Research and Teaching Statements

Aaron Massey
Fall 2011

Summary

Legal compliance in software systems is one of the most important problems in the field of software engineering. Laws, regulations, and organizational policies describe societal values that impose serious challenges and requirements to software engineers building systems. My dissertation examines how software engineers can ensure software requirements comply with relevant laws, regulations, and policies. Ensuring that systems comply with laws and regulations is (a) an opportunity to build trust between citizens and the institutions to whom they entrust their information; and (b) an ethical responsibility for any software engineering professional. Furthermore, as software becomes increasingly important to society, governments have increasingly begun to regulate software systems as they are built, deployed, and maintained. Methods, tools, and techniques for evaluating, establishing, or demonstrating legal compliance in software systems will only become more important for the foreseeable future.

My research focuses on developing methods and tools that enable software engineers to build software systems that demonstrably meet or exceed their legal obligations. This is a critical engineering challenge, and I am seeking an academic position at an institution that will enable me to continue to advance the science of regulatory compliance in software systems and legal informatics. My research shows that software engineers are currently ill-equipped to reason about legal compliance, and I am developing methods, tools, and techniques to address this in my research. Moreover, I look forward to educating and inspiring my future students to consciously focus on regulatory compliance as a professional responsibility as well as a form of ethical design. By discussing research focused on real-world problems in the classroom we can inspire our students to engage in scientific discovery and foster a desire to engineer new products that change the world for the better. I believe an academic position is a natural fit for my personality, my creative and analytical mind, and my sincere desire to improve the lives of others.

Dissertation Research

The guiding principle of my research was inspired by Richard Hamming's 1986 talk titled “You and Your Research,” in which he suggests that great scientists should ask themselves three guiding questions about their research:

1. What are the most important problems in your field?
2. Are you working on one of them?
3. Why not?

To this end, the main objective of my work is to aid software engineers ensure that software complies with law. Thus far I have developed empirically validated: (a) techniques for determining which software requirements are legally implementation ready (LIR); (b) metrics to automatically estimate which software requirements are LIR; and (c) tool support for identifying LIR software requirements using legal requirements metrics.

My research in support of these objective has focused on the following research questions:

RQ1: Can software engineers evaluate existing software requirements for legal compliance?
RQ2: Can metrics be used to automatically estimate the legal implementation readiness of software requirements?
RQ3: Does tool support improve the accuracy of human assessment of legal implementation readiness?
For my dissertation work, I conducted separate studies to examine each of these questions. For RQ1, I developed and validated a method for evaluating existing software requirements for legal compliance. This method was developed using the Inquiry Cycle Model [PTA94], ambiguity classification [AEC03], and relationship identification [AEC03]. It takes as an input a legal text and a set of requirements that must comply with that legal text. It produces as an output a set of documented legal compliance concerns that should be addressed to ensure the system meets its legal obligations. It also produces a traceability mapping of requirements to specific subsections within the legal text, which is then used as an input for RQ2.

To examine RQ2, I developed legal requirements metrics, which comprise attributes of a legal text and a set of requirements that must comply with that legal text, such as the percentage of sections within a legal text to which some requirement is traced. These metrics were validated in a case study against expert opinions and analyzed statistically using the Goal / Question / Metric paradigm [Bas95]. I have shown that models based on legal requirements metrics perform well at determining whether a requirement is LIR [MSO11].

As a part of my examination of RQ3, I developed Red Wolf, a legal requirements management tool, to support the use of legal requirements metrics as a part of the software engineering process. I am currently conducting a multi-case study [Yin03] to determine whether use of Red Wolf improves the accuracy of human assessment in three cases: (1) graduate students in software engineering; (2) professional software engineers; and (3) practitioners working in the field of HIPAA compliance of software systems. My data collection for this research is on schedule to be completed during the Fall semester of 2011 with publication to follow in the Spring.

Additional Research

Two themes that reflect my personal values underlie all of my research: engineering ethics and privacy-aware design. As recipient of the Wilkinson Research Ethics Fellowship, I examined the ethics of behavioral advertising. Behavioral advertising is a method for targeting advertisements to individuals based on behavior profiles, which are created by tracking user behavior over a period of time. Individually targeted advertising can significantly improve the effectiveness of advertising. However, behavioral advertising may have serious implications for civil liberties such as privacy. My research examined the ethics of behavioral advertising in the context of recent advances in technology, changes in the political and legal climate, and the history of advertising. By understanding the ethics of a particular technology, such as behavioral advertising, I was able to glean insight into challenges lawmakers face when specifying laws to protect citizens and entrepreneurs developing successful technologies in a competitive marketplace. This interest in privacy-aware design carries over into my dissertation research as evidenced by my focus on privacy- and security-related federal regulations as they govern software systems.

As a part of The Privacy Place research group, I redesigned and implemented a tool to support the analysis of privacy policies called the Privacy Goals Management Tool. Researchers can use this tool to extract high-level privacy goals as stated in online privacy policies, classify those goals, and ensure that these goals are accurately represented in the software system. An important part of this research is identifying goals that appear in numerous privacy policies. If these cross-domain goals can be identified, then they could be used to more clearly communicate actual privacy practices to users.

Future Work

Legal compliance in software systems is not limited to methods for examining existing requirements for compliance or the development of metrics to quickly assess legal compliance. In fact, potential opportunities for this research exist in numerous other areas. Natural extensions to this work include examining compliance in business policy documents, studying the evolution of legal compliance and legal disputes in previous software systems, and improving the tools of open government for the purposes of improved software engineering.
Open government, in particular, is an intriguing area for future work. In my dissertation research I developed an XML format for modeling the HIPAA regulation. Existing tools are not mature enough to be used in serious research projects, but that is changing rapidly. For example, President Obama’s Data.gov initiative has begun releasing the Federal Register, which is the official daily publication for rules, proposed rules, and notices of Federal agencies and organizations including the U.S. Department of Health and Human Services, in a machine-readable format. The research implications of this are enormous. What artifacts will prove helpful to software engineers seeking to build legally compliant software systems? Which will prove useful to policymakers as they attempt to specify laws for technologies they do not fully understand? These are questions that I hope to address in the future.

Teaching

One of the most important lessons I’ve learned in my time at North Carolina State University is an appreciation for excellence in teaching. Standing in front of a classroom and professing to be an expert with the explicit goal of preparing students to be professionals is an immense responsibility. It is not easy, but it is clearly one of the most important tasks our society faces today. In fact, as an educator, I believe education is one of the most powerful forces to impact positive change in society and the world at large.

Educators are responsible for preparing students for success in whatever field they may choose to follow. Prior to attending graduate school, I was a full time systems engineer for Advanced Micro Devices (AMD) working on MIPS-based embedded Linux systems. My three years as a full-time software engineer taught me countless aspects of life as a professional software engineer that I endeavor to pass along to my students. I believe my collective experiences to date, as a practitioner and researcher, uniquely position me to provide guidance based on personal experience about any potential future a computer science student may wish to pursue.

I am confident that I could teach any course in a computer science curriculum. I have direct experience and interest in teaching courses related to software engineering, computer security, digital privacy, requirements engineering, and introductory programming. Moreover, my work experience in systems programming qualifies me to teach courses on open source software development, embedded systems development, and operating systems.

I strongly believe in incorporating and highlighting the following strategies in my teaching:

**Active Learning.** Students must be engaged in hands-on, active exercises while in the classroom. These exercises are critical to ensuring that students retain the information presented and begin to see how it might be applied to their homework, projects, and ultimately to solve real-world problems.

**Team-based projects.** Computer science in general, and software engineering in particular, are team-based endeavors. Students must learn how to work in groups to build something that simply cannot be built by one person. In addition, collaborative projects often result in students learning more from one another than they could ever learn from any individual instructor.

**Real-world relevance.** Students can see the motivation to learn material that is clearly centered on real-world engineering challenges. People connect to material when they can see how it impacts the lives of others. This is why I choose relevant, real-world project domains, such as healthcare, mobile computing, or social networking, when designing in-class examples, homework material, and projects.

**Feedback.** Students learn best when they receive thoughtful feedback in a timely fashion on the work that they have done. When teaching CSC 216—*Programming Concepts in Java*, I worked with my TA to ensure that we provided grades as quickly as possible, particularly for the weekly
quizzes. Although the quizzes were not a large portion of the grade for the course, they were a weekly signpost to the students, the TA, and myself regarding the areas in which students were excelling and those in which they were struggling.

**Reflection.** Numerous important subjects fall through the cracks of the curriculum at almost all universities. As a result, I start each class with a short, interesting topic students may wish to explore further. The purpose of this is to inform students about an important part of computer science or software engineering that they may be absent from the curriculum and encouraging them to explore it further. Examples of these topics from my CSC 216 class include everything from high-level discussions on the differences between IDEs and programmatic text editors to the rationale for using version control software.

**Clarity and Humor.** I have been lucky to learn from an expert in clear communication punctuated with humor to ensure that ideas “sticks” with students. My advisor excels at using humor to clarify a difficult point. This is particularly powerful for those broader points about software engineering and society with which many college students struggle.

As a student at North Carolina State University, I thoughtfully sought teaching opportunities to prepare myself to become a professional educator. I served as a teaching assistant on three separate occasions, and on one of those occasions, I was directly involved with designing a new graduate-level course on Privacy Technology, Policy, and Law—a new and important area of study for graduate students focusing on software engineering. I was also invited as a guest lecturer on more than 10 separate occasions for courses ranging from an introductory undergraduate course on Java programming to several graduate courses on software engineering. Lectures included the following topics: an Introduction to Software Engineering, Software Processes, Software Planning, Risk Management, Project Scheduling, Project Estimation, Requirements Engineering, and Formal Methods.

This past summer, I taught CSC 216—Programming Concepts in Java as the instructor of record. CSC 216 is the second course in computing, intended for majors and students in the Computer Science Certificate Program. Emphasis is placed on encapsulation; methods and types; polymorphism; inheritance; interfaces; testing strategies; linked structures; and specification and implementation of finite-state machines. As the instructor of record, my responsibilities covered the breadth and depth of teaching: hiring, managing, and mentoring a teaching assistant; creating, implementing, and grading compelling and challenging course quizzes, projects, and exams; curriculum planning; lecturing; and assigning course grades.

I approach my responsibilities as an educator humbly recognizing that my knowledge has the power to dramatically improve the futures not only of the students that I teach, but also their families and by extension, our society as a whole. This is not a responsibility to be taken lightly. Educators must embrace it. They must recognize that they cannot blame earnest students for failing to learn in the same way authors cannot blame their readers for failing to grasp their message—the lion’s share of the responsibility lies with the communicator.

**References**


