Jane Hayes (publishes as Jane Huffman Hayes)
Department of Computer Science, hayes@cs.uky.edu
University of Kentucky, phone: (859) 257-3171
Hardymon Building 233, fax: (859) 323-3740
Lexington, KY, 40506-0495

http://www.cs.uky.edu/~hayes

<u>Research Statement – September 2011</u>

Table of Contents

Research

Software systems permeate our society and we entrust them with lives of everyday people on a daily basis. For example, a commuter on a train is trusting that the switching software correctly routes the trains; an airline passenger trusts that developers of air traffic control software and aviation flight control software have built the system correctly; the grocery shopper purchases produce that has been found to be safe and can be tracked back to the distributor and even to the farm using software developed to FDA standards; and patients in a hospital are monitored remotely by software systems. Software is also "big business" and software system issues can cost a company large amounts of lost time, lost current revenue, and lost future revenue due to damaged reputation.

Software is key to the economy of the United States and the World. The software and related industry employed 1.7 million people in the US in 2007, in jobs paying nearly twice the national average, and added more than $260 billion in value to the US economy [software-industry]. "The worldwide PC software market was $88 billion in 2008, representing 30% of the total packaged-software market, which was $297 billion." [software-industry] Kentucky understands that this is an area for potential economic development and has placed emphasis on it "Companies in the high-tech sector specializing in software development are a target industry for [Northern] Kentucky…" [beshear] It is clear that if Kentucky can attract software companies, economic growth will follow. As the land grant University, UK needs to educate the software engineers and researchers who will be employed by such companies. In addition, we need to develop techniques and process that support the development of reliable, dependable, safe software. My work supports this undertaking.

Software engineering is" the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software [sebok]." My research aims at improving the quality, and indirectly safety, of software systems through development of improved software engineering practices and techniques. My work can support the development of better software systems and thus help everyday people as well as businesses to continue their reliance on software to be available, dependable, and safe.

My industry background is in independent verification and validation of mission- and safety-critical software systems. I basically acted as a third party agent for a Government organization who was acquiring or regulating a critical software system (such as for a weapon system or nuclear power plant or electronic funds transfer system). My research follows these interests. In my undertakings, I always strive to develop and deliver a practical solution/approach to problems being experienced in industry/society.

I, along with my research collaborators and students, conduct research in the following areas related to Software Engineering, Software Reliability, and Software Maintainability:

- Traceability;
- Maintainability
- Testing and Reliability: and
- Software Engineering Education.

Traceability

Requirements traceability, defined as ‘the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)” [gotel] is a critical element of any rigorous software development process. For example, the U.S. Food and Drug Administration (FDA) states that traceability analysis must be used to verify that the software design implements all of the specified software requirements, that all aspects of the design are traceable to software requirements, and that all code is linked to established specifications and established test procedures. Similarly, the Federal Aviation Administration (FAA) has established DO-178B as the accepted means of certifying all new aviation software, and this standard specifies that at each and every stage of development “software developers must be able to demonstrate traceability of designs against requirements.” Software Process Improvement standards that are being adopted by many organizations, such as CMMI, require similar traceability practices. Traceability is broadly recognized as an important factor in building rigorous software systems, systems that are often taken for granted by everyday people (see first paragraph of research statement).

Unfortunately, despite this clear criticality, there is almost universal failure across both industry and government projects to implement successful and cost-effective traceability. These difficulties have been broadly attributed to problems associated with creating, maintaining, and utilizing requirements traceability matrices, and also attributed to the incorrect perception by many developers that the effort of tracing exceeds the benefits it returns. My work examines ways to automate traceability recovery to ease the burden of gathering traceability relationship information between various software engineering artifacts (such as between requirements and design, between test cases and code, etc.). I am considered an international expert in the area of software traceability.

Pre-Tenure Work (2001 – 2006)

Our research in traceability is in the application of information retrieval (IR) and data mining techniques to the Verification and Validation (V&V) and Independent Verification and Validation (IV&V) of software systems. The fundamental purpose of IV&V is to ensure that the correct processes have been used to build the right system. To that end, it must be verified that the approved processes and artifacts are guiding development during each lifecycle phase. This is often done by comparing a variety of existing project artifacts, such as requirements specifications, design specifications, user manuals, code, and bug reports to each other, with the purpose of establishing connections (traceability) between different components of these artifacts. For example, one may ensure that every requirement traces to at least one test case that was executed and passed.

We have concentrated on the problem of tracing between pairs of unstructured textual artifacts. Standard industry practice employed for the requirements tracing process is to perform it completely manually or to manually assign keywords to each element of the two document levels and then have software match the assigned keywords in order to generate lists of candidate links (so called until a human analyst approves them). These processes are boring, tedious and, unfortunately, error-prone. Our work offers a practical and effective solution to this tedium. Our approach to the problem is to use Information Retrieval (IR) methods to automatically determine words associated with individual elements and hence the similarity between artifact elements. Our early work adapted IR methods and tested them extensively. We developed Requirements Tracing On target (RETRO), a standalone requirements tracing tool, and released a version to the public via the NASA Goddard Space Flight Center Open Source website. In addition, the backend of RETRO was integrated with SuperTracePlus, an industry requirements and IV&V management tool.

We [RE2003[1]] compared information retrieval methods for tracing with a state-of-the-art process and IR methods were shown to be more robust. We developed requirements for an efficient and effective requirements tracing tool [RE2004] as well as measures to assess if a given tool meets the requirements. We showed how user feedback improves the quality of automated results [RE2004, PROMISE2005, SOFTWARE]. We developed an experimental framework for comparing different tracing experiments and applied it to existing research on automated tracing [IJSEKE]. We applied RETRO to the problem of tracing requirements to defect reports. The results showed that high recall and precision could be achieved (better than traditional methods) [ISSE-Yadla]. Finally, we started addressing the issue of the human analyst “in the loop” of the tracing task [MSR2005]. This work required Collaborative Investigator Training Initiative (CITI) certification as part of the University of Kentucky Institutional Review Board (IRB) process for research involving human subjects. It also requires that up to date and approved research plans and records be maintained.

We founded the Center of Excellence for Software Traceability (COEST) in 2006. The Center was initially funded by NASA through a grant to UK. We developed a mission, a vision, a business plan [COEST-Biz], and a website (www.traceabilitycenter.org). We elected officers. We held a workshop of international researchers and developed the first version of the grand challenges of traceability [Grd-Chlg-1].

[1] Lower case references indicate work of others. References can be found on pages 12 through 16.

Post-Tenure Work (2006 – July 2011)

Examining the tracing problem, it is clear that the human analyst has the final say on whether software engineering artifact elements trace to each other (such as for safety critical systems). In these cases, the analyst will perform tracing manually or will work with the results of an automated tracing tool. The resulting final Trace Matrix (TM) that has been 'approved' by the human is then used to support important activities such as change impact analysis, requirement satisfaction assessment, etc. Our community (including our research group) has been building automated traceability tools that generate high quality TMs (very high recall, good to high precision). It seems intuitive that a high quality tool-generated TM will lead to a higher quality final TM from an analyst. Surprisingly, our work has shown that this is not the case. Starting with an anecdotal study in 2005 [MSR05] we saw that analysts tend to make the final TM worse than the one suggested by the tool – they throw away correct links and keep incorrect links, for example. Ours was the first work to study the role of the human analyst in the tracing process. We coined the phrase "study of the analyst" to refer to this work, as opposed to "study of the method."

To pursue the study of the analyst, we have undertaken a series of experiments (some analysts used automated tools, some analysts performed manual tracing) where TMs of varying initial quality were given to analysts for vetting [RE2010C, RE2011]. These studies were conducted at two universities (UK and CalPoly) over a period of two years with 84 analysts participating. We found that no single participant recovered the perfect TM, though every link in the perfect TM was recovered by at least one analyst. We found that the TMs submitted by the analysts converged to a "hotspot" in the accuracy interval of [0.6, 0.75] f2-measure. Consensus is being formed, but in a region well below perfect accuracy. Analyst behavior is dependent on the quadrant of the starting TM (high recall, low precision – analysts weeded out bad links but also tossed out some good ones, yielding higher precision but lower recall; high recall, high precision – some analysts added bad links, others removed good links; low recall, low precision – analysts replaced bad links with good ones (the desired behavior); low recall, high precision – analysts added both good and bad links thus yielding higher recall but worse precision). Also, we again noted that the manual tracers produced TMs with higher precision (this was first noted in our earliest work [RE2003]).

In addition, statistical analysis was performed on the collected data to understand the factors may influence the final quality of the TM [RE2011], such as analyst tracing experience, effort expended on the task, etc. We found that the initial quality is the most important factor impacting the final quality of the TM [RE2011].

We also developed a model for examining the effort required to apply a TM vetting technique (such as examining the Top 2 candidate links in the list of each parent element, or ordering all candidate links by relevance weight and then examining each in descending order), thus supporting comparison of such techniques [TEFSEeffort]. Our earlier work showed, and newer work has reconfirmed, that the most important factor impacting effort is whether a human analyst knows when to stop examining candidate links (that is, the analyst feels that all links for a parent element have been found and stops looking further) [PROMISE-Larsen, TOSEM-in-preparation].

In addition to the work and publications mentioned above, we prepared a chapter on the study of the analyst for a book on traceability. It documented our prior work as well as some new work and insights into performing empirical study of the analyst role in tracing [Springer-Analyst].

Shifting gears to "study of the method," we developed REquirements TRacing On target (RETRO).NET [ISSE], a scaled down version of RETRO with improved usability. This has been made available to NASA as well as to select researchers. In addition, we developed a logging version of RETRO.NET and examined the individual steps taken by each analyst (after the experiments were run). We proposed a visualization scheme which shows the analysts 'moves' toward or away from a correct final TM [CHASE] – a decision to reject a correct link would constitute an analyst 'move' away from a correct final TM, for example. Based on our observations from these studies, we suggested a course of action for working with analysts: embrace the analyst (accept that the human analyst is fallible and attempt to predict and account for the fallibility); quarantine the analyst (perhaps prohibit the analyst from making certain types of decisions, treat the analyst as an "untrusted" component); change the analyst (educate and train analysts to decrease/remove the fallibility); and eliminate the analyst (use fully automated techniques, which may not be permitted in some circumstances and which is not yet feasible due to inaccuracy of such techniques) [NIER].

We developed a term proximity method for trace generation [EMPIRE]. We developed two swarm techniques for generating trace matrices [RE2010]. These methods were able to generate TMs of higher quality than those generated by the Vector Space Model with term frequency (tf)-inverse document frequency (idf) weighting for some datasets. This work was honored as one of the top five papers in the IEEE Conference on Requirements Engineering 2010 and appeared in a special issue of the Requirements Engineering Journal [REJ2011]. A shorter description of this work was one of 16 papers accepted to the ACM Student Research Competition of ICSE 2011 [ACM-SRC].

We developed two techniques to work with TMs. One technique evaluates TMs using a committee of methods [RE2007]. By building voting committees from five IR methods, we used a variety of voting schemes (such as majority, super-majority, consensus) to accept or reject links from given candidate TMs. We found that the method found and rejected a large percentage of the false positives in TMs generated by an automated method, but did not perform as well on the TMs generated manually by human analysts [RE2007]. We also developed a technique for assessing whether requirements were satisfied given an existing TM [RE2009]. Requirement satisfaction assessment is an important activity for mission- or safety-critical software systems.

It is often the case that heritage systems lack documentation (requirements, design, etc.) but are still being used and maintained. We developed a method for assisting engineers to recover requirements and then to decide if they should reuse parts of the heritage system or rewrite it from scratch in order to “keep up” with competitor systems [ICSM2008]. This work applied static and dynamic tracing techniques. We developed a number of secondary measures for evaluating TMs and also looked at the impact of vocabulary base on the LSI technique [REJ2010]. We found that traditional measures, such as recall and precision, might evaluate two candidate TMs as being of equal quality, but the secondary measures might find one TM of much higher quality. For example, the true links might be at the top of the lists for one TM but not for the other. The secondary measures evaluate TMs from the perspective of a human analyst who must work with the TM.

The Center of Excellence for Software traceability (COEST) has continued to play an important role in our work and in our research community. We received NSF funding to hold a follow-on grand challenges workshop in 2007 (a special offering of Traceability in Emerging Forms of Software Engineering (TEFSE)). A group of COEST researchers were able to obtain a large NSF grant, partly due to groundwork laid by the COEST. It became clear that there was a need for sharing of data, benchmarks, etc., we shared our ideas on this with the community at TEFSE 2007. [TEFSE2007-benchmk]. As a result, the community initiated a traceability challenge at the TEFSE workshop, akin to the challenges at the Text Retrieval Conference (TREC). Our research group won the first challenge at TEFSE 2009 with our RETRO.NET tool [TEFSE-Chlg-2009]. Subsequently, we updated the grand challenges to include research goals, evaluation plans, technology transfer plans, etc. [Springer-Chlg].

We are developing an intermediate language called Temporal Action Language (TAL) to assist with consistency checking of temporal requirements that are specified in natural language using answer set programming as the back end. The process proceeds as follows. A group of related requirements identified as temporal are retrieved from the trace matrix. Next, they are translated into Temporal Action Language (TAL). The TAL statements are then translated into programs for clingcon [clingcon], a combination of answer set programming and constraint solving. The programs are then run. If an answer set is found, the requirements are consistent for the specified time horizon. If no answer set can be found, the requirements are not consistent. We have based TAL on action languages by Baral and Gelfond [baral]. Our language is more readable than other formal languages such as Linear Temporal Logic [ltl] and action languages and permits specification of events more than one time step prior or subsequent. We are currently completing the syntax and semantics of TAL. Though this is work in progress, we have performed some early validation/proof of concept work. We have applied our proposed technique to a networking case study, inspired by a NASA software system. We manually translated the natural language requirements into TAL and manually translated TAL into clingcon programs. We then ran the programs in clingcon and examined the output. The technique was able to identify inconsistencies for certain time horizons (correctly) and did not find issues with other time horizons (correctly). To date one paper on this work has been accepted and one is in submission [ASE, CISE].

Contributions

The research contributions in this area are summarized here (as it is my main area). We were the first to introduce feedback to augment IR methods for trace generation. We were the first to undertake study of the analyst interacting with the results of an automated traceability tool. We introduced requirements for a requirements tracing tool as well as measures for their evaluation. We were the first to examine the impact of initial quality on the final quality of TMs. We were the first to use heatmaps to show the movement of TMs worked on by analysts (these now show up in other submissions on tracing) [RE2010C]. We were the first to propose ratings for the quality of initial traces (what constitutes "good" recall, "good" precision, etc.) as well as the first to propose categories for the size of datasets (small, large) [IJSEKE][TSE]. We introduced a committee of methods to evaluate a trace matrix generated by an automated trace tool. We examined different ways that an analyst might vet a trace matrix, with emphasis on reduced effort for the analyst. We developed secondary measures to help assess the internal quality of candidate link lists from the analyst's perspective [REJ2010].

We developed the concept of a Center of Excellence for Software Traceability. We pitched it to our colleagues, we got funding, we developed a mission, vision, business plan, developed a website, and grew the community. We assisted with the development of grand challenges, obtaining funding to support a grand challenges community meeting. We developed numerous data sets and made them available to the community.

Future Work

Future work will proceed in a number of directions. First, we will pursue the study of the analyst. This work will include controlled study of a number of factors that may influence analyst behavior on the small percentage of links that are not being retrieved by automated tools and/or is not being vetted properly by analysts, so called "hard" links or "stubborn" traces. We will examine the proposed strategies for dealing with the fallibility of analysts (change, quarantine, embrace, eliminate the analyst). Two, we will continue to purse applications of TMs along the lines of our work on requirements satisfaction assessment, reuse analysis, and TM evaluation. Possible areas for study include change impact analysis, criticality assessment, and regression testing.

Support and Results

Collaborators. This research is being conducted with a host of collaborators including, but not limited to, Alex Dekhtyar of CalPoly and his students, Giulio Antoniol and Yann-Gael Geuheunec of Ecole Polytechnique de Montreal and their students, Max di Penta of University of Sannio, Jane Cleland-Huang of DePaul University, Jonathan Maletic of Kent State University, Andrea Zisman of City University (London), Dan Port of University of Hawaii, and Allen Nikora of JPL/CalTech.

Student Involvement. Two PhD students who have graduated were involved in this work (Senthil Sundaram (Microsoft) and Ashlee Holbrook (Lexmark)). Six PhD students have worked or are working on this project at present: Wei-Keat Kong, Hakim Sultanov, Wenbin Li, Bill Kidwell, Mark Hays, and Jody Larsen. A number of M.S. students have worked on the project: Sravanthi Vadlamudi, Sarah Howard, and Suresh Yadla. Six undergraduate students have participated on this project, many of them as REU participants: James Osborne, Alex Wilson, Aaron Jones, Matt Smith, Marcus McAllister, and Jesse Yanelli.

Research dissemination. To date, this research has yielded nine journal papers [REJ2011, REJ2010, ISSE, ISSE-Yadla, TSE, SOFTWARE, IJSEKE], two book chapters [Springer-Analyst, Springer-Chlg], twenty-eight refereed conference and workshop papers (see CV), a number of technical reports ([COEST-Biz], for example), and several invited presentations ([ICSE-Tech-Brf], for example). Our work has received a few awards: NASA's The Buzz Award, NASA's Best Research Award, and NASA's Best Paper Award. We also had a paper selected as one of the Top 5 papers of RE 2010. One of our PhD students (Sultanov) was among the 16 students selected for the ICSE 2011 ACM Student Research Competition.

Support. This work was previously funded by five NASA grants and one NSF workshop grant. It is currently funded by two NSF grants, including a Major Research Instrumentation (MRI) grant. We have also had a number of REU supplements from NSF for undergraduate research.

Maintainability

"Our increasing dependence on software should be supported by highly reliable, dependable, easily modifiable applications, but this is not the case. The demand for software changes is becoming larger and faster, but we are not able to keep up. We only seem to be able to produce buggy, unreliable software, even if we are producing more of it faster. Web-based software has further exacerbated the problem by decreasing software development cycle times, challenging maintainers to modify running software (operating 24×7), increasing application complexity and performance requirements, etc. Similarly, the increased demand for software and a shortage of adequately trained personnel has further contributed to poor software quality.

Software maintainability is defined as 'the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment' [ieee]. Maintained software refers to software systems or components that are currently in use and are modified (we include perfective, adaptive and corrective maintenance) on a regular basis. Estimates from as far back as 1980 still hold true today; software maintenance typically accounts for at least 50% of the total lifetime cost of a software system [lientz]. Schach et al. found that 53–56% of personnel time is devoted to corrective maintenance [schach].

However, all is not lost. We can view maintainability as an opportunity, an opportunity to improve how we build software so that it is more easily maintained." [OMAJSME]

My interest in software maintainability developed in my first position after college: maintenance programmer at the Defense Intelligence Agency. I maintained a software system written in FORTRAN-66 that lacked documentation and lacked meaningful variable names. My research work in this area focuses on ways to automatically identify software that is not easy to maintain, primarily using models built based on static measures that can be retrieved from the code (such as cyclomatic complexity, number of comments, coupling between objects (CBO) [ckmetrics]).

Pre-Tenure Work (2001-2006)

For my Master's thesis, I developed a technique for determining whether sufficient in-line documentation exists in a piece of code to ease its future maintenance (understandability aspect of maintainability) [ICSM1988]. In 2003, we developed a lightweight improvement process for software maintenance called Observe-Mine-Adopt and applied it in industry. This work also included the development of two new measures for maintainability, one subjective (Perceived Maintainability) and one objective (Maintainability Product) [OMAJSME]. These measures have been adopted by one industrial partner and have been used in building several maintainability models.

In 2004, we began working on developing models of maintainability. One direction we examined was effort estimation for maintenance, specifically for software enhancements [CSMR2004]. We also examined whether or not we could build a model to predict code modules that would be easy to change (or hard to change) [ICSM2005]. We also examined maturity models for software maintenance with Alain April and Alain Abran in [JSME].

Post-Tenure Work (2006-2011)

Our industrial experience indicated that certain types of components tend to have certain types of faults, we call these relationships *fault links*. If we can understand the fault links that exist historically for a given software system or domain, we can use this knowledge to focus our code walkthroughs and unit tests based on the type of component (basically performing fault-based verification). We found evidence of fault links in an open source system [EDCC]. We then examined the benefit of fault links when performing code walkthroughs. We asked analysts to examine two different systems (some analysts used fault link walkthrough sheets, some did not) and found that analysts performing code reviews with fault link information were able to find more defects and more 'hard to detect' defects than those without [STVR-Fault].

We developed a method for determining what methods and classes require refactoring, and in what order based on static metrics and past history [ISSE-Refactor]. We developed a method for determining the defect type of a code change based on the history of the code revision (compares the current code to the code before the change and then uses a machine learner to classify the fault type) [ICSE-SRC].

Future Work

We will examine the broader context of fault links, something we call fault chains. These are basically fault links that occur across product lines (a fault that occurs in product 1 and then shows up again in downstream products 2 and 3) or in different versions of a software system (fault occurs in version 1 of software system X and then again in version 2 of software system X) or from one life cycle phase to another of a product (occurs in the requirements phase, occurs again (latent or re-introduced) in the code phase). If we can find evidence of fault chains, we can tailor our verification and validation techniques to focus on these faults.

Support and results

Collaborators. Collaborators include Alain April and Alain Abran of Ecole de Technologie Superieure (ETS) Universite du Quebec, Anneliese Andrews of University of Denver, and Max di Penta of University of Sannio.

Student Involvement. Two PhD students and one graduated PhD student (Liming Zhao, Billy Kidwell, and Dr. Ashlee Holbrook) as well as many MS students (Tina Gao, Naresh Mohamed, Sandip Patel, and Inies Chemmannoor) have worked in this area.

Research dissemination. To date, this research has yielded three journal papers [JSME, OMAJSME, ISSE-Refactor] and six conference or workshop papers. A PhD student (Kidwell) was one of the 16 students selected for the ICSE 2011 ACM Student Research Competition.

Support. This work was funded this year by NASA as part of a three year grant with JPL ($210K for three years). Previously the work was performed unfunded.

Testing and Reliability

"Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. [hetzel] Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation." [jiantao]

The latter part of my industry career involved performing and/or managing independent testing as part of Independent Verification and Validation (IV&V) of mission- and safety-critical systems such as nuclear power plant instrumentation and control subsystems and weapon systems. This interest translated into my PhD dissertation on input validation testing.

Pre-tenure Work (2001 – 2006)

I developed a technique for detecting input validation faults early in the lifecycle as well as demonstrated that such faults often go undetected [EMSE-IVT, 3 conference papers]. Dr. Offutt and I developed a semantic fault model that has been widely used in testing research. This characterizes a fault not by its syntactic size (number of tokens that must be corrected in order to obtain a "correct" program) but by its semantic size (estimated as the number of inputs for which incorrect behavior is observed) [ISSTA]. Separately from Dr. Offutt, I developed a fault-based method for testing object-oriented software systems [ISOOMS].

Post-tenure work (2006 – 2011)

The fault link papers mentioned above under Maintainability also include aspects of verification and validation, specifically their use for code walkthroughs and unit testing. The journal paper focused on verification of code modules based on fault links, and is counted under Testing and Reliability [STVR-Fault]. We proposed the Consistent Programmer Hypothesis that posits that experienced programmers exhibit a particular style or voice in the code that they write that can be detected using static and dynamic measures [STVR-CPH]. We studied professional programmers as well as graduate student programmers (some also professionals) and found evidence to support the hypothesis. We also examined small programs written by the authors and then attempted to identify other programs written by the authors (using PCA and LDA). We were able to correctly classify 88% (of experienced programmers) [IJCA]. This could be useful in authorship attribution for copyright protection, software forensics, etc.

Future Work

We have a PhD student early into his studies who is examining the semantic fault model as a way to determine overlap of testing techniques.

Support and results

Collaborators. Collaborators include Jeff Offutt of George Mason University and Dave Pruett of NASA.

Student Involvement. One graduated PhD student (Ashlee Holbrook), one current PhD student (Mark Hays), and one graduated MS student has supported this work (Inies Chemannoor).

Research dissemination. To date, this research has yielded four journal papers [STVR-Fault, EMSE-IVT, STVR-CPH, IJCA] and nine conference or workshop papers.

Support. The work was funded this year by an industrial partner (funded one PhD student for the calendar year). We also received one multiyear grant from JSC for $194K for fault based analysis.

Software engineering education

My interest in educating/training software engineers dates back to the early '90s when I began to deliver training courses for SAIC as part of the corporation's effort to attain a CMM Level 3 rating. Upon arriving at the University, I began to read software engineering education papers, websites, etc. to gain an understanding of the field as I worked to develop new courses and the software engineering curriculum. My work here focuses on engaging students through real world projects as well as examining how to use research as part of the learning experience for software project courses.

Pre-tenure work (2006 – 2011)

My research in this area has been multi-faceted. I became interested in using my course projects as real-world experiments when I first arrived at the University [CSEET2002]. This resulted in a project framework that I have since used for my experiments in all areas of research [see IJSEKE, for example].

I became interested in on-line course delivery and collaborated with one of my Ph.D. students [JAME]. As I have taught a number of courses with a semester-long team project component, I became interested in how to evaluate an individual student's contribution to a group project. This resulted in practical guidelines on grading such projects [ICSE2003]. Next, I became interested in whether or not current evaluation schemes adequately capture the potential of a student to succeed as a software engineer. We examined several potential measures and find that current grading schemes do NOT capture important information needed to predict student success [CSEET2006]. Therefore, we proposed more detailed measures to capture this information.

Post-tenure work (2006 – 2011)

We documented Nancy Mead's contributions to the software engineering education field, specifically focusing on her security method called System Quality Requirements Engineering (SQUARE) [CSEET2008].

Future work

Currently we have been incorporating professional certification preparation into our Computer Science courses (such as for the QAI's Certified Associate Software tester certification) and plan to continue this as well as evaluate its impact.

Support and results

Collaborators. Tim Lethbridge, Dan Port, and Alex Dekhtyar.

Student Involvement. The students working on traceability research also assisted here as did Mary Biddle (a PhD student in our department).

Research dissemination. To date, this research has yielded one journal paper and three conference or workshop papers.

Support. This work is not funded.

References

[RE2010C] David Cuddeback, Alex Dekhtyar, Jane Huffman Hayes, "Automated Requirements Traceability: the Study of Human Analysts," Proceedings of IEEE International Conference on Requirements Engineering (RE), September 2010, Sydney, Australia, RE 2010: 231-240.

[RE2011] Alex Dekhtyar, Olga Dekhtyar, Jeff Holden, Jane Huffman Hayes, David Cuddeback and Wei-Keat Kong, "On Human Analyst Performance in Assisted Requirements Tracing: Statistical Analysis," to appear in Proceedings of IEEE International Conference on Requirements Engineering (RE) 2011, Trento, Italy.

[MSR2005] Jane Huffman Hayes, Alex Dekhtyar, Senthil Sundaram, "Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results," Proceedings of Workshop on Mining of Software Repositories (MSR), associated with ICSE 2005, St. Louis, MO, May 2005.

[CHASE] Wei-Keat Kong, Jane Huffman Hayes, Alex Dekhtyar, Jeff Holden, "How Do We Trace Requirements? An Initial Study of Analyst Behavior in Trace Validation Tasks," in Proceedings of Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) 2011, an ICSE workshop.

[NIER] David Cuddeback, Alex Dekhtyar, Jane Huffman Hayes, Jeff Holden, Wei-Keat Kong, "Towards Overcoming Human Analyst Fallibility in the Requirements Tracing Process", in Proceedings of the International Conference on Software Engineering (ICSE 2011), New and Innovative Emerging Results (NIER) Track.

[TEFSEEffort] Alex Dekhtyar, Jane Huffman Hayes, Matt Smith, "Towards a Model of Analyst Effort for Traceability Research: A Position Paper," in Proceedings of Traceability of Emerging Forms of Software Engineering (TEFSE) 2011, an ICSE workshop.

[RE2007] Alex Dekhtyar, Jane Huffman Hayes, Senthil Sundaram, Ashlee Holbrook, Olga Dekhtyar, "Technique Integration for Requirements Assessment," pp.141-150, Proceedings of the 15th IEEE International Requirements Engineering Conference (RE 2007), 2007.

[RE2009] E. Ashlee Holbrook, Jane Huffman Hayes, Alex Dekhtyar, "Toward Automating Requirements Satisfaction Assessment," Proceedings of Requirements Engineering, IEEE International Conference on, pp. 149-158, 2009 17th IEEE International Requirements Engineering Conference, 2009.

[REJ2010] Senthil Karthikeyan Sundaram[2], Jane Huffman Hayes, Alex Dekhtyar, and E. Ashlee Holbrook[3], "Assessing Traceability of Software Engineering Artifacts," Requirements Engineering Journal, Volume 15, Issue 3 (2010), Page 313-335, August 2010.

[ICSE-SRC] Bill Kidwell, "A Decision Support System for the Classification of Software Coding Faults: A Research Abstract," ACM Student Research Competition, ICSE 2011.

[ISSE-Refactor] Liming Zhao, Jane Huffman Hayes, Rank-based Refactoring Decision Support: Two Studies, accepted to Innovations in Systems and Software Engineering: A NASA Journal (ISSE), July 2011.

[2] This was after Senthil Sundaram earned his PhD
[3] This was after Ashlee Holbrook earned her PhD

[RE2010] Hakim Sultanov, Jane Huffman Hayes, “Application of Swarm Techniques to Requirements Engineering: Requirements Tracing,” Proceedings of IEEE International Conference on Requirements Engineering (RE), September 2010, Sydney, Australia, RE 2010: 211-220.

[REJ2011] Hakim Sultanov, Jane Huffman Hayes, and Wei-Keat Kong, “Application of Swarm Techniques to Requirement Tracing,” Special Issue of Requirements Engineering Journal as one of the Top 5 papers at IEEE International Conference in Requirements Engineering (RE) 2010, accepted April 2011, online June 2011.

[ACM-SRC] Hakim Sultanov, “Requirements Tracing: Discovering Related Documents through Artificial Pheromones and Term Proximity,” ACM Student Research Competition, ICSE 2011.

[EMPIRE] Wei-Keat Kong and Jane Huffman Hayes, “Proximity-Based Traceability: An Empirical Validation using Ranked Retrieval and Set-based Measures,” to appear in Proceedings of Empirical Research in Requirements Engineering Workshop (EMPIRE 2011), associated with IEEE RE Conference.

[EDCC] Jane Huffman Hayes, Inies Chemannoor, Vinod Surisetty, and Anneliese Andrews, “Fault Links: Exploring the Relationship Between Module and Fault Types,” Proceedings of European Dependable Computing Conference (EDCC), Budapest, Hungary, April 2005.

[STVR-Fault] Jane Huffman Hayes, Inies R. Chemmannoor, and E. Ashlee Holbrook, "Improved Code Defect Detection with Fault Links," Journal of Software Testing Verification and Reliability, Article first published online: 15 JAN 2010, DOI: 10.1002/stvr.426.

[ISSE] Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram, E. Ashlee Holbrook, Sravanthi Vadlamudi, and Alain April, "REquirements TRacing On target (RETRO): Improving Software Maintenance Through Traceability Recovery," Innovations in Systems and Software Engineering: A NASA Journal (ISSE), Vol. 3, No. 3, pp. 193-202, 2007.

[RE2003] Jane Huffman Hayes, Alexander Dekhtyar, James Osbourne, “Improving Requirements Tracing via Information Retrieval,” in Proceedings of the International Conference on Requirements Engineering (RE), Monterey, California, September 2003, pp. 138 – 148.

[RE2004] Jane Huffman Hayes, Alexander Dekhtyar, Senthil Sundaram, Sarah Howard, “Helping Analysts Trace Requirements: An Objective Look,” in Proceedings of IEEE Requirements Engineering Conference (RE) 2004, Kyoto, Japan, September 2004, pp. 249-261.

[PROMISE2005] Senthil Sundaram, Jane Huffman Hayes, Alex Dekhtyar, ”Baselines in Requirements Tracing,” Proceedings of Workshop on Predictive Models of Software Engineering (PROMISE), associated with ICSE 2005, St. Louis, MO, May 2005.

[SOFTWARE] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan Sundaram, "Improving After-the-Fact Tracing and Mapping: Supporting Software Quality Predictions," IEEE Software, Vol. 22, No. 6, pp. 30-37, November/December 2005.

[PROMISE-Larsen] Alex Dekhtyar, Jane Huffman Hayes, Jody Larsen, “Make the Most of Your Time: How Should the Analyst Work with Automated Traceability Tools?” In Proceedings of the Third international Workshop on Predictor Models in Software Engineering (May 20 - 26, 2007), Minneapolis, MN, International Conference on Software Engineering.

[TOSEM-in-preparation] Alex Dekhtyar, Jane Huffman Hayes, Jody Larsen , “Effective Use of Analyst Effort in Automated Tracing,” to be submitted to TOSEM in Fall 2011.

[IJSEKE] Jane Huffman Hayes and Alex Dekhtyar, "A Framework for Comparing Requirements Tracing Experiments," International Journal on Software Engineering and Knowledge Engineering (IJSEKE), Vol. 15, No. 5, pp. 751-781, October 2005.

[ISSE-Yadla] Suresh Yadla, Jane Huffman Hayes, and Alex Dekhtyar, "Tracing Requirements to Defect Reports: An Application of Information Retrieval Techniques," Innovations in Systems and Software Engineering: A NASA Journal, Vol. 1, No. 2, pp. 116-124, September 2005.

[Springer-Analyst] Alex Dekhtyar and Jane Huffman Hayes, "Studying the Role of Humans in the Traceability Loop," (28 page chapter) accepted in June 2011 to "Software and Systems Traceability," edited book by Springer-Verlag, peer reviewed.

[ICSM2008] Giuliano Antoniol, Jane Huffman Hayes, Yann-Gaël Guéhéneuc, Massimiliano Di Penta: "Reuse or rewrite: Combining textual, static, and dynamic analyses to assess the cost of keeping a system up-to-date." Proceedings of International Conference on Software Maintenance (ICSM) ICSM 2008: 147-156.

[TEFSE-Chlg-2009] Jane Huffman Hayes, Wei-Keat Kong, Wenbin Li, Hakim Sultanov, Steven Alexander Wilson, Sami Taha, Jody Larsen, Senthil Sundaram., "Software Verification and Validation Research Laboratory (SVVRL) of the University of Kentucky: Traceability Challenge", (2009). Workshop paper published – UK "won" the Challenge. Held at 2009 Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE) (May 18 - 18, 2009), an ICSE workshop.

[Springer-Chlg] Orlena Gotel, Jane Cleland-Huang, Jane Huffman Hayes, Andrea Zisman, Alexander Egyed, Paul Gruenbacher, Alex Dekhtyar, Giulio Antoniol, Jonathan Maletic and Brian Berenbach, "The Grand Challenge of Traceability," (74 page chapter), accepted in June 2011 to "Software and Systems Traceability," edited book by Springer-Verlag, peer reviewed.

[Grd-Chlg-1] Orlena Gotel, Jane Cleland-Huang, Jane Huffman Hayes, Andrea Zisman, Alexander Egyed, Paul Gruenbacher, Alex Dekhtyar, Giulio Antoniol, Jonathan Maletic and Brian Berenbach, "Problem Statements and Grand Challenges," 08-23-2006, COET-GCT-06-01-0.5, available at www.coest.org.

[COEST-Biz] Jane Huffman Hayes, Alex Dekhtyar, Jane Cleland-Huang, Charter (Business Plan) for the Center of Excellence for Traceability, COET-CBP-07-02-1.0, 2/15/07.

[baral] C. Baral and M. Gelfond, "Reasoning agents in dynamic domains," Logic-based artificial intelligence, pp. 257–279, 2000.

[ltl] Yde Venema, Lou Goble, ed. The Blackwell Guide to Philosophical Logic. Black well, 2001

[clingcon] http://www.cs.uni-potsdam.de/clingcon/

[ASE] Wenbin Li, Toward Consistency Checking of Natural Language Temporal Requirements, to appear in proceedings of Doctoral Symposium of Automated Software Engineering (ASE) in November 2011.

[CISE] Wenbin Li, Mirek Truszczynski, Jane Huffman Hayes, "Toward Consistency Checking of Natural Language Temporal Requirements," submitted to 2011 International Conference on Computational Intelligence and Software Engineering (CiSE 2011).

[ICSE-Tech-Brf] Jane Cleland-Huang and Jane Hayes, "Requirements Traceability in Software Intensive Systems," Technical Briefing at International Conference on Software Engineering (ICSE), May 2011.

[ICSM1988] Jane Huffman (Hayes) and Cliff Burgess, "Partially Automated In-Line Documentation (PAID): Design and Implementation of a Software Maintenance Tool," published in The Proceedings of the 1988 IEEE Conference on Software Maintenance (ICSM), pages 60-65, Phoenix, Arizona, October 1988

[OMAJSME] Jane Huffman Hayes, Naresh Mohamed, Tina Hong Gao, "Observe-Mine-Adopt (OMA): An Agile Way to Enhance Software Maintainability," Journal of Software Maintenance and Evolution: Research and Practice, Vol. 15, No. 5, pp. 297-323, October 2003.

[CSMR2004] Jane Huffman Hayes, Sandip Patel, and Liming Zhao, "A Metrics-Based Software Maintenance Effort Model," In Proceedings of the 8th European Conference on Software Maintenance and Reengineering, Tampere, Finland, March 2004. pp. 254-258.

[ICSM2005] Jane Huffman Hayes, Liming Zhao, "Maintainability Prediction: A Regression Analysis of Measures of Evolving Systems," Proceedings of IEEE International Conference on Software Maintenance (ICSM) 2005, Budapest, Hungary, September, 2005.

[JSME] Alain April, Jane Huffman Hayes, Alain Abran, Reiner Dumke, "Software Maintenance Maturity Model (SM^{mm}): The Software Maintenance Process Model," Journal of Software Maintenance and Evolution: Research and Practice, Vol. 17, No. 3, pp. 197-223, May-June 2005.

[EMSE-IVT] Jane Huffman Hayes and Jeff Offutt[+], "Input Validation Analysis and Testing," Journal on Empirical Software Engineering, 11(4):493-522, December 2006.

[ISOOMS] Jane Huffman Hayes, "Testing Object-Oriented Systems: A Fault-Based Approach," published in The Proceedings of the International Symposium on Object-Oriented Methodologies and Systems (ISOOMS*)*, Springer-Verlag Lecture Notes on Computer Science series, Number 858, pages 205-220, Palermo, Italy, September 1994.

[ISSTA] Jeff Offutt[+] and Jane Huffman Hayes, "A Semantic Model of Program Faults," published in The Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), pages 195-200, ACM, San Diego, California, January 1996.

[STVR-CPH] Jane Huffman Hayes and Jeff Offutt□, "Recognizing Authors: An Examination of the Consistent Programmer Hypothesis," Journal of Software Testing Verification and Reliability, Article first published online: 16 JUL 2009, DOI: 10.1002/stvr.412, hardcopy Volume 20, Issue 4, pp. 329 - 356.

[IJCA] Jane Huffman Hayes, "Authorship Attribution: A Principal Component and Linear Discriminant Analysis of the Consistent Programmer Hypothesis," International Journal on Computers and their Applications (IJCA), Vol. 15, No. 2, pp. 79-99, 2008.

[CSEET2002] Jane Huffman Hayes, "Energizing Software Engineering Education through Real-World Projects as Experimental Studies," in Proceedings of the 15th *Conference on Software Engineering Education and Training (CSEET),* Covington, KY, February 2002, pp. 192-206.

[JAME] Sandy Patel and Jane Huffman Hayes, "Case Study: Teaching an Electronic Course," Journal for the Advancement of Marketing Education, Volume 2, Number 1, Summer 2002.

[ICSE2003] Jane Huffman Hayes, Tim Lethbridge, and Dan Port, "Evaluating Individual Contribution Toward Group Software Engineering Projects," in Proceedings of the *International Conference on Software Engineering (ICSE),* Portland, Oregon, May 2003, pp. 622-627.

[CSEET2006] Jane Huffman Hayes, Alex Dekhtyar, Ashlee Holbrook, Olga Dekhtyar, Senthil Sundaram, "Will Johnny/Joanie Make a Good Software Engineer?: Are Course Grades Showing the Whole Picture?," Proceedings of Conference on Software Engineering Education and Training (CSEET), Oahu, HI, April 2006.

[sebok] SWEBOK executive editors, Alain Abran, James W. Moore ; editors, Pierre Bourque, Robert Dupuis. (2004). Pierre Bourque and Robert Dupuis. ed. Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7.

[gotel] Gotel O.C.Z and Finklestein A.C.W., "An analysis of the requirements traceability problem", in Proceedings of ICRE94, 1st International Conference on Requirements Engineering, Colorado Springs, Co, IEEE CS Press, 1994

[+] My advisor

[ieee] IEEE. Authoritative Dictionary of IEEE Standards Terms, ANSI/IEEE Std. 100. Institute of Electrical and Electronic

Engineers, New York NY, 2000.

[linetz] Lientz BP, Swanson EB. Software Maintenance Management. Addison-Wesley: Reading MA, 1980.

[schach] Schach S, Jin B, Wright D, Heller G, Offutt AJ. Determining the distribution of maintenance categories: survey versus empirical study. Kluwer's Empirical Software Engineering 2003; 8(4) in press.

[ckmetrics] Chidamber Shyam, Kemerer Chris, "A metrics suite for object oriented design", IEEE Transactions on Software Engineering, June1994.

[jiantao] Pan, Jiantao, "Software testing," Carnegie Mellon University, http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/.

[hetzel] Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p. : ill ; 24 cm.

[beshear] http://www.arcronsystems.com/_htdoc/cscenter/notice_view.asp?cate=&keyword=&idx=53&page=1

[software-industry]

http://www.bsa.org/country/Public%20Policy/~/media/Files/Policy/Security/General/sw_factsfigures.ashx