These are the BI components of the Summary and the Project Description from a successful Career grant in Computer Science.

I confess that the summary looked very generic to me, and I was prepared not to be impressed. However, the BI discussion in the Project Description establishes that the proposed activities are more specific to his work. Note that in the summary, he used 3rd person, "the PI" because that becomes a public document. In the main narrative, he uses 1st person "I" which gives the descriptions more immediacy (in my opinion). Other strengths are his reference to general educational issues (with citations) to create the rationale for his choice of activities, and in one place, he extends his impact by doing something with the professional society. When talking about mentoring, he names names and gives details of the students' success to establish his "track record," but then he is specific about what resources he will use to continue and expand his mentoring. Another detail in the high school component is where he mentions participating in written advocacy to the state. It's not a major point, but establishes him as committed to what he is proposing. --bp

(Summary) **Broader Impacts**

The proposed work has four expected broader impact contributions:
1. The resulting algorithms and techniques will likely have significant industrial impact, as debugging is critical to industrial software development. The PI has a history of releasing open-source tools and of transitioning technology to industry, and has planned industrial collaborations.

2. The PI will develop an undergraduate software engineering course, changing lecture-based teaching into team-based, hands-on exercises that help students understand the importance of clear and ample communication, documentation, and planning. Additionally, the exercises will teach students how to debug, which is a critical, lacking aspect of today's software engineering education. The class will involve debugging and testing techniques from the proposed research.

3. The PI has an extensive track record of published research with undergraduates, resulting in students transitioning to premier PhD programs, which the project will further support.

4. The PI regularly mentors students at his former high school in STEM opportunities and careers.

In the **Project Description**

4 Educational Benefits, Outreach, and Broader Impacts

This section describes the proposed educational effort in advancing classroom education (Section 4.1), technology transfer (Section 4.2), undergraduate mentoring (Section 4.3), and high-school outreach (Section 4.4).

4.1 Classroom Education

Undergraduate education today is centered around lecturing. Considerable effort has been made to change this, but much more is needed to re-center our education on group work, active participation, and critical learning. To be successful after graduation, whether in industry or graduate school, the students must be able to work in teams, plan collaborative activities, deal with conflict, and be able to derive and seek out information rather than simply learn and repeat it.

I teach an undergraduate "Introduction to Software Engineering" course required for all Computer Science majors at UMass. This class is project-driven: teams of eight students solicit requirements, design, test, implement, and maintain a system. I will revamp this course to focus on active learning that helps students learn by doing. I will replace half of the lectures with in-class group activities that help

students understand the challenges of software development and the importance of clear and ample communication, documentation, and planning. The skills the students learn in these activities will be directly applicable to their projects, which will serve to reinforce the learning. Half of these activities will extend other educators' efforts [50, 100, 134] and others, including uses of model inference, will be developed from scratch. The activities will teach debugging — one especially important skill that today's software engineering courses do not teach sufficiently—through dedicated activities and group reflection sessions on debugging strategies.
Signifying the progress already made toward this goal, I will be giving a tutorial at the 2014 Software Engineering Educators Symposium (http://cs.nju.edu.cn/changxu/sees) held jointly with FSE 2014 on activity-based undergraduate education.

I now explain five examples of the activities proposed to be developed and adapted:

Activity model inference: A student performs a two-minute task, such as folding a paper airplane, while another student logs all actions. The class collectively examines all the logs and infers a model using temporal property model inference. This first model suffers from poor abstraction and diversity of logging granularity and nomenclature. The students discuss a common logging language and redo the two-minute task, regenerating the logs and the models. Discussing the variance in the paper-airplane-folding behavior model, the students identify behavior that results in poor airplanes and manually debug the model to disallow such behavior. Finally, the students use the model to identify a set of test airplane folds that determine correct behavior. This activity exposes the students to debugging and behavioral understanding challenges, as well as to model inference, comprehension, and test generation tools developed as part of this proposal.

Broken specification telephone: Groups of six students sit in a circle. One student is given a 25-line pseudocode program and writes its specification, choosing to use natural language or a previously taught formal specification. The next student uses the resulting specification to write a pseudocode program. The third student, again, writes a specification based on the pseudocode. The process repeats six times and the students collectively compare the original and the final pseudocode. This activity teaches the importance of specifying special corner cases and exceptional behavior, as most specifications will quickly lose those cases. Further, the activity teaches the ambiguity of specifications and the importance of formal, clear written communication and documentation.

Specification groupthink [50]: Teams of eight students are asked to complete a a partial specification of a phone system. Then, the students answer (without conferring) multiple choice questions about their system's behavior in various situations. There are no right and wrong answers, but each team gets more points when more of its members independently agree. The exercise teaches the complexity and ambiguity of specification and importance of communication. The activity runs as a game show, with multiple rounds and the ability to reflect, redesign, and re-strategize as a team. The highest-scoring and the most improved teams win prizes.

Pair programming [134]: Students individually design and draw a transportation device with four requirements and then, in groups of four, are asked to combine one device's propulsion, another's brakes, another's restraints, and another's appearance. Next, two pairs of students write short movie scripts and then combine one pair's characters and explosions with another's romance and plot twist. Finally, two pairs of students design part of a robotic classroom assistant, switch partners to design another part, and switch again to finalize the design. After each activity, the groups share their products with the class. The activity teaches the importance of frequent communication and early integration.

Extreme programming simulation [100]: The students compete in a group-based workplace simulation game SimSE. The game simulates a group of developers using extreme programming practices to develop

software. The students alternate completing game rounds and discussing strategies as a class. The activity teaches the values of frequent communication, pair programming, refactoring, and frequent debugging, as well as contrasts modern development practices to more traditional spiral and waterfall development models.

All activities (1) are preceded by short lectures that introduce the relevant concepts and involved techniques, materials, and tools, (2) involve mid-activity discussions of strategies that are working and not working, and (3) are followed by a reflective discussion of successes, failures, and lessons learned. Students' qualitative feedback during the reflective discussions will guide improving the activities and developing new ones. Further, student performance on homework assignments and exams on activity-relevant topics will be compared to prior terms' performance to demonstrate the activities' educational effects.

4.2 Technology Transfer

With the billion-dollar costs associated with debugging [19], tools that improve software quality and ease debugging and testing will likely have a significant global impact on society. I have a history of making all tools and source available publicly, e.g., [11, 12, 13, 14, 21, 23, 24, 81, 103] and of transitioning technology to industry. For example, a tool developed based on my prior work [23, 24, 25] has been used internally by Microsoft to reduce the negative effects of collaborative conflicts. The attached letter of collaboration from Microsoft Research speaks to the likely industrial broader impact.

4.3 Mentoring Undergraduates in Research

I have a track record of mentoring undergraduates, women, and minorities in research, often resulting in publications [11, 12, 14, 103, 104, 127, 1] and student transition to graduate school. Three representative examples are: Roykrong Sukkerd published as a first author [127] and is now pursuing her PhD at Carnegie Mellon University. Michael Herzberg and Sebastian Fiss, undergraduates visiting UMass from Germany, completed the preliminary work described in Section 2.1.4, to appear as a technical paper at ASE 2014 [103]. In the summer of 2013, I recruited and advised two undergraduates, Brandon McNew from Hendrix College in Arkansas, and Jeanderson Barros from Brazil, who normally do not have access to research opportunities. Today, Brandon is a full-time software developer at CUSi, and Jeanderson is an intern at Google, Inc. Jeanderson, Michael, and Sebastian are all applying to graduate schools next year.

Whenever I teach undergraduate Software Engineering courses, I run an honors section for students interested in research, introducing critical literature reading, research problem identification, and collaborative research. I will continue mentoring undergraduates, recruiting them to do research, and using NSF's REU opportunities to involve students who do not have access to research at their institutions.

4.4 High-School Outreach

I attended public K–12 schools in Massachusetts and benefited greatly from my education, particularly from my high school experience at the Massachusetts Academy of Math and Science. The school exposes its students to independent research and exciting STEM opportunities, such as FIRST robotics (http: //www.usfirst.org). Every year since returning to MA, I have visited to my high school to talk to the junior classes about careers in Computer Science, exciting research in STEM fields, and summer research programs. Additionally, I have written letters to the MA Congress in support of funding Mass Academy (and public education), doing my small part to help reverse a large-scale planned cut to that funding in 2012. I believe STEM fields lose many talented students before college, and find it critical to reach out to younger students and encourage them to explore STEM. As part this project, I will continue to mentor Mass Academy and expand my visits to local high schools and middle schools.