranking value of 16, which is the sum of their respective individual ranking values, 1+5+10 for the Fingerprint authentication and 1+4+11 for the Retina authentication respectively.

The equal cumulative ranking implies that both alternatives are equally desirable. As a result, either alternative can be selected. It can also mean that there is not enough criteria to differentiate the two alternatives. In that case, additional side-effect goals may be added and the selection is repeated. The selection then continues upward to select between *Encrypt ID/password* and *Biometric authentication*.

To capture the rank-based selection, the applicable refinement rules include: (i) *RankAssignement*, a rule that assigns a ranking value to all contributions according to a pre-defined ranking scheme; and (ii) *RankBasedSelection*, a rule that selects an alternative with the highest cumulative ranking.

IV. NFR Pattern Organization

As the number of NFR patterns in a catalog grows, it would become increasingly more important to organize the patterns. This NFR Pattern approach organizes patterns along the three knowledge organizational dimensions, including the generalization, aggregation, and classification dimensions [24] through pattern specialization, composition, and instantiation operations respectively.

A. Pattern Specialization

Knowledge of NFR captured in an NFR pattern may be specialized for a more specific situation. For example, Fig. 9 shows that pattern *P1* is a specialized (sub-)pattern of pattern *P2* where *P1* captures a definition of security for the Payment Card Industry (PCI) [13] that is more specific than the general definition defined by a US law [6]. In this example, PCI is concerned with only the confidentiality aspect of security that is defined by the US law as confidentiality, integrity, and availability.

To ensure that a specialization is meaningful, sub-pattern *P1* is considered a specialized pattern of super-pattern *P2* if *P1* consists of a refinement rule graph that is more specific in breadth, in depth, or both, than the refinement rule graph of the super-pattern. Refinement rule graph is a graph that consists of refinement rules related to each other by *succeed* relationships. To be more specific in breadth, refinement rule

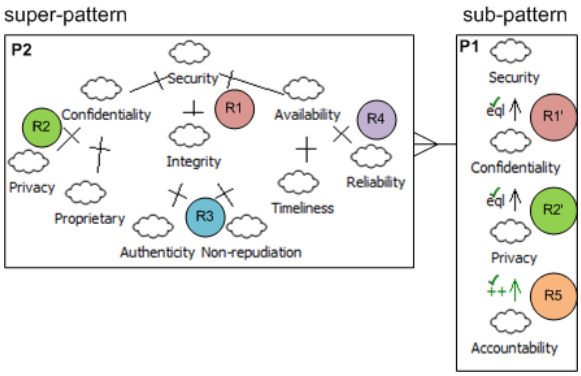

Figure 9. A pattern specialization

graph *G1* must have matching refinement rules in *G2* where each has a more limited refinement. To be more specific in depth, refinement rule graph *G1* must more hierarchical levels than refinement rule graph *G2*.

Using Fig. 9 as an example, pattern *P1* is a valid specialized sub-pattern of pattern *P2* because the refinement rule graph of *P1* is more specific in breadth as rule *R1'* and *R2'* of pattern *P1* have narrower definitions of Security and Confidentiality than that of of *R1* and *R2* of *P2* rules respectively. *P1* also has a deeper definition for Security with an additional refinement rule *R5* that further refines Privacy as Accountability.

Pattern specialization relationship is partial order – reflexive, anti-symmetric, and transitive. As a result, a specialized pattern may be further specialized by other patterns.

B. Pattern Composition

A pattern composition pre-assembles multiple patterns in a new pattern to form a larger chunk of knowledge. The smaller "part-of" patterns are related by *succeed* relationships to define the order in which the patterns should be applied when the "whole" pattern is applied.

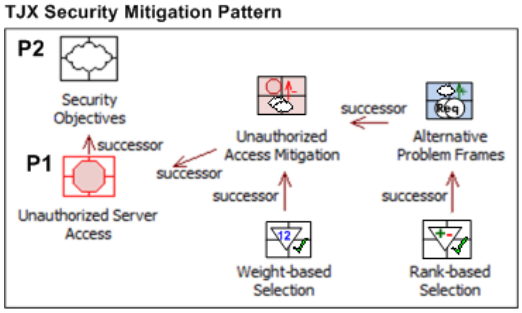

Figure 10. A pattern composition

Using Fig. 10 as an example, *Security Objectives*, *Unauthorized Server Access*, and other patterns are assembled as "part-of" patterns to form the "whole" *TJX Security Mitigation* pattern. When the "whole" pattern is applied, the refinement rules of the root pattern (*Security Objectives* pattern) are applied, followed by the application of refinement rules in the *Unauthorized Server Access* pattern, and so on and so forth until all the assembled patterns are applied, in the orders defined by the *succeed* relationships.

To ensure that the composition is meaningful, pattern *P1* may succeed pattern *P2* in a composition if the Initial proposition of *P1* also exists in the Result section of pattern *P2*. This constraint ensures that when each non-root pattern is applied, the required Initial proposition has been generated by the preceding pattern.

The "part-of" relationship between a pattern and its composite pattern is partial order, that is, reflexive, anti-symmetric, and transitive. As a result, a composite pattern can be in turn be assembled to be a part of another larger composite pattern.

C. Pattern Instantiation

A piece of knowledge may be applicable to a similar situation. For example, some security measures for protecting credit card confidentiality may also be applicable to protecting the confidentiality of patient information. Pattern instantiation allows a pattern to be used as a template for creating a new pattern.

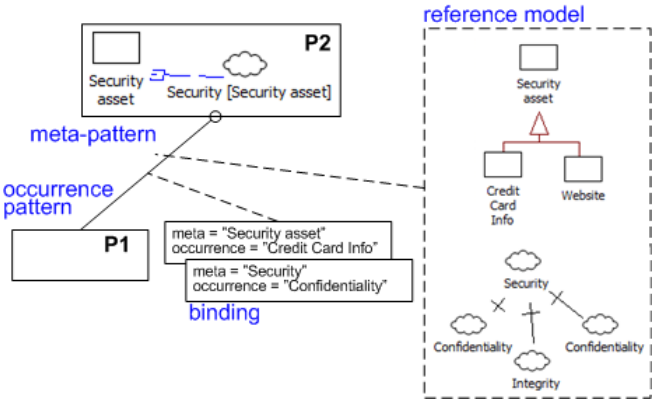

Figure 11. A pattern instantiation

Using Fig. 11 as an example, pattern *P1* is a new pattern created by customizing an existing pattern through two binding specifications. Note that pattern *P1* does not explicitly capture any knowledge (e.g. Initial and Resultant). Knowledge of pattern *P1* is inherited from pattern *P2*. When pattern *P1* is applied, the refinement rules of pattern *P2* are applied to inherit the knowledge. The results from applying *P1* are customized according to the binding specifications. The new pattern *P1* is called an occurrence pattern [25] of the existing pattern *P2*, which serves as the pattern for pattern P1. A pattern of pattern is also known as a meta-pattern [26]. The occurrence pattern in turn may be used as a meta-pattern to create other occurrence patterns.

To ensure that the binding specifications are meaningful, each *occurrence* element in the binding specification must be either a specialized entity or an offspring goal or problem of the corresponding *meta* element in the binding. The *occurrence* element defines the target of the source element defined by the *meta* element. The relationship between the meta and occurrence elements are verified against a reference model, which in practice may be an enterprise model. For example, the binding specification in Fig. 11 maps *Security asset* resource to one of its specialized resource, *Credit Card info*, and maps *Security* softgoal to one of its offspring goal, *Confidentiality*.

V. An Empirical Study: The TJX Case

The NFR Pattern approach has been applied in an empirical qualitative study based on the TJX case, with an objective to study whether the approach could help capture and reuse relevant NFR knowledge in a similar organization or project.

We studied several sources of information, including the US Federal Information Security Management Act (FISMA) [6], the Payment Card Industry (PCI)'s Data Security Standard (DSS) [13], an investigation report on the TJX case [3], a federal court indictment of the hacker [2], computer security literature, such as that related to the wireless network used by TJX [27], and over 30 news articles, such as the Wall Street Journal article that broke the store [1], among others. The large amount of real-world knowledge was analyzed and pieced together to relate relevant knowledge of NFR objectives, problems, and alternatives.

Figure 12 and 13 illustrate a build for reuse and a build with reuse scenarios respectively in the study. The first scenario shows that knowledge was first modeled, then captured as patterns via the *patternize* operation, organized via the *specialize*, *compose*, and *instantiate* operations. The second scenario shows that various patterns were used to gradually refine a target model.

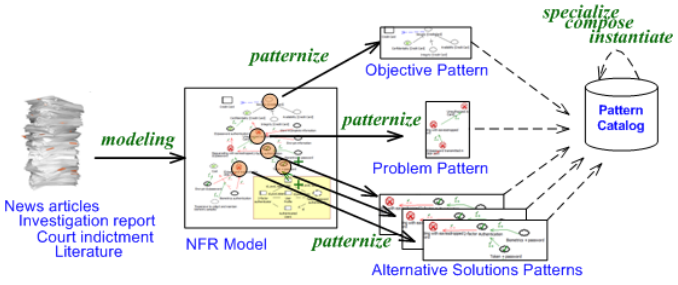

Figure 12. Capturing and organizing NFR patterns in the case study

We developed two tool prototypes to verify the approach and support the case study, including the RE-Tools [28] and the NFR Pattern Assistant [29]. The RE-Tools supports the modeling of the NFR Framework [4], the i* Framework [7], the Problem Independent Graph [14], Problem Frame [30], and UML [25] in an integrated manner. The NFR Pattern Assistance implemented the NFR Pattern approach, including 25 refinement rules: 2 for objective pattern, 8 for problem pattern, 10 for alternatives pattern, and 5 for selection, as well as 5 pattern operations: patternize, specialize, compose, instantiate, and apply.

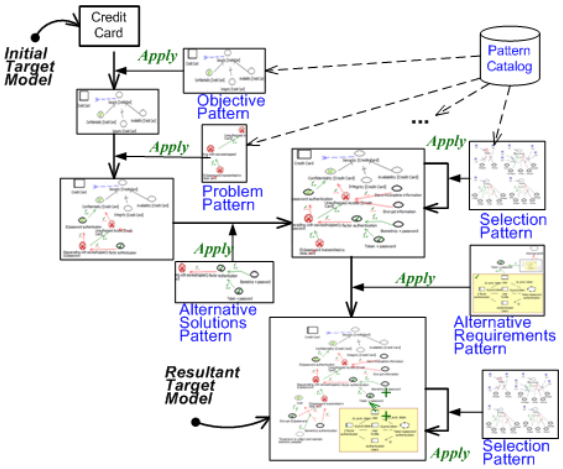

Figure 13. Reusing NFR patterns in the case study

Figure 14 shows a manually created model to capture the knowledge related to the unauthorized server access problem, whose small excerpts are used as examples throughout this paper. Figure 15 shows a corresponding target model that was refined by applying the captured patterns.

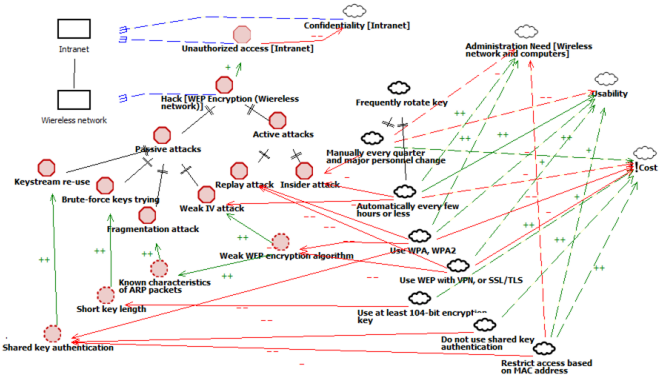

Figure 14. A model representing the real-world knowledge

The comparison of the two models found that the two model are almost identical semantically, but visually different. Semantically, the target model consists of 25 of 27 model elements (93%) from the source model. Visually, the models are different because the NFR Pattern Assistant prototype used an automated layout feature to place the generated model elements while the elements in the source model were manually placed.

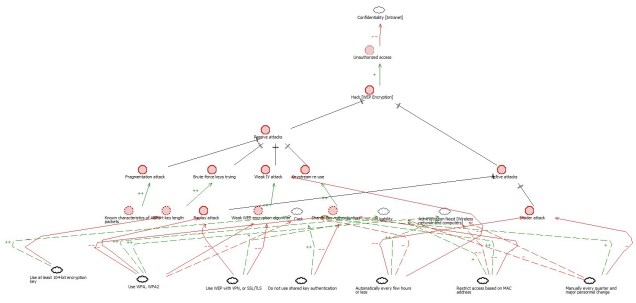

Figure 15. A model generated from pattern applications

Nevertheless, the study provides a preliminary evidence that the NFR Pattern approach could help capture, organize, and reuse a large percentage of knowledge of NFRs in model- and tool-based requirements engineering.

However, there are a number of threats to validity. For example, the accuracy of real-world knowledge representation in the patterns cannot be easily determined as the source models were subject to human's interpretation of real-world information. On the other hand, even if knowledge representation is accurate, the patterns may be used in an unintended application.

These two concerns may be partially alleviated by the use of the credential and applicability information in the patterns. For example, the knowledge representation in a pattern may be considered trustworthy if the credential information indicates that the pattern was authored by an expert in the field or it received many positive reviews from experts or past users. In another example, a military organization may exercise caution if the *where* attribute in the pattern indicates that retail commerce is an intended domain. However, the intended utility of the supplemental information was not verified in the case study as it was performed in a controlled setting by the authors.

Additionally, the case study did not validate whether the captured knowledge is applicable to other security cases because the source real-world information was obtained from only one security incident. The case study also did not validate whether the NFR Pattern approach is indeed applicable to all NFRs as intended because the case dealt with only security, cost, and trustworthiness NFRs.

VI. Related Work and Discussion

Related works discussed briefly in this section are in the areas of requirements patterns, and NFRs. For a more detailed survey, readers are referred to [31].

The analysis and business patterns have been used to capture functional requirements [32][33]. Darimont and van Lamsweerde [34] have identified various patterns for the refinement of functional goals. Similarly, Jackson has proposed various classes of software problems, called Problem Frames [30]. There are several extensions of goal-based and Problem Frames-based patterns. Hatebur [15], for instance, shows how Problem Frames can be used to solve problems in the development of secure software systems. Knowledge of NFRs has been captured as patterns in the form of informal guidelines [35][36], architecture and design [36]. Although similar in some aspects, the emphasis of our work lies in the capture of knowledge of NFRs that are treated as goals to be achieved, and it is ground in the NFR Framework [4] and the ability of the framework to manage synergy and conflict among NFRs. The patterns in our work are intended for model-based and tool-assisted requirements engineering.

There are also extensive works on using and extending the NFR Framework. For example, NFRs have been used as criteria for selecting among tasks [8], architectural alternatives [37], and relating NFRs to design through patterns [38]. Patterns proposed in this work can be used to capture such knowledge. Knowledge of agent-specific dependencies on NFRs have been captured by organizational and social patterns [39][40], and patterns for mapping agents concerns to requirements [41]. Our patterns complement such patterns by capturing deeper NFR-specific knowledge. Our alternative requirements patterns are similar in spirit to the work on mapping from goals, to means, then to a design [42][43], but ours additionally allow for the capture of multiple mappings as well as the knowledge of side-effects.

Similarly, our work is also an extension of the NFR Framework. Although the extension could make the overall framework more complex, NFR patterns could also make the NFR Framework easier to use where a large number of requirements engineers can concentrate on the application at hand and use the NFR patterns provided by a few subject matter and modeling experts to produce well-formed models in a time-saving manner without having to master the NFR modeling details.

Regarding pattern organization, Yang and Liu [44] classify requirements patterns by layers of goals, tasks and resources, and relate them by goal-oriented contributions, such as hurt and help. Our approach classifies patterns by kinds of NFR knowledge and relates them along the three knowledge organizational dimensions.

Since the works on requirements patterns have not been adequately extended to the large body of knowledge of NFRs, and vice versa, we believe that this work fills a small but important gap in the requirements engineering literature.

VII. Conclusion

In this paper we have presented a pattern-based approach to dealing with knowledge of NFRs. More specifically, we have presented four types of NFR patterns (objective pattern, problem pattern, alternatives pattern, and selection pattern), three ways to organize patterns (specialization, composition, and abstraction), and pattern representation using model refinement rules. NFR patterns are intended to capture and reuse specific aspects of knowledge NFRs, i.e, goals,

problems, alternative means/solutions/requirements, and selection in a structured and modular manner. Organization of patterns are intended to capture knowledge of NFRs in an understandable and manageable way. Finally, refinement rule-based representation and the constraints for pattern organizations allow for a tool implementation. Although we have observed some threats to validity in the case study, we felt the four kind of visually represented and organized along the organizational concepts patterns, helped us better understand and manage a large body of NFRs.

There are several lines of future work. Firstly, more case studies would be needed to better understand the strengths and weaknesses, in particular concerning the general applicability of the approach and the completeness of the defined refinement rules. A variety of different kinds of patterns would also need to be captured and reused, other than security, cost, and trustworthiness. The tool that we have implemented and used in our study was valuable in verifying the concepts behind the approach, but it is of a prototype quality. It needs to be enhanced for better usability and performance.

ACKNOWLEDGMENT

The authors would like to thank Bashar Nuseibeh for his insightful comments on an earlier draft. The authors also thank Rutvij Mehta for his help with copyediting, and the anonymous reviewers of RE'10 and MaRK'09 workshop, and its participants, particularly Martin Glinz and Robyn Lutz for their valuable comments. The BR-based author acknowledges the financial support of CNPq (Grants: 557.128/2009-9 and 304852/2009-0) and Faperj (Grant: Cientista do Nosso Estado and E-26/170028/2008). The UK-based author acknowledges the financial support of the EU project SecureChange.

REFERENCES

[1] J. Pereira, "Breaking the code: How credit-card data went out wireless door," *The Wall Street Journal*, May 4 2007.

[2] United States District Court of Massachusetts, "United States of America v. Albert Gonzalez," 18 U.S.C. §371, Aug. 5 2008.

[3] Office Of The Privacy Commissioner Of Canada And Office Of The Information And Privacy Commissioner Of Alberta, "Report of an Investigation into the Security, Collection and Retention of Personal Information of TJX Companies Inc. and Winners Merchant International L.P." Sep. 2007.

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

[5] L. Chung and J. do Prado Leite, "On Non-Functional requirements in software engineering," in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S.Yu, Eds., 2009, pp. 363–379.

[6] "The federal information security management act of 2002," 44 U.S.C. §3541, 2002.

[7] E. Yu and J. Mylopoulos, "Understanding why in software process modelling, analysis, and design," in *Proc. 16th Intl. Conf. on Soft. Eng.*, May 16-21, 1994, pp. 159–168.

[8] J. Leite, Y. Yu, L. Liu, E. Yu, and J. Mylopoulos, "Quality-based software reuse," *Advanced Information Systems Engineering*, pp. 535–550, 2005.

[9] C. Gunter, E. Gunter, M. Jackson, and P. Zave, "A Reference Model for Requirements and Specifications," *IEEE Software*, pp. 37–43, 2000.

[10] T. Tun and J. Hall, "Developer requirements in the PF approach," in *Proceedings of the 2006 Intl. workshop on advances and applications of problem frames*, 2006, p. 90.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[12] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

[13] Payment Card Industry, "Data security standard v1.1," Sep. 2006.

[14] S. Supakkul and L. Chung, "Extending problem frames to deal with stakeholder problems: An agent- and goal-oriented approach," in *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, 2009, pp. 389–394.

[15] D. Hatebur, M. Heisel, and H. Schmidt, "Security engineering using problem frames," *Lecture Notes in Computer Science*, vol. 3995, p. 238, 2006.

[16] J. Horn, "Multicriterion decision making," *Handbook of Evolutionary Computation, chap. F1.9*, 1997.

[17] C. Weber, J. Current, and W. Benton, "Vendor selection criteria and methods." *European Journal of Operational Research*, vol. 50, no. 1, pp. 2–18, 1991.

[18] R. Keeney and H. Raiffa, *Decisions with multiple objectives: Preferences and value tradeoffs*. Cambridge Univ Pr, 1993.

[19] J. Grefenstette, "Rank-based selection," *Handbook of Evolutionary Computation, chap. C2.4*, 1997.

[20] D. Whitley *et al.*, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proceedings of the third international conference on Genetic algorithms*, vol. 1. Sanfrancisco: Morgam Kaufm Publish, 1989, pp. 116–121.

[21] A. van Lamsweerde, "Reasoning about alternative requirements options," in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S.Yu, Eds., 2009, pp. 380–397.

[22] H. Xie, L. Liu, and J. Yang, "i*-prefer: optimizing requirements elicitation process based on actor preferences," in *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*. New York, NY, USA: ACM, 2009, pp. 347–354.

[23] G. Elahi and E. Yu, "Trust Trade-off Analysis for Security Requirements Engineering," in *Proc. of the 2009 17th IEEE Intl. Requirements Engineering Conf.*, 2009, pp. 243–248.

[24] R. Hull and R. King, "Semantic database modeling: Survey, application and research issues," *ACM Comp. Survey*, vol. 19, no. 3, pp. 201–260, 1987.

[25] Object Management Group, "UML Superstructure, V2.1.2," http://www.omg.org/docs/formal/07-11-02.pdf.

[26] G. Meszaros and J. Doble, "A pattern language for pattern writing," *Pattern languages of program design*, vol. 3, pp. 529–574, 1998.

[27] A. Bittau, M. Handley, and J. Lackey, "The final nail in WEP's coffin," in *IEEE Symposium on Security and Privacy*. Citeseer, 2006.

[28] "The RE-Tools," www.utdallas.edu/∼supakkul/tools/RE-Tools.

[29] "The NFR Pattern Assistant," www.utdallas.edu/∼supakkul/tools/NFRPassist.

[30] M. Jackson, *Problem frames: analyzing and structuring software development problems*. Addison-Wesley, 2000.

[31] S. Supakkul, "Capturing, Organizing, and Reusing Knowledge of NFRs: An NFR Pattern Approach," Ph.D. dissertation, University of Texas at Dallas, 2010.

[32] M. Fowler, *Analysis Patterns: reusable object models*. Addison-Wesley, 2000.

[33] H. Kilov, *Business Specifications: The Key to Successful Software Engineering*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[34] R. Darimont and A. Van Lamsweerde, "Formal refinement patterns for goal-driven requirements elaboration," *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 6, p. 190, 1996.

[35] S. Withall, *Software requirement patterns*. Microsoft Press Redmond, WA, USA, 2007.

[36] M. Markus, E. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and System Engineering*. John Wiley & Sons, 2006.

[37] L. Chung, B. Nixon, and E. Yu, "Using non-functional requirements to systematically select among alternatives in architectural design," in *Proc. 1st Int. Workshop on Architectures for Software Systems, Seattle, Washington*. Citeseer, 1995, pp. 31–43.

[38] D. Gross and E. Yu, "From non-functional requirements to design through patterns," *Requirements Engineering Journal*, vol. 6, no. 1, pp. 18–36, 2001.

[39] M. Kolp, P. Giorgini, and J. Mylopoulos, "Organizational patterns for early requirements analysis," *Lecture Notes in Computer Science*, pp. 617–632, 2003.

[40] T. Do, Kolp, M., and A. Pirotte, "Social patterns for designing multi-agent systems," in *Proceedings of the 15th intl. conf. on software engineering and knowledge engineering (SEKE-2003)*, 2003.

[41] N. Maiden, S. Manning, S. Jones, and J. Greenwood, "Generating requirements from systems models using patterns: a case study," *Requirements Engineering*, vol. 10, no. 4, pp. 276–288, 2005.

[42] S. Konrad and B. Cheng, "Requirements patterns for embedded systems," in *Proc. of the IEEE Joint Intl. Conf. on Requirements Engineering (RE02)*, pp. 127–136.

[43] J. Fletcher and J. Cleland-Huang, "Softgoal traceability patterns," in *Software Reliability Engineering, 2006. ISSRE'06. 17th Intl . Symposium on*, 2006, pp. 363–374.

[44] J. Yang and L. Liu, "Modelling requirements patterns with a goal and PF integrated analysis approach," in *Proc. of 32nd Annual IEEE Intl . Computer Software and Applications Conference COMPSAC'08*, 2008, pp. 239–246.