Experimental Evaluation of a Lightweight Method for Augmenting Requirements Analysis

Carlos Jensen  
School of EECS  
Oregon State University  
Corvallis, OR 97331  
+1-541-737-2555  
cjensen@eecs.oregonstate.edu

Colin Potts  
College of Computing  
Georgia Institute of Technology  
Atlanta, GA 30332  
+1-404-894-5551  
potts@cc.gatech.edu

ABSTRACT
We introduce the concept of augmentation methods, methods that complement other methods by addressing specific non-functional requirements (NFRs). Since most projects do not have dedicated expertise in all relevant NFRs most team members may be comparative novices for that class of NFR. STRAP is a lightweight goal-refinement method for analyzing privacy NFRs. We describe it briefly and then present three experiments to assess its effectiveness and that of several existing privacy frameworks. We analyze the results in terms of method efficiency: the number of analysts needed to find a given proportion of benchmark problems. The alternative methods are generally effective in identifying privacy vulnerabilities but they are inefficient, since the average analyst misses many potential problems. In three distinct application domains, STRAP led to equal or better identification of privacy vulnerabilities and was in all cases more efficient. We conclude that a combination of lightweight structuring and heuristic appropriateness is the reason for these advantages.

Categories and Subject Descriptors

General Terms

Keywords

1. INTRODUCTION
Non-functional requirements (NFRs) are requirements of manner or quality. Thus, if a desired function is to accomplish X, an NFR might be to accomplish X with a certain degree of security or within performance limits. In this sense, NFRs are subsidiary to functional requirements.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Because NFRs are subsidiary in this sense, their analysis must augment an existing analysis of functional requirements. But this raises a problem. If different classes of NFR have their own analysis methods, how can team members, most of whom have no expertise in the area of the NFR understand the implications for the system? For example, Yu & Cysneiros’s method for analyzing privacy NFRs [15] requires its users to be familiar with the requirements modeling language i*, which is a significant training investment. If the project is using UML, say, to model the application, the language gap could cause inefficiencies and misunderstandings. In this paper we focus on methods requiring little or no previous training or experience, intentionally setting the threshold low to demonstrate the feasibility of making an impact.

A good example of this philosophy applies to usability. As Nielsen and Molich [8] have shown, novice analysts can identify many usability issues from an application description if they follow a systematic set of heuristics. Such heuristics are surprisingly effective compared to analyses based on human factors engineering principles and much more efficient in project effort.

Privacy requirements are an increasingly important class of NFR. In this paper, we address an empirical evaluation of methods for identifying potential privacy vulnerabilities in an application. In the spirit of Nielsen and Molich, we advocate methods that are effective but also efficient in analyst time and effort. We do not assume that project teams contain privacy experts. As with Nielsen and Molich’s work, we embrace the possibility of ecologically valid evaluations being possible with novice subjects as opposed to specialists in the area.

Several design and analysis frameworks for privacy have been proposed. The first to go beyond a set of guidelines was Bellotti and Sellen [2] (henceforth B&S). The B&S framework was designed to deal with privacy concerns emerging from ubiquitous environments. In the B&S framework, designers ask key questions meant to alert designers to potential privacy problems. Potential problems are then addressed and evaluated by 19 heuristics.

The framework most closely resembling B&S is the ‘Risk Model’ framework by Hong et al. [3] (henceforth Hong). The Hong framework consists of three steps: (1) A risk analysis driven by a set of social, organizational, and technology questions; (2) Risk management, where the potential for financial and legal harm are assessed; (3) Prioritization of problems and identification and evaluation of potential design solutions.
Patrick & Kenny’s framework [10] (henceforth P&K), is based on legal analysis and requirements, primarily the OECD guidelines [9]. It also borrows elements from requirements and software engineering techniques to structure the analysis process. The method consists of three steps: (1) Derive use-cases (UCs) for the system; (2) From these, generate object-sequenec diagrams (OSDs); (3) Apply privacy heuristics to identify and address privacy issues. P&K therefore augments existing UML representations and scenario-based analysis methods.

In the next section, we present a lightweight method, STRAP, adapted from goal-oriented refinement. STRAP is intended to augment existing methods rather than to stand alone. In other words, it is not meant to replace other, more general, requirements analysis methods. We then report the results of three experiments designed to study the relative effectiveness and efficiency of STRAP and the frameworks mentioned above. Eighty-five novice subjects took part, analyzing systems from three different application domains. We believe that this paper presents one of the first detailed experimental comparisons of such methods.

2. STRAP: Structured Analysis of Privacy

The goal of STRAP is to address some of the critiques of privacy analysis frameworks. STRAP combines elements of goal-oriented analysis and heuristic evaluation in an effort to achieve effectiveness while keeping costs low. An essential property of STRAP is that it does not require analysts to learn new or different skills. It is primarily aimed at supporting analysts in identifying privacy and security concerns, including related usability concerns, during early design. It is an add-on method, and does not address other parts of the design process. A detailed description is available [4].

STRAP builds on the GBRAM goal-oriented analysis technique [1] and its extension to account for obstacles [11]. It also owes much to the KAOS family of languages [7] and i* [15]. But any goal-analysis technique could be used in place of GBRAM as long as it produces a structured visual artifact to anchor the analysis.

The visual artifact (a goal-tree like Figure 1) serves as a guide to the identification of potential privacy problems. A set of questions are then used to identify potential problems. For each goal and sub-goal, the analyst asks the following to better understand the information lifecycle:

- What information is captured, accessed, processed, transmitted, received, or stored in meeting this goal?
- How is information captured, accessed, processed, transmitted, received, or stored in meeting this goal?
- Who are the actors involved in the capture, access, processing, reception, or storage?
- What knowledge is derived from this information?
- What happens to the raw data once the goal is met?

These questions help identify the potential for privacy problems. It is essential to strive for exhaustiveness and not discard potential problems prematurely.

Refinement starts by identifying those vulnerabilities that can be eliminated and those that must be mitigated. Take a vulnerability stemming from a goal requiring the storage of personal data to support customization. This vulnerability can be eliminated in many ways. For example, functionality could be sacrificed (no customization) or weakened (only aggregate data stored), or the vulnerability could be mitigated (database encryption).

When evaluating alternative solutions one needs to weigh three factors; the costs associated with the risks eliminated or mitigated, how well these risks are addressed, and the costs associated with each solution. While methods exist to weigh risks and estimate costs of proposed system changes, it is often difficult to weigh the utility or completeness of mitigation strategies objectively, especially those involving end-users in the decision-making process. In lieu of objective analysis, in STRAP we combine heuristics from an analysis of the OECD guidelines [9] and FIPs [12], with principles derived from a number of studies [2, 3, 5, 6, 8]. Keeping the list of heuristics small is an important factor in achieving efficiency [8].

<table>
<thead>
<tr>
<th>Table 1: STRAP heuristics, organized by FIP category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notice / awareness</td>
</tr>
<tr>
<td>Available, Accessible and Clear</td>
</tr>
<tr>
<td>Presented in context</td>
</tr>
<tr>
<td>Not over-burdening</td>
</tr>
</tbody>
</table>
In experiments 1 & 2, we recruited subjects from an undergraduate HCI class. In experiment 3, subjects were recruited from an ethics and social issues in computing class. Subjects were randomly assigned to experimental conditions, and distributed among the experimental conditions.

In experiment 1, subjects attended a two-hour lecture explaining the two techniques and the application they were to analyze. In experiments 2 & 3, subjects were only given written descriptions of the methods and the applications they were to analyze. In order to minimize bias, subjects were given copies of method descriptions written by their authors. Subjects were asked to track their time as they completed the analysis, and returned the completed assignments when done.

In experiment 1, a panel of experts evaluated the same system as the participants. This expert review was used as a benchmark.

In experiment 3, six subjects were also interviewed for about thirty minutes after the experiment. Two subjects, a high achiever and low achiever, from each experimental condition were interviewed. The results of the interviews are not given here for space limitations but are available separately [5].

### 3.2 Results

**Experiment 1: Augur Calendar System**

The expert analysis of Augur yielded 36 vulnerabilities, half of which were unique. The subjects’ reports were evaluated and normalized independently by the reviewers who also marked false positives and mapped remaining vulnerabilities to their own analyses. Where the reviewers disagreed, the subject was given the benefit of the doubt, and the vulnerability was counted. Table 3 summarizes the results.

| Table 3: Results Experiment 1 (Std dev in parentheses). |
|---|---|---|
| Min. on task | Unfiltered vuln. | Filtered vuln. |
| STRAP | 88.77 (25.82) | 6.86 (2.28) | 5.07 (2.20) |
| B&S | 101.18 (43.46) | 6.53 (2.50) | 3.53 (1.41) |
| Change | +13.49 | -0.33 | -1.54 |

**Task effort:** The time spent on analysis was the same between groups (df=22, t=0.831, n.s.).

**Task performance (quantitative):** There was no difference in the total number of vulnerabilities discovered (df=27, t= 0.364, n.s.), but in the B&S group, 31.7% of the vulnerabilities were determined to be general HCI issues rather than privacy issues compared to 14.58% non-privacy vulnerabilities for STRAP. When these “false positives” were removed from the analysis, we found a difference in how many privacy vulnerabilities were discovered (df=27, t= 2.225, p<0.05), with the STRAP subjects reporting 43.6% more vulnerabilities than B&S subjects.

STRAP subjects discovered a maximum of nine vulnerabilities, with the average being 5.07 (2.20 stdev), and the median four. Overall, subjects discovered 17 of the 18 vulnerabilities discovered by the experts (94.4%). The B&S subjects discovered a maximum of six vulnerabilities with an average of 3.53 avg (1.41 stdev), and the median three. Overall, subjects discovered 13 of the 18 vulnerabilities discovered by the experts (72.2%).

**Task performance (qualitative):** Assessing subjects’ understanding of the system is more straightforward in STRAP.
than B&S, because the goal trees that students produced can be matched against the benchmark tree and those of other subjects. A composite of these diagrams is shown in Figure 2. In the first three levels of this decomposition we see that over 50% of subjects identify the majority of system goals.

Figure 2: Analysis Quality of Augur Using STRAP

Color intensity and number indicates the number of subjects identifying these goals as part of their STRAP analysis of Augur

Task effort: There were no differences in the time spent on the analysis by the analysts in the three conditions (F(2,24)=0.45, n.s.).

Task performance (quantitative): The total number of reported vulnerabilities differed across the three conditions (F(2,31)=3.70, p<0.05): STRAP analysts were again more productive than B&S analysts (df=22, t=2.856, p<0.01), but did not differ from Hong analysts (df=22, t=0.852, n.s.). A marginal but statistically difference occurred between B&S and Hong analysts (df=20, t=1.841, p<0.1).

Experiment 3: Teamspace

Table 5: Results Experiment 3 (Std dev in parentheses).

<table>
<thead>
<tr>
<th></th>
<th>Task time (min.)</th>
<th>Vuln.</th>
<th>Vuln/hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRAP</td>
<td>161.5 (70.2)</td>
<td>4.50</td>
<td>1.75 (0.96)</td>
</tr>
</tbody>
</table>

Method efficiency: Figure 4 shows how the likelihood of discovering all vulnerabilities increases with the number of independent analysts. Figure 3 & 4 show similar patterns.
Experience, and only needed to put in a very minimal effort. As part of the design phase, and 2) that our subjects had little or no experience, and only needed to put in a very minimal effort. Overall, this experiment showed that with relatively little cost and effort, a team of untrained analysts can discover a reasonable number of privacy vulnerabilities using the augmentation method STRAP. These results are especially striking when one considers two key facts; 1) that privacy is not typically considered formally as part of the design phase, and 2) that our subjects had little or no experience, and only needed to put in a very minimal effort.

<table>
<thead>
<tr>
<th></th>
<th>Hong</th>
<th>P&amp;K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128.00 (52.1)</td>
<td>135.00 (34.4)</td>
</tr>
<tr>
<td>2.33 (1.21)</td>
<td>2.60 (1.14)</td>
<td>1.77 (0.86)</td>
</tr>
<tr>
<td>1.27 (0.85)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Task effort:** Subjects reported spending over 2.5 hours on the analysis, longer than in experiments 1 and 2. There was no difference in analysis time among the groups (F(2,14)=0.62, n.s.). Subjects in the two conditions requiring problem structuring (goal-oriented analysis in STRAP; UCs and OSDs in P&K) spent the same time (about 40%) on that phase. P&K subjects returned an average of one short scenario, and one very basic OSD per analysis. This is a shallower analysis than P&K advocates, and less detailed than STRAP subjects’ goal trees.

**Figure 5:** Vulnerability identification per analyst.

**Task performance (quantitative):** There was a difference in the number of reported vulnerabilities (F(2,14)=5.74, p<0.02). This derives from two component differences: between STRAP and P&K (df=10, t=2.602, p<0.05), and between STRAP and Hong (df=11, t=3.0219, p<0.05).

**Task efficiency:** The correlation between time on task and vulnerability discovery was much weaker than in Experiment 2 (r = 0.475 compared with r = 0.801). Though STRAP subjects reported more vulnerabilities, they also spent longer on their analysis. There was no difference in the discovery rates (F(2,14)=1.56, n.s.).

**Method efficiency:** A total of eleven unique vulnerabilities were discovered. STRAP subjects discovered all eleven, with each analyst on average reporting 4.5 (1.22 stdev), and the median four. P&K subjects discovered six (54.5%) unique vulnerabilities, with the average being 2.60 (1.14 stdev), and the median three. Hong subjects discovered seven (63.6%) unique vulnerabilities, with the average being 2.32 (1.21 stdev), and the median 2.5. Figure 5 shows how the likelihood of discovering all vulnerabilities increases with the number of independent analysts. The results are in line with Experiment 2, and P&K is similar to the Hong data.

4. DISCUSSION

Overall, subjects using STRAP performed best. This was not entirely unexpected, as STRAP builds on the other frameworks. While we had expected STRAP and P&K to lead to better results than the less structured B&S and Hong frameworks, we expected these gains to come at the price of more time on task, given the preliminary analysis steps. This did not seem to be the case. It is possible that the less structured frameworks are misleadingly simple; a lack of structure to guide the analysis results in analysts wasting time determining what to look at next, or keeping track of what has already been considered. This lack of structure requires analysts to rely on their own cognitive resources rather than external artifacts.

P&K and STRAP share many theoretical commonalities. It was therefore surprising how differently subjects performed with these two methods. Subjects using P&K performed similarly to those using Hong, and the method was rated poorly by subjects. These findings are especially surprising, given that Teamspace was documented in a scenario style that should have been compatible with P&K. The interview data indicates that as an augmentation method, the structuring it requires too much extra work for too little perceived gain [4]. Unless a suitable analysis has already been performed as part of the RE process, this extra work may prevent successful adoption.

5. CONCLUSIONS

This paper has been about a specific method for elucidating requirements issues, STRAP, but it has also used STRAP as an example of an augmentation method for NFRs for use by intelligent people who have no prior training in goal-oriented analysis or privacy considerations. Although the conclusions with which we are most comfortable apply to privacy analysis, it is quite possible that the strategy we adopted in devising STRAP could be used for other NFRs and for augmentation of methods other than those based on goal refinement. We caution against applying a lightweight augmentation philosophy indiscriminately to critical non-functional requirements such as safety or mission-critical security requirements. However, privacy is a serious matter, too, and it is surprising how well novice analysts can identify privacy issues using STRAP. Methods like this may prove an ideal way to check enterprise-rich NFRs with domain experts outside the development organization.

Nothing in STRAP is specific to GBRAM or ScenIC. It could augment other goal-oriented methods, and probably is compatible with teleological actor frameworks such as i* and also with basic structured analysis and top-down decomposition of function. Experiment 3 demonstrated, however, that commercial methods based on UML may be too heavy-weight for augmentation in this way unless they are already in use and accepted by the project team.

In contrast to STRAP, guidelines-based methods appear to provide too little structure and leave the novice analyst awash and prone to design fixations (e.g. usability or legal requirements) that may be important but are not the objective of the analysis. What seems to be lacking in these methods is not thoughtfulness in the identification of guidelines. They are generally principled and sound. Rather, what is lacking is a way to relate the problems identified by the framework back to artifacts about the system that can be amended to reflect the new insights.
flaws in system requirements may strike some readers as a scurrilous suggestion. We should emphasize that we are not advocating requirements analysis be done by unskilled staff with no training in formal or structured techniques. Nor are we arguing that sets of checklists and sticky notes attached to documents can supplant rigor and tool support in mission-critical systems. However, for many systems, the implications of NFRs to users are chronic rather than acute, burdensome and inconvenient rather than dangerous or fraudulent, dehumanizing rather than life-threatening. It is for these systems that we claim that STRAP and augmentation methods developed from a similar philosophy are applicable.

6. ACKNOWLEDGMENTS
This work was funded by National Science Foundation ITR Grant #0325269. We thank Joe Tullio, Heather Richter, Carl Cox, and Chris Morris for their help and feedback.

7. REFERENCES