Discovery in Negligence Analysis: Evolution of a Sufficiently Safe Spec

Foaad Khosmood, Clark Savage Turner, J.D., Ph.D.
Department of Computer Science
California Polytechnic State University
San Luis Obispo, CA. 93407 / (805)756-6133
fkhosmoo@calpoly.edu, csturner@calpoly.edu.

Abstract

Negligence liability is a rapidly increasing area of concern for software engineers and designers. The implications of negligence are sufficiently severe that most companies can no longer ignore the legal consequences of engineering design decisions [1][2]. As engineers and industry participants, it is our duty to balance the interests of public safety against our own profits.

This paper continues previous work [3] toward a comprehensive software process augmented to include preliminary legal tasks and milestones. Specifically the implementation of the first part in the proposed model is further explored and suggestions are given. We shall call this step "discovery." Discovery [1] consists firstly of the negligence analysis, before the design work begins. Secondly, it involves the recording of subsequent changes and tradeoffs and the updating of the negligence analysis during the design process itself. This method provides a foundation upon which the legal consequences of design decisions can be judged. Once the product is released, this method will help control and manage possible negligence claims rather than react to them.

Keywords

Software process, negligence, workflow modeling, liability

Introduction

The general social understanding of negligence and the accepted legal processes about negligence occurring within the realm of software engineering are constantly evolving. In fact the collection of software processes and those of the negligence processes can be shown to have a symbiotic relationship as described in [3].

There are typically two ways for a company to face liability in a lawsuit: Strict liability and negligence [2]. Our present focus is the latter in which no contracts exist with end users of a product.[2]

Negligence lawsuits are always concerned with awareness and reasonability of tradeoffs where public safety is at stake. In negligence cases it is often the social reasonability of the specs that is legally in question. The first half of the process described in [3] –called Discovery- is explored here in order to defend the reasonability of the tradeoffs made in the spec.

Traditional safety analysis in industrial production and post-hoc accident investigations provides a rich source of well-established methods for prevention of hazards [6]. These lessons and techniques can be adapted to the realm of negligence analysis in software engineering.

A carefully drafted spec along with equally negligence-conscious software processes will address many of the issues raised here and in [3]. Our contribution is to organize the recording and linking of software processes to legal principles, sources and decisions in order to rationalize the spec to any reader at a later time. In addition we hope to standardize an evolutionary negligence analysis strategy lending itself to automation.

Discovery

Much like the development of software itself, the formulation of a legal strategy should follow evolutionary rationalization as articulated in [4]. The law contains safety principles developed over hundreds of years through the testimony of many engineering experts. These principles should be treated as additional "requirements" for safety critical product design.

The first section of the process laid forth in [3] is the discovery. Keeping in mind that discovery should be

[1] Discovery here is meant in the same sense as the legal term "discovery," or "pretrial discovery," whereby an effort is made to gather as much relevant information as possible before the formation of a legal position.

[2] Warnings and disclaimers do exist on products but these are generally not effective to avoid liability.

conducted at every stage of software development, the goals of this step are:

- first, to consider the impending software development stage as a whole. Gather relevant industrial, academic and social standards. Investigate the codification of these standards into law either as legislation or as prior case law and study the resulting constraints upon engineering processes [5]. The idea is to formulate a legal strategy to exhibit rational tradeoffs respecting public safety; and,
- second, as the development stage is specified, the same research is recursively applied and applicable constraints are updated for each sub-component of that development stage.

The first goal should ideally be realized before engineering work can begin in the development stage in question. What is learned from the legal research could conceivably render the development stage too risky to perform or it could reassess the scope or other characteristics of the project. Preserving this order should be reflected in the design schedules and project timeline calculations. Understanding this strategy to be necessarily imperfect, it should be articulated in such a way as to make future clarifications and modifications [4].

The second goal is realized in form of an interrupt-driven process that runs during the engineering development stage itself. Upon tackling design issues, one would go back and record the background research for any particular methodology one employs. The process closely resembles evolutionary rationalization [4]. Just as the design and specifications evolve during implementation, negligence strategies and legal documentation, too, evolve during the design process. In fact, in both cases, changes should continue to trickle in until the product is released and -in the case of legal analysis - even after that point.

Negligence "Safety Analysis"

Traditionally, failure or negligence analysis occurs post-design: At best before product release and at worst after disaster occurs or negligence is claimed [3]. While this approach is clearly undesirable, it does have the added advantage of hindsight. It is much easier to ask the "right questions" once negligence is already claimed or disaster struck. A fault-tree analysis could easily be constructed once the top node is identified. Such an analysis has already been considered for software testing processes [5]. At its heart discovery is concerned with making the right inquiries and formulating the right responses as early as possible. But how do we predict and judge future negligence outcomes before they occur? What are the right questions to ask? Is there a method or process for determining future undesirable outcomes and guarding against them in the present?

In [6], Suokas lays out several approaches for "search strategies" in industrial safety analysis. With little adjustment, these approaches can be adopted to our discovery process which has a legal focus –specifically negligence- as apposed to a human-safety focus, yielding "negligence safety analysis."

Suokas defines safety analysis as "systematic examination of the structure and functions of a system aiming at identifying accident contributors, modeling potential accidents, and finding risk-reducing measures [6]." With regard to negligence safety analysis, we adopt the same definition replacing the word "accidents" with "negligence."

The main search strategies in safety analysis are forward, backward and morphological analysis [6]. Forward and backward analysis are described as holistic approaches which either project forward possibilities of safety hazards or work backwards from those possibilities identifying potential pitfalls along the way. These approaches are better suited if the analysis is done as a separate entity with the software design and implementation processes frozen in time. Our process – specifically discovery- demands a more fluid and evolutionary approach grown together with all the other aspects of the software production. Morphological search attempts to "concentrate on the factors having the most significant influence on safety [6]." A search for hazard sources is conducted and potential targets are identified and potential troubling paths are constructed and traversed [6].

While Suokas doesn't present the morphological search as being in-order, it is nevertheless well suited for a template driven, recursive strategy such as discovery. In other words it can find potential negligence hot-spots in both high level and low level stages of software development. Instances of the morphological search approach have been standardized in the safety analysis industry and implementation procedures are widely available.

We can revisit the first and the second goals of discovery with morphological search in mind. The first goal aims to do the background research and formulate a broad legal strategy against negligence claims. But how does one even begin to formulate a strategy? The essence of the design stage in question must be examined even though the details are not yet known. This examination provides areas or sub-stages that can be evaluated for their potential legal "hazards." Rough outlines of scenarios can be made to show potential paths to danger and thus a broad strategy is born where certain pitfalls are avoided by changing the direction of the design altogether. One need not be perfect at this stage since the pursuit of the second goal of discovery will necessitate the revisiting of this strategy numerous times. Fine tuning of the

strategy and filling of gaps is to occur during the second goal of discovery coinciding with the engineering design stage itself.

The second goal seeks to identify specific trouble spots in the software design or production stage as it unfolds. An implementation of the second goal of discovery would first seek areas in the design stage that could result in potential negligence cases: Broadly speaking areas leading to user-interaction or potential deviations from a contract or regulation. Once these areas are identified, specific scenarios are explored and corrective action is taken to minimize their possibility of occurrence. When such an action requires a change in the legal strategy or even the specification itself, that change can be accomplished using the feedback loop laid forth in the discovery model.

A Fictional Example

Company X receives requirements from a customer for an inter-hospital wireless patient stats transfer system. The customer would like to record vital statistics and information of an incoming hospital patient electronically and transmit that information to a central server wirelessly. Using hand-held PDA or tablet-PC type devices, the nurses or emergency personnel can enter the data and send it to the central system. A human-assisted decision maker then assigns the load to a doctor on duty and the information gets transmitted to that doctor's hand-held device. Thus by the time the doctor arrives, he would've had a chance to review the information. After the doctor arrives at the scene, further analysis can be done using the system. The doctor, for example can check for potential adverse drug interactions for that particular patient using the handheld interface [7].

Company X creates high level requirements for the user interface (UI) subsystem and development begins. If company X adopts the model in [3], then initial discovery begins before any development. The following could be a subset of questions that must be answered for the first goal of the negligence analysis and research process.

- Is company X legally liable if the UI fails to perform? [8] If so to what extent? Have specific boundaries of liability been negotiated with the customer? Are there contracts available that can clarify this liability domain? What is the standard practice for medical equipment manufacturers and contractors?
- Are there or will there be enough instructions to make sure a reasonable employee uses the system correctly? [8] Does company X have the budget or the means to develop the documentation and training or perform the necessary testing in order to satisfy this reasonability concern? Are there industry certifications that could verify company X compliance in that regard?
- Is there any recent case law where computer UI designers or medical equipment manufacturers have been sued due to malfunction? If yes, was it judged to be negligence? If yes, did the product operations pass the reasonability standard? What was shown to be deficient that could possibly apply to this product? What was the ruling and the remedy recommended by the judge? (Risk analysis.)
- To what extent is the hardware manufacturer or another supplier of the UI device components is liable in case of malfunction? Does company X's contract with its sub-contractors and suppliers cover liability issues as well?
- Are there insurance policies, either locally or as a result of an agreement with another company that could cover liability due to the UI?

No such set of questions can be complete, but as the Therac25 incidents proved they are often not even asked [9]. The idea is to ask the relevant questions and gather the relevant data before the main development activities begin. These questions may not be sufficient or relevant to the final product but they should be as comprehensive as possible. Once that work has begun the second goal of discovery can be realized. The second goal is to make sure the *details* of the development stage and major decisions are recorded and adjustments to the negligence analysis are made in an evolutionary fashion. To continue the previous scenario, company X will make several design decisions once development has started. For instance the designers could decide to use an html based web-page for the UI interface as opposed to a java applet or a proprietary GUI. Once this decision has been made, some additional information is now known about the type of liability that this product may assume. A morphological search can be conducted for the specific area of html web pages in the context of this project. Thus we can revisit the previous analysis and ask additional question or filter out some that are no longer relevant due to this decision. For example, the morphological search could identify "html standardization" as an area of potential negligence hazard. This is because not all browsers interpret html standards alike. A potential scenario can develop whereby a web browser on a particular handheld device is unable to correctly interpret a particular html tag as it was intended and as a result some information does not get displayed and a doctor will not be receiving it. If such information is critical –for example existence of severe allergies detected by the backend database from the patient's files- then the consequences can be devastating. The consideration of this scenario at this level in design will result in corrective action. For example, additional constraints could be placed on the handheld device browsers. Perhaps only one type of browser would be allowed and only certain tags would be utilized by the web server. Testing cases could

be augmented to include every type of transaction just to make sure the displays behave correctly.

For another example, with regard to the case law research, additional searches can be done using new concepts such as "web" and "html" which were previously not known. Likewise liability cases that were pulled before could be reexamined. The ones utilizing an HTML based GUI could be given preference while the ones that had other types of GUI may be discarded or de-emphasized. One may find for example, that a certain case that was deemed relevant before was specific to using a java-based GUI. And a negligence case will not have been contested had the defendant in that case not used Java. Therefore that case is now much less important to company X's legal analysis and research.

Conclusion and Future Work

A software process enhanced to handle negligence liability requires a robust and comprehensive discovery. The strategy and documentation necessary for discovery must be constructed before each design development stage and continually augmented and revised during the development stage. In essence a Parnas style evolutionary tradeoff and documentation process is required for the most detailed and comprehensive results.

Safety analysis provides the language and the processing tools necessary to better achieve the first and second goals of discovery. Many time-honored approaches exist in the area of safety analysis at various stages of development [6]. The application of these safety analysis tools to negligence liability analysis requires a shift of focus from human safety to legal liability. Additionally a shift of implementation from a holistic approach, whereby the whole product or process is examined post-production to a localized approach where every development stage is individually examined recursively. Morphological searching approach can be used as implementation of discovery in order to find potential negligence pitfalls in the design stage.

Future work will continue with the second part of the enhanced development process discussed in [3]. This step which occurs after each traditional development stage, involves evidence generation, archiving and traceability. The discussion in this paper together with the next could serve as the blueprints for the construction of an automated legal assistance system. Such a system could help software designers in liability cases over the lifetime of their products.

References

[1] Turner, Richardson, Software and Strict Products liability: Technical Challenges to Legal Notions of Responsibility, *Proceedings of the IASTED International Conference on Law and Technology*, San Francisco, USA, Oct. 2000.

[2] Turner, C., and Fox, J.K., When Bad Code Comes From Good Specs, *Proceedings of the Sixth IASTED International Conference on Software Engineering and Applications,* Cambridge, USA, Nov. 2002.

[3] Turner, C. and Khosmood, F. Rethinking Software Process: The Key to Negligence Liability, *Proceedings of the Fifth IASTED International Conference on Software Engineering and Applications*, Anaheim, USA, Aug. 2001.

[4] Parnas, A rational Design Process: How and Why to Fake It, *IEEE Transactions on Software Engineering*, No. 2, Feb. 1986.

[5] Turner, Richardson, King, Legal Sufficiency of Testing Processes, *Proceedings of the 15th International Conference on Computer Safety, Reliability and Security*, Vienna, Austria, Oct 1996.

[6] Suokas, J., The Role of Safety Analysis in Accident Prevention, *Accident Analysis and Prevention*, Vol 20. No. 1. , 1988, 67-68.

[7] Buchholz, E., MedApp / Handheld Distributed Computing for Medical Practitioners, unpublished manuscript available from Professor Clark Turner (csturner@calpoly.edu), California Polytechnic State University, Winter 2003.

[8] Kaner, Software Quality & the Law, *The Gate, the newsletter of the San Francisco Section of the American Society for Quality control*, July 1995, 1.

[9] Leveson, Turner, An Investigation of the Therac-25 Accidents, *IEEE Computer*, Vol. 26 No. 7, July 1993.