Research Statement

Irwin Kwan

Building software no longer rests solely in the hands of trained software programmers. *End users* with no computer science training are developing software to fulfill their needs—today's end-user developer is a business owner tracking finances on spreadsheets, a tour guide customizing an interactive dashboard for a museum, or an avid gamer creating game add-ons. In fact, the number of end-user developers predicted for 2012 was on an upward trajectory, outnumbering professional developers by a factor of almost five [22]. The software engineering research community is beginning to recognize this population's need to develop complex and reliable programs: its premiere conference, the International Conference on Software Engineering, lists "end-user software engineering" as a topic of interest and "end-user software engineering" will be presented at the upcoming Future of Software Engineering track alongside topics such as ultra large-scale systems, software services, and big data.

The end-user developer is building programs of increasing complexity and often cannot solve all of his or her problems alone [11, 17]. Consequently, they collaborate with others: In the 90s, this occurred primarily in their co-located social environments [18] but today's mobile computing and Internet allows remote collaboration over large distances. The problem is that managing remote collaboration is challenging—even professional programmers, let alone end-user developers, have problems with the communication and coordination required to build complex programs. Some of the problems in this context include end-user developers making undocumented assumptions about their programs [18], developers being unaware or even overwhelmed by others' actions [7], difficulty understanding technical and domain concepts over limited communication channels [24], and programs having dependencies that are difficult to trace [1, 21]. End-user developers are not aware of the software engineering practices used to mitigate these issues, such as knowledge management, awareness notifications, and dependency-tracking.

My research goal is to design software engineering practices and tools to *help end-user software developers who collaborate in geographically-distributed environments* develop *complex and reliable programs.* One of the key components toward this goal is to design an approach that entices end-user developers to adopt and use good collaborative software-engineering practices, especially in remote settings. Because many end-user developers are *active users*—defined as those who unwillingly learn just enough to solve their most immediate issues [5], we cannot use traditional pedagogical techniques like guided tutorials or online lessons. Instead, the approach must enable end-user developers to acquire collaborative software-engineering practices gradually on their own, just in time, in the context of their own tasks and goals. I plan to identify appropriate collaborative software-engineering practices for end users, then present them using an adaptation of the *Idea Garden approach* to create the *Idea Garden for collaboration.* The Idea Garden approach extends a programming environment to present relevant information in the context of the end user developer's current task, explaining *concepts*, *strategies*, and *mini design patterns.* It is not a tutorial nor does

it automatically resolve problems for the user [3]. The Idea Garden approach is a theory-based approach based on minimalist learning theory [4] and has helped end-user developers transfer their learning into practice [2]. However, the existing Idea Garden approach has considered only a single end-user developer learning programming concepts for the first time. In this work, we must modify the Idea Garden approach to tackle new problems, such as adapting content for *multiple end-user developers* that may each have different knowledge, motivations, and environments and providing *practices*, rather than concepts, to the end-user developers.

As we do not know much about how end-user developers work together on their programs and what practices they use, some formative work is required to understand which collaborative software-engineering practices are appropriate and effective for end user developers. First, I plan to identify which collaboration practices work for end-user developers by evaluating their effectiveness empirically in a laboratory setting. I will frame the findings using existing theories and constructs from human-computer interaction and collaborative software engineering, such as information foraging theory [20] and awareness [7] and develop a methodology for adapting these practices for end-user developers. Second, I plan to use iterative design to adapt the Idea Garden approach to use contexts from multiple collaborating end-user developers. Because the end-user developers are working as a team, significant changes must be made to make the presentation of collaborative practices relevant not only to individuals, but also to the team as a whole.

This research enables the growing population of remotely-collaborating end-user developers to improve the quality of their programs by introducing collaborative software-engineering practices. The form of these practices tailored to their motivations and expertise. The Idea Garden approach will need to be adapted to use multiple workspaces and contexts entice end-user developers to improve information-seeking behavior, awareness, and dependency management. The effects of this work are not necessarily limited to end-user software engineering either: it can push the boundaries of collaborative software development for professionals as well. If the Idea Garden for Collaboration approach can transfer good collaborative software-engineering practices to an end-user developer with little computing experience, then the approach may also enable transfer of collaborative practices to professional software engineers as well.

My expertise in empirical software engineering, end-user software engineering, and collaborative software engineering makes me qualified to carry out this research goal. In my postdoctoral work, I advised the design of the Idea Garden approach [3, 2] and led the integration of the Idea Garden into a debugging game for high school students; in addition, I am co-researching end-user programming using a platform called Gidget that teaches programming using a debugging-first approach [16]. I studied how explanations of decisions by intelligent assistant interfaces affected the mental models of end users of these systems [12, 25]. In the area of software engineering, I extended Information Foraging Theory to explain how debugging software developers forage for information [19] and proposed design patterns for software engineering tools based on Information Foraging Theory principles [10]. I have experience in collaborative software engineering from my Ph.D by investigating socio-technical congruence and its effect on global software teams [15, 13, 6] and investigating communication and awareness using requirement-driven social networks within geographically-distributed teams [7, 9, 8, 14, 23]. Table 1 summarizes these projects. My involvement with end-user software engineering and collaborative software development has resulted in publications at venues such as ACM Conference on Human Factors in Computing (CHI) (where I earned a best paper honourable mention), the International Conference on Software Engineering (ICSE), International Conference on Education Research (ICER) (where I earned a people's

Name	Description	Publications
Idea Gardening	Entices end-user programmers to learn and use concepts and strategies in the context of their own tasks	[2, 3]
Gidget	A "debugging-first" game that teaches end users how to program through debugging puzzles	[16]
Mental models of intelligent assistants	Identifying how explanations affect end user's mental models of how intelligent assistants work	[12, 25]
Information Foraging Theory in Software Engineering	Extending information foraging theory to better understand how developers look for information while developing software	[10, 19]
Requirements-driven social networks in global software teams	How global software teams communicate, coordinate, and maintain awareness around requirements work	[7, 9, 8, 14, 23]
Socio-technical congruence and team performance	Effects of aligning team communication and software dependencies on global software team performance	[15, 13]

Table 1: My research project experience

choice best paper award), and the IEEE Symposium in Visual Languages and Human-Centric Computing (VLHCC), as well as in the IEEE Transactions of Software Engineering (TSE), the ACM Transactions on Software Engineering and Methodology (TOSEM), the ACM Transactions on Computer-Human Interaction (TOCHI), and IEEE Software. In addition I have co-written a funded NSF grant about learning and Information Foraging Theory. My breath of knowledge and depth of expertise provides me with the experience to accomplish my goal of enticing end-user developers to learn and use collaborative software-engineering practices. The Idea Garden for Collaboration is the first step to helping this growing population of programmers collaborate to build complex, reliable programs.

References

[1] Margaret Burnett, Curtis Cook, and Gregg Rothermel. End-user software engineering. *Commun. ACM*, 47(9):53–58, September 2004.

[2] Jill Cao, Irwin Kwan, Faezeh Bahmani, Margaret Burnett, Scott D. Fleming, Josh Jordahl, Amber Horvath, and Sherry Yang. End-user programmers in trouble: Can the idea garden help them to help themselves? In *2013 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 151–158, 2013.

[3] Jill Cao, Irwin Kwan, Rachel White, Scott D. Fleming, Margaret Burnett, and Christopher Scaffidi. From barriers to learning in the idea garden. In *IEEE Symposium on Visual Languages and Human-centric Computing 2012*, pages 59–66, Innsbruck, Austria, 2012.

[4] John M. Carroll. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. The MIT Press, June 1990.

[5] John M. Carroll and Mary Beth Rosson. Paradox of the active user. In John M. Carroll, editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, pages 80–111. MIT Press, Cambridge, MA, USA, 1987.

[6] Daniela Damian, Remko Helms, Irwin Kwan, Sabrina Marczak, and Benjamin Koelewijn. The role of domain knowledge and cross-functional communication in socio-technical coordination. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 442–451, Piscataway, NJ, USA, 2013. IEEE Press.

[7] Daniela Damian, Luis Izquierdo, Janice Singer, and Irwin Kwan. Awareness in the wild: Why communication breakdowns occur. In *Intl Conf on Global Software Engineering, Münich, Germany*, pages 81–90, August 2007.

[8] Daniela Damian, Irwin Kwan, and Sabrina Marczak. Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In Jim Whitehead, Ivan Mistrík, John Grundy, and Andr'e van der Hoek, editors, *Collaborative Software Engineering*, chapter 3, pages 57–76. Springer-Verlag, 2010.

[9] Daniela Damian, Sabrina Marczak, and Irwin Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centered Social Networks. In *Intl Requirements Engineering Conference, New Delhi, India*, pages 59–68, October 2007.

[10] Scott D. Fleming, Christopher Scaffidi, David Piorkowski, Margaret Burnett, Rachel Bellamy, Joseph Lawrance, and Irwin Kwan. An information theory perspective on tools for debugging, refactoring, and reuse tasks. *Transactions on Software Engineering and Methodology*, 22(2), 2013.

[11] Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan F. Blackwell, Margaret M. Burnett, Martin Erwig, Christopher Scaffidi, Joseph Lawrance, Henry Lieberman, Brad A. Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. The state of the art in end user programming. *ACM Computing Surveys*, 43(3):21:1–21:44, April 2011.

[12] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more? the effects of mental model soundness on personalizing an intelligent agent. In *ACM Conference on Human-Computer Interaction*, pages 1–10, Austin, USA, May 2012.

[13] Irwin Kwan, Marcelo Cataldo, and Daniela Damian. Conway's law revisited: A task-based view. *IEEE Software*, 29(1), Jan/Feb 2012.

[14] Irwin Kwan and Daniela Damian. The Hidden Experts in Software-Engineering Communication (NIER Track). In *ICSE'11: Proceedings of the Intl Conf on Software Engineering, Honolulu, USA*, pages 800–803, 2011.

[15] Irwin Kwan, Adrian Schroter, and Daniela Damian. Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project. *IEEE Transactions on Software Engineering*, 37:307–324, 2011.

[16] Michael J. Lee, Andrew J. Ko, and Irwin Kwan. In-game assessments increase novice programmers' engagement and level completion speed. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 153–160, New York, NY, USA, 2013. ACM.

[17] Bonni A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing.* MIT Press, Cambridge, Massachusetts, 1993.

[18] Bonni A. Nardi and James R. Miller. Twinkling lights and nested loops: distributed problem solving and spreadsheet development. *Intl J. Man-Machine Studies*, 34:161–184, 1991.

[19] David Piorkowski, Scott D. Fleming, Margaret Burnett, Rachel Bellamy, Irwin Kwan, Christopher Scaffidi, and Josh Jordhal. The Hows and Whats of Programmers Diet. In *ACM Conference on Human-Computer Interaction*, 2013.

[20] Peter Pirolli. *Information foraging theory: adaptive interaction with information.* Oxford University Press, April 2007.

[21] Gregg Rothermel, Margaret Burnett, Lixin Li, Christopher Dupuis, and Andrei Sheretov. A methodology for testing spreadsheets. *ACM Trans. Softw. Eng. Methodol.*, 10(1):110–147, January 2001.

[22] C. Scaffidi, M. Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 207–214, 2005.

[23] Adrian Schröter, Jorge Aranda, Daniela Damian, and Irwin Kwan. To talk or not to talk: Factors that influence communication around changesets. In *Proc. CSCW '12*, 2012.

[24] Bikram Sengupta, Satish Chandra, and Vibha Sinha. A research agenda for distributed software development. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 731–740, New York, NY, USA, 2006. ACM.

[25] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, San Jose, CA, USA, September 2013.