UNIVERSITY OF CALIFORNIA,
IRVINE

Supporting Trust in Globally Distributed Software Teams: The Impact of Visualized Collaborative Traces on Perceived Trustworthiness

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Erik Harrison Trainer

Dissertation Committee:
Professor David F. Redmiles, Chair
Professor André van der Hoek
Assistant Professor James A. Jones

2012

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

DEDICATION

TO MY FAMILY

TO MY FRIENDS

TO MY MENTORS

I couldn’t have done this without you.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

ACKNOWLEDGMENTS

I'll always remember this slice out of my life.

I would like to express the most sincere gratitude to my faculty advisor and committee chair, Professor David Redmiles. Without his steady guidance and patience, this dissertation would not have been possible.

I thank my committee members, Professor André van der Hoek and Professor James A. Jones, whose irreplaceable feedback and advice have served me well. Professor van der Hoek was always one of my biggest skeptics, questioning nearly all of my assumptions. My work is undeniably better because of him. Professor Jones's unwavering encouragement was a true source of inspiration and motivation.

Various researchers at Microsoft Research Silicon Valley, especially Cathy Marshall and Doug Terry, gave me the opportunity to get professional research experience by funding an internship in the summer of 2009.

I have been incredibly lucky to have a fantastic collection of mentors throughout the years. Specifically, Doctors Matthew Bietz, Ban Al-Ani, and Cleidson de Souza were always there to help me flesh out a thought, critique my work, and advise me in the scholarly arts. My graduate years might have been very different were it not for them.

I very much appreciate Gary and Judy Olson's Hana Lab for providing the laptops, recording equipment, and access to their human subjects pool for my laboratory experiment.

Many amazing friends and colleagues have accompanied me on my journey (and helped keep me sane along the way!). The motley crew, in alphabetical order: Steve Abrams, Nadine Amsel, Michael Bebenita, Gerald Bortis, Michael Caldera, Leyna Cotran, Christoph Dorn, Tobias Hildenbrand, Nityananda Jayadevaprakash, Benjamin Koehne, Leslie Liu, Kristina and Ramzi Nasr, Ko Nee, Wiwat Ruengmee, Anita Sarma, Raj Sawhney, Bryan Semaan, Patrick Shih, Jungmin Shin, Roberto Silveira Silva Filho, Kyle Strasser, Norman Su, Oliver Yi Wang, and Yongjie Zheng. I want to recognize Stephen Quirk for his early work on Ariadne and the ideas we shared during the first two years of our PhDs.

I will be forever grateful to my family: my mother, father, and brother who showed me unconditional comfort and love during the most stressful times. They patiently supported me through many, many years of schooling. And last but not least, Giselle Su, my treasure.

Financial support for this work was provided by the U.S. National Science Foundation under grants 0943262, 0808783 and 1111446. Additional support was provided by the Department of Education under the GAANN Fellowship as well as an Eclipse Innovation Grant and Jazz Innovation Award.

CURRICULUM VITAE

Erik Harrison Trainer

EDUCATION

Irvine, CA **UC Irvine** **June 2007-December 2012**

- PhD, Information and Computer Science. Advisor: David Redmiles. GPA: 3.97

Irvine, CA **UC Irvine** **June 2005-June 2007**

- MS, Information and Computer Science. GPA: 3.89

Irvine, CA **UC Irvine** **June 2001-June 2005**

- BS, Information and Computer Science. GPA: 3.6

RESEARCH INTERESTS

- Computer-supported Cooperative Work (CSCW)
- Human-Computer Interaction (HCI)
- Global Software Engineering (GSE)
- Trust
- Visual interface design
- Prototyping
- Interactive and collaborative interfaces

EXPERIENCE

Grad. Student Researcher **UC Irvine** **June 2007-June 2012**

- Showed, through proof-of-concept laboratory experiment, that a visual interface can support trust in globally distributed software teams.
- Designed, implemented, and evaluated Ariadne and Theseus (in Java, GWT), two interactive and collaborative visual interfaces to support awareness and the development of trust in globally distributed software teams.
- Disseminated tools and results through the publication of research papers (see below) and construction of posters that showcase the work.

Research Intern **Microsoft Research** **June 2009-Sept. 2009**

Community Information Management group. Advisor: Catherine C. Marshall.

- Designed and implemented (in C#) the first prototype of *CIMBib*, a peer-to-peer application for sharing scholarly references (e.g., papers, bibliographies, and annotations) and replicating them among personal computing devices.

Teacher's Assistant **UC Irvine** **April 2009-June 2009**

Information Visualization.

- Drafted homework assignments, led discussion sections, and graded students.

Teacher's Assistant **UC Irvine** **April 2008-June 2009**

Information Visualization.

- Drafted homework assignments and graded students.

Student Ethnographer Qualitative Research Class January 2006-March 2006

- Performed a research study of street performers in Santa Monica, CA.
- Developed an interview protocol and interviewed 12 informants.
- Observed musical performers licensed by the city and interviewed them.
- Evaluated the interview data using techniques grounded in theory.
- Published a research report at the end of the school quarter.

OTHER WORK EXPERIENCE

Web Dev Intern WallStreet University, Inc Sept. 2004-May 2009

- Customized and deployed Moodle, an open-source course management system for integrating online educational investing materials.
- Managed the company's web development and programming team (4 people), as well as remote programming teams located at Cal State Fullerton, CA (2 people) and The University of the West, Rosemead, CA (1 person).
- Implemented front-end e-commerce shopping cart in PHP, Javascript and HTML.
- Developed 5 Adobe Flash courses with animation and action script.
- Wrote small PHP modules for company web site to handle customer mailing lists, course announcements, and conversion rates.

Intern Pilot Chemical Company June 2001-Sept. 2002

- Initiated use of maintenance database software in the plant.
- Developed monthly spreadsheets documenting electric and gas usage.
- Reorganized and implemented material safety data sheets (MSDS) file system for the plant.
- Assisted plant manager in updating and computerizing ISO 9000 documents.
- Initiated work on Microsoft Access Database for the Customer Complaint system.

Intern Pac Comm Technologies June 2000-Sept. 2000

- Collaboratively learned Java programming language with other interns.
- Designed and developed web pages and updated existing sections of the company web page.

PROJECTS

- **Theseus** (http://www.ics.uci.edu/~etrainer/theseus). Web application with visualization widgets that keep globally distributed developers aware of their colleagues and promote the development of trust. Java, GWT.
- **Ariadne** (http://awareness.ics.uci.edu/~ariadne)**.** Eclipse plug-in that visualizes the socio-technical network of a project, including inter-dependencies between software developers. Java, JUNG, Protovis.

STUDENT MEMBERSHIPS

- Association of Computing and Machinery (ACM)
- Institute of Electrical and Electronics Engineers (IEEE)
- ACM Special Interest Group on Software Engineering (SIGSOFT)

GRANTS & AWARDS

- Department of Education (GAANN) Fellowship **2005-2011**
- IBM Jazz Innovation Grant **2008**
- IBM Eclipse Innovation Grant **2005**
- UCI Undergraduate Research Opportunities Program (UROP) Funding **2005**

COMMUNITY SERVICE

Undergrad Outreach Representative. **September 2010-July 2011**

- Attended monthly meetings with ICS faculty and administrators.
- Designed fliers and posters for undergraduate recruitment.

Member, Organizing Committee for Grad Student Symposium. **May 2008**

- Solicited graduate students to submit papers to the symposium.
- Organized hotel arrangements for speakers.

Student Reviewer. **May 2008**

- Peer-reviewed papers from VLHCC 2008
- Peer-reviewed Springer Book Chapter on Collaborative Software Engineering
- Organized hotel arrangements for speakers.

Research Conference Volunteer **November 2005-February 2010**

- 2010 ACM Conference on Computer-Supported Cooperative Work (CSCW 2010)
- Student Volunteer, 20th IEEE/ACM Conference on Automated Software Engineering (ASE 2005)

Rhythm Guitarist **Orange Solution** **June 2009-Sept. 2009**

- Play shows once per month at local venues, including Whisky a Go Go and Detroit Bar
- Organiz.ed canned food drives (i.e. discount admission to shows with a non-perishable food item) in partnership with the Orange County Food Bank to help Orange County families in need.
- Donated more than 200 lbs of food thus far.

CONFERENCE PUBLICATIONS

Al-Ani, B., Bietz, M., Wang, Y., Trainer, E., Koehne, B., Marczak, S., Redmiles, D.F., Prikladnicki, R. (2013): 'Globally Distributed System Developers: Their Trust Expectations and Processes', In Proceedings of the 2013 ACM Conference on Computer-supported Cooperative Work (CSCW 2013), 10 pp., in press.

Al-Ani, B., Wang, Y., Marczak, S., Trainer, E., and Redmiles, D.F. (2012): 'Distributed Development Teams and Non-Use of the Web 2.0 Technologies: A Proclivity Framework', In Proceedings of the 2012 International Conference on Global Software Engineering (ICGSE 2012), 10 pp., in press.

Wang, Y., Trainer, E., Al-Ani, B., Redmiles, D., and Marczak, S. (2012): 'Attitude and Usage of Collaboration Tools in GSE: A Practitioner Oriented Theory', In Proceedings of the 2012

International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), held in conjunction with the 2012 International Conference on Software Engineering (ICSE 2012), 3 pp., in press.

Trainer, E. and Redmiles, D.F. (2012): 'Foundations for the Design of Visualizations that Support Trust in Distributed Teams', In Proceedings of the 2012 ACM International Conference on Advanced Visual Interfaces (AVI 2012), pp. 34-41.

Al-Ani, B., Trainer, E., Redmiles, D.F., and Simmons, E. (2012): 'Trust and Surprise in Distributed Teams: Towards an Understanding of Expectations and Adaptations', The 4th ACM International Conference on Intercultural Collaboration (ICIC 2012), pp. 97-106.

Al-Ani, B., Marczak, S., Trainer, E., Redmiles, D.F., and Prikladnicki, R. (2012): 'Distributed Developers' Perspectives of Web 2.0 Technologies in Supporting the Development of Trust', The Future of Collaborative Software Development Workshop, held in conjunction with the 2012 Conference on Computer-supported Cooperative Work (CSCW 2012).

Trainer, E., Al-Ani, B., and Redmiles, D.F. (2011): 'Impact of Collaborative Traces on Trustworthiness', In Proceedings of the 2011 International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), held in conjunction with The 2011 International Conference on Software Engineering (ICSE 2011), pp. 40-47.

Trainer, E., Quirk, S., de Souza, C.R.B., and Redmiles, D.F. (2008): 'Analyzing a Socio-Technical Visualization Tool Using Usability Inspection Methods', In Proceedings of the IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC 2008), pp. 78-81.

Trainer, E. and Redmiles, D.F. (2008): 'Towards an Infrastructure for Software Visualization Research', In First International Workshop on Infrastructure for Research in Collaborative Software Engineering (IReCoSE), held in conjunction with The 16th International Symposium on the Foundations of Software Engineering (FSE), available at http://home.segal.uvic.ca/~IRCoSE-2008/.

Al-Ani, B., Trainer, E., Ripley, R., Sarma, A., van der Hoek, A., Redmiles, D.F. (2008): 'Continuous Coordination within the Context of Cooperative and Human Aspects of Software Engineering', In Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), held in conjunction with The 2008 International Conference on Software Engineering (ICSE 2008), pp. 1-4.

Trainer, E. (2008): 'Connecting the Social and Technical Aspects of Computing with Visualization', In Proceedings of the IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC 2008), pp. 272-273.

de Souza, C.R.B., Quirk, S., Trainer, E., and Redmiles, D.F. (2007): 'Supporting Collaborative Software Development through the Visualization of Socio-Technical Dependencies', In Proceedings of the 2007 International ACM Conference on Supporting Group Work (GROUP 2007), pp. 147-156.

Trainer, E., Quirk, S., de Souza, C. R. B., Redmiles, D.F. (2005): 'Bridging the Gap between Technical and Social Dependencies with Ariadne', In Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange, (OOPSLA ETX 2005), pp. 26-30.

de Souza, C.R.B., Dourish, P., Redmiles, D.F., Quirk, S., and Trainer, E. (2004): 'From Technical Dependencies to Social Dependencies', In Proceedings of the 2004 Workshop on Social Networks, held in conjunction with The 2004 Conference on Computer-Supported Cooperative Work (CSCW 2004), available at http://www.ischool.washington.edu/mcdonald/cscw04/

JOURNAL PUBLICATIONS

Al-Ani, B., Redmiles, D., van der Hoek, A., Alvim, M., da Silva, I., Mangano, N., Trainer, E., Sarma, A. (2008): 'Continuous Coordination within Software Engineering Teams: Concepts and Tool Support', *Journal of Computer Science and Engineering in Arabic, Special Issue on Software Engineering*, vol. 1, no. 3, 2008, pp. 10-33.

Redmiles, D., van der Hoek, A., Al-Ani, B., Hildenbrand, T., Quirk, S., Sarma, A., Silveira Silva Filho, R., de Souza, C., Trainer, E. (2007): 'Continuous Coordination: A New Paradigm to Support Globally Distributed Software Development Projects', *Wirtschaftsinformatik, Special Issue on the Industrialization of Software Development*, vol. 49, 2007, pp. 28-38.

BOOK CHAPTERS

Sarma, A., Al-Ani, B., Trainer, E., Silva Filho, R.S., da Silva, I., Redmiles, D., van der Hoek, A. Continuous Coordination Tools and their Evaluation, in I. Mistrík, J. Grundy, A. van der Hoek, J. Whitehead (eds.), Collaborative *Software Engineering*, Springer, Ch. 8, pp. 153-178.

TECHNICAL REPORTS

Trainer, E., Quirk, S., de Souza, C.R.B., and Redmiles, D.F. (2012): 'Usability Inspection Method-based Analysis of a Socio-Technical Visualization Tool', Technical Report UCI-ISR-12-6. University of California, Irvine, Institute for Software Research.

Trainer, E. and Redmiles, D.F. (2010): 'Initial Successes and Failures Prototyping Socio-technical Visualizations Using a Collaboration Infrastructure', Technical Report UCI-ISR-10-5. University of California, Irvine, Institute for Software Research.

Trainer, E. and Redmiles, D.F. (2009): 'A Survey of Visualization Tools that Promote Awareness of Software Development Activities', Technical Report UCI-ISR-09-5. University of California, Irvine, Institute for Software Research.

ABSTRACT OF THE DISSERTATION

Supporting Trust in Globally Distributed Software Teams: The Impact of Visualized Collaborative Traces on Perceived Trustworthiness

By

Erik Harrison Trainer

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 2012

Professor David F. Redmiles, Chair

Trust plays an important role in collaborations because it creates an environment in which people can openly exchange ideas and information with one another and engineer innovative solutions together with less perceived risk. The rise in globally distributed software development has created an environment in which workers are likely to have less information and lower quality information about their remote colleagues. As such, the likelihood of coordination breakdowns increases. Observers of a breakdown are more likely to erroneously attribute the cause to personal characteristics (dispositional attributions) of the persons involved, rather than characteristics of the situation (situational attributions). Data collected from globally distributed software teams show that such breakdowns can negatively impact trust between the parties involved, as well as the perceived quality of the collaboration.

At the same time, software engineering research has resulted in a rich set of tools for globally distributed software developers. These tools support the smooth flow of work across remote sites by visualizing data extracted from projects. They help developers understand changes made to the system, identify experts, determine the availability of their colleagues, and track the activities of developers on whose code they depend.

Synthesizing literature on trust, tools that support awareness of development activities, and visualization, this dissertation asks whether a tool can usefully support the development of trust in globally distributed teams. It presents the design of a tool called Theseus and two evaluations that assess Theseus' usefulness. In the first evaluation, Theseus' interface was analyzed using three usability inspection methods. In the second, Theseus was assessed in a laboratory experiment with 28 graduate students and 12 professional software developers.

The results show Theseus is highly usable and that it has a significant effect on distributed developers' perceived trustworthiness toward others. Participants quickly became immersed in the information the tool provides. In situational conditions, Theseus resulted in higher perceived trustworthiness and more situational attributions than dispositional ones. These results support the thesis statement of this dissertation and open up interesting areas for future research, especially understanding how collaboration tools can potentially shape distributed software developers' sense of trust toward one another.

Chapter 1. Introduction

Trust plays a significant role in the quality--and ultimate success--of collaborations. Trust is a pre-requisite for many collaborative activities: idea and information exchange, decision-making, and innovation. It is especially critical for teams where team members have never met face-to-face, or will never have the opportunity to meet face-to-face due to limited project resources. Existing literature on trust has shown that "swift trust," trust that is pre-supposed at the beginning of new collaborations, can lead to productive and innovative results but is fragile and easily broken (Meyerson et al., 1996; Jarvenpaa and Leidner, 1999). In traditional face-to-face teams, team members have the time to assess one another based on personal interaction and shared experience (Wilson et al., 2006). In these situations, trust tends to develop more quickly and is more durable over the long-term (Meyerson et al., 1996).

Now more than ever, software engineering involves working with remote team members located in different geographic regions. A specific example is a company we are currently studying that has sites in the U.S., Brazil, Costa Rica, Ireland, Israel, Mexico, Poland, China, Taiwan, and Malaysia. The experience of working remotely often leads to various coordination and communication challenges documented by many researchers (e.g., Olson and Olson, 2000, Herbsleb and Grinter, 1999, Cramton, 2001; Cramton 2002; Herbsleb and Mockus, 2006).

Collaboration tools help to manage the complexities of remote communication, for instance e-mail, mailing lists, teleconferencing tools such as *Live Meeting*, and wikis. Other tools support software development specific activities. Awareness tools and issue trackers, respectively, summarize status of the project and support the assignment of currently incomplete tasks and enhancements to the developers working on the project.

The human aspects of coordination, including expectations about behavior, must be considered alongside technologies. When a coordination breakdown occurs, a failure to meet a development milestone for instance, people's future expectations of their collaborators change. Suppose a developer is not getting an e-mail response from a new colleague. Without the knowledge that a distributed collaborator is involved in multiple projects and located several time zones away and working on their own separate deadline, the developer may unfairly assume the latter is not a responsive person, perceive them to be untrustworthy, and have less trust in them in the future.

This research aims to overcome distributed developers' lack of knowledge about their colleagues over the course of a remote collaboration by exposing important information that can help build trust. This dissertation shows that it is possible for a technology, or software interface, to increase or decrease a remote team member's perceived trustworthiness of their colleague. The interface exposes situational factors, preventing a potential loss of perceived trustworthiness *when losing that trust is unwarranted*. This shifts how we view the development of trust in its early stages from a focus on assumptions about new team members that leads to a "fragile" and "weak" temporary trust (Meyerson et al., 1996), to a focus on the role of everyday technology in setting the foundations upon which more long-term trust rests.

In the remainder of this chapter, I introduce trust as the subject of study by motivating it through a set of scenarios. I conclude the chapter with an outline of the dissertation. Chapter 2 continues with a description of the overarching approach of this work.

1.1 Scenarios of Lost Trust

Communication breakdowns can impact globally distributed software engineers' sense of trust toward their remote colleagues. This section presents three scenarios that illustrate this

point. The scenarios are drawn from interviews in which my colleagues and I interviewed systems engineers at a Fortune 500 technology company about their experiences working with remote team members on various projects. Only names are anonymized.

1.1.1 Poor Response Time

Ray, a developer with 11 years experience as a systems engineer, currently works in Costa Rica. He collaborates with remote teams in Arizona, China and Malaysia. He recalls that sometimes the Malaysian and Chinese teams don't give him information in a timely manner:

> "You start to assume that everything's going on fine, and then suddenly you get a huge escalation from your manager saying, 'Hey, what's the problem here? This has been going on for one or two weeks, and then nobody's helping out,' type of thing. And then you're saying, 'What? I didn't know anything about that.'"

The lack of responsiveness has a negative impact on his trust toward these teams:

> "You don't trust that person as much. I think I tend to be more pressing. 'How's it going? Do you have any issues? Is this done? Have you done this? Have you done that?'"

1.1.2 Hidden Agendas

James, a software engineer with 12 years experience working with remote team members recalls that openness is an important pre-requisite for trust when he works with his sites in Israel and the US:

> "If they say something and you find out that it is not true or when you find out that they have hidden agendas then automatically they become low. I trust people when I can talk with them about the problems and they will give me feedback that I can understand and I can see that what they're telling is happening in reality. They don't hide information."

1.1.3 "If You're Busy, Tell Me!"

Peggy a systems engineer based in California with 10 years experience working in software development, talks about how she values the ability of her remote team members to respond promptly (even if they are busy):

> "So, if they've done something where they're not delivering or they're not responding or they're - and they don't have a good reason for that, right, I don't get a response back saying that, 'Hey, I'm out of town' or 'I'm not going to be able to respond to emails for a while,' that tends to drop them pretty quickly in my eyes of what I can expect out of them. If I'm not getting a lot of response, then the trust level drops pretty quickly."

1.1.4 Summary

The deep claim of this research is that the problems the quotations above reflect are fundamentally related. Common to all three scenarios is the problem of uncertainty with regards to what expectations to have of one's distributed colleagues. The overarching approach of this dissertation is to provide a tool that identifies and exposes existing, but not easily accessible, information that can help form these expectations upon which trust rests. For instance, knowing that a colleague is busy fixing a bug the week before a deadline can explain why they haven't answered a recent e-mail. Similarly, an e-mail response is likely to arrive sooner from a colleague in the same time zone (maybe even same day) than from a team member located 15 hours away. The value the tool provides is revealing this information about the trustee. The tool prevents the trustor from lowering their perceived trustworthiness of them when situational factors, i.e., factors beyond the trustee's control, are at work.

1.2 Dissertation Outline

The remainder of this dissertation is structured as follows. Chapter 2 motivates the approach of this dissertation. In distributed teams, trust is fragile and easily broken. In addition, it is slow and takes time. Software engineering tools provide a swath of information that describes software development activities. Chapter 2 also describes the approach itself: identify information that reduces uncertainty and leads to the formation of expectations upon which trust rests, and then summarize this information in a visual interface of a tool prototype called Theseus. In its concluding sections, Chapter 2 formally states the thesis statement and research questions that guide this dissertation. It also summarizes the contribution of this work.

Chapter 3 presents previous research by others as well as me into the areas of trust, relevant data that can influence trust, and visualization. It explores the origins of trust between globally distributed team members as well as what data have typically been used by tools that promote the awareness of software development activities. In addition, it foreshadows the impact of visualizing data on perceived trustworthiness by reviewing results from an early pilot study conducted with Ariadne, the predecessor to the Theseus tool. Chapter 3 also presents a "design space" that distills literature at the intersection of trust, information visualization, and software engineering "awareness" tools into a framework for tool design.

Chapter 4 outlines the design of the Theseus tool and the first version of its interface. Theseus provides relevant information that can influence perceived trustworthiness by extracting collaborative traces from existing development repositories and generating data that cannot be easily acquired in sufficient quantity and quality. It then visualizes this information in an appropriately designed and usable way, based on scholarly literature from software engineering and information visualization.

Chapter 5 presents results from two phases of evaluation of the Theseus prototype. The first phase consisted of applying three usability inspection methods to the interface: Heuristic Evaluation, Cognitive Dimensions of Notations, and Cognitive Walkthrough. Based on the results of these inspection methods, help and documentation as well as drag and drop functionality were added to the interface.

In the second phase of the evaluation, a laboratory experiment was designed to evaluate the tool first for utility and second for usability. Theseus' utility was demonstrated by measuring changes in perceived trustworthiness as well as the quality of insights made by subjects. Theseus' usability was demonstrated by analyzing results from usability questionnaires as well as video footage of human subjects.

Chapter 6 summarizes the contributions of this dissertation and concludes with a set of design principles for tools to support the development of trust in globally distributed software development teams. The principles derive from the model of the design space as well as results from the laboratory experiment. Chapter 6 also discusses limitations of the work. For instance, one point of concern is that the way in which this dissertation operationalizes trust is only a starting point. In order to push this work further, future directions for this research should include long-term follow-up studies, extending the design space to support concrete tasks, and new designs and implementations of visual interfaces.

Chapter 2. Research Overview and Approach

This dissertation explores the phenomena of trust between globally distributed software developers and whether tools can influence that trust. This chapter describes trust in the context of globally distributed software engineering and explains why having an accurate perception of trust can be problematic in such contexts. It formally states the thesis statement of this dissertation, formulates the research questions, and foreshadows the overarching approach this dissertation takes to supporting the development of trust between globally distributed software developers.

Software development activities are well documented in the software engineering research literature. Typical software development activities include writing source-code, submitting changes, triaging bugs, reproducing failures, understanding execution behavior, and reasoning about design (Ko et al., 2007). These activities typically produce a "project memory," (Cubranic and Murphy, 2003) of archival artifacts such as source-code, e-mail lists, design documentation, problem reports and change histories of all this information that developers then use to coordinate and accomplish work. "Work" is usually defined as adding new features to the code or fixing bugs in the code.

For example*, issue and bug trackers* provide a history of all assigned and completed work on the project. *Change management systems* store all revisions of source-code files, the changes made in each revision, and who completed the change. *E-mail*, *instant messaging tools*, and *video conferencing tools* provide mediums through which developers can communicate their expectations about the work to be completed, for instance what exactly will be done, when it will be done, and who will do it. Similarly, issue and bug trackers indicate who is responsible for fixing bugs and implementing new features. These systems also indicate the priority of the

changes and an estimated time of completion, letting others on the team know when to expect a resolution.

Whether the work is actually completed as such can be another issue altogether. In order to avoid having to constantly check up on their colleagues, globally distributed team members often must place "trust" in them to do what is expected. Trust can be defined as one party's expectations of another (Furst, 1999; Sabherwal, 1999). When a deviation from the expectations occurs, the trustor can experience a loss of perceived trustworthiness in the trustee. In fact, participants in my research group's ongoing studies of globally distributed software development teams (Al-Ani and Redmiles, 2009) frequently mentioned the concept of "trust" or more specifically, their "perceived trustworthiness" of others as influencing their sense of others' abilities to deliver work on time, be available, and be transparent about their intentions, among others.

Setting expectations of others and thus calibrating trust can be a human-intensive process. New collaborations often require individuals to call and e-mail colleagues from the past and present and navigate their personal social networks for "reputational" information and shared work experiences (Ehrlich and Chang, 2006). Remembering who was involved during which times and who worked on what can be a timely process that requires individuals to search their e-mail inboxes, schedules, calendars, and other documents across a number of projects. Building trust requires time because it is based on one's accumulated experiences and interactions. In interviews of globally distributed software development teams, my research group found that people form expectations about their collaborators before a project begins, after a project ends, or when a project is ongoing and their collaborators exhibit behavior that surprises them. For example, Alex may have trusted Chris at the beginning of the project because he was a senior

developer, but less after the former failed to respond to his e-mails before a crucial release deadline. Past activities are the backdrop on which future expectations are set.

The lack of information about distributed colleagues that is characteristic of globally distributed software development increases the likelihood of coordination breakdowns (Herbsleb and Mockus, 2006), which can work against the development of trust. Remote workers are likely to have much less information and lower quality information about their remote colleagues (Cramton, 2001; 2002). Using the previous example, if Chris had known Alex was working on three projects, the past two months of which he was in Brazil working on fixing bugs, he could have modified his expectations with respect to his availability. Cramton (2001; 2002) observed that the lack of situational knowledge and the reduced ability to process it effectively can cause individuals in remote teams to attribute breakdowns to the individual rather than the situation itself, eroding team cohesion and lasting solutions. In this instance, a loss of trust is more difficult to repair. Without knowledge and expectations of the situation, trust can be slow to build between collaborators.

2.1 Approach through Tool Prototyping

There are several aspects of the process of setting expectations in the context of globally distributed software development that can be addressed and improved. First, developing a sense of trust can take time due to the lack of information about other colleagues' activities and the lack of ways to manage this information in order to set realistic expectations. Distributed developers can ultimately reach the same levels of trust as collocated team members; it just takes longer (Wilson et al., 2006). A tool that can partially automate this process can provide time savings.

Second, in software development, the traces of activity that help set expectations are typically hidden in project repositories or incomplete documents over time. Further, a lack of situational information about colleagues can negatively and more importantly, inaccurately, bias trust judgments. If a developer is involved in multiple projects, one shouldn't expect them to be particularly responsive, for example. An approach that renders this information explicitly from project and team artifacts can prepare people to make trust judgments based on meaningful data.

Third, the sheer number of interactions and project specific information and number of artifacts involved makes this data, if presented in textual form, difficult to interpret. Visualizations can be more effective in revealing and summarizing this information.

Fourth, trustworthiness is a vague term without a precise meaning and needs to be unpacked in order to operationalize it for use in software tool research. An approach that dissects and enumerates a specific meaning of trustworthiness and in which artifacts evidence of it can be found is a crucial first step in the process of engendering trustworthiness in distributed teams.

To address these issues I propose Theseus, a research prototype for globally distributed software developers that visualizes information that can positively influence developers' perceived trustworthiness of one another. The information Theseus visualizes, the way in which it visualizes it, and the aspects of trust it supports are grounded in the research literature on trustworthiness, information visualization, and software engineering awareness tools. The following subsections formulate the thesis statement, research questions, and overall contribution of this dissertation.

2.2. Thesis Statement

The thesis statement of this dissertation is that a software tool can usefully provide information that engenders perceived trustworthiness among distributed team members. The evaluation of the Theseus tool, presented in Chapters 4 and 5, proves this statement to be true.

2.2.1 Interpretation

The interpretation of "usefully" is that the tool is both usable and has utility (Nielsen, 1993). Usability refers to how well users can use the functionality provided in the tool. In this dissertation, usability is operationalized as an overall "usability score," calculated from an existing usability survey instrument (Lewis, 1993). Utility, on the other hand, refers to the tool's ability to do what is needed, i.e., "engender perceived trustworthiness," producing high or low perceived trustworthiness in the appropriate circumstance. This dissertation approximates perceived trustworthiness in two measurable ways: first, as an "inter-personal trust score," also calculated from an existing survey instrument (Johnson-George and Swap, 1982), and second, as an "attribution score" that refers to whether an event is believed by an observer to be caused more by characteristics of the situation, or by an individual's personal traits. "Insights," meaningful and quantifiable observations users make with Theseus, are additional measures of the tool's utility.

2.2.2 Research Questions

The thesis statement led to two research questions that guide this dissertation:

First, what information affects distributed team members' perceptions of others' trustworthiness?

Second, what are the design principles by which a tool can engender perceived trustworthiness among distributed team members?

A solution to the first question is shown with a scholarly review of the theory on trust and the identification of factors that affect its development. In addition, results of a pilot study are used.

The second question is answered with a design space that maps the identification of data sources from which information in the first question can be derived, to a set of factors that influence trust. The design space also guides tool designers by specifying which visual representations are appropriate for which data. Research Question 2 is also answered in part by the Theseus software system, which is designed from this space and subsequently evaluated with human subjects. Chapter 6 synthesizes these ideas into a list of design principles for tools that support trust.

2.3 Contribution

This dissertation contributes a proof-of-concept prototype to test whether software can usefully engender perceived trustworthiness among globally distributed team members. It also contributes the design and results of a laboratory experiment with software development practitioners as subjects. The results show that the proof-of-concept worked. Beyond the Theseus prototype, this dissertation contributes a new way to think about trust, namely through the lens of a design space from which tools can be designed. The design space provides a rich theoretical grounding in the literature on trustworthiness, awareness tools, and visual interfaces. Figure 2-1 uses a metaphor to show Theseus' link to existing theory.

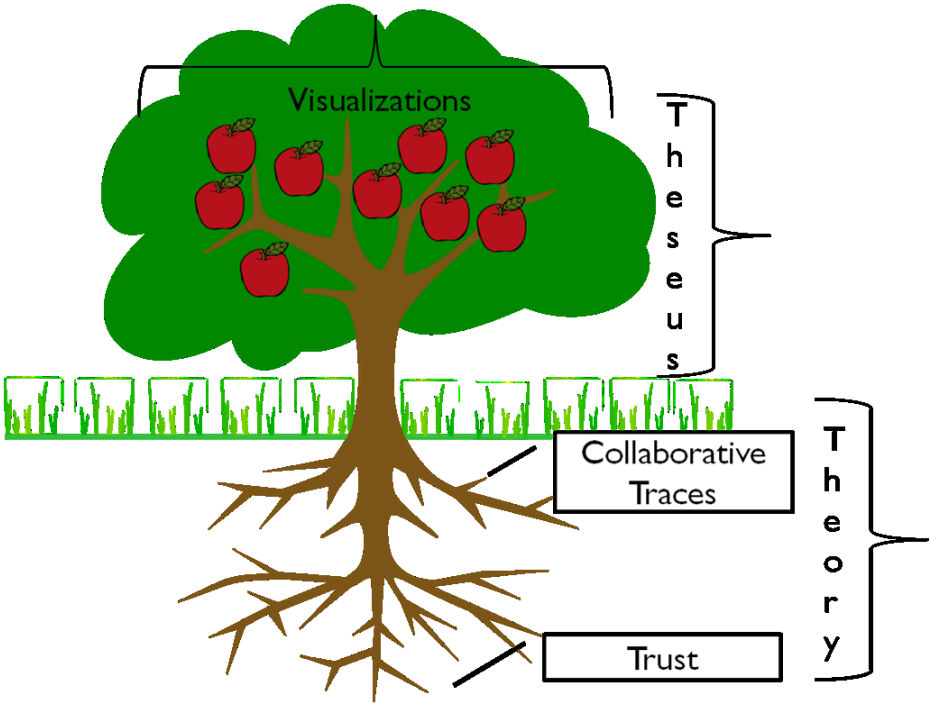


Figure 2-1. A metaphor illustrates the contribution of this dissertation.

The evaluation of Theseus shows evidence that software tools can shape distributed developers' sense of trust toward each other.

2.4 Summary

This chapter motivated and formally stated the central claim of this dissertation: *a tool can engender perceived trustworthiness among distributed team members*. Building a tool is a promising approach because tools can expose important information that helps build expectations upon which trust rests. A visual interface can summarize great quantities of data generated by collaboration tools developers already use and rely upon. Answering the central research questions of this dissertation requires identifying what information is relevant, articulating how to design a tool to expose the information, and evaluating the tool to see if it leads to higher perceived trustworthiness. The next chapter provides a scholarly review of related research work and situates the work in this dissertation within that space.

Chapter 3. Review of Trust, Traces and Visualization

Chapter 2 outlined the approach of this dissertation: summarize data in visual interfaces in a way that increases perceived trustworthiness between globally distributed software engineers. This approach requires understanding more precisely what is meant by trust, what data are relevant, and how the data will be visually presented to developers. This chapter sets the stage for the dissertation: it reviews previous literature by others on trust, *collaborative traces* (defined as such by me in Section 3.2), and visualization. It also presents previous results from a pilot experiment (Trainer et al., 2011) that suggest visualizations of collaborative traces can influence distributed team members' perceived trustworthiness toward others. The pilot study and results from it are significant because they strongly motivated the formulation of the research questions of this dissertation and the subsequent investigation of whether tools can engender trust among distributed team members. As such, the pilot study is reported in this Chapter as background.

3.1 Trust

Trust, or more precisely perceived trustworthiness, is a crucial ingredient of effective and productive collaborations (Handy, 1995; Jarvenpaa and Leidner, 1999; Piccoli and Ives, 2003; Zolin et al., 2004; Wilson et al., 2006). Trust is often defined in terms of one party's expectations of another. Furst et al's literature review of trust (1999) led them to conclude that an individual's trust in their team refers to the likelihood that team members will live up to expectations. Likewise, Sabherwal (1999) considers trust a "state involving confident positive expectations about another's motives with respect to oneself in situations entailing risk." Trust can take two forms: cognitive trust and affective trust. Cognitive trust refers to beliefs about others' competence and reliability, which can lead to individuals engaging in less self-protective actions

and being more likely to take risks. Affective trust refers to trust that arises from emotional ties among group members and reflects beliefs about reciprocated care and concerns (McAllister, 1995, Wilson et al., 2006). A programmer with many years of experience who meets deadlines and rarely introduces bugs will gain another programmer's cognitive trust, while a programmer who is quick to respond to other developers' questions, and sends e-mails to coordinate check-ins will gain another programmer's affective trust.

3.1.1 Availability

Earlier in Sections 1.1.1-1.1.3, this dissertation presented scenarios of lost trust, data from my research team's ongoing studies of distributed software developers (Al-Ani and Redmiles, 2009; Al-Ani et al. 2011). A theme central to each of the scenarios is availability. Moreover, as this chapter will show through literature review, availability is an important aspect of awareness tools for software development. One's availability has been shown to influence others' perceived trustworthiness of them (Levin and Cross, 2004; Butler, 1991; Jarvenpaa and Leidner, 1999). In general there are two aspects of availability: *accessibility* and *responsiveness* (Ackerman et al., 2003).

A user study by McDonald and Ackerman revealed that employees prefer to connect with people who are physically or organizationally close (McDonald and Ackerman, 1998, 2000). These preferences imply two aspects of accessibility. The first is practical accessibility. Practical accessibility refers to the individual's work unit location and their workload (e.g., number of parallel projects). The second aspect of accessibility is organizational accessibility, which refers to their position in the hierarchy, their job title, and job description.

The notion of responsiveness, or willingness to respond, relates to a person's accessibility. Willingness might depend on the helper's demeanor, their history of interactions

with the information-seeker, and any perceived benefits as a result of helping. Asking someone for help usually comes at a cost: the disruption of flow and continuity of ongoing work, which can decrease the productivity of the helper (Szóstek and Markopoulos, 2006). As such, helping may require precious time for little reward, Responsiveness implies reciprocity over the course of initiations and responses in which an individual who answers a question one day is likely to be the questioner at a later time. Jarvenpaa and Leidner (1999) found that initiations and response can influence perceived notions of affective trustworthiness.

3.1.2 Other Antecedents of Trust

Other researchers have reported findings about the antecedents, or sources, of trust as well. A review of factors by Al-Ani and Redmiles led to a "tug-of-war" model of trust (Al-Ani and Redmiles, 2009; see Figure 3-1). Study data indicated that a leader who demonstrates anticipated leadership qualities and adequate time allocation can act as two forces that positively influence trust in a distributed team. Conversely, a large team size, high team diversity, and challenging project deliverable can have a negative influence on trust within a distributed team. This tug-of-war analogy leads to the conclusion that negative forces can be overcome if the opposing forces (leadership and time) are of sufficient strength and quantity.

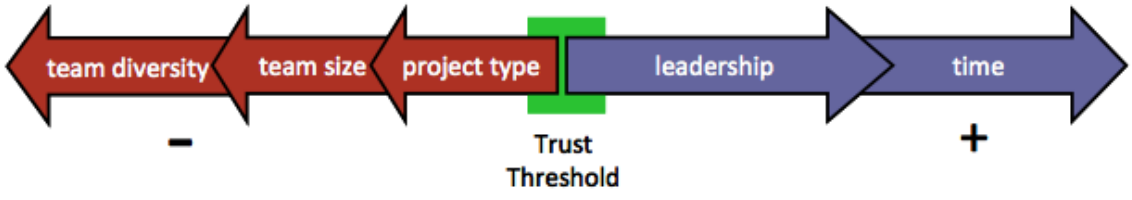


Figure 3-1. Trust Tug-of-War Model.

An additional review of the literature on trustworthiness was conducted by (Trainer et al., 2011). Table 3-1 lists the factors that influence trust found in that review.

Table 3-1. Trust Factors from Literature Review.

Trust Factors	Reference(s)
Frequent initiations and responses	(Butler and Cantrell, 1994; Iacono and Weisband, 1997; Jarvenpaa and Leidner, 1999)
Frequent updates of project progress	(Jarvenpaa and Leidner, 1999; Pyysiainen, 2003)
Reputation	(Ye et al., 2007; Murphy et al., 2002)
Use of multiple communication media	(Bos et al., 2002; Wilson et al., 2006)
Same location	(Olson et al., 2002)
Role	(Meyerson et al, 1996; Pyysiainen, 2003)
Availability	(Butler, 1991; Jarvenpaa and Leidner, 1999; Levin and Cross, 2004)
Leadership	(Al-Ani and Redmiles, 2009)
Frequent meetings	(Johnson and Johnson, 1989; Jarvenpaa and Leidner, 1999)
Years of Experience	(Iacono and Weisband, 2007)
Homophily	(Jarvenpaa et al. 1999; Tseng and Fogg, 1999; Hetzum et al., 2002)
Shared information	(Al-Ani et al., 2011)
Shared photographs	(Bos et al, 2002; Fogg, 2002; Steinbrueck et al., 2002)
Team diversity	(Jarvenpaa and Leidner, 1999; Al-Ani and Redmiles, 2009)
Monitoring	(Ferrin et al., 2007; Moe and Smite, 2007)
High/low Team Size	(Al-Ani and Redmiles, 2009)
Project size/type	(Al-Ani and Redmiles, 2009)
Expertise	(Artz and Gil, 2007)
Gender	(Johnson-George and Swap, 1982; Jeanquart-Barone, 1993; Chaudhuri and Gangadharan, 2003; Maddux and Brewer, 2005; Sun et al., 2006; Buchan et al., 2008)
Language	(Al-Ani et al., 2012)

The literature review of trust factors resulted in a set of constructs shown to influence the perceived trustworthiness between trustor and trustee. The overarching goal of this dissertation is to provide a tool that partially automates the collection and presentation of information that increases one developer's perceived trustworthiness of the other. The question then becomes what information in the globally distributed software developer's environment correspond to the factors that influence perceived trustworthiness in Table 3-1. Section 3.2 introduces the term *collaborative trace* to refer to such information.

3.2 Collaborative Traces

Software design and development activities typically produce a project memory of archival artifacts such as source-code, e-mail lists, design documentation, bug reports and change histories of all this information that developers use as a means toward coordinating the smooth flow of work. For example, dependency call-graphs annotated with authorship information at the line level for each artifact can be acquired from a project's Configuration Management (CM) repository. This information can help developers assess the impact of changes to source-code and whether they will need to coordinate with other developers as a result (de Souza et al., 2007).

Software engineering researchers have created and evaluated a host of awareness tools. In earlier work (Trainer and Redmiles, 2009), I surveyed forty (40) different visualization tools in the research areas of: software tools and environments, empirical software engineering, computer-supported cooperative work, awareness, and visualization (e.g., at the ICSE, CSCW, ECSCW, SOFTVIS, AVI, and VL/HCC venues). Common goals of these tools include providing information about module dependencies and real-time developer activity to proactively avoid check-in conflicts and "breaking the code," understanding change management and evolution of code, recommending the right people and the right artifacts at the right time to

match the task at-hand, and determining opportunistic times to contact developers (and the latter's willingness, in turn) for help, all of which change over the course of development. Research has found that not providing this information can increase the chance of behavioral invisibility and the possibility of reducing trust (Jarvenpaa and Leidner, 1999).

In 2011, I introduced the term "collaborative trace," which refers to the information visualized by these tools (Trainer et al., 2011), most of which were surveyed in my earlier work (Trainer and Redmiles, 2009). Collaborative traces are representations of past and current activity by a group of developers manipulating software development artifacts. For example, collaborative traces include source-code change sets, source-code call graphs and interdependencies annotated with authorship information, work items (such as bugs and feature requests), and e-mail and chat messages. They are generated by collaborative events such as writing source-code, fixing bugs, and correspondence. Table 3-2 lists example collaborative traces organized by the tasks they support and examples of research tools that utilize them.

Table 3-2. Tasks and example tools that support them with corresponding collaborative traces and input sources.

Tasks	Tools	Collaborative Traces	Input Sources
Understand Change Management and Evolution	*ELVIN* (Fitzpatrick et al. 2006), *Command Console* (O'Reilly et al. 2005), *softCHANGE* (German et al. 2004), *SeeSoft* (Eick et al. 1992), *Tesseract* (Sarma et al. 2010)	Change-sets/commit logs and associated meta-data (e.g. authorship, size, descriptions, and time), file revisions, source-code call-graphs annotated with authorship, source-code authorship distribution, bug reports, e-mail messages	CM Repository, E-mail Database, Mailing List, Bug Tracker

Understand Developer Activities	*FASTDash* (Biehl et al. 2002), *SHO* (Ellis et al. 2007), *SeeSoft*, *Jazz* (Hupfer et al. 2004), *Tesseract*	Change-sets/commit logs and meta-data, file revisions, source-code authorship distribution, work items, bug reports, ChangesInProgress, e-mail messages	CM Repository, E-mail Database, Mailing List, Bug Tracker, Workspace
Find Relevant People and Artifacts	*ExpertiseRecommender* (McDonald and Ackerman 2000), *Answer Garden* (Ackerman and Malone 1990), *Hipikat(Mylyn)* (Kersten and Murphy 2006), *Team Tracks* (DeLine et al. 2005), *Tesseract*	Change sets/commit logs, source-code call graphs, source-code authorship distribution documentation, bug reports, developer navigation patterns, developer skill sets, e-mail messages	CM Repository, E-mail Database, Bug Tracker, Employee Directory, Expertise Database, Organizational Chart
Avoid Check-in Conflicts	*CollabVS* (Dewan and Hegde 2007), *TUKAN* (Schummer and Haake 2001), *Palantír* (Sarma et al. 2003)	ChangesInProgress, ChangesCommitted, source-code call-graphs	CM Repository, Workspace
Determine Availability	*Awarenex* (Begole et al. 2002), *CommunityBar* (Tee et al. 2006), *Jazz*	Idle time, artifacts in current workspace, location coordinates, meetings in progress	GPS, Keyboard Input, Video, Shared Calendar, Workspace

For instance, the SeeSoft tool used change sets and authorship information from the CM repository to show who worked on what parts of a source-code file. The Tesseract tool used change sets from the CM repository as well, but to determine dependencies between the source-code. Palantír used change sets for a completely different purpose, to avoid check-in conflicts to the CM repository. In contrast, Awarenex, a tool to show a colleague's "work rhythms used other collaborative traces such as text from the keyboard and meeting information from shared calendars to indicate the availability of different team members.

While tools generally use the same set of collaborative traces, the ways in which they present information from the traces vary more greatly. The discipline concerned with the

presentation of information is referred to as information visualization, or more simply, visualization.

3.3 Visualization

The previous sub-section of this chapter described collaborative traces, information that could show aspects of factors that influence perceived trustworthiness. This sub-section introduces the subject of visualization, reviews existing visualizations to promote awareness, and presents a model of the design space for building visualizations to support trustworthiness.

Said another way, a goal of this dissertation is to expose relevant collaborative traces that influence trust using a visual interface. The visual interface is comprised of various visual representations. Visual representations can be understood as data structures for expressing knowledge. They include, but are not limited to, graphs, tables, maps, diagrams, charts, and icons (Lohse et al., 1994). Visual representations are composed of graphical symbols, or visual elements that are arranged according to a layout. The visual elements communicate visual "sentences" and their meanings (Moody, 2008). Visual elements include text labels, horizontal and vertical bars in a bar chart, and nodes and edges of a graph.

One example of design guidelines for giving meaning to different forms of visual elements, or encoding value, is provided by Bertin's "guidance" (Bertin, 1983; Figure 3-2). Bertin's "guidance" shows which encoding mechanisms (rows) can be used on some simple visual elements to perform basic tasks (columns) such as "association" and "quantity." Other researchers have addressed the encoding of quantitative data using "position," "length," "angle/slope," "area," "volume," "density," and "shape" ordered from most effective to least effective (Cleveland and McGill, 1984; Mackinlay, 1986). Like Bertin, their guidance suggests "color" is better suited for encoding ordinal and categorical data rather than quantitative data.

Figure 3-3 shows Mackinlay's ranking of encoding mechanisms for different types of data from most useful at the top, to least useful at the bottom. The terms visual representations, visual elements, and layout, as well as design guidance for encoding value provide a vocabulary for discussing visualizations and their properties.

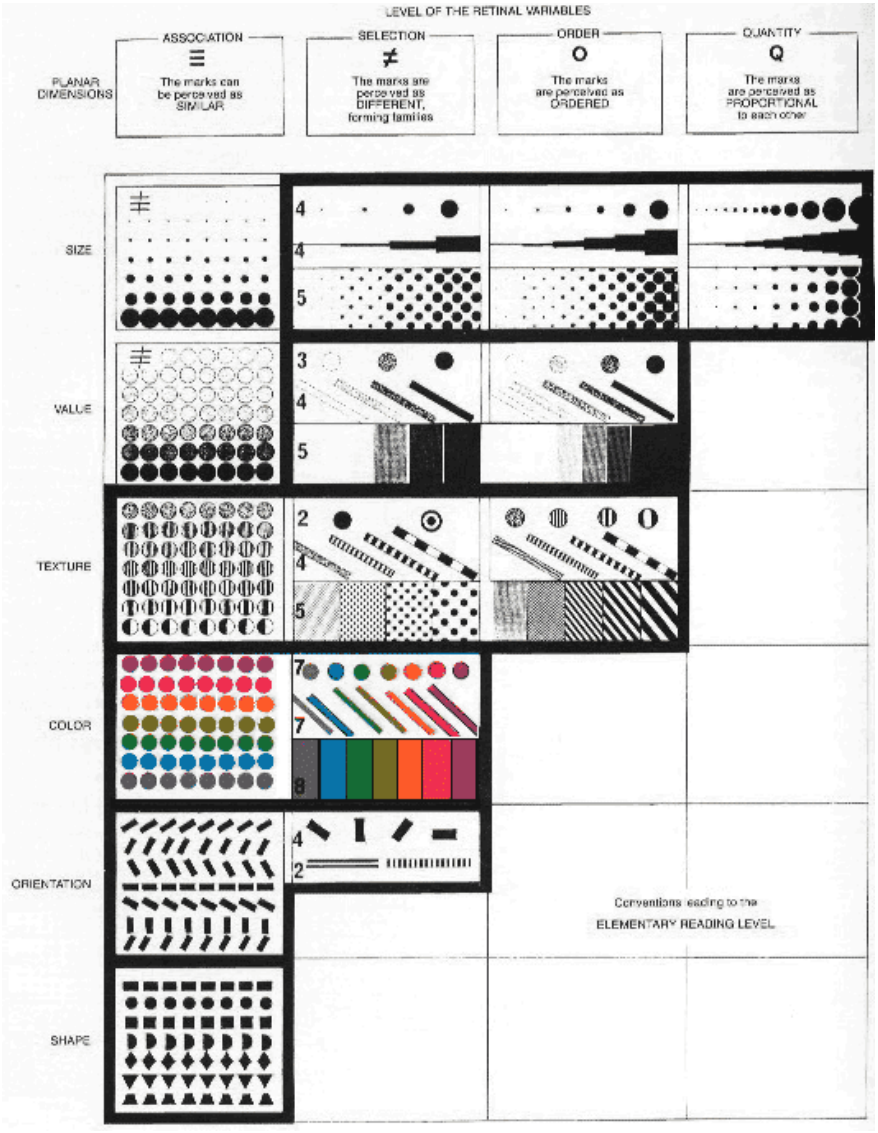


Figure 3-2. An Interpretation of Bertin's guidance from (Bertin, 1983): columns indicate general tasks while rows indicate retinal variables, or encoding mechanisms for those tasks.

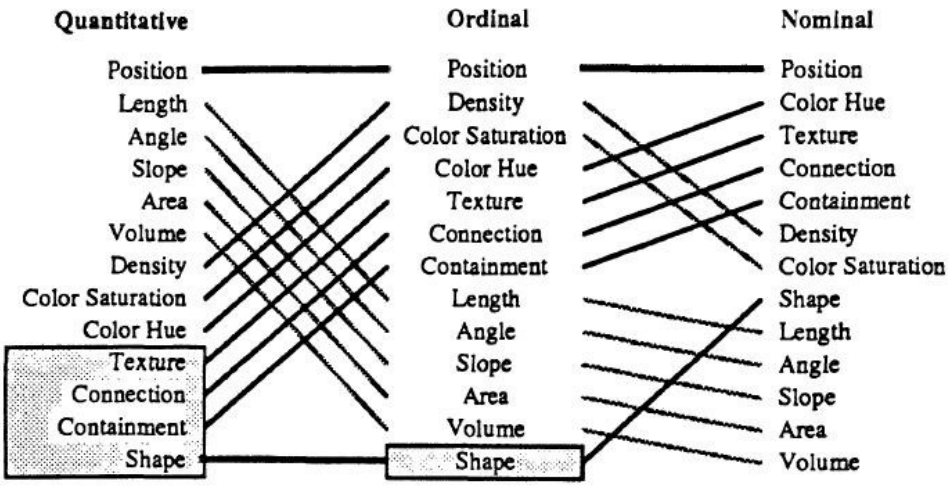


Figure 3-3. Mackinlay's list of encoding mechanisms for quantitative, ordinal, and nominal data, ordered by effectiveness from top to bottom (Mackinlay, 1986).

Nowadays, guidelines for visualizing data are available to developers through web-based applications (e.g. ManyEyes, Swivel, Google Chart Tools), and typically include a baseline set of visual representations organized by tasks they support. Examples include node-edge diagrams and matrices for analyzing relationships between entities, bar charts for comparing values, circle packing to examine hierarchical relationships, and scatterplots to determine the relationship between two variables.

Once a visualization is designed, tool designers typically integrate the visualization into a tool. Some visualizations act as small decorators in the development environment that can be turned on and off, while others are more immersive and require a standalone monitor. In any case, the visualization comprises the user interface of the tool, and is the primary medium through which end users interact. The following sub-section presents a literature review of such tools. Common to all tools is the idea of supporting some aspect of the software engineering process.

3.3.1 Visual Representations in Awareness Tools

In previous work (Trainer and Redmiles, 2009), I identified more than forty tools that visually support the awareness of software development activities. This sub-section summarizes a sub-section of these tools, with particular emphasis on the visual representations used in the interfaces. The sub-headings *Content* and *Visual Representation* are shown for each tool for two reasons. The first is to have an organizing mechanism for understanding the tool's purpose and contribution, while the second is to foreshadow the importance of constructs the sub-headings correspond to (i.e., collaborative traces and visual representations) to the design of Theseus and other tools that engender trust.

Understand Change Management

ELVIN/Tickertape

ELVIN/Tickertape is a lightweight chat mechanism and publish/subscribe notification tickertape integrated into CVS, the popular control versioning software (Fitzpatrick et al., 2006). Developers subscribe to groups (e.g. structured by software component) and messages are generated and sent by the system to the tickertapes belonging to each individual in the group whenever code is committed to the repository (see Figure 3-4). Developers can compose custom messages and send them to members of their group or others'. Developers use the messages sent to their tickertape to initiate chat dialogs with other developers in response to the CVS message (Figure 3-5). ELVIN/Tickertape was evaluated using quantitative statistical analysis of the CVS logs combined with a qualitative analysis of chat logs and interviews. There was evidence of the tool supporting stimulated focused discussion around the changes and supplementing log information with contextual pieces of information such as significance of the changes and

corresponding developer discussions. The tool plays an important role in supporting coordination by combining changes to the code and dialog around those changes.

Figure 3-4. Tickertape message of the form: group: CVS committer: file modified: comment.

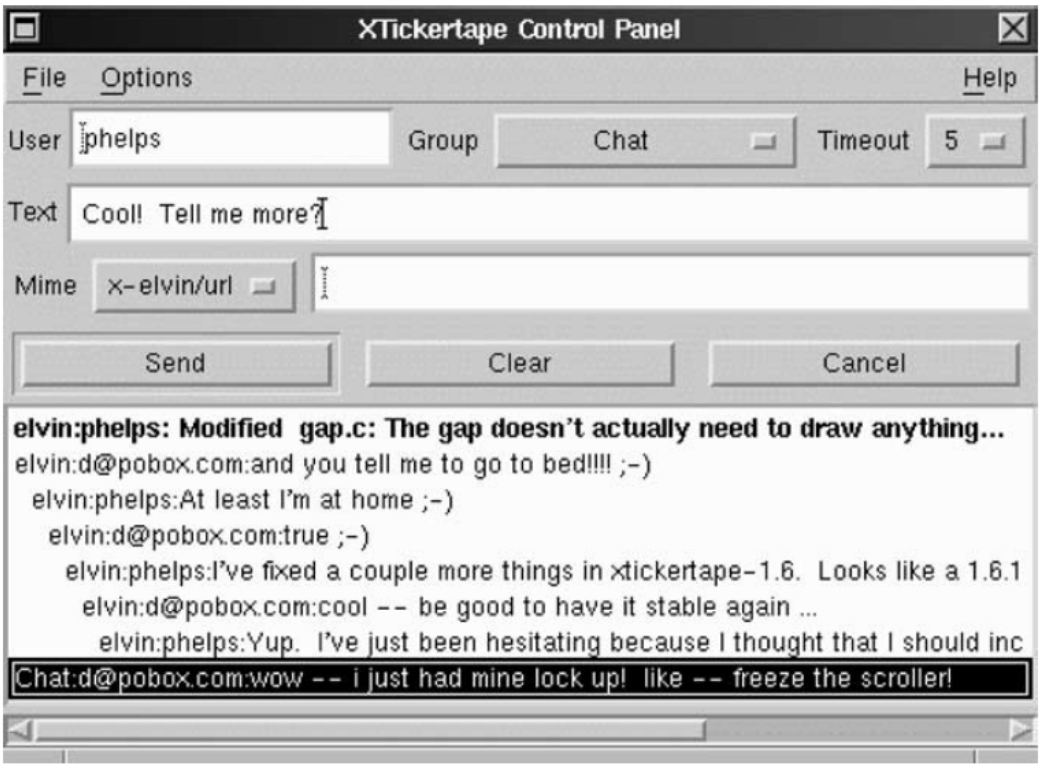


Figure 3-5. Threaded chat message around an ELVIN/Tickertape commit message service.

Content: The tickertape displays a message every time a developer commits code to the repository noting the developer who made the change, the source-code artifact that was changed, and the commit comment. Users can also open a dialog box to start a threaded conversation related to the content of the tickertape message. The dialog box serves as a mechanism to associate individuals with the source-code they are working on. Dependencies between people and between source-code must be inferred by the user.

Visual Representations: ELVIN uses text and simple dialog boxes only. It displays a scrolling "ticker" window with messages and a threaded chat messaging log showing conversations relating to the commits. The representations are light-weight and the tickertape metaphor is well-suited toward displaying abbreviated notifications of all types (e.g., stock prices and sports scores). The representations do not distract greatly from developer work because reading CVS commit messages and project mailing lists are familiar activities.

Command Console

Command Console is a set of linked visualizations on eight consoles designed to help gauge project progress, reveal conflicts, and build a shared understanding of software development activities (O'Reilly et al., 2005). The consoles update in real time in accordance with developers' activities. The system was evaluated during a 5-week long study of an industrial-sized software project. Project managers said that it gave a good high-level view of where action happens in the code and that it helped bring new developers up to speed. Command Console also helped project managers "understand the impact that changes were making." (O'Reilly et al., 2005).

Figure 3-6. The Command Console Display.

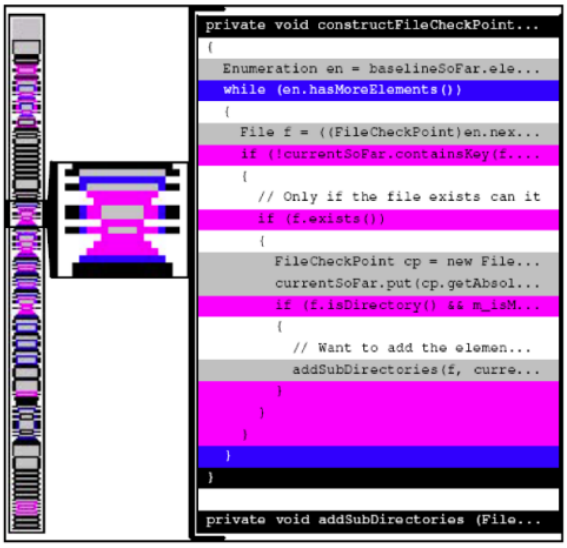


Figure 3-7. The "Complexity Thumbprint," a visualization that displays source-code size and structure.

Content: The linked displays show attributes of source-code revisions, including size and structure, using "complexity thumbprints" (Figure 3-7). One display shows groups of complexity thumbprints organized by the modules and system-level components they belong to

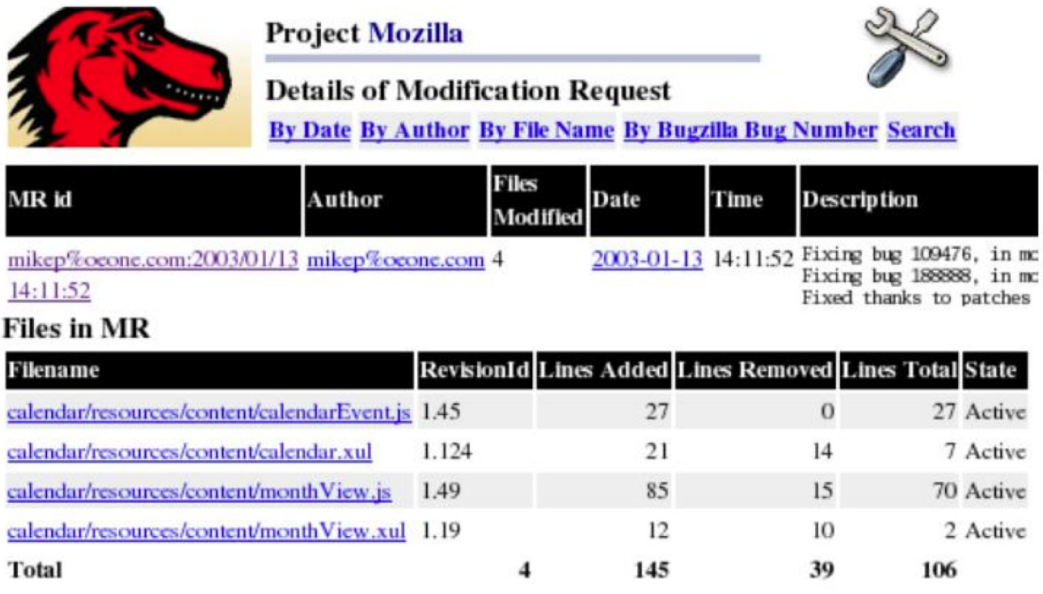
while another displays the ongoing changes to artifacts and warnings of any potential conflicts as a result. It is possible to color the complexity thumbprints by authorship and relate developers to portions of the code they implement, yet dependencies must be inferred from the visualizations.

Visual Representations: Command Console uses graphics and text to convey awareness information. It lays out artifacts in familiar hierarchical list views but makes use of unconventional display setups and representations of source-code developers do not typically use. The tool uses a "war room" shared display, best-suited for co-located teams, yet today's teams are increasingly becoming distributed. It is not clear how or even, if, the Command Console could work in such configurations of teams. Assigning a Command Console unit to each location would be costly. One solution might be to broadcast the large image from a central location to computers that can simply project the image on shared walls or projector screens belonging to remote team members. On the other hand the decrease in resolution associated with most projectors might make it difficult to see and interpret important details in the visualization. The "complexity thumbprints" convey size, structure, and ongoing changes in developers' workspaces. They are novel representations and require some initial learning. However when aggregated, they, crucially, show important patterns in structure at the system-level in concert with tasks color-coded with the source-code that references them.

SoftCHANGE

SoftCHANGE (German et al., 2004) extracts metadata from a CVS repository and a team's corresponding issue tracking system and correlates both. It analyzes different revisions of the same files to determine the exact nature of the changes made and attempts to classify changes based on the issues they address (e.g. new features, code defects). The tool provides graphical

views of source-code, authors, and their relationships, such as what source-code was modified together and who modified what files when.



Project Mozilla

Details of Modification Request

By Date | By Author | By File Name | By Bugzilla Bug Number | Search

MR id	Author	Files Modified	Date	Time	Description
mikep%oeone.com:2003/01/13 14:11:52	mikep%oeone.com	4	2003-01-13	14:11:52	Fixing bug 109476, in mc Fixing bug 188888, in mc Fixed thanks to patches

Files in MR

Filename	RevisionId	Lines Added	Lines Removed	Lines Total	State
calendar/resources/content/calendarEvent.js	1.45	27	0	27	Active
calendar/resources/content/calendar.xul	1.124	21	14	7	Active
calendar/resources/content/monthView.js	1.49	85	15	70	Active
calendar/resources/content/monthView.xul	1.19	12	10	2	Active
Total		**4**	**145**	**39**	**106**

Figure 3-8. A hypertext view of the details of a change request.

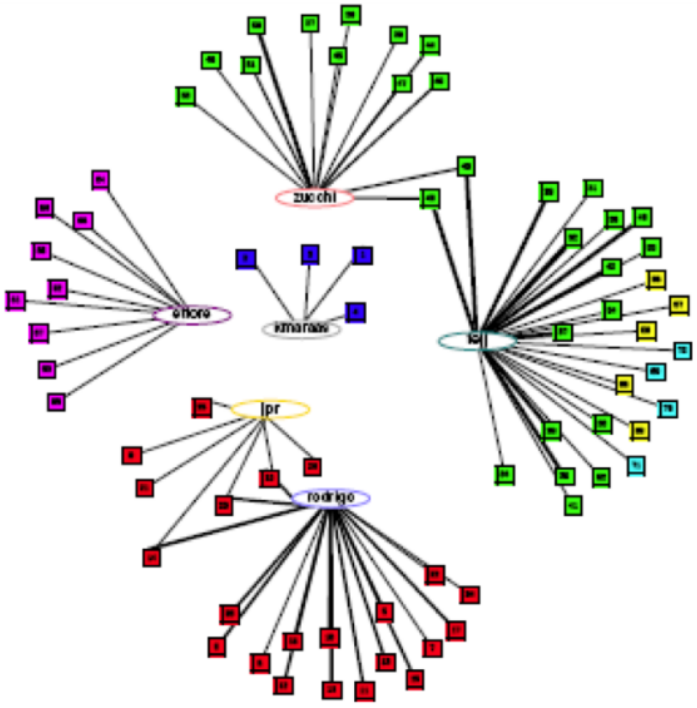


Figure 3-9. A network graph of authors and the files they modified, color-coded by module.

Content: SoftCHANGE displays changes from CVS along with issues managed by a project issue tracker and the developers involved. The tool displays relationships between them, such as relationships between number of files and number of open issues, numbers of functions added versus number and types of change issues, and developer activity compared with the number of change issues. The graphs can also be used to show who is working on what code, but not who depends on whose code.

Visual Representations: SoftCHANGE uses a combination of textual descriptions and graphical charts to convey information. It uses a hypertext view to show details related to specific change requests, including when and why a change was made, the type of change, and if it was fixed. Graphical views such as scatter plots of time ordered information allow users to inspect relationships between bug rates and attributes of files changed. Color-coded network diagrams are used to show relationships between developers and the files they have modified. It is not clear whether these visualizations are linked together or how they are used in concert to reason about changes made to the system.

Find Relevant People and Artifacts

Expertise Recommender

Expertise Recommender is a tool for locating expertise needed to solve difficult technical problems (McDonald and Ackerman, 2000). Development of the tool was preceded by a field study that resulted in analytical guidelines for locating expertise: expertise identification, expertise selection, and escalation. Expertise Recommender provides support for this model of expertise. A user specifies a knowledge request via a client interface (Figure 3-10) and sends it to a server that processes the request and makes recommends of individuals with matching

expertise. Before sending the request, the user selects a location heuristic (e.g. "change management," "tech support") to define the repository that should be searched (CM system or tech support database respectively) and a filter (e.g. "social network," or by department) to filter out people not associated with particular groups in the organization. If results returned by the tool are not sufficient, the expertise-seeker can "escalate" the request to other contacts in different departments who have higher authority or access to more resources.

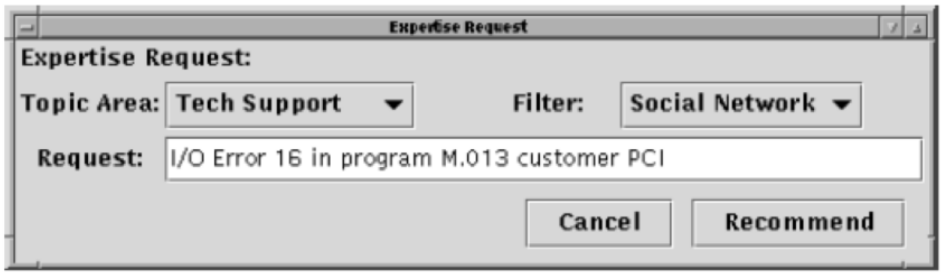


Figure 3-10. An expertise request dialog window.

Figure 3-11. Recommendation results returned by Expertise Recommender with the option to escalate the request.

Content: Expertise Recommender links people and the extent with which they have worked on technical artifacts to offer recommendations of people who are best-suited to address problems and requests defined by the expertise-seeker. Recommendations are based on data contained in CM repositories and technical support databases as well as individuals' social networks. The recommendations returned by the tool include contact information for each individual and their location yet no description of the work they have performed relative to the technical issue. It also gives no indication of whether or not that person may be currently available.

Visual Representations: The tool's UI makes very little use of graphics, displaying small dialog windows with the familiar "look and feel" (text fields and buttons) for filling out and sending expertise requests. It returns recommendations via another dialog window in a scrollable list with textual descriptions of experts and their contact information. It is a stand-alone system instead of integrated into the tools used by technical support representatives or the development environments used by developers. On the other hand, it is relatively lightweight as well. No usability issues are reported by the authors.

Hipikat

Hipikat is an awareness tool that leverages project archives to make recommendations of related artifacts related to developer tasks, such as changing a piece of source-code, to support developer productivity (Cubranic and Murphy, 2003). Given a change request, a developer queries the Hipikat interface for related artifacts and is returned a list of related source-code and bug reports (Figure 3-12). Upon inspecting the results and their relevance criteria, the developer drills down to each recommendation to find the information of interest.

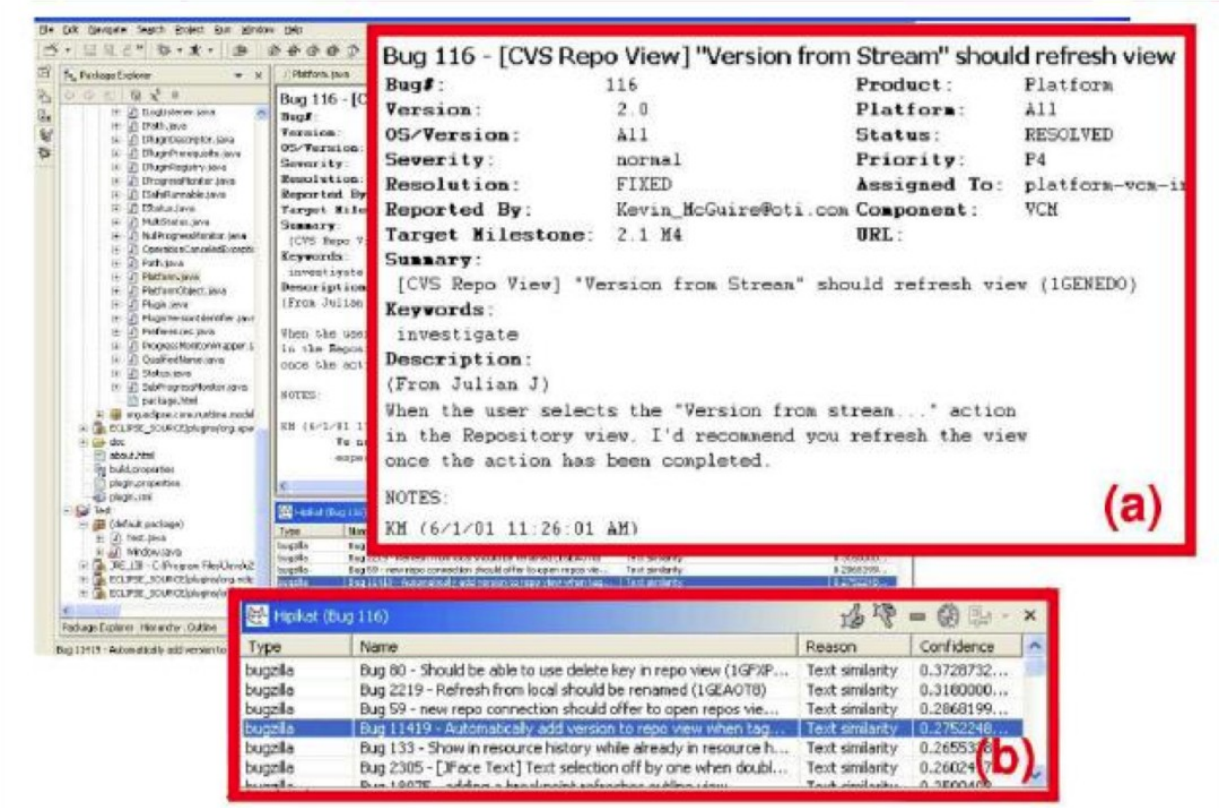


Figure 3-12. Hipikat UI integrated into Eclipse displaying the change task (a) and a list of related artifacts (b). © 2003 IEEE.

Content: Hipikat crawls CM repositories and issue tracking systems and recommends related artifacts (e.g. source-code, change requests) based on the task the user (i.e. developer) is currently performing. It explicitly models relationships between versions of source-code and bugs but source-code authorship and bug assignment information is not shown. These relationships must be inferred.

Visual Representations: Hipikat primarily uses textual decorators within the development environment to convey recommendations. It uses lists and windows fully integrated into the Eclipse development environment, thus the interface has the same "look and feel." Queries are performed by right-clicking on artifacts in the familiar hierarchical source-code view and entering search terms into Eclipse dialog boxes. Results and confidence are expressed via

textual descriptions in the same list dialogs used by Eclipse. Because of this design, context-switching is noticeably reduced and no substantial time spent "learning" the interface is required.

Team Tracks

Team Tracks is a visualization that unveils developers' patterns of navigation through source-code in an effort to support comprehension of that code by users who are new to the code (DeLine et al., 2005). The visualizations are integrated into Microsoft Visual Studio dialogs much in the same way Hipikat is integrated with Eclipse. Team Tracks is based on two assumptions: 1) parts of the code developers visit more frequently are more important to someone new to the code and 2) the more two snippets of code visited are visited in succession, the more likely they are to be related. As such, the visualization is composed of two displays within the development environment: "Code favorites" (e.g. frequently visited code) and "Related Items." The tool was evaluated quantitatively through a program comprehension quiz and user satisfaction ratings as well as qualitatively through interviews with developers who used it to complete typical development tasks such as change requests.

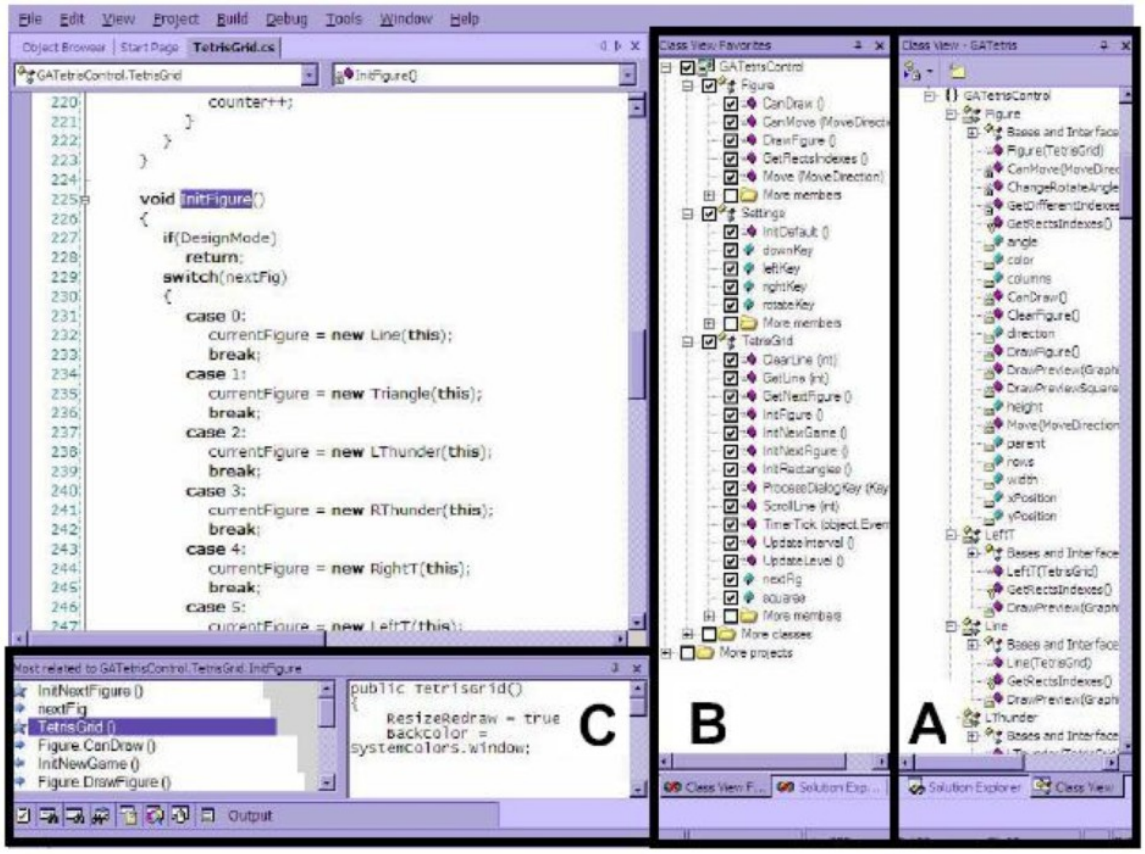


Figure 3-13. Team Tracks interface in Visual Studio displaying A) A standard directory/file view, B) Code Favorites and C) Related Items to source-code selected in the development editor. © 2005 IEEE.

Content: Team Tracks uses windows integrated into Visual Studio to display the source-code most viewed by developers and rank-ordered related methods to the source-code currently in focus in the development editor. The tool uses icons to show whether the methods have incoming or outgoing dependencies to the source-code in the editor but the call graph for a method is not displayed. This could make traversing the call path easier for the user. No authorship information is shown by the tool so it is not possible to determine who navigated dependent code without querying a CM repository. Navigation data from a senior architect, for example, would possibly be more revealing than a programmer.

Visual Representations: Team Tracks uses a combination of textual decorators and visual icons to convey awareness information. Like Hipikat, Team Tracks' visualizations are integrated into familiar windows part of the everyday implementation work of software developers. The views are anchored around where developers spend most of their time in the editor: the source-code. The "Class Favorites" window displays favorite code items the same way in which the Visual Studio environment hierarchically displays directory structures. The "Related Items" view uses list structures also found in the environment interface. In addition, the interface uses good information presentation principles: small horizontal bars appear next to source-code items indicate relative ranking, inviting direct comparisons and contrasts to other ranked items (Tufte, 1990; 2006).

Avoid Conflicts

Palantír

Palantír is an Eclipse plug-in that supports developers in identifying and avoiding conflicts that arise from committing different versions of the same file to a CM repository (Sarma et al., 2003). The tool increases awareness by continuously sharing information regarding other developers' actions on files in the workspace, for example the potential for conflicts and the severity and impact of changes to those files. The tool was empirically evaluated using a lab experiment (Sarma et al., 2008b). The results indicate that Palantír increased self-coordination among users and, as such, led to fewer conflicts.

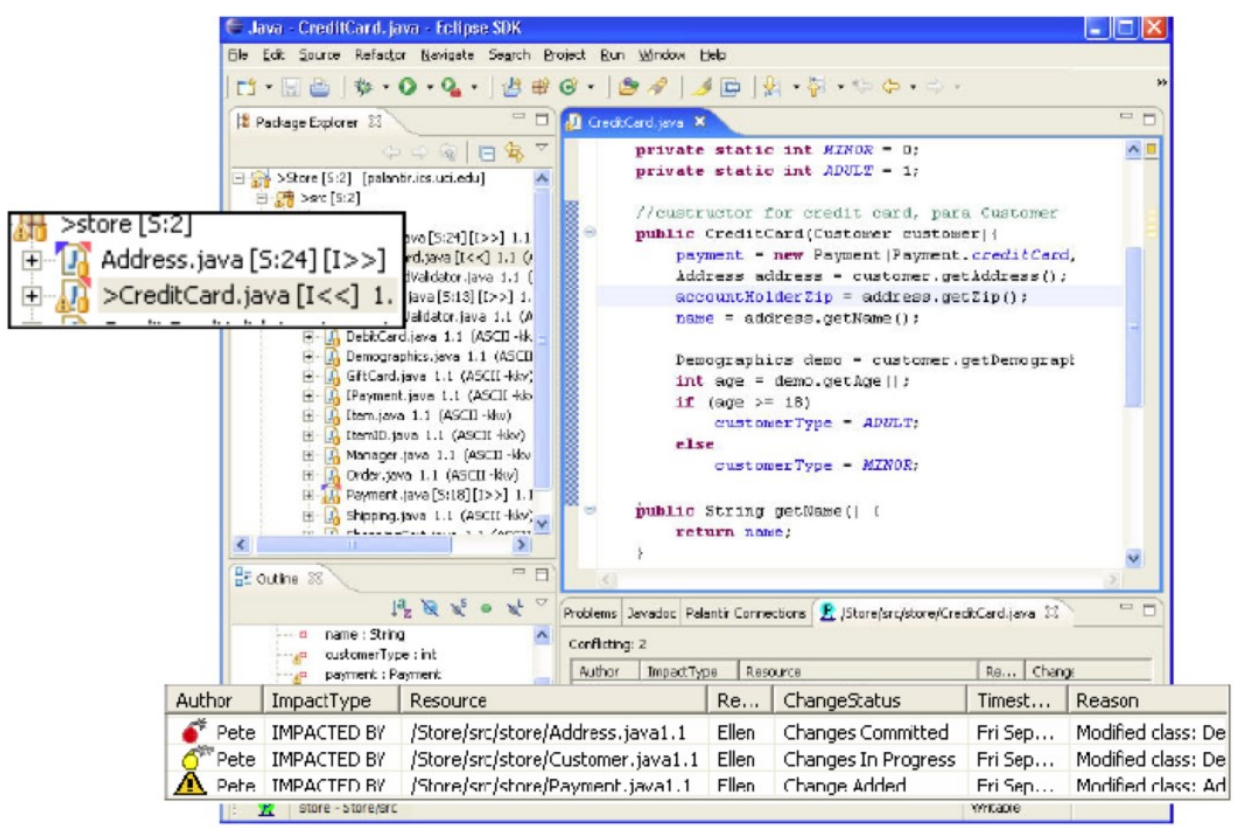


Figure 3-14. Palantír-enhanced Eclipse workspace with annotations to files in conflict (left) and description of impacts of changes (bottom). © 2003 IEEE.

Content: Palantír extracts information from a number of control versioning repositories and monitors activities in the workspace for changes to the local and repository versions of source-code. It analyzes the differences between files in order to compare the number of lines changed and calculate a measure indicating the severity of the changes. The tool then translates these activities to events that it subsequently shares with all affected user workspaces. Palantír shows textual descriptions of these events, including who is changing what, whose code will be impacted by the change, and when the change happened or is currently in progress. These events describe dependencies of the type "impacted by" rather than dependencies of the type "is called by."

Visual Representations: Like Hipikat and Team Tracks, Palantír's interface is integrated with the development environment so as not to distract from current tasks. The source-code editor anchors interactions with Palantír because much of the work involved in analyzing conflicts revolves around looking at source-code. It uses visual decorators in the resource view of Eclipse to indicate types and sizes of emerging conflicts and textual representations to indicate impact severity and give details about who is involved, what code was affected, and when the changes occurred.

TUKAN

TUKAN is a collaborative development tool with the broad goal of orienting developers around code and promoting awareness of other developers' activities (Schummer and Haake, 2001). It displays graphs of artifacts that are semantically related and extracts versioning information from a CM repository to determine the severity of potential conflicts.

Content: TUKAN connects to a source-code repository and semantically analyzes source-code artifacts and determine which ones are related due to dependencies, use, and inheritance relationships. It collects information from developers' Smalltalk workspaces to identify who is working on what files and whether conflicts will emerge. It annotates the graphs of source-code artifacts with icons signifying active developers and weather icons corresponding to the severity of parallel work on those same artifacts. Developers can get a feel for who will be impacted to changes to their code and whether they will be affected by changes to others' code.

Visual Representations: TUKAN primarily uses graphics to convey awareness information. It is integrated with an online SmallTalk development editor so as not to distract from development activities by way of context-switching. The tool uses notations familiar to software developers

like dependency graphs and graphs showing use and inheritance relationships. The graphs are annotated with intuitive visual decorators like "people" icons showing developers who are working on source-code as well as weather metaphors that convey "sunny" (and thus positive) or "stormy" (and thus negative) states with respect to potential conflicts.

Determine Individual Availability

Awarenex

Awarenex (Begole et al., 2002) is a visual awareness tool that reveals patterns in people's work schedules such as: where they are during times of the day, what times of the day they are available, when they are busy with appointments, what times they arrive and depart for work, what time zones they are working in, and when they break for lunch. The tool logs input received by the user's keyboard to determine when they are active and available for contact and inactive. It collects data from online calendars to infer when users are in appointments. The visualizations show activity over a 12 hour day during the 10 months the data were collected.

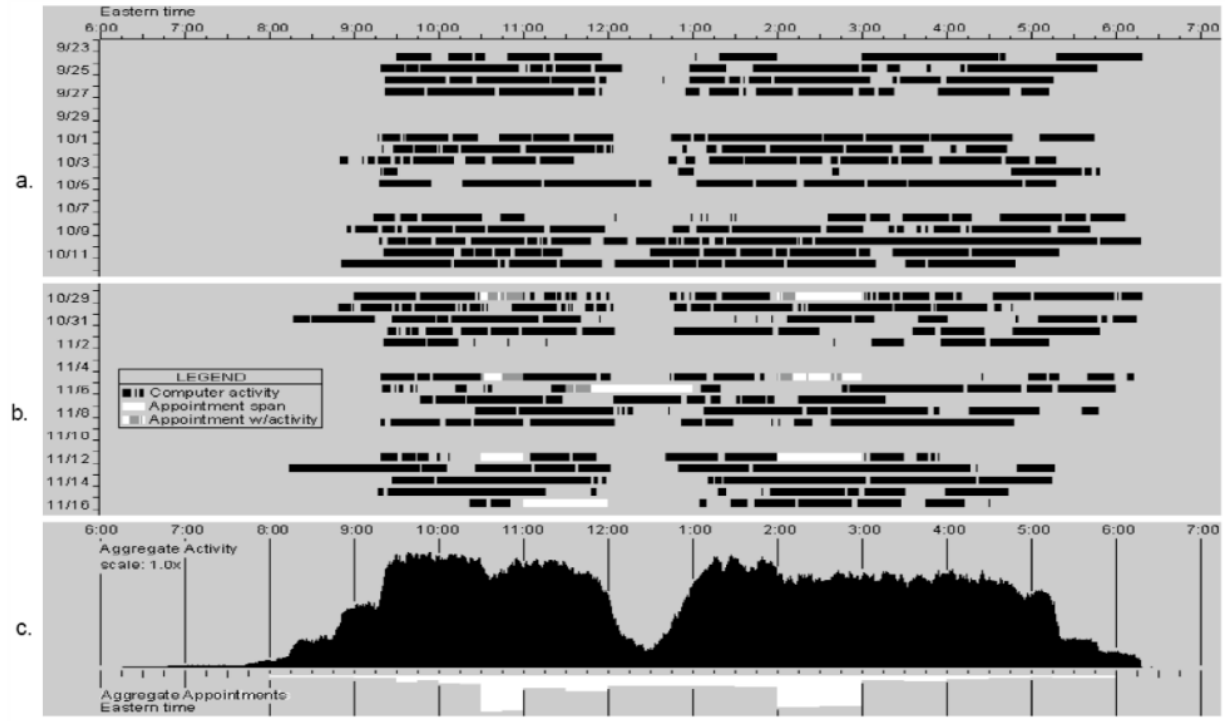


Figure 3-15. Visual interface showing a) 3 weeks of activity with active periods in black, b) another 3 weeks with white indicating appointments, and c) an aggregate view over 10 months of daily activity. Arrivals and departures can be seen in a) and b).

Content: Awarenex uses input from the keyboard to determine when users are active or inactive. It logs whether the user is reading or sending e-mails and analyzes their online calendar to determine when the user is in an appointment and unreachable. Location data is also collected (e.g. office, home, lab). People can be mapped to their general work activities as well as when and where they do work. Thus their availabilities can be inferred from the visualizations.

Visual Representations: Awarenex uses graphics and minimal text to visualize activity using a time-ordered x-y axis. The x axis shows a 12 hour day while the y axis shows days increasing from top to bottom (Figure 3-15). Activity is graphically represented by horizontal bars with length representing time and different colors indicating certain types of activity (e.g.

appointments, reading e-mail, etc.). Reading downward, the horizontal bars invite direct comparisons and contrasts (Tufte, 1990; 2006) in availability and types of activity day-to-day.

Community Bar

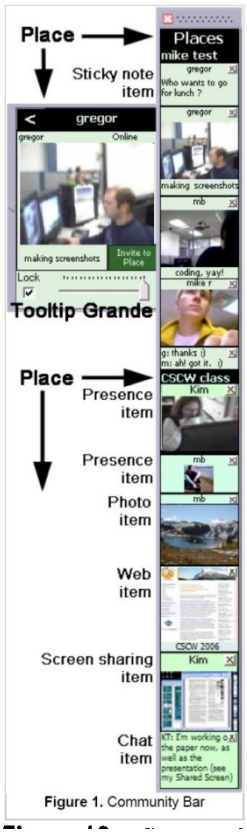


Figure 3-16. Community Bar content is composed of places, presence information, chat dialogs, sticky notes, photo items, and web items (e.g., webpages).

Community Bar (Tee et al., 2006) is an application that allows users to share their screens with one another, engage in real time chat messaging, share video information about where they are, and post digital objects with which they are working. The tool supports progressive visibility of information users choose to reveal. Users can control the visibility of their information by blurring sensitive parts of content. Feedback from usage of the tool suggested

that it was useful for opportunistic interactions, monitoring when people could be interrupted, and to measure progress on collaborative tasks.

Content: Community Bar allows the sharing of multiple types of artifacts such as video frames, images, web resources and even the users' screens (Figure 3-16). It shows individuals and the artifacts they choose to share but not who is monitoring, and thus dependent on, the availability of whose artifacts. Shared content is visible to everyone connected to Community Bar except when privacy controls are used to selectively display the content. For example, someone might want to share an unfinished draft of a document only with their collaborator(s) and then release it with full visibility once it is completed. One's availability can be determined by looking at their desktop screen to determine what they are working on, their status message, or their webcam.

Visual Representations: Community Bar's interface consists of one long vertical window that lays out posted artifacts from top to bottom. It uses graphics and when applicable, text corresponding to peoples' status, short announcements to others in the group, and chat logs.

Understand Developer Activities

FASTDash

FASTDash (Biehl et al., 2007) is a visual "dashboard" widget embedded on a single monitor or large, shared screen that allows software developers and their teams to monitor all ongoing activities including what files are checked out, what is being changed and by whom, what files are being viewed, and what files are undergoing debugging. The tool's requirements were gathered from interviews with 13 developers. In response to these interviews the tool was built and qualitatively and quantitatively evaluated using 6 of the original 13 programmers as

users. It was shown to improve team awareness, reduce reliance on shared artifacts, and increase team communication.

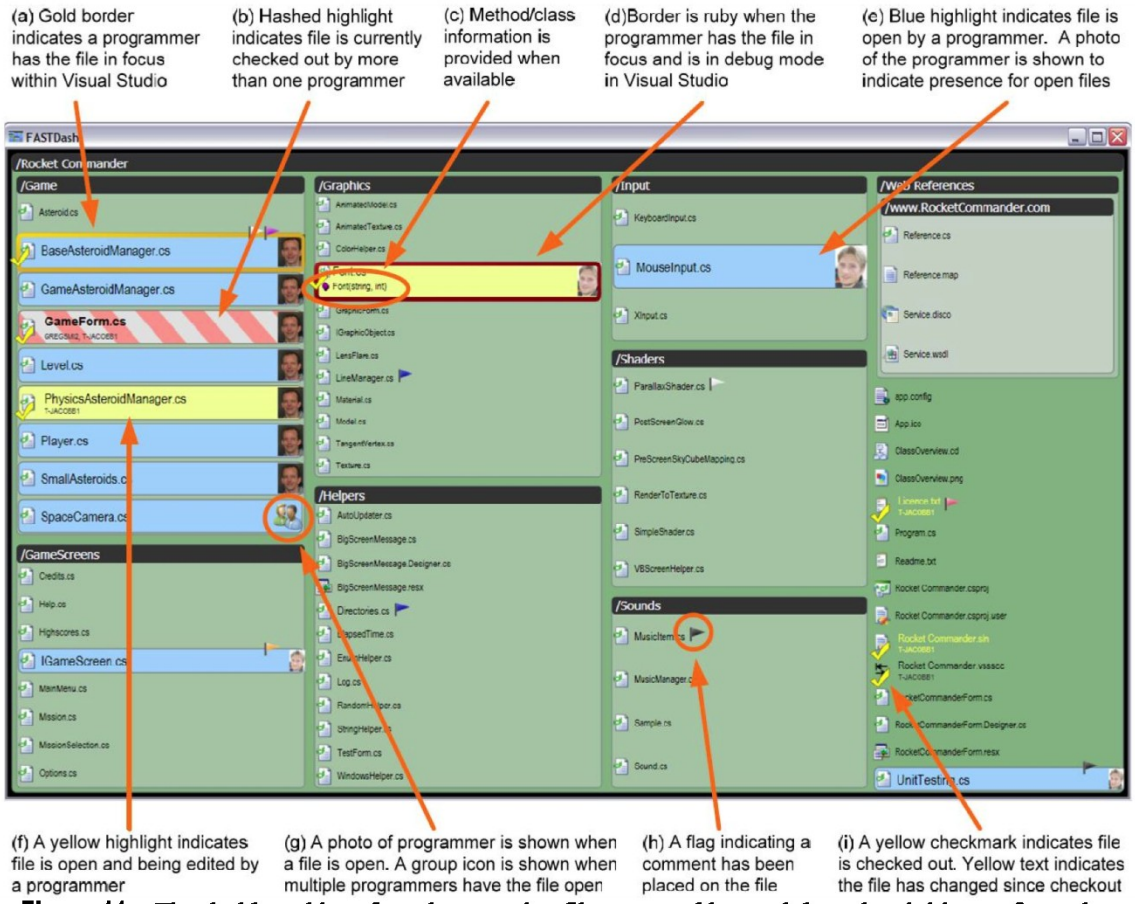


Figure 3-17. The dashboard interface shows active files grouped by module and activities performed on them by active developers.

Content: FASTDash displays a software project's files organized by module and graphical annotations to the files corresponding to activities being performed on them by developers, such as making changes, viewing the file, debugging it, adding documentation, etc. The interface alerts the user to who is working on what but gives no evidence of whether that work will affect the former's own work. The user must infer that themselves.

Visual Representations: FASTDash uses a combination of graphics and text, but mostly graphics, to show awareness information (see Figure 3-17). Unlike other visualizations such as Palantír, Team Tracks, or Hipikat, FASTDash exists as a standalone application. It uses familiar file icons for resources in the workspace (e.g. photos, .source-code files, authors). However a problematic aspect of the interface design is that it uses inconsistent forms of visual cues to indicate the different states of files: hashed highlighting to indicate files checked out by multiple developers and thus potential sources of conflict, yellow highlighting to indicate files that are open and being edited, gold borders to indicate files that are in focus in the editor, and checkmarks superimposed on files to indicate files that are checked out. The variety in different graphical annotations for related states of activity makes it difficult to remember the meaning of each. This is an interface that will require time to learn before it can be used efficiently by teams, yet this observation was omitted from the authors' field study of the tool.

Jazz

Jazz is a collaborative development environment (Booch and Brown, 2003) that enhances the existing Eclipse platform with collaboration mechanisms for use in small-team settings (Hupfer et al., 2004). It extracts activities from user interactions with the interface and the local history of source-code and work items (i.e. reports that detail work to be accomplished) to 1) monitor what and how source-code is being changed and 2) push this information to other developers in the workspace who have subscribed to be notified of these ongoing changes. The Jazz Band (bottom of Figure 3-18) is a shared buddy-list from which users can initiate interactions with others (e.g. screen sharing, chat sessions) without the overhead of leaving the IDE and launching other applications.

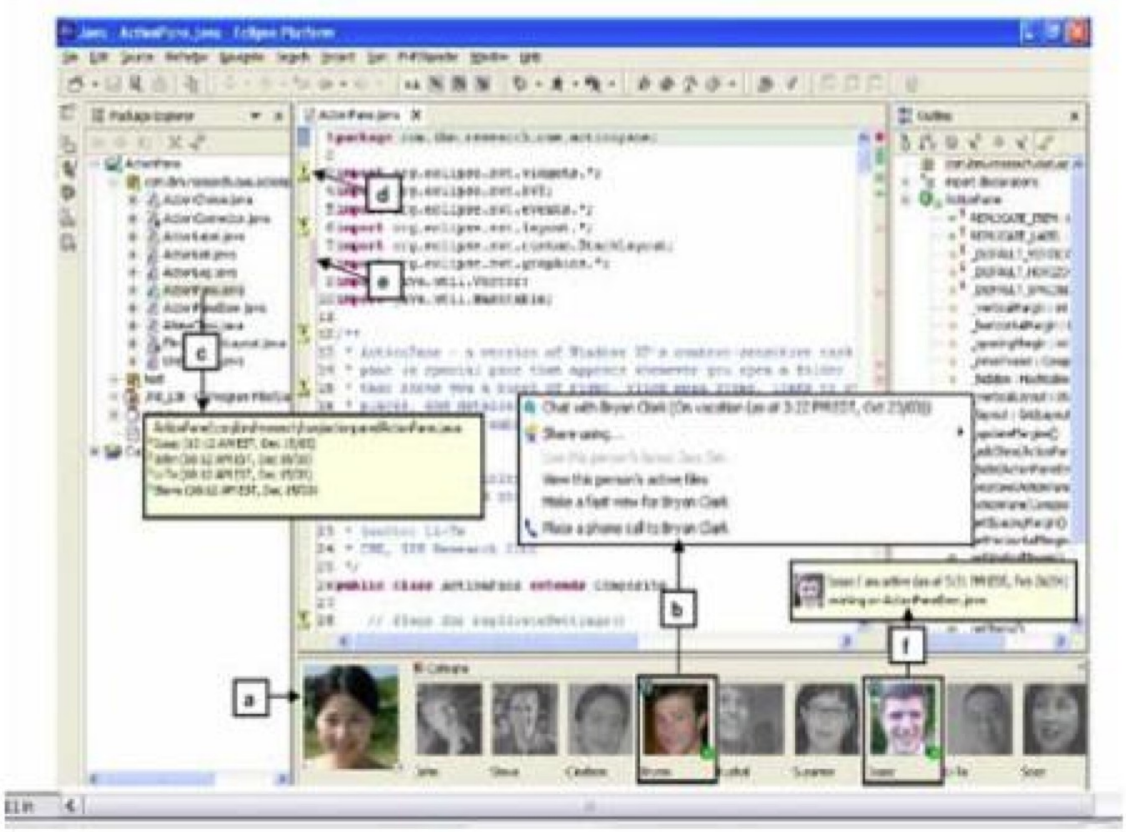


Figure 3-18. The Jazz environment showing a) team members, b) communication options for interacting with a team member, c) workspace files and resources annotating their current states and who is changing them, d) an anchor for a chat transcript pertaining to the opened code, e) a recently modified portion of code, and f) a team member's status/location.

Content: Jazz displays ongoing activities performed by team members within the development environment, including the changes they make to source-code, comments and documentation relevant to the code, and focused group discussion. By structuring activities around team mates, Jazz enhances developer awareness of others' contextually relevant interactions with the code and each other. When a developer selects a snippet of source-code, for example, and initiates a chat everyone else immediately knows broadly what will be discussed. No one needs to direct them to another area in the code. Jazz can also be used to gauge developer availability through

the use of Jazz Band buddy list status messages and tooltips in the hierarchical source-code window describing who is changing what files.

Visual Representations: Like Palantír, Team Tracks, and Hipikat, Jazz visualizes awareness data in the context of current development activities unraveling in the development environment. FASTDash and Jazz are similar in their goals and motivations yet FASTDash uses novel visualizations in place of visualization that have the same "look and feel" as the environment. Jazz, on the other hand, uses easily identifiable visual cues and annotations to source-code in the editor (as in Palantír) to signify the status of files, chat logs around the code, and documentation. The biggest enhancement to the existing editor is the Jazz Band (Figure 3-18a), which features avatars of team members augmented with mechanisms for different forms communication. The Jazz Band gives users a choice for what communication medium to use (e.g. chat, phone (VOIP), or e-mail). Providing different options is critical—yet not addressed in the majority of the tools surveyed in this paper—since it has been shown that different users prefer to be contacted different ways (Herbsleb and Grinter, 1999). For example, non-native speakers prefer e-mail because it gives them time to compose their thoughts and responses. Responding in real time can become time consuming and frustrating.

This sub-section summarized visual awareness tools to support collaborative software development activities. By systematically reviewing existing tools from the literature, has built up a vocabulary for describing visual representations. However, results from studies with the tools above did not include developers' perceptions of others' trustworthiness. As explained in this dissertation, the subject of study is trustworthiness. Accordingly, the author sought to understand the impact of visualizing collaborative activities on trustworthiness. The next sub-section describes this investigation.

3.4 Impact of Visualizing Collaborative Traces on Trust

In earlier research (Trainer et al., 2011), I conducted a pilot study to investigate a variation of this dissertation's thesis statement: can visualizations of collaborative traces influence a developer's sense of team members' trustworthiness? The experiment suggested that they could, but that certain information, such as a developer's ongoing assignments, could be more useful than other information, such as the inter-dependencies between other team members and them. More profoundly, the pilot study served as inspiration to explore the inter-relationships between trust, collaborative traces, and visual representations, leading to a design space (an organizing mechanism) from which to build tools to support trust. The design space is presented beginning in Section 3.5.

3.4.1 Pilot Study

My research colleagues and I recruited six UCI Informatics graduate students for the study. Each participant was presented with two scenarios and corresponding visualizations. In the first scenario, the participants were presented with three Ariadne visualizations that consisted of names of male developers within the organization and who were distributed across multiple sites. Participants were informed that all developers listed in the visualizations were of the same gender (male) to avoid gender bias. The visualizations presented to participants represented a history of work by the members in different teams. None of the developers in the visualization were collocated with any other member to eliminate bias toward locality. Each study participant was asked to assume that they were part of a development team and that they needed to assume that they will implement features throughout the program's source code.

Participants were also informed that they needed to assume they were having difficulty implementing their work and would need to find others in the organization that were

knowledgeable about the code to help them. We asked subjects who they perceived as competent and possessing the necessary technical expertise. In addition, we asked which developers might be receptive toward helping as well as whom the subjects would prefer working with. We explained that they could only contact someone listed in one of the three visualizations.

In the second scenario the participants were presented with two new Bracket visualizations which represented the interdependencies in the team the participants were currently assigned to, and a different list of developers. They were informed that their role in this team was to re-use existing code and to utilize code written by other members of their team. Here, the participants were also informed that they were having difficulty using the main code module and that the documentation was not very helpful. Participants were asked to choose a team member to contact for help. Once again, they were informed that all of the male team members were distributed in different countries and time zones but that the visualizations represent the current work being performed on their team's modules.

3.4.1.1 Study Results

When asked about assessing a developer's technical expertise (i.e. cognitive trust), participants used the number of incoming dependencies to determine whether the developer was trustworthy relative to others on the team. Similarly, when asked about meeting deadlines and commitments (i.e. affective trust), participants cited the incoming connections to a developer when making their choice. However, when asked about the contributions of others in more detail (e.g. whose work should be checked for accuracy or who would they trust to use their own code), the visualizations fell short. Participants remarked they would need to see more information, such as developers' work in other projects, views of the source-code in question, and the rate at which developers introduced bugs into the system.

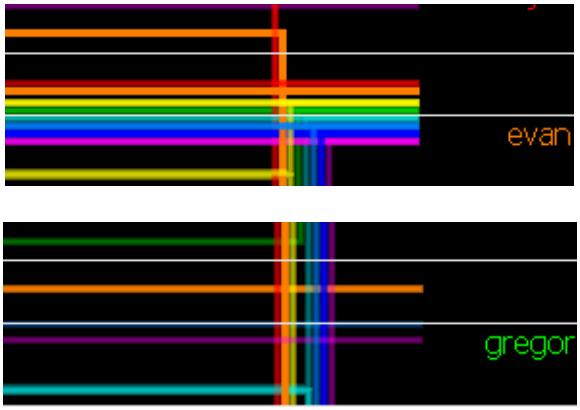


Figure 3-19. Participants chose developers with larger dependency bands (left), rather than smaller ones (right), as possessing technical expertise and meeting deadlines and commitments.

Visualizing only a small subset of collaborative traces was not considered useful. For example, participants in the Ariadne pilot experiment used interdependencies from a call-graph annotated with authorship information as the sole collaborative trace in order to appraise developers' cognitive and affective trust. Developers with many incoming interdependencies were viewed as trustworthy. The remaining developers were not. It may have been the case that these individuals were involved heavily in the design of the code, rather than its implementation, as senior architects. Alternatively, it may have been the case that they were more heavily involved in other projects going on at the same time. In this instance, views of these individuals' other activities would have been quite useful as some of the participants in our study suggested. Despite these flaws, the experiment provided evidence that visualizing collaborative traces can impact people's sense of trustworthiness toward others. From just a brief glance at incoming dependencies, participants quickly formed impressions of developers' trustworthiness.

3.5 Collaborative Traces for Trust

Combining the review of trust factors and the idea of collaborative traces, Table 3-3 was produced. The columns of Table 3-3 are trust factors, i.e. information that affects people's perceptions of others' trustworthiness. The authors acquired these factors from the previously mentioned review of the literature on trustworthiness (Table 3-1). Rows in the matrix are collaborative traces as well as other data that, although do not fit the definition of a collaborative trace because they are not an outcome of work, provide information about an individual or the organization, such as an org. chart or the time zone in which a developer is located.

For example, change-sets and authorship from source-code can reveal information about developers' expertise and their general development activity. A developer whose code is used by everyone on the team can be perceived as having expertise in that area of the codebase (Trainer et al., 2011). In addition, the rate at which developers send out and respond to e-mails and instant messages can signal their overall willingness to initiate and respond to the cares and concerns of others. Developers located in different time zones but who are quicker to respond to e-mails than developers in the same time zone may be perceived as especially trustworthy. Developers who resolve other developers' work items may be perceived as particularly benevolent and as a consequence, gain their colleagues' affective trust. Lastly, developers who use multiple communication media: are available by chat or are active in the project's mailing list rather than just e-mail alone, may be perceived by some to be particularly trustworthy.

Table 3-3. Collaborative traces and other data (rows) mapped to trust factors identified in the literature (column).																		
	Initiations and Response	Same Location	Role	Expertise	Reputation	Availability	Leadership	Years Experience	Communication Media	Homophily	Shared Information	Shared Photographs	Frequency of Meetings	Team Diversity	Project Updates	Team Size	Project Size	Monitoring
Project descriptions			X			X	X			X						X	X	X
Chat thread	X								X		X							X
Instant messages	X					X			X		X							X
E-mail messages	X					X			X		X	X			X			X
Mailing list postings	X					X			X		X				X			X
Calendar						X							X		X			
Keyboard input						X												
Time zone		X				X				X				X				
Personnel profiles		X	X	X		X	X	X		X		X		X				
Work items	X		X	X	X	X				X				X	X			
Source-code			X	X	X					X				X	X			
Org. chart		X	X		X	X	X			X				X				
Change sets				X						X					X			

3.6 Visual Representations for Trust

Visual representations summarize the information provided by collaborative traces. Initial research in this area has shown that visual representations of traces such as dependencies created from source-code and change-sets can influence developers' perceptions of their team members' trustworthiness (Trainer et al., 2011).

Choosing the appropriate visual representation for one's data is of great importance. Nowadays, guidelines for visualizing data are available to developers through web-based applications (e.g. ManyEyes, Swivel, Google Chart Tools), and typically include a baseline set of visual representations organized by tasks they support. Examples include node-edge diagrams and matrices for analyzing relationships between entities, bar charts for comparing values, circle packing to examine hierarchical relationships, and scatterplots to determine the relationship between two variables.

Prior to the emergence of web-based advice, researchers sought to understand the underlying perceptual reasons behind the suitability of different kinds of visual representations for data. As such, they lifted the level of discussion regarding the design of visual interfaces (Bertin, 1983; Mackinlay, 1986). While the aforementioned web-based tools have succeeded in making the high-level advice more accessible, they have not lifted the level of discussion. In contrast, our work seeks to lift the level of discussion by providing advice on the design of visual interfaces to support trust. Table 3-4 is a matrix of visual representations (rows) and trust factors (columns) that provides advice for looking at a trust factor and seeing whether it can be represented by a particular form. Table 3-5 is a matrix of visual representations (rows) and collaborative traces (columns) that provides advice for looking at a trace and seeing data-wise, whether it can be potentially represented in a particular visual form.

Table 3-4. Visual representations (rows) mapped to trust factors identified in the literature (column).																		
	Initiations and Response	Same Location	Role	Expertise	Reputation	Availability	Leadership	Years Experience	Communication Media	Homophily	Shared Information	Shared Photographs	Frequency of Meetings	Team Diversity	Project Updates	Team Size	Project Size	Monitoring
Node-edge				X			X				X			X		X		
Matrix	X				X	X					X							
Scatterplot	X					X												
Circle Packing		X	X	X		X				X								
Sunburst		X	X	X						X								
Treemap		X	X	X						X								
Indented text		X	X	X						X								
Line Chart	X					X			X		X		X					
Bar Chart	X				X			X										
Spreadsheet		X	X	X			X	X	X									
Map	X	X				X								X				

Table 3-5. Visual representations (rows) mapped to collaborative traces and other data (column).													
	Project descriptions	Chat thread	Instant messages	E-mail messages	Mailing list postings	Calendar	Keyboard input	Time zone	Personnel profiles	Work items	Source-code	Org. chart	Change sets
Node-edge		X	X	X	X					X	X		X
Matrix		X	X	X	X					X	X		X
Scatterplot		X	X	X	X	X	X	X		X	X	X	X
Circle Packing								X	X	X	X	X	X
Sunburst								X	X	X	X	X	X
Treemap								X	X	X	X	X	X
Indented text	X							X	X	X	X	X	X
Line Chart			X	X	X					X	X		X
Bar Chart		X	X	X	X		X		X	X	X		X
Spreadsheet	X					X	X	X	X	X	X	X	X
Map		X	X	X	X			X		X	X		X

For example, by flattening the hierarchy elements in CirclePacking (Wang et al., 2006) visual representations, developers can be grouped by their similarities, such as their areas of expertise in a particular field, or their location. Identifying this homophily can influence developers' sense of trust toward one another. Information about expertise can be obtained from collaborative traces such as source-code and change-sets, as well as other data such as org. charts and personnel profiles.

Additionally, circles can show all developers who have been assigned to or resolved the same work item, a collaborative trace representing the description of work to be done. Circles can pack developers by location as well. Location data can be derived from developers' work sites, personnel profiles, and time zones. In bar chart visualizations, source-code could be used as a collaborative trace to plot the number of modules authored per developer in a prject. The length of bars can be rapidly compared to make quick judgments about who contributes in the project. Alternatively, other information that influences trustworthiness such as the developer's years of experience, or the developer's daily response rate to e-mail or instant messages could be shown using length of bar as the encoding mechanism. The number of work items resolved per developer could also be easily shown with a bar chart.

Understanding which trust factors to present, how to present them in a visual interface, and where the data will come from is a necessary step in designing tools to support the development of trust. We propose the following model for the design space of visual interfaces to support trustworthiness between distributed team members:

Model of Design Space =

{ Trust factors, Visual representation, Collaborative traces }

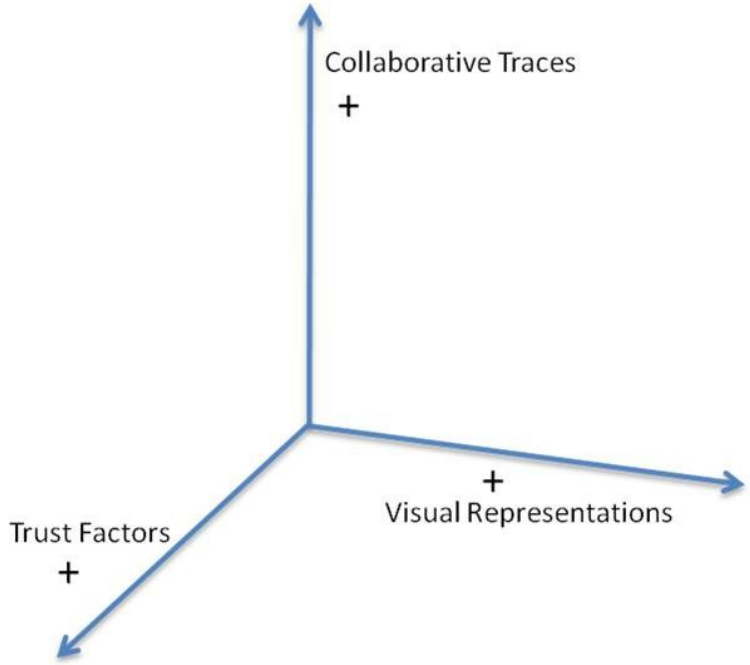


Figure 3-20. Axes of the "design space" of visualizations to support trust.

That is, to support the development of trust, tool designers should pick trust factors they would like to maximize (e.g., responsiveness, expertise, or location from the columns in Table 3-3). They should then choose visual representations for those factors (e.g., bar charts, spreadsheets, or maps from Table 3-4), and collaborative traces and other data (e.g., source-code, work items, or time zones from the rows in Table 3-3 and columns in Table 3-5) that provide information about the trust factors. The intersection of all three forms a triple in 3-D space (Figure 3-20). We are careful to acknowledge that data on each axis are not continuous; the purpose of the axis model is to illustrate the design space only.

Thus, each point in the space represents a particular combination of traces, trust factors, and visual representations. Because multiple collaborative traces can be used for a single trust factor (Table 3-3), points along the collaborative trace axis represent a

combination of traces, where the order of traces does not matter. Thus {e-mail, instant message} is the same as {instant message, e-mail}. An example triple in the space could be ({availability},{e-mail, instant message},{bar chart}). This point is distinct from ({availability},{e-mail},{bar chart}) but the same as ({availability},{instant message, e-mail},{bar chart}). Awarenex, presented in the review of tools in this section (Begole et al., 2002), fits into the three-dimensional space. It uses keyboard input and calendar information as its collaborative traces and an actogram as the visual representation. Other tools surveyed in previous work (Trainer and Redmiles, 2010) include Ariadne versions 1 (de Souza et al. 2005) and 2 (Trainer et al., 2008), Tesseract (Sarma et al., 2009), SoftCHANGE (German et al., 2004), ELVIN (Fitzpatrick et al, 2006) and SeeSoft (Eick et al., 1992). This set of existing tools fits into the design space of the model.

3.7 Summary

This chapter provided a scholarly review of trust, collaborative traces, and visualization. These three components make up the research questions of this dissertation, which can be stated another way as one central question: can visualizations of relevant collaborative traces influence ones perceived trustworthiness? The exploratory pilot experiment presented in this chapter suggests that the answer is *yes*. Chapter 4 details the design and implementation of a prototype that uses elements from the design space of tools to support trust.

Chapter 4. Building Theseus

The matrices of the design space in the previous chapter synthesized research literature on collaboration tools, trust, and visual representations. This chapter presents **Theseus**, a software prototype that traces one path through this space. The next few sub-sections of this chapter address which elements of the space are used, how information flows through Theseus, and the visualizations that comprise Theseus' interface.

As shown by the grouped X's in Table 3-3, there are a variety of collaborative traces that can show aspects of availability relative to the other aspects of trust identified. The number of X's in the columns marked other aspects of availability such as initiations and response, general activity, and same location imply an opportunity for research in this area. Designing to show information that can affect a trust factor (in this case, availability) translates into the following set of requirements for Theseus.

Requirement 1: Provide indications of an individual's location, workload, and role.

Requirement 2: Provide indications of an individual's responsiveness: their initiations and responses in project correspondence.

The collaborative traces that correspond to availability are email messages, instant messages, and mailing list postings. Other data include organizational charts, the number of projects in which a developer is involved, and work site locations and time zones.

Benevolence

The collaborative traces that correspond to benevolence are work items and mailing list postings.

Requirement 3: Show indications of who resolves whose work items as well as who replies to solutions in mailing list postings.

The collaborative traces relevant here are source-code and change-sets, as well as other data such as org. charts and personnel profiles.

Theseus departs from traditional expertise recommender such as Expertise Recommender (McDonald and Ackerman, 2000), Expertise Browser (Mockus and Herbsleb, 2002), Contact Map (Nardi et al., 2002), MII Expert Finder (Mattox et al., 1998), and XperNet (Maybury et al., 2000) by proposing that results returned from the tool be used to explore connections between individuals through artifacts they share in common rather than be used at face value. Recommender systems assume the user can effectively convert their problems into a series of search statements—what the information retrieval literature refers to as formulation (Kuhlthau, 1993). In contrast, Theseus gives the developer the ability to draw these same conclusions by exploring artifacts that are meaningful to them, artifacts with which they personally have worked in the past. While other systems derive search results from a broad graph of connections, Theseus uses the links which users themselves have created through e-mail correspondence, work items, and change sets.

While it is one aspect of the tool, the primary idea behind Theseus is not about recommending (e.g. Facebook or LinkedIn) trustworthy colleagues based on only shared ties in a socio-technical network. It is not difficult to see that as more people are involved in the referral process (i.e. Alice asks Bob who asks Chris who asks David for a recommendation), it is likely that the semantic meaning of the trust exchanged between each pair of people is likely to become inconsistent as people bring their own interpretations to bear (much like the children's game of "Telephone"). Indeed a participant in our study mentioned that while he asks colleagues for recommendation, he would not rely on referrals made by colleagues more than one separation away because "I can't know their trust baseline for others." For this reason, Theseus does not try to calculate transitive trust paths to infer trust relationships and make recommendations of trustworthy colleagues. Thus a requirement for Theseus is as follows:

Requirement 4: Given a new contact, calculate all paths to that contact starting from the user, and emphasize those who are along the shortest path to the new contact.

Relevant collaborative traces for relationships include: e-mails, source-code authorship, and work items.

4.1 Information flow

Figure 4-11 illustrates that information flows through Theseus in four separate phases. Designing Theseus as a web application with server and client side eliminates the need to install software on the developer's computer, making experimentation and deployment of successive iterations of the interface easier.

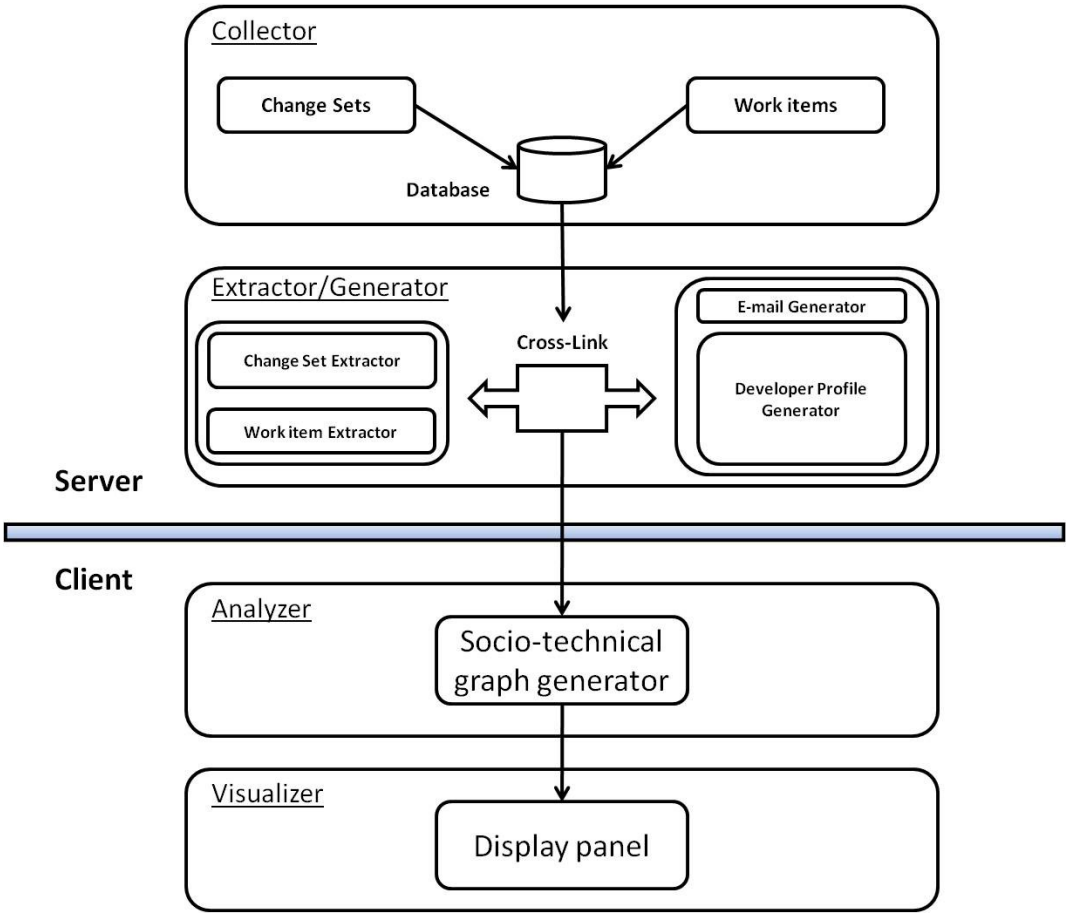


Figure 4-1. Theseus' architecture.

4.1.1 Collector

Most projects, open-source and commercial, use a set of internal tools to track development: SCM systems, work item and issue trackers, and project mailing lists. Theseus assumes the existence of these repositories prior to the extraction of their data. The collector mines artifacts from these repositories and stores them in a database from which they can be saved across successive sessions.

4.1.2 Extractor/Generator

Theseus incorporates an extraction step to ensure that it can inter-operate with collaborative traces from multiple types of repositories (e.g. CVS vs. Subversion vs. Jazz SCM, and BugZilla vs. Jazz work item tracker). A series of adapters for each type of repository is used.

The Extractor/Generator module cross-links developers found in the repositories with *generated* profiles and e-mail messages. Section 4.2 discusses this idea of generating data in more detail. The output of the extractor/generator is a small set of XML files. These files enumerate all the traces, extracted, and generated which are needed to describe relationships between developers, e-mail messages, mailing list messages, work items, roles, and developers' profiles.

4.1.3 Analyzer

The analysis component of Theseus calculates relationships between developers based on the information contained in the XML files, such as who resolves whose work items and who communicates with whom. The output from the analyzer is a socio-technical graph describing communication and associated contacts and artifacts.

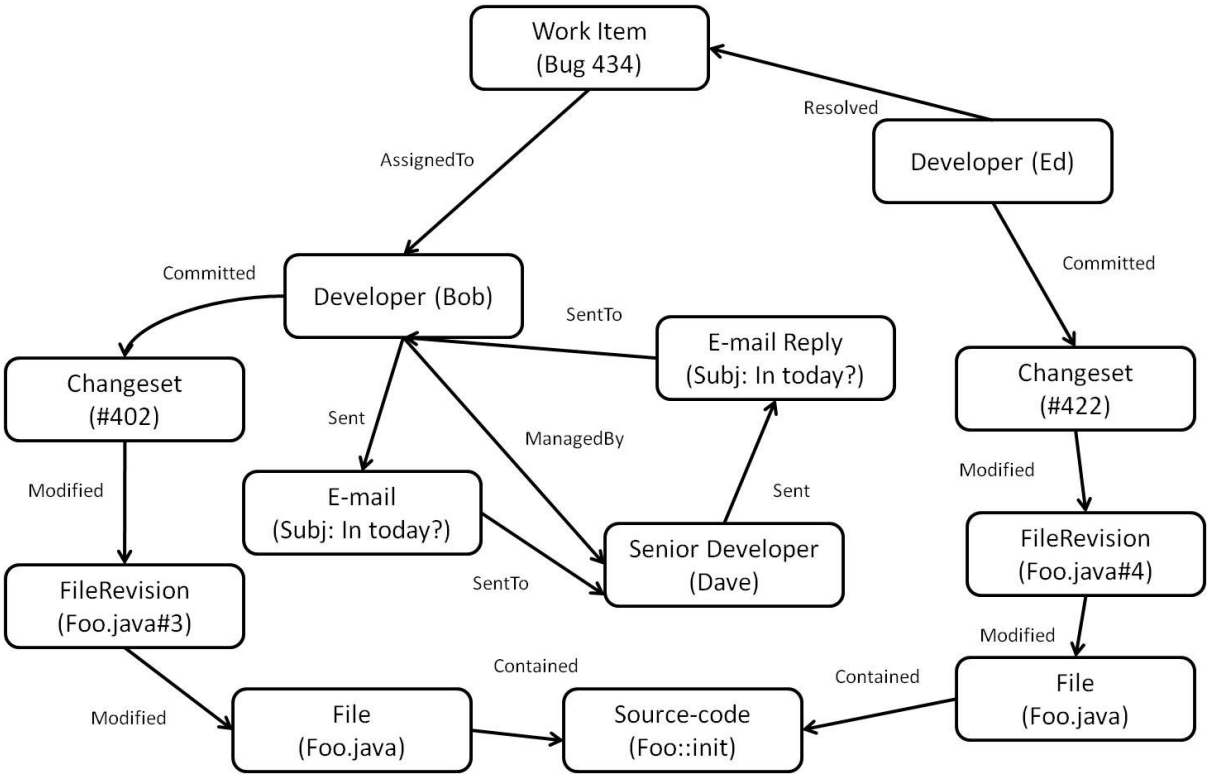


Figure 4-2. Structure of the socio-technical graph used as input to Theseus Visualizer.

As an example, in the graph above, we see that although Bob was assigned a bug, Ed fixed it for him. Also of importance is that Ed committed a change set that modified source-code modified by a changeset committed by Bob. Theseus considers these developers connected as a result.

We also see that Bob has sent Dave, his manager, an e-mail and that Dave has responded to it. Theseus considers the two connected by their relationship in the hierarchy and this e-mail.

4.1.4 Visualizer

The last step in the Theseus information flow is visualizing the developer's contact lists, availability of colleagues, and their benevolence using collaborative traces from the socio-technical graph created in the previous step. In the following section, I describe Theseus's user-interface and visualizations.

The main interface consists of four sections. These sections are: a panel of reference contacts, from which the user can add and delete individuals (a), a panel which displays

search results and visualizations of contact availability and social network connections (b), the availability radar, which tracks the cumulative availabilities of searched contacts (c), and a display of the collaborative traces which involve the searched contacts in question (d).

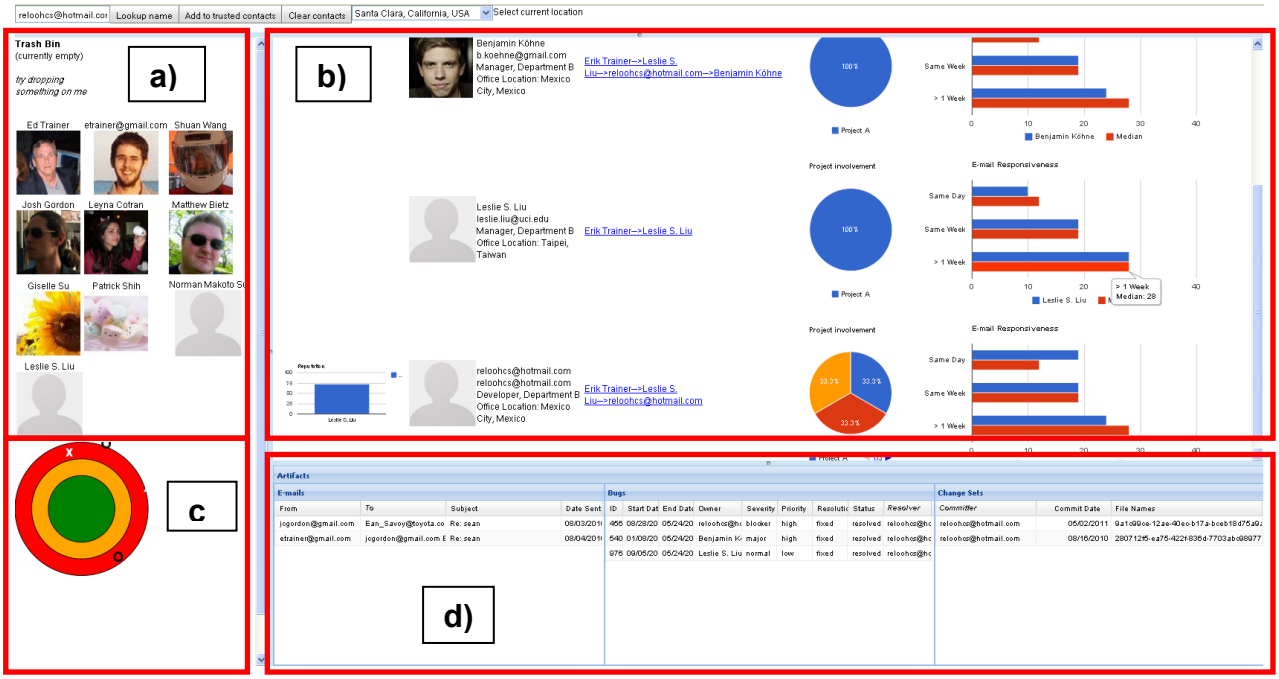


Figure 4-3. The Theseus user interface.

Upon loading, Theseus gives the developer the option to input their current team's location from a drop down list of generated locations. They then authorize the tool to collect contact information and project artifacts by clicking the "Authorize Theseus" button.

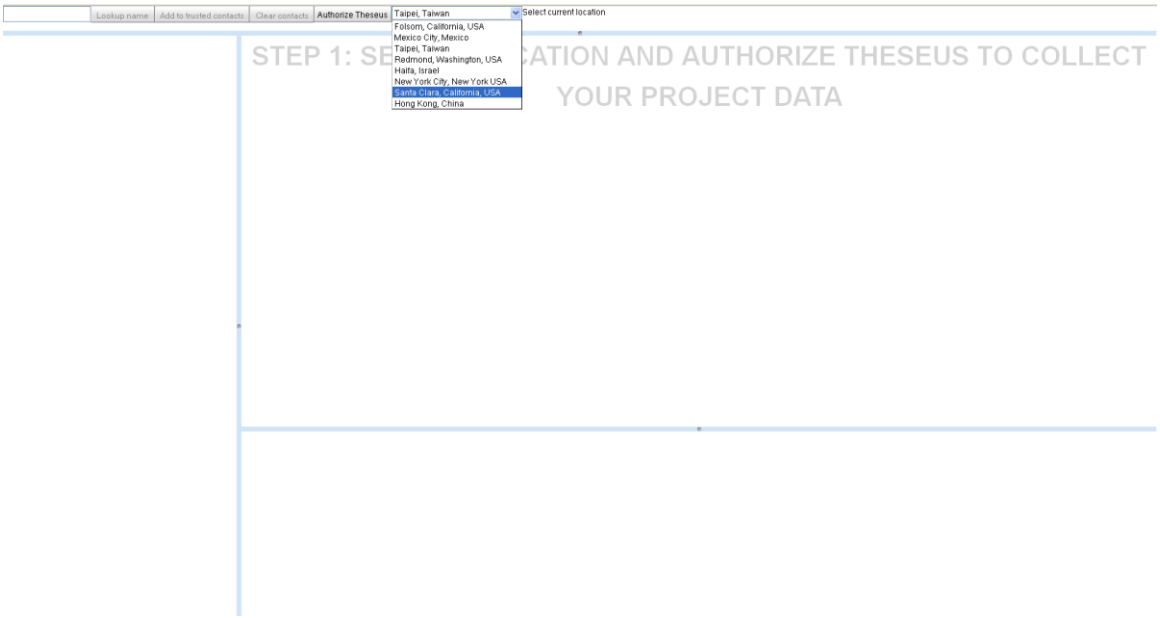


Figure 4-4. The user begins by selecting their physical location using a drop down menu (top).

Once authorized, Theseus connects to the developer's e-mail account (Gmail is supported in the current implementation), stores e-mail messages in a locally created database, and populates the left pane with the most frequently e-mailed colleagues over the past year. Colleagues can be removed from the pane or added at any time by dragging a contact's image to the "trash bin" (upper left corner).

The panel on the bottom of the screen displays three grids of project artifacts: e-mail, work items, and change sets. As developers search for new colleagues, the displays automatically filter to show traces in which the latter are included in the *to* field (email), the *resolver* field (work items) and *committer* field (change sets). Initially, it is populated with *all* e-mails, work items, and change sets.

Below the list of contacts (left) is the **availability radar**, which is activated and updates when new contacts are searched.

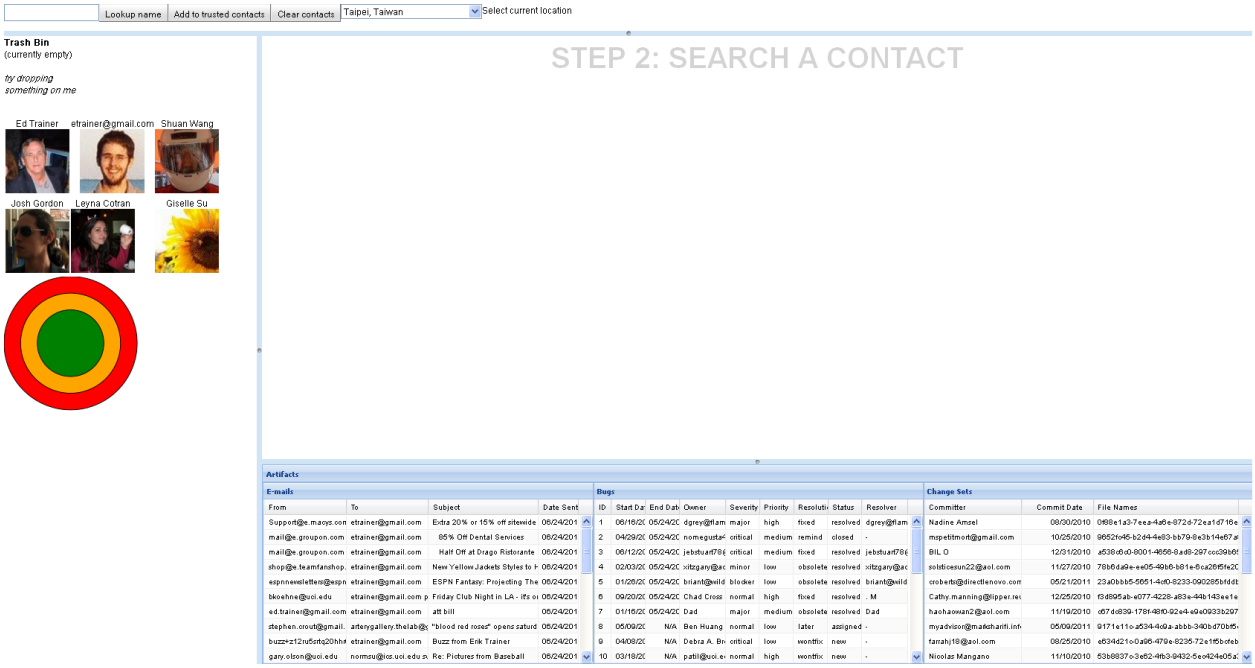


Figure 4-5. The user's trusted contacts loads (left) along with all collaborative traces from the project repository (bottom).

The next step is to type in the name of a new collaborator in the search text box, located in the upper left corner. If the name is found in Theseus's socio-technical graph, Theseus auto-suggests the name in the text box. After the user clicks "Lookup name", the **availability radar** is populated (bottom left) as well as other visualizations that serve as indicators of *referral trust*, *accessibility*, and *responsiveness*.

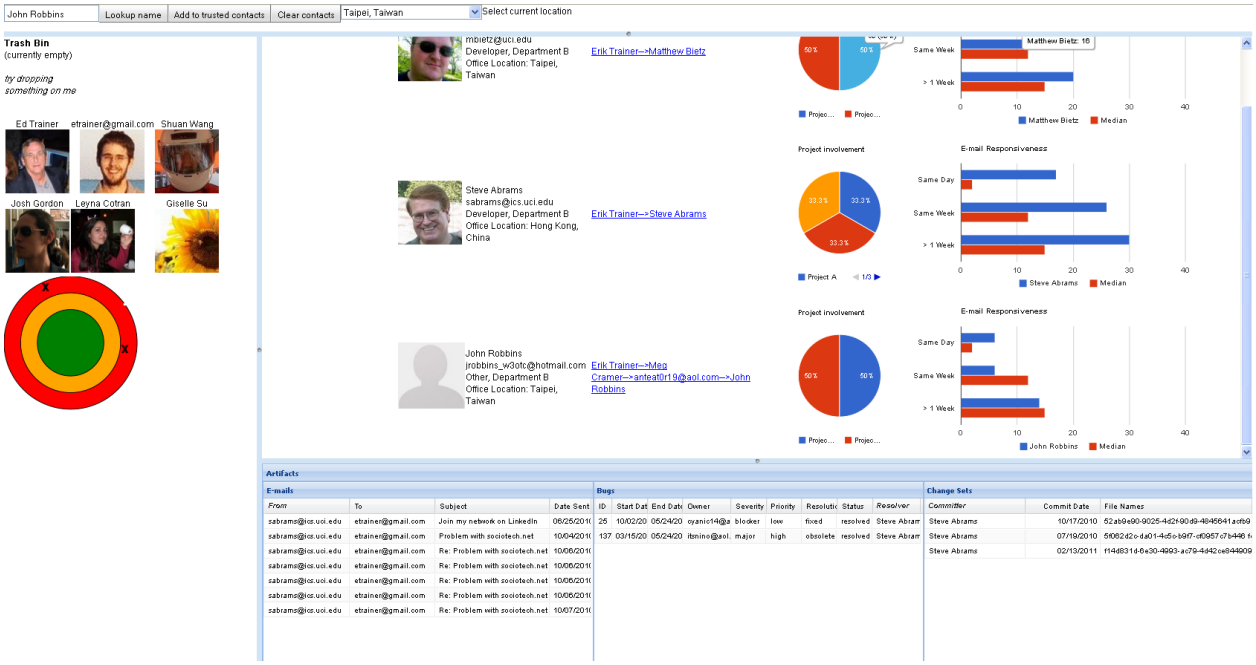


Figure 4-6. Visualizations populate as the user searches for contacts, and collaborative traces in which the user has participated appear below.

In the top pane, a textual view of the shortest path between the user and the contact in question, and at bottom, a sortable and filterable view of project artifacts provided by the socio-technical graph. The textual description is presented as a hyperlink that when clicked, filters the bottom view to show artifacts which involve the contact of interest.

Erik Trainer–>Steve Abrams	Erik Trainer–>Meg Cramer–>anteat0r19@aol.com–>John Robbins
A direct connection	**An indirect connection (3-degrees away)**

Figure 4-7. Textual descriptions of connections to the contact of interest. Direct and indirect connections are shown in this example.

The bottom displays shows artifacts that reference the contact in question. The current implementation displays e-mail, work items, and change sets. The screen capture below shows the bottom display panel when the contact "Steve Abrams" is clicked: a filter on the *from* field is automatically applied to show all e-mails from them, the *resolver* filter on bugs is applied, and the *committer* field is applied on change sets.

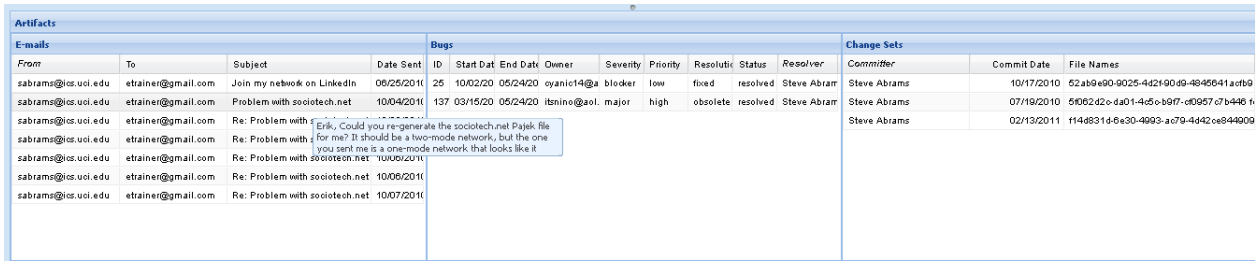


Figure 4-8. A filter is applied to show collaborative traces involving the direct contact, in this case "Steve Abrams."

When the hyperlink on the right (i.e. for contact "John Robbins") is clicked, another filter to the artifact data is applied. No results come up in the e-mail artifacts, but "anteat0r19" shows up as an owner of a work item that "John Robbins" resolved:

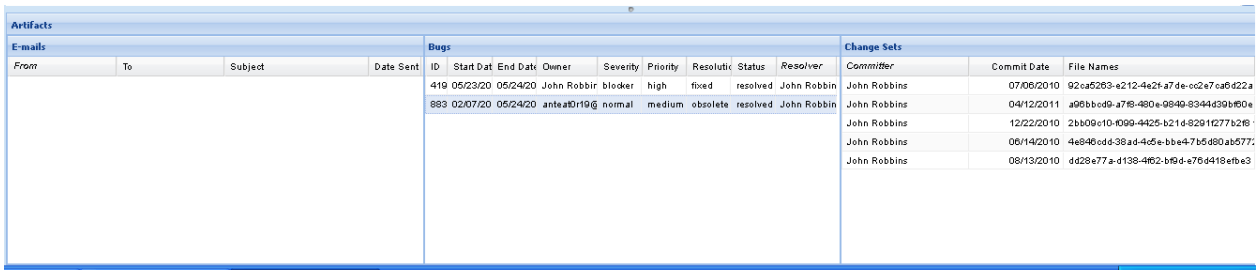


Figure 4-9. A filter is applied to show collaborative traces involving the indirect contact, "John Robbins."

Backing up and searching "anteat0r19" as a contact results in the following filters:

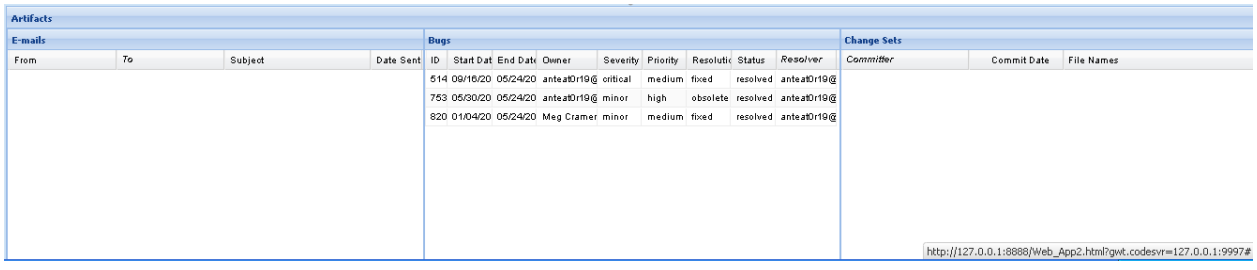


Figure 4-10. "anteat0r19" is connected to Meg Cramer because the former resolved the latter's work item.

When the filters are applied again, "anteat0r19" shows up as a resolver of "Meg Cramer's" work item.

The developer concludes that “John Robbins” is connected to contact “Meg Cramer” because the former resolved a work item owned by another individual who resolved “Meg Cramer’s” work item. As such, no context exists for the user to initiate a conversation with “John Robbins,” even through “Meg Cramer” as an intermediary. No referral trust should be inferred.

Instead, the developer can determine who else in their trusted contacts may know “John Robbins” by determining who else is involved in the projects he is working on. This is accomplished by clicking on each piece in the contact’s pie chart visualization and viewing contacts that appear highlighted in red:

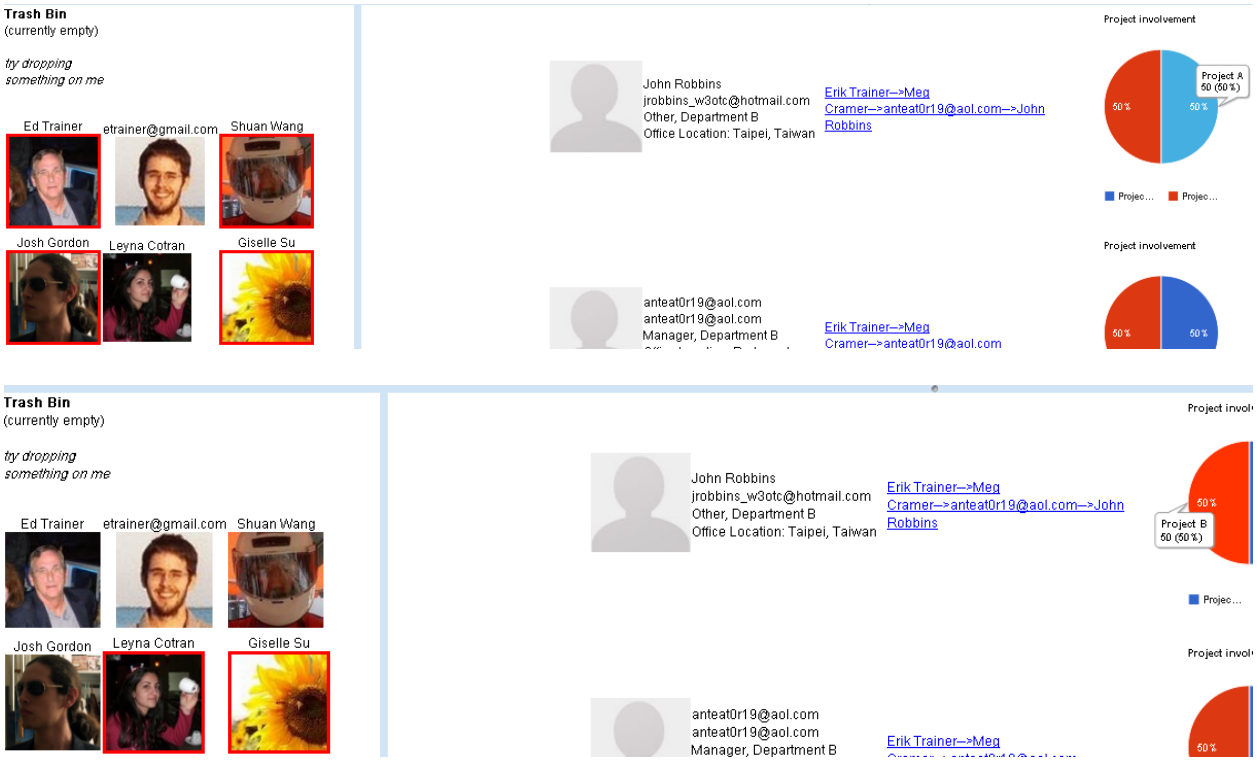


Figure 4-11. Red boxes around a contact indicate they are involved in the selected project.

A comparison shows that contact “Giselle Su” is involved in both projects and may be able to provide additional information about “John Robbins.”

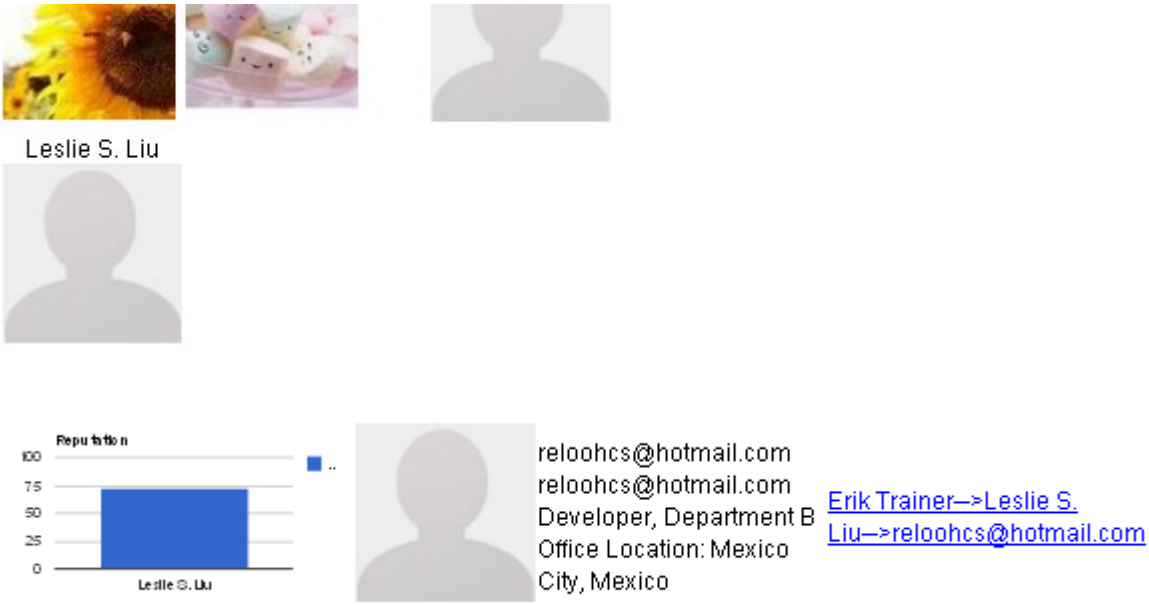


Figure 4-12. The reputation bar indicates how trusted contacts have rated the contact in question. In this example, trusted contact "Leslie Liu" has rated "reloohcs@hotmail.com" 75 out of 100.

The reputation bar appears next to a search contact if any links to that contact are in the user's trusted set of contact. The user can see how their trusted contacts rated that contact's trustworthiness, on a scale from 0-100. The inclusion of this visualization was inspired by online communities such as ***Slashdot*** and ***GIThub*** that use reputation as a social currency for rating community members' value.

Theseus provides three views of *accessibility*. The first is a text only description next to the contact, which summarizes their role and location in the organization.

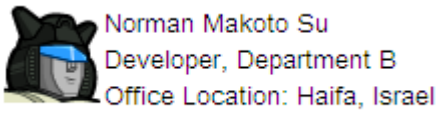


Figure 4-13. An image of the contact (left) and a textual description of their role and location (right).

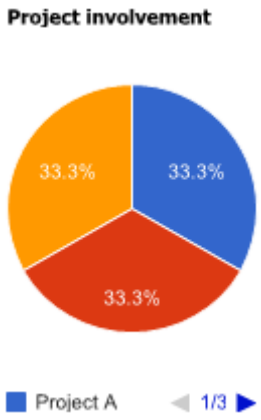


Figure 4-14. A pie-chart visualization indicating a contact's involvement in different projects.

The second is a graphic indicating the number of projects in which the individual is involved with each slice of the "pie" calculated by their source-code authorship percentage calculated by:

Percentage = Number of collaborative traces involving contact/ Number of total collaborative traces

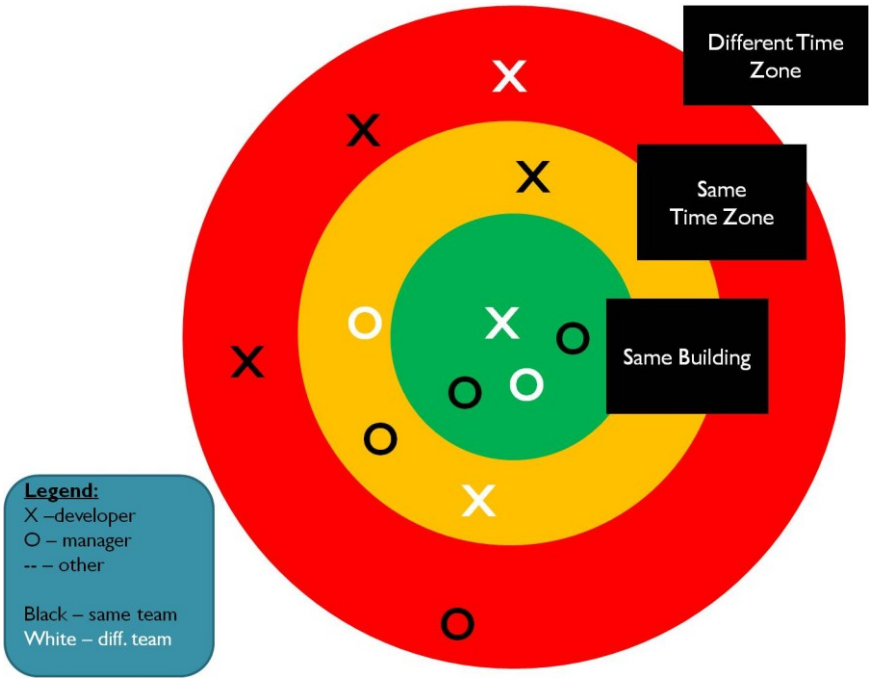


Figure 4-15. The availability radar visualization classifies individuals' location into three levels and indicates their role in the team.

The third is a visualization of that information which can be used to compare multiple individuals, the availability "Radar." This visualization uses a level-based indication of availability because the influence of location is very significant for distributed teams (Olson and Olson, 2000). For example, consider person A and B in Irvine, person C in New York, and person D in Israel. For A, the availability <A, B> is much greater than <A, C> and <A, D>. Literature has suggested that there may be no big difference between <A, C> and <A, D> although the physical distances vary greatly (Allen, 1984; Bird et al., 2009). As such, in the radar visualization, physical distances do not define the distance categories. Instead, categories, as represented by the different colors of the three concentric circles, are based on the results of the aforementioned studies.

The first visualization we created using the design space is the "Availability Radar" visualization (Figure 3). The "Availability Radar" groups developers by their proximity to the user of the visualization. However, the groupings are not defined by physical location alone. They take into account time zone as well, which is just as important. As such, the groupings are defined as follows:

Green: Same building

Yellow: Same time zone

Red: Different time zone

The further someone is physically from the user's current location, the farther out they are placed along the X and Y axis from the center of the visualization. A black "X" in the visualization represents a manager whereas a white "X" in the visualization indicates a non-manager. We use color to encode role in the organization because of evidence from our data on trust that indicates that people perceive managers to be less available in hierarchical organizations.

"Availability Radar" was inspired by the design of the CirclePacking visual representation and modified to show "levels" of availability without a strict hierarchy. The circles are used to group "children" nodes, i.e. developers together that have the same properties, i.e. same location. The collaborative traces that provide the data needed for the visualization are organizational charts and time zones.

As contacts are added, the availability radar animates to show the contact added to the radar visualization. Hovering over individual data points allows the developer to see the individual's location and how many hours away they are:

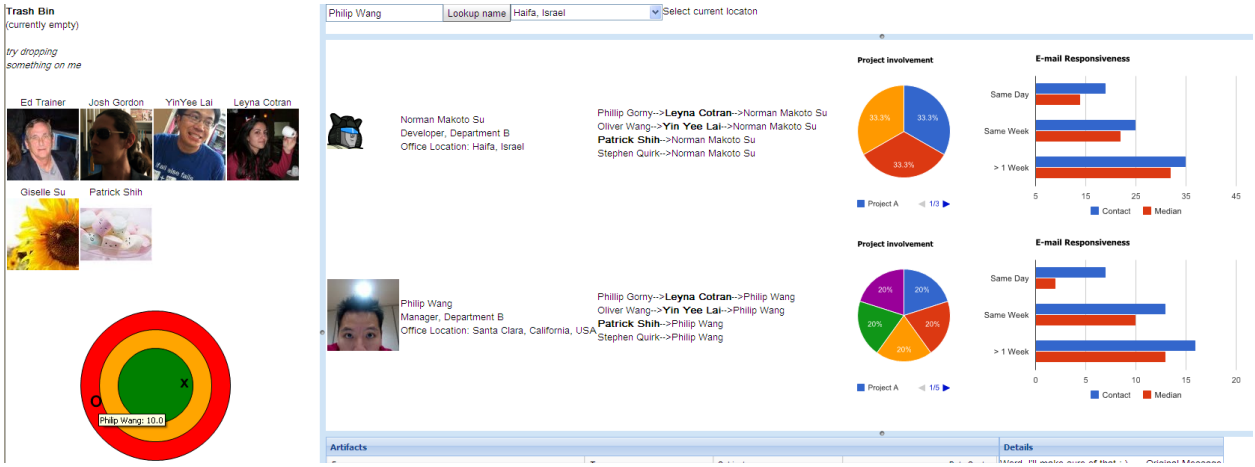


Figure 4-16. The user "Philip Wang" is located in a different time zone (10 hr. difference) than the current user of Theseus (left). In contrast, "Norman Makoto Su" is located in the same building. Therefore the latter may be considered more accessible.

E-mail *responsiveness* is binned into three separate categories in order to provide an at-a-glance view of how often individuals reply to e-mails sent to them. The interface also shows the median response time rather than the mean time in order to control for highly skewed responders (e.g., auto-responders or dead e-mail addresses). The bar chart indicates the number of e-mails replied to and the median values:

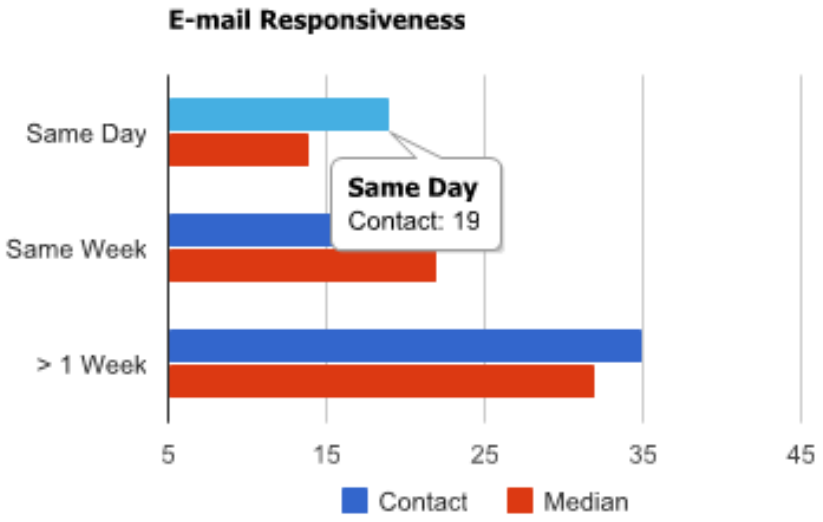


Figure 4-17. An early prototype of a responsiveness visualization, showing the contact in question's average number of e-mails replied to within the same day, week, or more than a week.

Studies of real-world email datasets, such as the ENRON corpus, have concluded that the average corporate reply time is about 1 day. One quantitative study of the ENRON data (Kalman and Rafaeli, 2005) found that the average response time of over 16,000 e-mails was 28.8 hours. The researchers found that a typical user responded to 84% of messages within a day and 95% of messages within 5 days. Another quantitative study (Karagiannis and Vojnovic, 2009) analyzed 315 million e-mails over a 3 month period and found that the median reply time to a message was approximately 1 day. Qualitative studies (e.g., Tyler and Tang, 2003) have found that individuals *expect* others to reply to them within 1 day. When questioned about their reply times, participants maintained informal "Responsiveness Images," which projected to others that they were likely to reply within the work day or 24 hours of receiving a message.

In response, a more refined version of the previous visualization may use different categories for responsiveness:

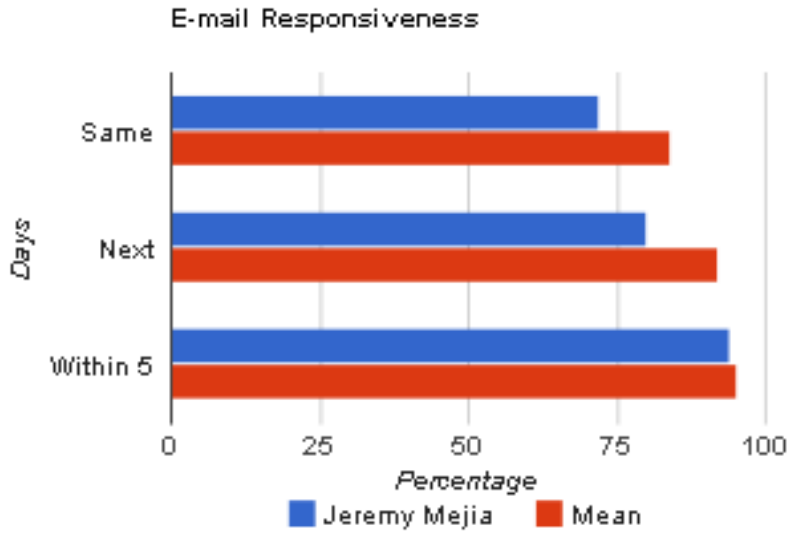


Figure 4-18. A refined version of the previous visualization, with categories drawn from the research literature on corporate e-mail responsiveness.

The second visualization (Figure 4-18) shows developer's responsiveness, the second aspect of availability. The insights about levels of responsiveness described in Section 5.2 were the basis from which the categories of "same," "next" and "within 5" days were designed. The *bar chart* visualization was chosen because it allows for quick comparisons between numerical data the user in question's response rate and the average reply rate using *length* as an encoding mechanism. We see in the example below that the contact's response rate is slightly below average. The collaborative trace that provides this visualization with data is *e-mail*. One could imagine the same visualization using *instant messages* or *mailing list postings*.

The third and final visualization example (Figure 4-19) is an alternative to the previous visualization. It is also designed to show responsiveness, but shows individuals' average reply time to e-mail relative to the mean reply time of all individuals. For example, suppose person A replied to 10 emails in following time interval (1, 2, 1, 4, 2, 9, 23, 11, 8, 3) <in hours>. Their average reply time is 6.4 hours. After calculating every contact's average reply time, standardize them according to their distribution. Then calculate standard scores

for each individual's average reply time and plot them along a horizontal axis to show how far, to the left or right, the scores are from the mean average reply time, using the formula:

$$z = \frac{\chi - \mu}{\sigma}$$

where:

χ = person A's average reply time

μ = mean of population's average reply times

σ = standard deviation of population's average reply times

The inspiration for this visualization is the scatterplot. It looks almost exactly like the scatterplot, except that it has been modified to hide the y-axis. Stacking the horizontal axes on top of one another for each contact searched affords quick comparisons using *position* as an encoding mechanism to determine each contact's reply time relative to everyone else.

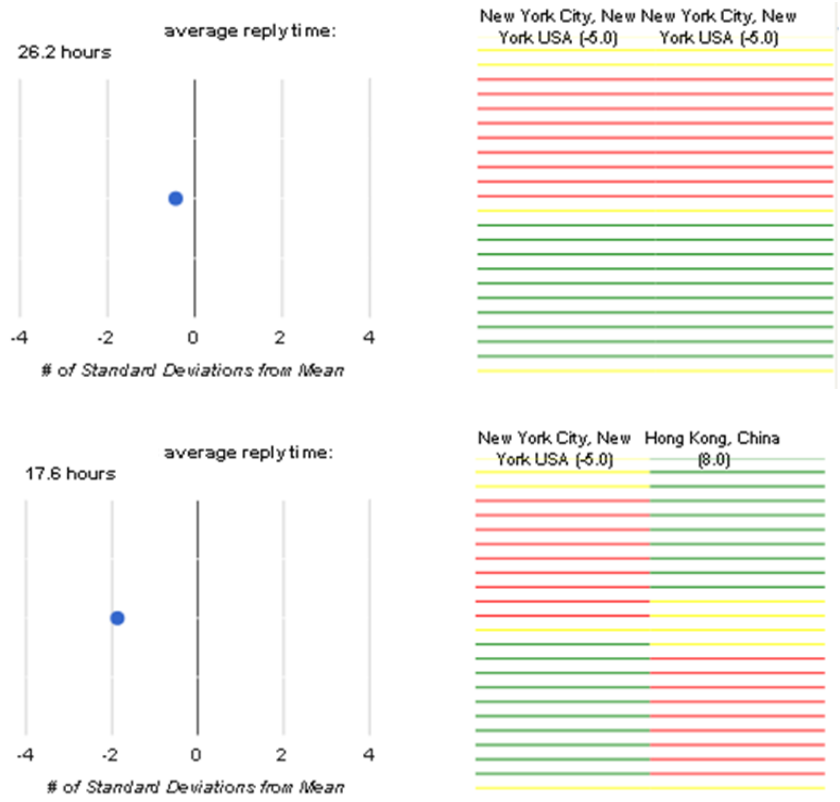


Figure 4-19. The bottom contact responds to e-mail more quickly than the top, despite being 13 hours ahead of him.

In distributed teams, an individual's reply time to a message is likely to be influenced by time available in the day for work. What is 1 pm over the course of a normal work day for one developer may be 3 am for another developer. As such, one should not expect a response within the same work day. Using this insight, we augment the reply time visualization with a visualization of the time zone overlap between the user and the contact in question. The left half of the visualization consists of rows corresponding to each hour in the day for the user, while the right half shows all hours in the day for the contact in question. A green line indicates normal working hours (8 am – 5 pm). A yellow line indicates hours during the day where the individual is likely to be active, but not necessarily at work (7 am and 6 pm – 9 pm). A red line indicates normal sleeping hours (10 pm – 6 am). Overlaps (and the lack of overlap) in working and non-working hours are immediately obvious given this color coding.

This visualization was inspired by row/column Spreadsheet visualizations, in which color is typically used to encode attributes of the cells. Although a Map visualization could also show time zone information (Table 3-5), it was not chosen for two reasons. First, there is not enough screen real-estate to show the whole map. Second, Map-based visualizations typically indicate differences in time zone by shading dark for night time and light for day time. The beginning of one person's work day may be the end of another's. By breaking down the time zone overlap by hour, users of the visualization can intuit good times to meet and follow-up on correspondence.

Each "cell" is color-coded by the time of the day in the particular region. When a single row is viewed, it is obvious whether the two times belong to the same time group. In this design, the cells are "squashed" to the point where each row is simply a line because what is necessary to show is the overlap, not the reading of a particular numerical value, i.e. the exact time. The collaborative traces required are e-mail and time zone. Like the previous example, one could imagine instant messages and mailing list postings in addition to e-mail.

One idea to indicate benevolence of a globally distributed software developer is to show whose work items they resolve. If they resolve other people's work items, other developers may perceive them to be more trustworthy. In the visualization depicted by Figure 4-20, trusted contacts are represented as circles, with the areas of those circles corresponding to the number of bugs they have fixed that were not originally assigned to them. I anticipate some example patterns to take the following form:

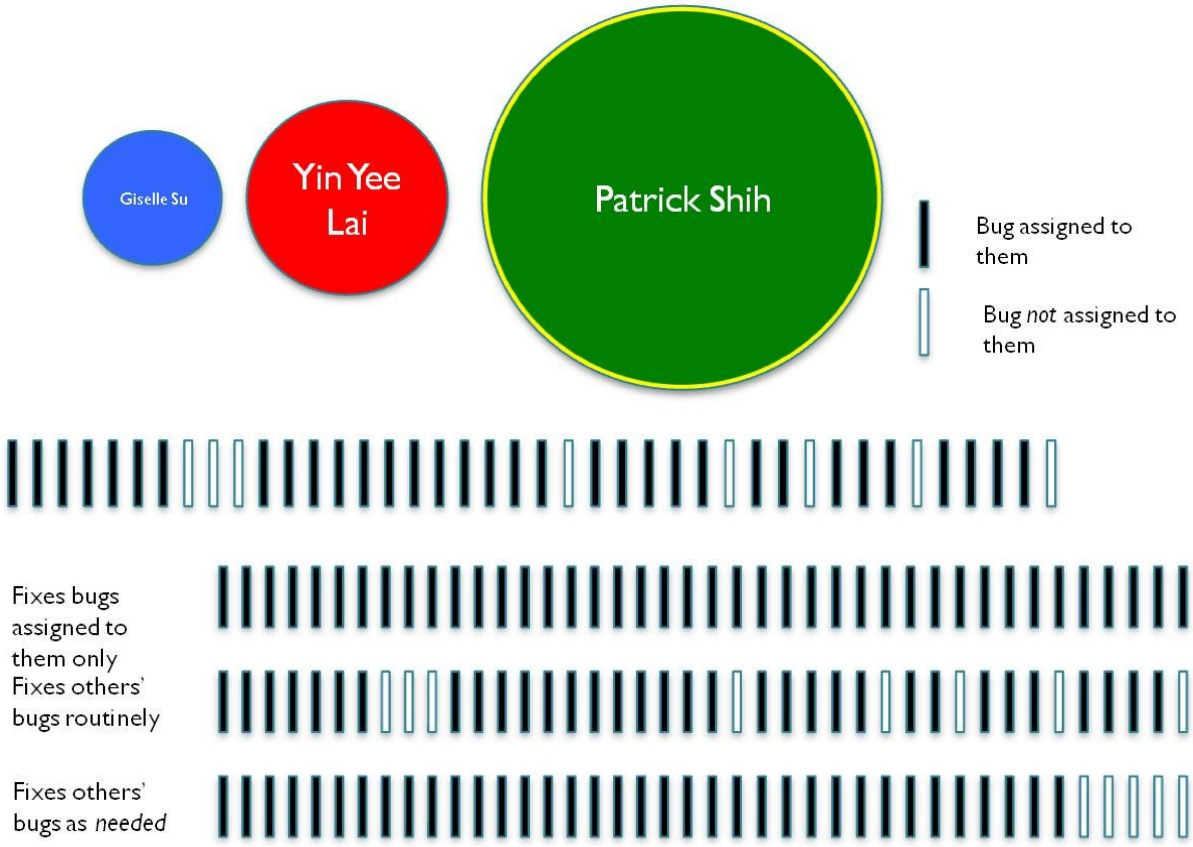


Figure 4-20. A visualization showing who tends to resolve others' work item assignments, which is one way to operationalize benevolence.

The absence of filled lines are used to pre-attentively process and identify bugs which were not assigned to, but fixed by the individual in question.

4.2 Data Generation

In the last sub-section, the visualizations that comprise Theseus' interface were presented. The usefulness of a particular visualization rests on the data it visualizes; in some of the above visualizations such as e-mail responsiveness, the best data is not easily available, and this becomes problematic for Theseus' design. This sub-section introduces the idea of generating data not typically available to drive visualizations and address the question of whether it necessarily needs to be realistic or not.

Good data is critical to visualization design. Edward Tufte, an expert in the visual presentation of information states, "Above all else show the data" (Tufte, 2006). For data to be good, it must be available in sufficient quantity and quality. An example of sufficient quantity would be the following: if I want to visualize all the change requests for a project in the past year, I would need every change request from the past year. If the repository was disabled for 3 months out of the year, I would not have sufficient quantity because of the missing data points. Further, if the names of everyone who submitted the change request was anonymized or sanitized in any way, as is sometimes the case (see the Jazz project for an example), I would not have sufficient quality, where "completeness" is an interpretation of quality. Data that is not in sufficient quantity or quality is by definition, not good.

A related point to quantity and quality is the availability of the data. Is the data freely available or is it locked? Typically, open-source project data is available to anyone who wants it by nature of the provisions in open-source license agreements. As such, researchers interested in open-source project data typically write scripts that extract collaborative traces from repositories belonging to open-source communities, such as SourceForge, GitHub, and others. On the other hand, corporate data is much more difficult to gain access to. Access to data in corporate settings typically requires long-term corporate-university partnerships and financial investments that fall beyond the scope of the individual researcher's capabilities.

Moreover, non-disclosure agreements that may not guarantee access to all the data the researcher needs. For example, researchers may be able to gain access to the source-code repository but not the bug-tracking repository. Or, researchers may get information about the number of teams developing a product, but not where those team members are located.

This limited access is at odds with the need in research to prototype and experiment under the different conditions and parameters afforded by rich and varied data. How much more could we know if all that data was made available and easy to access? To address these issues, software engineers in different domains have used data generation, which is effectively an input stimulus to a device under test.

One wide use of generators is in the area of large scale database testing and Quality Assurance (QA). Currently, database developers and administrators often have to spend hours of dull work to create test data sets before examining database or application performance. A database developer should populate the test database with millions of realistic data records and run the application again and again. It is the only way to ensure that your users will get an application with the necessary performance and response time. Data generators support this process by generating database records, tables, procedures and views for common procedures in the testing phase: performance testing, QA testing, loading tests and usability testing. For example, a database administrator may want to test that the database schema can handle records of different types. A data generator will give them the ability to quickly fill in tables with the types they desire to test: the integer data generator generates random integer values, the string generator generates random strings, and the regular expression generator generates strings that match a pattern that you specify.

Other uses for data generators range from functional verification of microprocessors to statistical analysis. In the field of microprocessors, Bhaskar et al. (2005) propose a template that captures the commonalities among the different random testing tools and

enables the user to quickly design a random test generator by adding product-specific details and using most of the methods available in the template. Using the generator in this way leads to high degree of code reuse, less debugging of the random tool and huge reduction in design-cycle time. In statistics, the Apache Commons collection, an Application Programming Interface (API) for programmers, uses data generators to generate random sequences of numbers that follow particular probability distributions. These distributions are especially important when conducting statistical simulations of systems such as fluids, disordered materials, strongly coupled solids, and cellular structures using Monte-Carlo methods. For example, when using the built-in JDK function Math.random(), sequences of values generated follow the Uniform Distribution, which means that the values are evenly spread over the interval between 0 and 1, with no sub-interval having a greater probability of containing generated values than any other interval of the same length.

What these applications of data generators all have in common is that *they seed systems* with data *in the early design stages.* When prototyping tools that will eventually require real-world data, access to data may be uncertain, or it may be dangerous to use real world information if the system is not stable, as in the case of database design. Alternatively, when testing statistical models in theory, the only feasible avenue may be experimentation with Monte-Carlo simulation.

While there is a substantial body of research on mining collaborative traces such as work items, mailing list postings and source-code versions, access to developer profiles, organizational charts, e-mail, work site locations have proved to be not as accessible due to concerns of privacy and confidentiality. Moreover, contact information and developer profiles are rarely kept up to date (Ehrlich, 2003, Ackerman et al., 2003) because people do not usually perceive immediate rewards for keeping this information current. Most data made available for research use is from open-source projects, whose roles, hierarchies, and

location may not correspond to organizational structures and attributes by virtue of the fact that most open-source developers are not circumscribed by the structures and sense of place that "organization" implies. Information about correspondence, such as the number of messages sent between two developers, or type of communication media different developers use is typically not made public.

In this work, missing data points for input to Theseus's visualizations are generated for the purposes of experimentation, however real-world information is used as well. Best practices for most open-source and commercial projects imply tools to manage development, namely source-code versioning systems (SCM), developer mailing lists, and issue tracker and work item databases. In previous work I have developed source-code extractors for different SCMs and issue trackers.

Work items and information about change sets were mined and extracted from 2 projects: MIRTH, an open-source healthcare management system, and the Mozilla Bugzilla project. This information was cross-linked with generated developer names and contacts. Theseus generates the following information:

Developer attributes:

Name

E-mail address

Org. role (e.g., developer, manager)

Location

Photograph

Number of projects

Average response time (in hours) to e-mail.

4.3 Summary

This chapter presented the design of the Theseus tool, which consists of multiple visualizations. Data for the visualizations come from real-world issue trackers, change management repositories, and mailing lists. In addition, some data is generated: e-mail responsiveness, developer locations, and roles. However this data is based on real-world data sets and observations about responsiveness in software development teams. In that respect, it is realistic.

The next chapter presents two different phases of evaluation of the Theseus prototype. The first phase assessed Theseus' usability through the application of usability inspection techniques, while the second phase assessed Theseus' utility and usability in a human subjects experiment. The method and results for each phase are discussed in turn.

Chapter 5. Evaluating Theseus

Accurately evaluating visualization tools is a difficult challenge and subject of ongoing study in the visualization community (e.g., the BELIEV workshop at the Conference of Advanced Visual Interfaces, AVI). One reason is that only measuring time and errors, by far the most common quantitative approach, can be misleading. For instance, a longer time may not necessarily be bad; the user could simply be more immersed in the information presented to them, and possibly learn more than they would have otherwise. Another issue is that few tools address the same problem, making it difficult to perform comparative evaluations. Indeed, Theseus is the only tool to my knowledge with the objective of supporting trust in globally distributed teams using visualizations of collaborative activity.

One of the big challenges relevant to this dissertation is that it is difficult to articulate quantifiable tasks for information visualizations tools because they support general browsing and data exploration rather than solve specific problems. As such, traditional performance evaluations are difficult to perform. When it is difficult to articulate the task, it is difficult to specify criteria for evaluating the system and what should be measured. New evaluation techniques beyond measuring correct task completion, time to task completion and number of errors are slowly emerging. One such example is the identification of low-level analytic tasks (Stasko, 2006) and the development of insight-based evaluations (Saraiya et al., 2005). Both techniques are used in the evaluation of the Theseus prototype described here.

The goal of Theseus's evaluation is to show that the tool can "usefully engender accurate perceived trustworthiness among distributed team members." According to Nielsen, the usefulness of a system is composed of its usability and utility (Nielsen, 1993). Following this definition, this chapter assesses both the usability and utility of Theseus. Usability is a quality attribute that assesses how easy user interfaces are to use. Utility refers to the design's functionality: does it do what users need? (Nielsen, 1993). Theseus's usability is assessed in

Section 5.1 of this chapter and its utility (and to a smaller degree, its usability again) is assessed in Section 5.3.

5.1 Phase 1: Usability Inspection

Goal: *Assess the usability of Theseus with usability inspection methods.*

Designing visualizations is a creative process that has been described by experts as a "craft" (Shneiderman, 2003) and requires incremental refinement and improvement. Lightweight inspection methods can help me refine the initial design and work out low-level tasks users will perform with the tool and whether the visualizations help with these tasks. First, the author checked Theseus's interface against well-established usability principles with Nielsen's Heuristic Evaluation (Nielsen, 1994). Second, the interface was evaluated with the Cognitive Walkthrough (Wharton et al., 1994), a method that is particularly good at focusing on the user's role, a priori assumptions, and what they can accomplish with and without training. Third, the Cognitive Dimensions of Notations Framework (Green, 1989) (referred to hereafter as "Cognitive Dimensions") was applied to uncover operations with the interface that are mentally demanding. While the Cognitive Walkthrough is task specific, the Heuristic Evaluation takes a holistic view of usability problems and may catch issues the Cognitive Walkthrough and other techniques, like the Cognitive Dimensions, miss. The application of all three methods increases the chances of the tool being more usable for human subjects testing. The following sub-sections present each evaluation of the Theseus interface in turn.

5.1.1 Heuristic Evaluation

The Heuristic Evaluation (Nielsen, 1993) is a usability inspection method that assesses an interface's usability against a prescribed "checklist" of aspects of usability. In this section I analyze whether Theseus's interface fails to meet, moderately meets, or meets well the heuristics.

5.1.1.1 Visibility of system status

This heuristic indicates, “The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.” (Nielsen, 1994). On start-up, Theseus’ visual interface prompts the user to select their location and “authorize” the tool to collect project information from Google, indicating that the tool is waiting for user action. After the user selects “Authorize”, the visual interface displays a common loading gif animation familiar to users of web technologies (Figure 5-1), indicating the tool is doing processing work. In addition to the .gif, background text on the screen indicates what steps the tool is taking, namely: connecting to the e-mail client, connecting to the work item repository, and connecting to the user’s client. Subsequent user actions show the loading .gif if the tool is processing data. After collaborative traces are loaded into the bottom grids, the interface displays a message to begin searching for contacts. When filters are applied in the bottom grid, the same .gifs appear on the grids themselves, indicating that filtering is being performed. Thus we can say Theseus shows good visibility of system status.

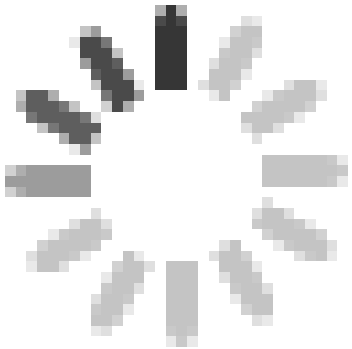


Figure 5-1. Universal “Loading” image used by Theseus to indicate the tool is busy processing data and that users must wait.

5.1.1.2 Match between system and real world

“The system,” Nielsen recommends, “should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.” Theseus

"speaks the user's language" in that it uses language and terms familiar to the user as well as visual representations similar to the user. On the one hand. users of the tool will recognize terms such as "E-mails," "Bugs," and "Change Sets" (and their fields, e.g. "From," "subject," "priority" "status," "commit") because these terms are extremely common in software development work. "Contact" is a term frequently used in e-mail clients to refer to a person with whom one may communicate. On the other hand, the contact list in the left pane that shows thumbnail pictures of contact arranged by row and column should be familiar to users of social networking sites like Facebook and Google+ and contact management features of Skype. The visual representations in the main pane are common to anyone who has worked with charts in Excel. As such we can say Theseus shows a good match between system and real world.

5.1.1.3 User control and freedom

This heuristic suggests that users "will need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialogue. Support undo and redo." Undo and redo are not supported in the Theseus interface. However, users can apply or re-apply filters in the collaborative trace grid easily be switching off and on checkboxes. Moreover, users can close the tool's window by clicking the X in the top right or left. When managing contacts on the left side, users can use the drag-and-drop metaphor to remove contacts they have added on accident. Users can also clear individual searched contacts on the main pane if they have searched a contact that is no longer useful to them. Thus we can say Theseus's compliance with the "user control and freedom" heuristic is moderate because it allows users to remove information from the interface but not to automatically re-do the changes.

5.1.1.4 Consistency and standards

According to this heuristic, "Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions." Theseus does not use different words or actions to refer to the same thing. For instance, individuals are always referred to "contacts" in the user interface, consistent with most address management software such as e-mail. Terms like "Emails," "Bugs," and "Change Sets" are consistent with the terms used in communication tools and software development tools such as bug trackers and control management systems. The collaborative trace grids show the same fields and properties as these tools. Text fields and buttons follow user interface standards defined in google's web application APIs, meaning they have the same "look and feel" as Google products such as GMail and Google Charts. We can say that Theseus satisfies this heuristic well.

5.1.1.5 Error prevention

This heuristic encourages the designer to "either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action." The Theseus interface pre-empts some erroneous user actions by disabling certain functionality. For example, the text box for searching and adding contacts to the left pane is disabled upon load of the interface, making the only option available to "Authorize" the tool and connect to project repositories. The same text box has an auto-complete feature that can be used to quickly select existing contacts in the user's Google Contacts account. This feature helps users avoid mis-spelling certain names. A failure to prevent errors is evident in the removal of a contact, however. When users drag a contact to the trash bin, there is no confirmation option before the contact is deleted. This feature can be easily added to the current version of the prototype. As such, Theseus moderately satisfies the "error prevention" heuristic.

5.1.1.6 Recognition rather than recall

This heuristic suggests, “Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.” Instructions for use of the Theseus interface are explicitly stated on the screen display. All objects and options are displayed on the interface at all times and there are not multiple dialogs that users must traverse. As such, users do not have to remember how to perform certain actions from one window to another. Clickable objects, such as paths to a contact through other contacts are highlighted in blue, much like hyperlinks on the web. Further, when users hover over non-textual representations such as charts and image thumbnails, the cursor changes to indicate that the object is clickable. We can therefore say that Theseus satisfies the heuristic.

5.1.1.7 Flexibility and efficiency of use

This heuristic refers to the system’s ability to “cater to both inexperienced and experienced users. Allow users to tailor frequent actions.” Currently, there are no accelerators in the form of keyboard shortcuts or key bindings. However, filtering can be chained together using Boolean operators or by manipulating a series of checkboxes, which turns out to be faster. Moreover, buttons are tab indexed so the user can press “enter” instead of selecting a button with the mouse and clicking it directly. Additionally, the interface offers flexibility with respect to deleting contacts. There are multiple ways to delete contacts from the left hand pane. The user can click a button to remove the contact of interest or drag a thumbnail image of that contact to the trash. As such we can say Theseus moderately satisfies this heuristic.

5.1.1.8 Aesthetic and minimalist design

"Every extra unit of information in a dialogue," Nielsen writes, "competes with the relevant units of information and diminishes their relative visibility." Each visualization in the interface of exists to serve a purpose. There are no irrelevant units of information in the interface. In some instances, labels and legends for pie charts and bar graphs are needed to interpret the graphs. Information that can be contained in pop ups on hovering events is embedded in those popups, such as the work items pie chart and the availability radar. However, one concern is that the scatter plot chart in the time zone is disproportionately sized to the pie chart visualization for example, and may take up too much horizontal space. Thus we can say Theseus has moderate aesthetic and minimalist design.

5.1.1.9 Help users recognize, diagnose, and recover from errors

This heuristic states, "Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution." Possible errors do suggest practical and actionable solutions rather than displaying error codes. For example, if Theseus fails to authorize the user, it will display a standard login window asking the user to check their name and password and re-submit the information. If a user filters the grids and no results are returned, the grid tells the user that no matches were found and to re-input the filter. In this sense, Theseus satisfies this heuristic well.

5.1.1.10 Help and documentation

"Help and documentation," the heuristic suggests, "should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. There is no help and documentation beyond the instructions that appear in the user interface that state in what order tasks should be performed. Yet using some of the visualizations such as the availability radar and time zone overlap require initial instruction. Knowing what it means to be in the third level of the radar or why the overlap is all green is crucial for interpreting the

information these respective visualizations convey. At the very least, a help dialog should be available that shows explicit instructions for use should be included. Thus, we can say Theseus offers poor help and documentation.

5.1.1.11 Summary

Theseus satisfies the majority of the heuristics specified by the Heuristic Evaluation. Theseus performed the poorest at help and documentation, so instructions will be added to the interface. In order to avoid the instructions interfering with use of the tool (minimalist design heuristic), a possible solution is the addition of a "?" button or icon next to each visualization. When clicked, a popup will display instructions for using the respective visual representations. Minor modifications, as indicated in the analysis of each heuristic where the tool performed moderately well, will also be made to the tool before it is assessed in the laboratory evaluation in Phase 2.

5.1.2 Cognitive Walkthrough

In addition, the Cognitive Walkthrough (Wharton et al., 1994) was used to analyze the usability of the Theseus prototype. The Cognitive Walkthrough is a usability inspection method that entails two activities: 1) the designer does a task analysis that specifies the sequence of steps or actions required by a user to accomplish a task, and the system responses to those actions, and 2) the designer walks through each task and, for each subtask, asks themselves a set of four questions (Wharton et al., 1994) that are designed to uncover usability issues. The four questions are:

1. Will the user try to achieve the effect that the subtask has?
2. Will the user notice that the correct action is available?
3. Will the user understand that the wanted subtask can be achieved by the action?
4. Does the user get feedback?

In the following subsections, the Cognitive Walkthrough is applied to Theseus using three analytical tasks useful for evaluating information visualizations (Stasko, 2007). It is assumed that for each task presented below, the user of the system is given a list of developers with whom they will collaborate.

5.1.2.1 "Order developers by the number of hours away from you"

The action sequence for this task is:

1. *Authorize Theseus*
    1.1. Select current location from dropdown list
    1.2. Click "Authorize Theseus" button
2. For i = 1 to number of contacts, *Search Contact*
    2.1. Type contact's name in the text field in the upper left
    2.2. Click "Lookup Name" button
    2.3. Examine spot on availability radar where contact appears
3. *Use Availability Radar to Compare Contacts*
    3.1. Visually compare position of contacts on radar
    3.2. Hover over text for the number of time zone hrs away from user

Analysis:

1.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user will understand that they must select a location from the drop-down list. This information is necessary to "initialize" the tool, so to speak. The user will interpret this as a necessary sub-task because the instructions on the interface indicate it as such.
2. *Will the user notice that the correct action is available?* Yes, the user will know that this action is available because there is a drop down box with the same "look and

feel" as other user interface elements in other applications. In addition, next to the drop down box is a label that tells the user to select the location from the list.

3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the user will understand that clicking items in the drop down list will allow them to select their current location. The items in the drop down box are geographical locations, which correspond to the user's subtask of literally "selecting a location."
4. *Does the user get feedback?* Yes, the user gets good feedback. After they select a location, a dialog box appears on the screen telling them that, in fact, they have selected a location.

1.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user will understand that they must authorize the tool to collect project relevant information to build up their contacts, e-mails, work items, and change sets. They will understand the effect of authorizing the tool because of the instructions on the interface that tell them this is a necessary step.
2. *Will the user notice that the correct action is available?* Yes, there is a clearly visible button that reads, "Authorize Theseus." It is the only button on the interface that is enabled at this point in the task.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the text on the button reads, "Authorize Theseus." This text matches the instructions on the interface, so it is obvious that this button will connect to the project repository and allow the user to proceed to the next step in the task.
4. *Does the user get feedback?* Yes, the user will get good feedback for two reasons. First, the tool will display a login window to Google to authorize the user. This step lets the user know that no information will be displayed without their authorization.

Also, it indicates that the user is making progress toward their goal. Second, after the user logins, the user interface will display a standard "loading" animated image. While this image displays, text on the interface will indicate what task the tool is performing (e.g., what step in the data collection phase it is in).

2.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user will understand that they should type in the name of a contact to search. This is for two reasons: the focus of the tool is to explore contacts, and instructions in the main pane of the interface tell the user to "Search for a contact" as the second step in using the tool.
2. *Will the user notice that the correct action is available?* Yes, the user will notice that the action is available. After step 1, the textbox and button for searching a name become enabled and thus more visible to the user.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the text box implies that the user should type something in the box. The placement of the "Lookup Name" button indicates that the tool will search for whatever the user enter types.
4. *Does the user get feedback?* Yes, the user will see that progress is being made as they type characters in the box. Whatever they type will appear in the box, character by character. If the tool finds the contact in the contact database, it will auto-complete the text box with the contact's name, saving the user time.

2.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, as explained in 2.1, the user knows that they are searching for information about a contact. The user understands that they must provide the tool with input and explicitly ask the tool to

search for what they have entered. Clicking a button is a very common way of issuing a query.

2. *Will the user notice that the correct action is available?* Yes, the user will know they can click a button to search for a contact because the button is visible next to the input text box. After step 1, the button becomes enabled.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the user will know to click the button because of the text on the button and its placement next to the input box. The words "Lookup Contact" are actionable and imply the tool will perform a search operation as soon as the button is clicked.
4. *Does the user get feedback?* The user will get moderately good feedback because the interface will respond by filling the interface with different visualizations, one of which is a picture of the contact if it is available, i.e., exists in the user's set of contacts. However, there will be some delay until all elements fill the screen. This may confuse the user because the standard way for showing progress is to use the loading animation image. A way to improve the feedback would be to use the standard loading image until all panels of the visualization are filled.

2.3

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user will understand that they should examine the availability radar to determine where the contact falls, to the extent they know the use of the availability radar. To ensure that the user knows how the availability radar works, instructions will be included about how to use it, especially since this issue was brought up in the Heuristic Evaluation.
2. *Will the user notice that the correct action is available?* Yes, the user will know that they can look at the availability radar because an animation will gradually show the

contact falling from the "sky" onto the availability radar. This animation captures the user's attention and draws them to the availability radar.

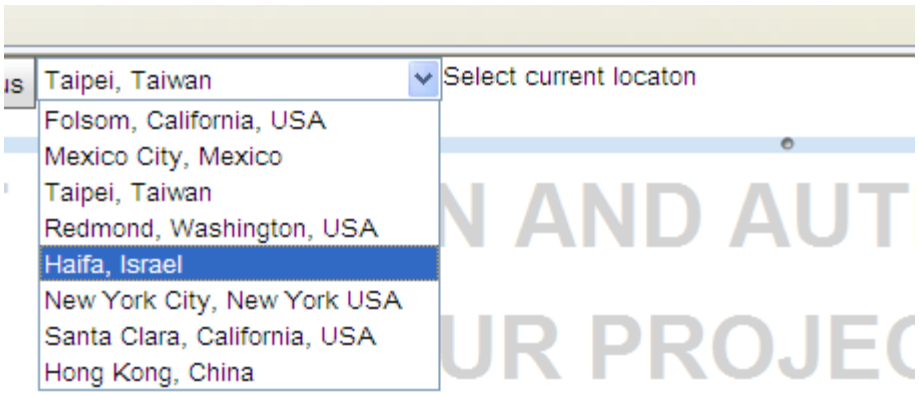
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the user will know to associate the placement of the contact in the radar with how many time zones away the contact is, based on the levels encoded into the visualization. As in question 1, this assumes the user knows the purpose of the visualization and how to interpret it.
4. *Does the user get feedback?* Because the user's action consists of examining the placement of a visual representation, there is no feedback from the user interface in the traditional sense. However, learning does occur on the part of the user. At this point, the user learns that for each contact searched, a new spot in the availability radar will be created. Subsequent glances at the availability radar will thus be easier.

3.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, because the task requires the user to compare multiple compacts, and the user learns that each contact searched appears on the availability radar, the user will know that as each contact is added, it can be compared to the one before it.
2. *Will the user notice that the correct action is available?* Yes, as each contact is added, it appears on the availability radar.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, as each contact appears in a different position on the radar, the user will recognize they can be compared by position.
4. *Does the user get feedback?* As in 2.3, there is no direct feedback because the subtask requires no physical action or manipulation with the interface.

3.2

1. *Will the user try to achieve the effect that the subtask has?* No, until the user hovers over an icon, for the first time, they may not know they can get additional information about each contact.
2. *Will the user notice that the correct action is available?* No, the user will probably not know this action is available because tool tips do not appear until the user hovers over the items. Any help and documentation included for the availability radar should clearly state this action is available.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Maybe. In visualizations it is common to provide methods for obtaining more detail about a visual representation through the "hover" action. As such Theseus users may have this intuition, or they may not have it.
4. *Does the user get feedback?* Yes, a tool tip appears over the contact and displays the number of hours away the contact is from the user. In addition, at this point in the task, the user learns that they can hover over the icons to display further information about a contact. Subsequent subtasks of this type should not be problematic.



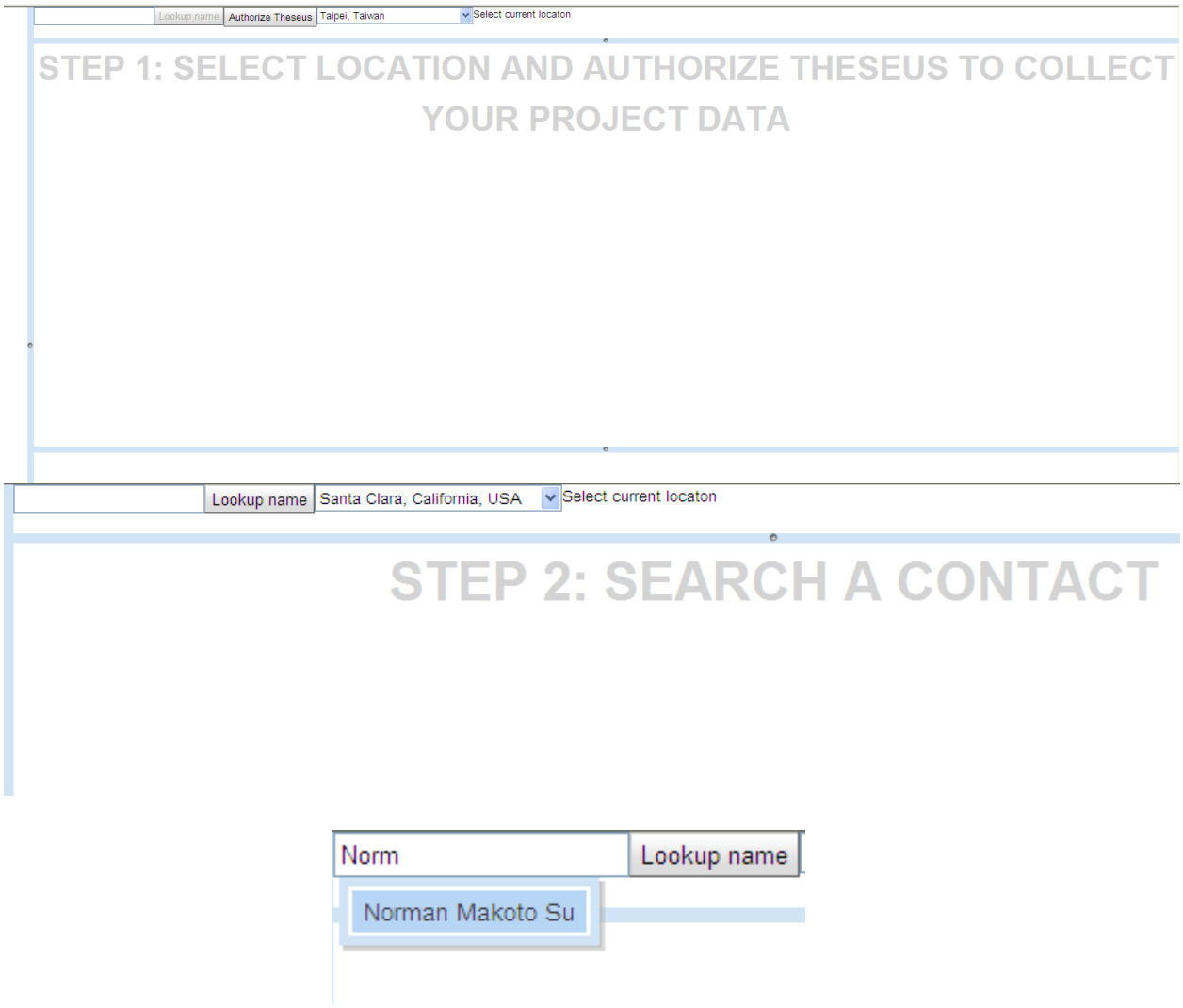
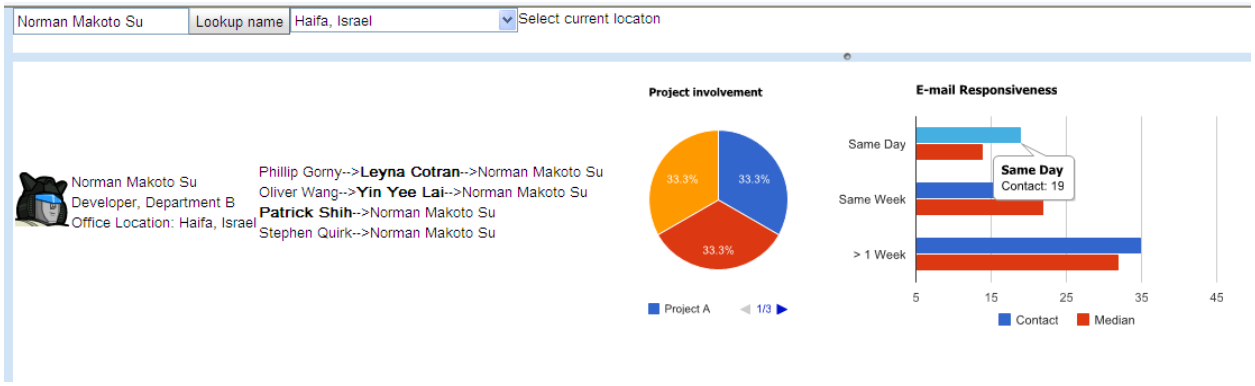


Figure 5-2. Selecting a location and searching for a contact.



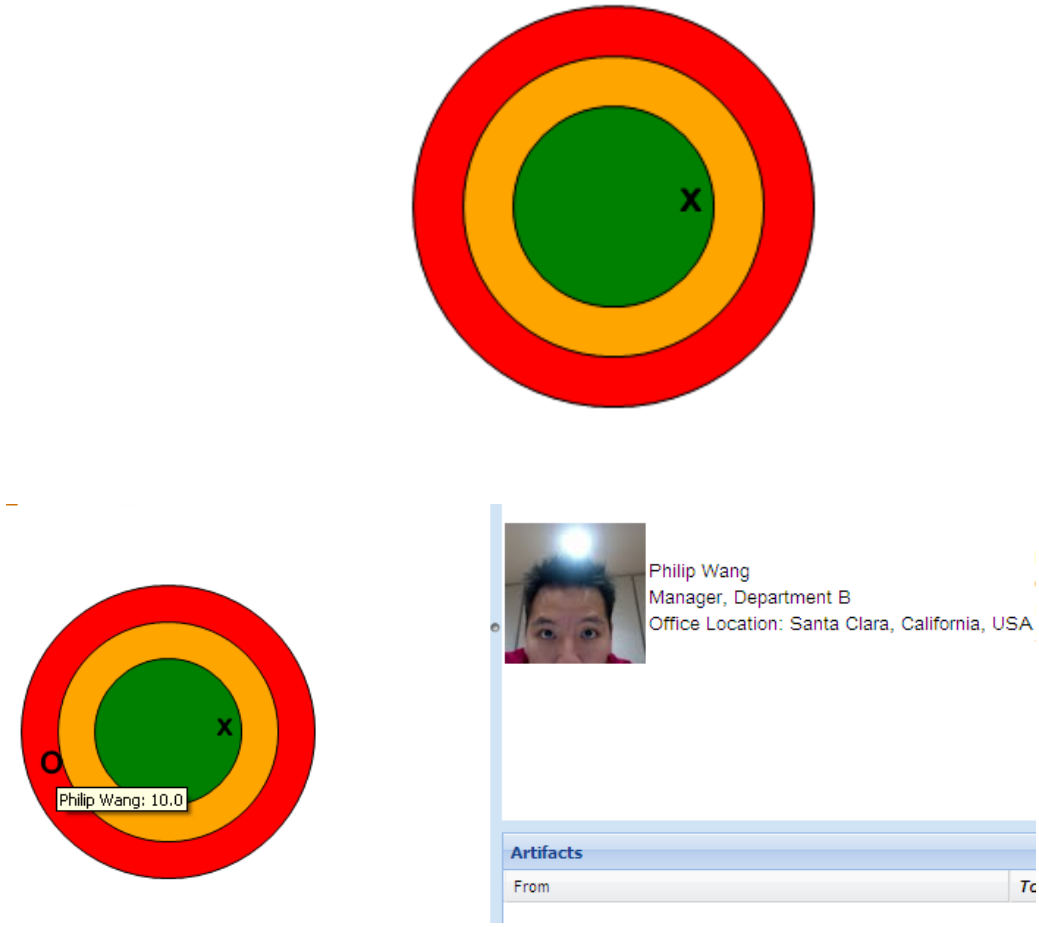


Figure 5-3. Comparing location between 2 developers.

5.1.2.2 "Sort developers by the number of other work items they have resolved"

The action sequence for this task is:

1. *Authorize Theseus*
   1.1. Select current location from dropdown list
   1.2. Click "Authorize Theseus" button
2. For i = 1 to number of contacts, *Search Contact*
   2.1. Type contact's name in the text field in the upper left
   2.2. Click "Lookup Name" button
   2.3. Count number of work items using Bugs grid
3. Identify Resolved Work Items

3.1. Launch benevolence visualization

3.1.1. Drag relevant contacts to visualization

3.1.2. Compare area of circles for selected developers

OR (not optimal! But possible)

3.2. Compare number of work items for each contact, as counted in step 2.3

Analysis:

1.1 through 2.2 are covered in the previous task

2.3

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user will know to look at the work items view because the purpose of the task is to count resolved work items by the contact in question.
2. *Will the user notice that the correct action is available?* Yes, the user should understand that this action is available because the work item collaborative trace display is visible on the bottom of the screen. In addition, every time a contact is searched, the display updates and automatically filters on the "resolver" field with the contact's name.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the user will know that counting the work items will result in the number of work items resolved by the contact. Each work item takes up a row in the table. Therefore, the user can count the number of rows in the table. Each row has a "resolver" column, which corresponds to the name of the person who resolved the work item.
4. *Does the user get feedback?* There is no direct feedback because the user is performing a count of the number of rows in the table. However, there is implicit feedback; a text label at the bottom of the table does indicate the number of rows presented. The user can check that their count matches this number.

3.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user should know that this subtask is needed to reach the goal because the benevolence visualization shows who resolves whose work items and who replies to whose messages. To make this more clear, help and documentation will be added to the visualization as suggested earlier.
2. *Will the user notice that the correct action is available?* Yes, a button is visible on the collaborative trace grid that launches the visualization.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the text on the button clearly states "Launch Benevolence Visualization"
4. *Does the user get feedback?* Yes, after the user clicks a button, a separate panel displays with the benevolence visualization. This indicates to the user that the operation has completed successfully.

3.1.1

1. *Will the user try to achieve the effect that the subtask has?*
2. *Will the user notice that the correct action is available?*
3. *Will the user understand that the wanted subtask can be achieved by the action?*
4. *Does the user get feedback?*

3.1.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user is interested in comparing the number of work items different developers have resolved. Because the areas of circles in the visualization correspond to these numbers, users will look at the relative sizes of the circles in the visualization.
2. *Will the user notice that the correct action is available?* Yes, the circles in the visualization are the main focus. They are clearly visible.

3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the circles are labeled with the developer's name in the center of the circle. The circles will also have different areas from which users can compare resolved work items by developer.
4. *Does the user get feedback?* There is no direct feedback because the developer is making area judgments.

3.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, if the user does not use the benevolence visualization to compare developers, they will most likely count the rows in the work item grid for each developer and then compare them after the fact. However, this is not the optimal path for the task.
2. *Will the user notice that the correct action is available?* No, the action to compare resolved number of work items for multiple developers is not in the interface. This is a step that must be done manually, without the aid of the tool.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, because the user can get a count of work items for each developer, it is intuitive that they can compare each of those numbers for every developer.
4. *Does the user get feedback?* No, the user does not get feedback because this is a manual task.

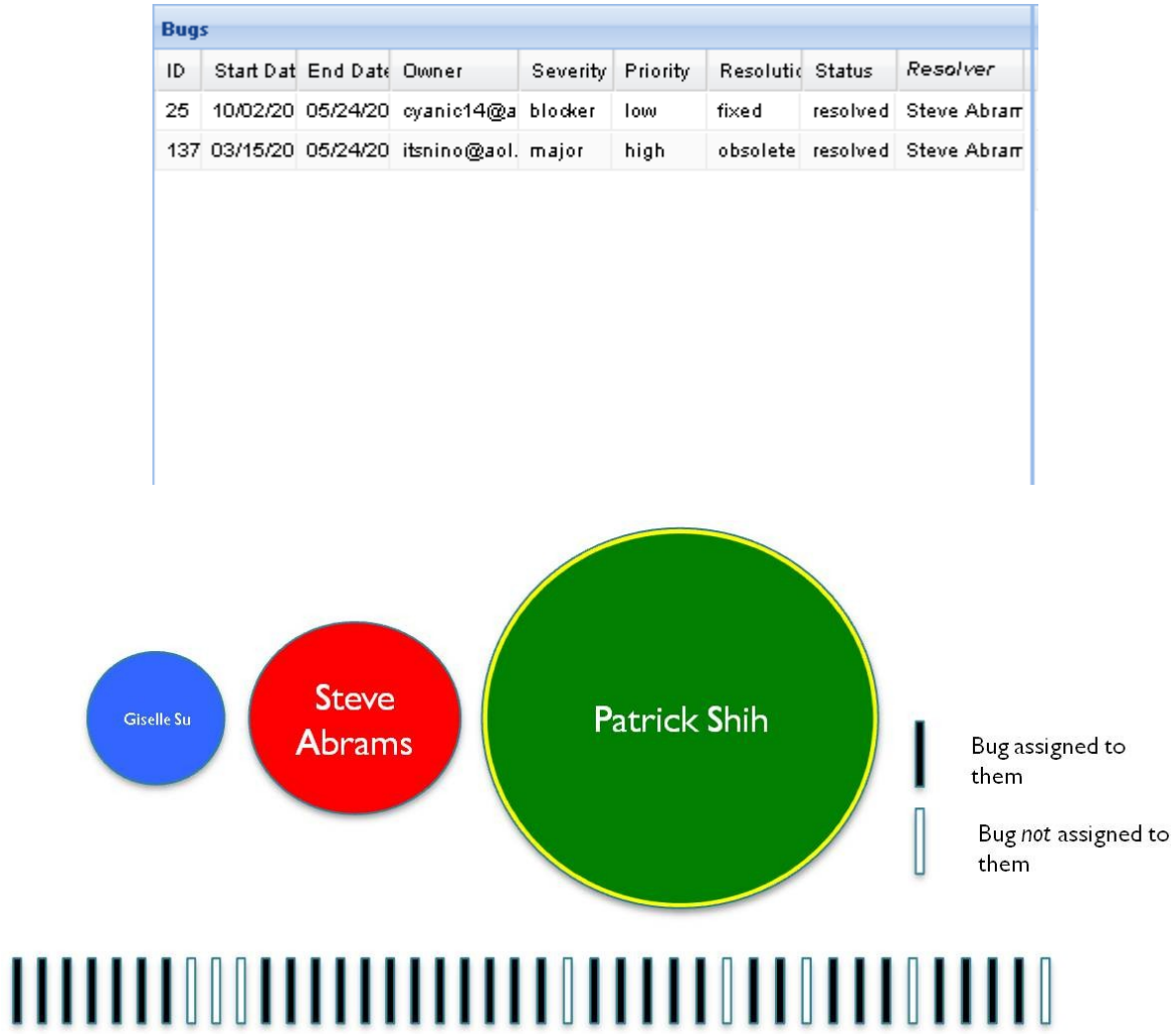


Figure 5-4. 2 views of the benevolence visualization.

5.1.2.3 "Identify the developer least likely to respond in the same day"

The action sequence for this task is:

1. *Authorize Theseus*
   1.1. Select current location from dropdown list
   1.2. Click "Authorize Theseus" button
2. For i = 1 to number of contacts, *Search Contact*
   2.1. Type contact's name in the text field in the upper left
   2.2. Click "Lookup Name" button
3. Compare Developer Responsiveness using Responsiveness Bars

3.1. Look at consecutive developers' graphs of response bars

3.2. Visually compare bars in the "Same day" category across contacts

4. Compare Responsiveness with Scatterplot + Time Zone Overlap

4.1. Look at consecutive developers' graphs of reply time + time zone overlap

4.2. Visually compare them

5. Compare Resonsiveness Bars and Scatterplot + Time Zone Overlap

5.1. Visually compare the graphs

Analysis:

1.2 through 2.2 are covered in the previous two tasks

3.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user knows that to find who is least likely to respond in the same day, they must compare responsiveness times. To the extent that the user knows what each visualization is for, they will know to look at the visualizations in the main pane.
2. *Will the user notice that the correct action is available?* Yes, after the user searches for a contact, graphs for that contact appear in the main pane. They are clearly visible.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the graph is clearly labeled with the title "Average Reply Time."
4. *Does the user get feedback?* The user does not get feedback because the action is to read the graphs; no user interface action is expected.

3.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, because the task is framed same day, the user will know to determine the number of e-mails sent within the same day, and therefore look at the bars whose labels indicate that as such.

2. *Will the user notice that the correct action is available?* Yes, because contacts appear one on top of the other and the graphs of responsiveness have the same representation, the user will know they can compare the values for "Same Day." However, if more than 2 contacts are to be compared, the user might not easily see them if they do not appear consecutively (i.e., they were not searched one after the other). In this case, a drag and drop should be implemented that allows contacts to be moved according to a user-defined order.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, because the visual representations are the same for responsiveness for each contact, the user will see they can compare the values for "Same Day." Horizontal bars afford easy comparisons (Mackinlay, 1986).
4. *Does the user get feedback?* The user does not get feedback because this is a visual comparison task.

4.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, the user knows that to find who is least likely to respond in the same day, they must compare responsiveness times. To the extent that the user knows what each visualization is for, they will know to look at the visualizations in the main pane.
2. *Will the user notice that the correct action is available?* Yes, after the user searches for a contact, graphs for that contact appear in the main pane. They are clearly visible.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the graph is clearly labeled with the title "Average Reply Time."
4. *Does the user get feedback?* The user does not get feedback because the action is to read the graphs; no user interface action is expected.

4.2

1. *Will the user try to achieve the effect that the subtask has?* Yes, because the task is framed same day, the user will know to determine whether the average reply time is less than 24 hours, and therefore look at visualization.
2. *Will the user notice that the correct action is available?* Yes, because contacts appear one on top of the other and the graphs of reply time have the same representation, the user will know they can compare the values of reply time across the horizontal axis. However, if more than 2 contacts are to be compared, the user might not easily see them if they do not appear consecutively (i.e., they were not searched one after the other). In this case, a drag and drop should be implemented that allows contacts to be moved according to a user-defined order.
3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, because the visual representations are the same for reply time for each contact, the user will see they can compare the values using horizontal position. Position is an encoding mechanism that affords easy comparisons (Mackinlay, 1986).
4. *Does the user get feedback?* The user does not get feedback because this is a visual comparison task.

5.1

1. *Will the user try to achieve the effect that the subtask has?* Yes, because both visualizations indicate a different dimension of responsiveness (percentage answered within several time periods, and average reply time per message), the user realizes the visualizations are related. The user will therefore perform some comparison between the two.
2. *Will the user notice that the correct action is available?* Yes, both visualizations appear side by side in the interface.

3. *Will the user understand that the wanted subtask can be achieved by the action?* Yes, the user knows that the graphs can be visually compared because they present similar information.

4. *Does the user get feedback?* There is no feedback because this is a visual comparison subtask.

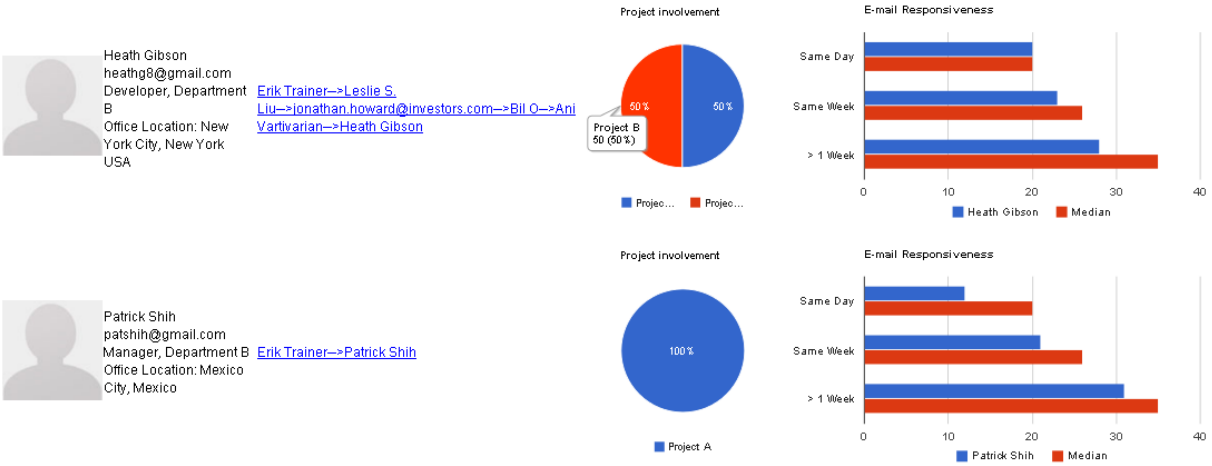


Figure 5-5. Comparing the e-mail responsiveness of 2 developers.

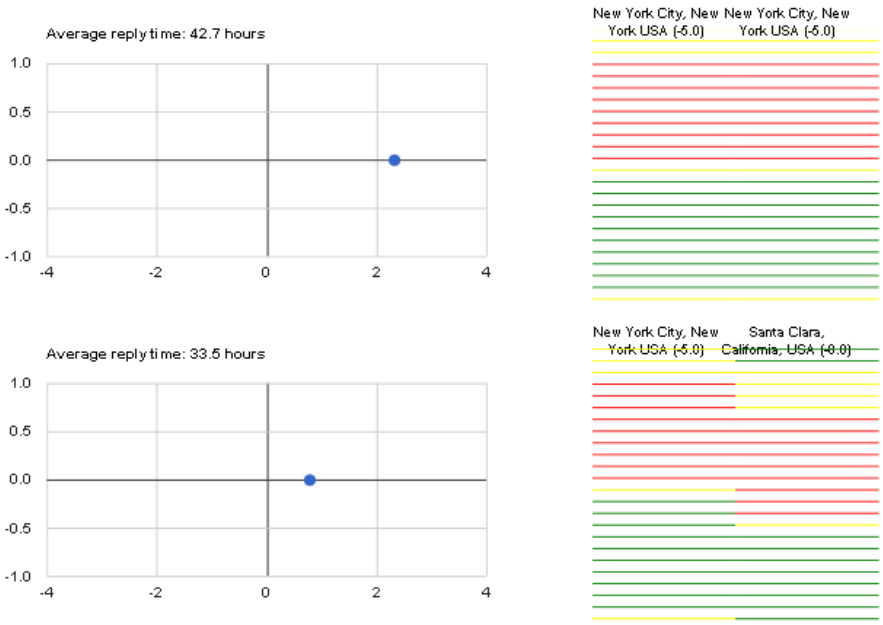


Figure 5-6. Comparing the e-mail responsiveness of 2 developers using the aid of a time zone overlap visualization.

5.1.2.4 Summary

After applying the Cognitive Walkthrough, we can say Theseus has several usability problems. First, it is obvious that help and documentation need to be added to the different visualizations so that users know tasks can be completed using them. This issue was highlighted in the Heuristic Evaluation as well, and should be considered a major priority. For example, the levels of the availability radar need to be clearly designated. The first question about knowing what subtask is needed to reach the goal revealed this problem in several of the subtasks analyzed. Second, some actions are not visible, such as hovering over a point in the availability radar. Instructions that talk about hovering will improve this aspect of the interface. The third usability concern has to do with making comparisons of developers' responsiveness. When 2 developers are to be compared, the situation is manageable, but when there are more than 2 developers to be compared, the user can easily lose sight of the comparison. Therefore, a drag and drop solution should be implemented so that users can group together the contacts to be compared.

5.1.3 Cognitive Dimensions of Notations

The Cognitive Walkthrough uncovered issues that made learning how to use the visualization difficult. I chose to complement this analysis with the Cognitive Dimensions of Notations Framework (Green, 1989) in order to further uncover mentally demanding operations in the visual interface. Cognitive Dimensions provides a vocabulary for analyzing the usability of tools, programming languages, and environments. Like the other inspection methods presented here, Cognitive Dimensions are designed for non-usability specialists and can be applied in the early stages of design before experiments with human subjects.

Similar to other work (Khazaei and Triffitt, 2002), the "Cognitive Dimensions Questionnaire Optimised for Users" (Blackwell and Green, 2000) was used as a starting point to identify the relevant dimensions as a basis for the evaluation. The questionnaire clearly

presents the concepts and introduces a set of questions that map to each cognitive dimension. I used these questions and the details from (Green, 1989) to complete the analysis. This section presents the results.

5.1.3.1 Visibility and juxtaposability

This dimension states that it should be easy to tell what has been changed or created. Different elements in the interface should be visible at the same time if the user needs to compare them. Theseus indicates changes to the interface in several ways. When contacts are searched, for example, a new row with the contact's information appears in the middle pane. In addition, an animation of the contact being placed on the availability radar appears shortly thereafter. Moreover, when the user filters collaborative traces, the grids refresh to reveal the results.

There are no additional windows or dialogs that obstruct any one unit of information in the interface. As the user searches contacts, information appears below the previously searched contact to facilitate comparisons of the contacts' information. For example, responsiveness can be directly compared by looking at how far to the left or right the contact appears in the scatterplot visualization. Moreover, in the radar visualization multiple contacts can be compared in terms of their proximity to the user by judging their distance from the center of the radar. However, it is difficult to compare contacts that do not appear consecutively, one after the other, in the main pane. A practical solution may be to enable a drag-and-drop feature where users can drag-and-drop a contact on top of or below the contact to which the comparison should be made. As such we can say Theseus offers good visibility and juxtaposibility.

5.1.3.2 Viscosity

The "viscosity" dimension tests how difficult it is to make changes to previous work in the interface and whether there are particular changes that are more difficult to make than

others. Theseus offers some support in terms of viscosity, but because the interface is intended to display static information about contacts, the extent to which users can modify information in the interface is limited. For example, the tool allows users to remove individual contacts, either from the search window or the trusted set of contacts in the left pane. However, there is no option to remove all contacts, or "reset" the list. Moreover, once a contact appears on the availability radar, they cannot be removed. An obvious solution would be to allow the user to right-click, or through some other interaction, and delete the symbol for the contact in question. Users can easily modify filters in the collaborative trace grid by unchecking previous filters and typing new ones into the filtering text field. Thus we can say Theseus moderately satisfies the viscosity dimension. In order to improve the tool along this dimension, the availability radar should be modified to allow deletion of contacts.

5.1.3.3 Diffuseness

"Diffuseness" tests whether visual representations in the interface concisely convey the intended information instead of being long-winded. The visualizations in Theseus were carefully designed to convey only information relevant to affective trustworthiness. The image thumbnails in the left pane minimally convey the identities of the user's trusted contacts by displaying an image of the contact and their name. Conveying other contact identifiable information such as age, location, and interests would be distracting from the user's task, which is to find information about contacts they do *not* know so well. Therefore only 2 representations are appropriate. Although responsiveness is conveyed using multiple visual representations such as the responsiveness bars and scatterplot and therefore may be viewed as long-winded at first glance, these representations show responisveness from two different perspectives. As such they are not redundant. Thus we can say that Theseus satisfies the diffuseness dimension.

5.1.3.4 Hard mental operations

This dimension assesses whether the interface requires heavy mental effort on behalf of the user. We can say that Theseus requires no more than minimal mental effort. The action that requires the most mental effort is comparing information from two contacts. The visual representations ease the mental load on the user. For example, horizontal bars are used to facilitate comparisons between response times based on the number of days passed, scatterplot points are used to compare average reply times, and color is used to show overlap in time zones. Juxtaposition is related to hard mental operations when we think about comparisons made between contacts. As such, just as in Section 5.1.3.1, comparing contacts who do not appear consecutively is a mentally demanding task. As in 5.1.3.1, this problem suggests that adding a drag-and-drop feature may fix the problem.

5.1.3.5 Closeness of mapping

"Closeness of mapping" refers to how closely related the visual notations and representations are to what is being described in the interface. There are several things described in the interface: contacts, work artifacts, responsiveness, and availability. The visual representations for contacts consist of images of the contacts and textual descriptions such as name and location. These are appropriate considering traditional representations of this information, as found in address books and social networking pages. Similarly, the textual representations of e-mails, bugs, and change sets follow standards set in e-mail clients, bug trackers, and CM systems. The same fields found in these tools such as "status" and "subject" for bugs and e-mails respectively, are present in Theseus' interface.

Responsiveness and availability are more like attributes that can be conveyed through common charts and graphs. They do not have standard representations, so it is difficult to say whether the visual representations in Theseus are correct or not. On the other hand, the visual representations exploit encoding rules established by Bertin (1983) and Mackinlay (1986).

For example, position is used in the scatterplot to judge how responsive a developer is. Color is used in the time overlap visualization to indicate times of the day. Horizontal bars are used to compare a contact's reply rate in the responsiveness bars visualization. As such, we can say Theseus has a good "closeness of mapping."

5.1.3.6 Role expressiveness

This dimension asks whether it is easy to tell what role each visual representation plays in the overall scheme of the interface. With the exception of the availability radar, it is easy to tell what the role of each representation is in the interface. Visualizations in the main pane correspond to information about the contact being searched. The pie chart shows the contact's parallel projects, the responsiveness bars indicate their time to reply, the time zone overlap indicates their overlap with the user. The availability radar, however shows information about the contacts as an aggregate. Because its location in the interface is separate from the main search pane, users may question its relevance and relationship to the other representations. Because changes to this visual representation appear after each contact is searched in turn, its purpose may gradually become clear to the user. A straightforward solution may be to label the availability radar with instructions or a short description of its purpose in the form of a text caption. Therefore we can say Theseus moderately satisfies role expressiveness.

5.1.3.7 Hidden dependencies

"Hidden dependencies" assesses the following: if there are dependent operations in interface, that is if some changes affect other parts of the interface, are these dependencies visible? The interface indicates, through textual displays, that authorizing the tool is required before the user can start performing other tasks. It is implied that searching a contact will display information about that contact in the center display pane. It is not obvious, however, that searching a contact will populate the availability radar or filter the collaborative trace

grid at the bottom of the interface. It is not until the action of searching for a contact is completed that the interface provides appropriate feedback by populating the visualizations with data for that contact. Providing help and documentation as per Section 5.1.1.10 can expose these hidden dependencies. Thus with some slight modifications to the interface, Theseus has no hidden dependencies.

5.1.3.8 Premature commitment

Premature commitment tests whether the user can perform tasks in any order they like, or the interface imposes a certain order of operations. If the latter, the order should be obvious and should not cause problems using the interface. There is a specified order for tasks performed in the interface. In particular there is an order that operations must follow: authorize the tool, select a location, and search a contact. The order is made obvious by textual descriptions of each step on the center display of the interface. Disabled buttons prevent the user from performing tasks out of order. Once the first two operations are performed, the user is free to search as many contacts as they wish. Thus Theseus satisfies the premature commitment dimension.

5.1.3.9 Consistency

This dimension states that if different visual representations mean similar things, the similarity should be clear from the way the representations appear. Theseus uses a scatterplot and a bar chart to convey similar aspects of responsiveness, namely reply time in terms of days as well as average hours to reply, respectively. The similarity is clear because the scatterplot and bar chart appear next to each other in the interface and each of them is labeled on top with a description of what each conveys. As such, Theseus has good consistency.

5.1.3.10 Summary

Theseus satisfies the majority of the Cognitive Dimensions, save for Viscosity and Role Expressiveness. A solution to the Viscosity dimension suggests adding the ability to

delete contacts from the search list. A possible solution to Role Expressiveness is to provide help and documentation indicating the role of particular visualizations such as the availability radar.

5.1.4 Phase 1 Summary of Results

In summary, Theseus' visual interface satisfies the majority of the Heuristic Evaluation heuristics and Cognitive Dimensions. The Cognitive Walkthrough confirmed problems revealed by both inspection methods. For example, the question of knowing which visualization to use for a subtask is related to the Role Expressiveness Cognitive Dimension. In this case, the user may not know what the availability radar is for, relative to the other visualizations in the interface. Overall, however, the interface meets 90% (9/10) of the heuristics defined in the Heuristic Evaluation and 100% (9/9) of the Cognitive Dimensions. Of the requirements met, 44% (4/9) of the Heuristic Evaluation heuristics and 22% (2/9) Cognitive Dimensions are moderately satisfied.

Table 5-1. Results of Heuristic Evaluation and Cognitive Dimensions.

Method	Item	Compliance		
Heuristic Evaluation		Low	Moderate	High
	Visibility of System Status			X
	Match Between System and Real World			X
	User Control and Freedom		X	
	Consistency and Standards			X
	Error Prevention		X	
	Recognition Rather than Recall			X
	Flexibility and Efficiency of Use		X	
	Aesthetic and Minimalist Design		X	
	Help Users Recognize, Diagnose, and Recover from Errors			X
	Help and Documentation	X		
Cognitive Dimensions				
	Visibility and Juxtaposability			X
	Viscosity		X	

	Diffuseness			X
	Hard Mental Operations			X
	Closeness of Mapping			X
	Role Expressiveness		X	
	Hidden Dependencies			X
	Premature Commitment			X
	Consistency			X

However, the biggest usability concern is the lack of help and documentation for the different visualizations, as revealed by the Heuristic Evaluation inspection, Cognitive Walkthrough, and Cognitive Dimensions. As such, instructions will be added to the visualizations in the style described in the Help and Documentation analysis of the Heuristic Evaluation A second concern revealed by the Cognitive Walkthrough and the Cognitive Dimensions is the lack of support for comparing more than two developers in the main panel. A suggested solution is to implement a drag-and-drop functionality that allows users to re-arrange developers on the panel so that direct comparisons are less mentally demanding, i.e. less scrolling between contacts. As such, both issues are of high priority and were added to the interface before the human subjects evaluation began. In Phase 2, these particular usability issues should not arise, therefore validating the evaluation performed in Phase 1.

5.1.5 Interface Re-design

Based on this report of Theseus' usability, I made several modifications to Theseus' user interface. In particular, to address the issue of help and documentation, I added help icons to the interface that display detailed instructions for interpreting each visualization. In addition, to address the problems of aesthetic and minimalist design and role expressiveness, I removed the availability radar, responsiveness bars, and collaborative trace grid. The availability radar, for example, displays information in a corner of the screen then makes that information available only on a hover-over from the user. This is problematic because it becomes mentally demanding to track this activity while comparing contacts in the main

view panel. The responsiveness bars were removed because they are redundant; the reply time scale already gives an indication of a contact's responsiveness. As such, their role is not clear. Visiting researchers and members of my research group agreed that the time zone overlap and average reply time visualizations could potentially be the most useful. As such, these visualizations remained in the interface. Further, I implemented a drag-and-drop solution to improve the user's experience of comparing contacts several rows apart in the main display panel.

5.2 Phase 2: Laboratory Experiment

Goal: *Assess the utility of the Theseus prototype and re-assess its usability with human subjects.*

This laboratory experiment evaluates the utility of Theseus by measuring perceived trustworthiness scores and insight characteristics. The experiment tests several hypotheses about how using Theseus can result in higher perceived trustworthiness. Usability is also re-assessed using standard questionnaires and analyzing participants' think aloud observations from the video recordings.

5.2.1 Conceptual Design

The conceptual goal of the experiment is to prove the over-arching thesis statement posed by this dissertation: *A tool can usefully engender perceived trustworthiness.*
This sub-section presents a conceptual overview of an experiment that meets this goal. It describes the central task, what is measured in the experiment, and the particular hypotheses checked.

5.2.1.1 Task: Failure to Deliver

This study presents human subjects with the task of making sense of a particular communication break-down: failure to deliver on time. In interviews with distributed team members, we asked participants when they were surprised in a collaboration. Failing to

deliver on time or to meet commitments, emerged as a prominent example of such a surprise (Al-Ani and Trainer 2011). Typically, this had a negative result on the interviewee's sense of trust in that collaborator. In every case, the participant gave an explanation or attribution for the person's inability to meet the deadline. Following this example, participants in this experiment are asked to choose likely attributions for the developer in question's failure to deliver on time.

Attributions

Remember that attributions play a key role in assessing someone's perceived trustworthiness. "Attribution is the process by which people make inferences about the causes of events...when things go wrong, people may blame individuals rather than examine characteristics of the situation. This directs attention away from team-level learning and lasting solutions and harms team cohesion and cooperation" (Cramton, 2001; Cramton, 2002). Additional psychological research on trust has examined in great detail the role that cognitive processes such as attributions, expectations, and individual differences play in the development of trust and distrust (Deutsch, 1958; Rotter, 1967; Wrightsman, 1991). The authors find that *dispositional attributions reflect low perceived trustworthiness while situational attributions reflect high perceived trustworthiness.* The lack of information in distributed teams leads to a tendency to over-emphasize personal characteristics when a failure occurs, leading to fundamental attribution error. When a person is making fundamental attribution error, that person is really saying, "This person doesn't seem trustworthy at this moment."

As such, the experiment proposed here uses attribution quality as a measure of perceived trustworthiness. Section 5.3.2.2 presents details of the attributions and a specific description of the experiment task.

5.2.1.2 Summary of Measures

The independent variable (control) in this experiment is the presence or absence of the Theseus tool. To prove the hypothesis, the tool must yield significantly higher trust compared with the no tool condition. Thus there are at least two different conditions: 1) no Thesus tool and 2) Theseus tool. Because attributions may be situational, dispositional, or a combination of both, condition 2 was further subdivided into three conditions: 1) Theseus situational, 2) Theseus moderate situational, and 3) Theseus dispositional.

There are four conditions in total:

a. **No Theseus**
b. **Theseus situational**
c. **Theseus moderate situational**
d. **Theseus dispositional**

The dependent variables in the experiment are *usability scores*, *trust scores*, *attribution scores*, and *insight characteristics*. The following sub-sections provide an overview of how each variable was measured.

Usability scores were measured using questionnaires and by analyzing the video recordings of each participant's session. Usability questions were selected from standard usability instruments (e.g., (Lewis, 1993; Chin et al., 1988) as well as the Technology Acceptance Model framework (Venkatesh & Davis, 2000). The video recordings produced by the commercial usability program *Morae* were viewed to flag usability problems in the tool and errors expressed by participants.

Trust scores were measured in two ways:

1. **Standard specific inter-personal trust scores** (Johnson-George and Swap, 1982), a survey instrument that measures one's perceived trustworthiness toward a specific individual.

2. **Attribution scores** (Cramton, 2001; 2002). Situational and dispositional attributions reflect the individual's sense of perceived trustworthiness to another individual. Situational attributions are characteristic of high perceived trustworthiness. Dispositional attributions are characteristic of low perceived trustworthiness.

The data collected about trust and usability rely on subjects self-reporting. Self-reports are often the best way to assess psychological states (Clore, 1994). Self-reports have the highest fidelity when they are most accessible (Robinson and Clore, 2002). So, for example, if an emotion has faded over time, subjects will report how they think they should have felt. As such, in this experiment subjects were given questionnaires that measure trust immediately after each task, when the thoughts should have been fresh in their minds.

Generating insight is one of the primary purposes of visualization (Card et al. 1999; Spence, 2001). An insight is defined as "an individual observation about the data by the participant, a unit of discovery" (Saraiya et al., 2005). Suddenly seeing something previously unnoticed or seeing familiar from a new perspective can affect how we understand the world around us. One of the claims of this dissertation is that having this information when a communication breakdown occurs can reduce the chances of trust eroding when it is unwarranted. In order to measure insights users can potentially glean from Theseus, a coding scheme originally developed by Saraiya et al. for the insight-based evaluation (2005) was followed. Several characteristics of insights were measured:

Table 5-2. Insight characteristics and meanings.

Insight Characteristic	Meaning
Observation	The actual finding about the data, an expectation enumerated. Keep track of distinct data observations by each participant.
Time to first insight	The amount of time taken to reach the insight.
Total time	The amount of time until no more insights can be discovered.
Correctness	Incorrect observations that result from misinterpreting the visualization. All insights coded as "correct" or "incorrect."

5.2.1.3 Hypotheses

The research hypotheses of the experiment are two-fold:

Table 5-3. Research Hypotheses for Effect on Trust.

Trust Measure	Null Hypothesis (H_0)	Alternative Hypothesis (H_a)
Specific Inter-personal Trust	Using Theseus does not result in higher specific inter-personal trust compared with not using Theseus.	Using Theseus results in higher specific inter-personal trust compared with not using Theseus.
Attribution Type	Using Theseus does not result in more situational attributions compared with not using Theseus.	Using Theseus results in more situational attributions compared with not using Theseus.

Based on the above conditions and hypotheses, the experiment has the following additional research hypotheses:

- ***HYPOTHESIS 1-1:*** *Specific inter-personal trust scores are higher in Theseus situational (b) than No Theseus (a).*
- ***HYPOTHESIS 1-2:*** *Specific inter-personal trust scores are higher in Theseus situational (b) than Theseus moderate situational (c).*
- ***HYPOTHESIS 1-3:*** *Specific inter-personal trust scores are lower in Theseus dispositional (d) than both Theseus situational (b) and Theseus moderate situational (c).*

- ***HYPOTHESIS 2-1:*** *Attributions are more situational in Theseus situational (b) than No Theseus (a).*
- ***HYPOTHESIS 2-2:*** *Attributions are more situational in Theseus situational (b) than Theseus moderate situational (c).*
- ***HYPOTHESIS 2-3:*** *Attributions are more dispositional in Theseus dispositional (d) than both Theseus situational (b) and Theseus moderate situational (c).*

In sum, the conceptual goal of the experiment is to show that Theseus can usefully support the development of trust. The experiment measures whether Theseus is usable and whether it has utility, i.e., whether it can influence participants' perceived trustworthiness

toward distributed team members. As such, three dependent variables are measured: usability, trust, and insights. Usability is assessed by analyzing video recordings of participants and calculating usability scores from a questionnaire given at the end of the experiment. Trust is assessed at several points in the experiment, always after the subject's completion of a task that involves making attributions about a team member's failure to deliver. Trust was measured using standard trust instruments previously developed by other researchers, as well as classifying participants' attribution rankings as either situational or dispositional. Insights were measured by analyzing video recordings of participants according to the format of the insight-based evaluation.

5.2.2 Experiment Design

The experiment followed a within-subjects (repeated measures) design with one independent variable and four dependent variables:

Independent Variable:

1. Theseus intervention, two levels:
    a. Present (three experimental conditions)
    b. Absent (one condition)

Dependent Variables:

1 Usability scores
2 Trust scores
3 Attribution scores
4 Insight characteristics

The experiment consisted of three tasks, and followed the formats of the insight-based evaluation (Saraiya et al. 2005) and think aloud protocol (Lewis, 1982) techniques, whereby human subjects use a software tool and think aloud, discovering insights, as they perform a set of specified tasks. Users are asked to say whatever they are looking at, thinking, doing, and feeling, as they go about their task.

5.2.2.1 Participants

The sample size required for the experiment was *N*=40 subjects. The number was estimated by using comparable experiments as guidelines as well as statistical rules of-thumb to demonstrate statistical significance. Laboratory experiments of tools that provide awareness visualizations to software developers range from as little as *N*=1 to *N*=26. For example, Sarma's Palantir experiment used *N*=26 participants (Sarma et al., 2008) while the evaluation of a subsequent tool used *N*=5 participants (Sarma et al., 2009). Saraiya's evaluation of visual interfaces using the insight-based evaluation, from which the evaluation in this document is based, used *N*=30 for 2 control variables and 7 dependent variables in a 3 x 5 experimental setup (Saraiya et al., 2005).

A rule of thumb is 10 subjects per construct (Chin and Newsted, 1999). The experiment in this dissertation has a total of 4 constructs (1 independent variable and 3 dependent variables). As such 10 x 4 = roughly 40 subjects were determined to be adequate to demonstrate significance. This number is comparable with sizes in related studies shown above.

Forty (40) human subjects were recruited from the University of California, Irvine as well as a software development company nearby campus. The recruitment process consisted of posting fliers around the university's computer science buildings, sending e-mails to computer science department mailing lists, and arranging meetings with professional software developers through two contacts at local software companies in Irvine, California. Twelve (12) participants were professional software developers while the other twenty-eight (28) were graduate and undergraduate students studying computer science.

Only males were eligible to participate in this study for two reasons. First, various studies have shown there are interaction effects of gender of trustor and gender of trustee on trust (e.g., Johnson-George and Swap, 1982; Jeanquart-Barone, 1993; Chaudhuri and

Gangadharan, 2003; Maddux and Brewer, 2005; Sun et al., 2006; Buchan et al., 2008). Second, this experiment only tests a proof-of-concept, and does not seek to make claims about gender. Thus, we conclude that only one gender is necessary and we choose to include males only.

The average participant age was 27 years (median 27 years), ranging from 21 to 37 years. The average years experience working in distributed software development was 3 years (median 3 years), ranging from 0 years to 8 years. Nineteen participants (42%) were Caucasian, thirteen participants (27%) were Asian (Indian), eight (20%) were Asian (Chinese), one (3%) was Hispanic, one (3%) was African-American, one (3%) indicated they were both Asian and Hispanic , and one (2%) indicated they were Caucasian and Hispanic.

The highest level of education was graduate degree for 22 subjects, undergraduate degree for 17 subjects, and high school diploma for 1 subject.

5.2.2.2 Protocol and Measures

This section details the process by which the laboratory experiment was performed. For step-by-step instructions, see the Experiment Protocol Summary (Appendix 1.).

The average duration of the study was 78 minutes per subject. The study took place in the Hana Lab of the 6th floor of the Donald Bren Information and Computer Sciences building and on location at a software development company in Irvine, CA. The experiment was performed on a Dell Studio XPS laptop with an Intel Core 2 Duo 2.4 GHz processor, 4 GB RAM, and Windows 7 OS.

Before the first task, subjects were required to complete a demographics survey. The purpose of this survey was to potentially correlate any demographic factors with the participants' trust ratings in the subsequent tasks. In addition, participants completed a trust questionnaire (Rotter, 1967) (Appendix 2). Items in the scale deal with social agents including parents, salespeople, the judiciary, people in general, political figures, and members

of the news media. The scale gives an indication of the participant's general propensity to trust others, which were thought to potentially correlate with their trust scores in the study tasks.

Afterward, subjects began Task 1. First, they received the name of a distributed developer on their team in addition to the task 1 scenario:

> "You have to come into the office this weekend to work on the 'MIRTH' project. **Victor Ward**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule. Which explanation best describes why he was unable to deliver on time?"

Subjects received a printed list of possible explanations, or attributions, with instructions to rank them from top to bottom, from most likely reason to least likely reason. This produced a ranked list of attributions that was scored, in the analysis phase, to produce a single value that is either more situational (high quality attribution) or dispositional (low quality attribution) (Appendix 14).

Then, subjects were asked to fill out a standard specific inter-personal trust instrument (Johnson-George and Swap, 1982), which was scored to produce a value indicating their trust toward that person in particular (Appendix 4). Thus the purpose of this task was to yield a measure of trust, based on attributions as well as responses to previously validated trust instruments, without the intervention, i.e., Theseus.

On completion of this task, subjects were given a 5 minute break, after which they were taken into a new room. Subjects then received training on Theseus (Appendix 5). Then they began Task 2, their first task with the tool (Appendix 6):

> "From the following list of developers, use Theseus to identify who you believe to be most available to reply to e-mail. Identify who is least so. As you use the tool, think aloud, mention individual observations about each individual, and justify your selections."

Task 2 was intended to act as a training period for subjects, in which they started vocalizing insights and observations made while using the interface.

After completing Task 2, subjects begin the third task, which is composed of evaluating three different subtasks (Appendices 7-12). In each sub-task, the subject ranks a different developer. Subjects are expected to make a highly situational attribution for the first developer, a moderately situational attribution for the second developer, and a dispositional attribution for the third developer. The instructions for the task are essentially the same as Task 1, except that the subject has the intervention (i.e., Theseus).

> "You have to come into the office this weekend to work on the 'MIRTH' project. **[Developer Name]**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule.
>
> Use Theseus to determine which explanation most likely describes why **[Developer Name]** was unable to deliver on time:"

Again, subjects were told to think aloud and verbalize their observations with the tool. For each of the three developers, the subject ranked a list of attributions from most likely to least likely (the same list from Task 1). Again, I scored the attribution list. After the subject ranked the list of attributions, they were asked to rate their trust level toward the developer, using the same questionnaire as in Task 1. The purpose of Tasks 3a, 3b, and 3c are to measure trust *with the intervention*, as opposed to without it (i.e., Task 1). Task 1 and Tasks 3a, 3b, and 3c correspond to the No Theseus, Theseus situational, Theseus moderate, and Theseus dispositional conditions, respectively.

Tasks 2 and 3 were counterbalanced with one group (N=20) receiving task 2 first and task 3 second, and the other group (N=20) receiving task 3 first and task 2 second. The reason for this ordering was to minimize practice effects throughout the tasks.

Following the protocol of both the insight-based evaluation and think aloud procedures, during tasks 2 and 3, I asked subjects to comment on observations, inferences, and conclusions they made about the distributed developers in question. The subjects examined the data with the tool until they felt they could not gain more insight. The insight characteristics that were coded by me during this time are shown in Table 5-3. Subjects were allowed to ask clarifications on task instructions and features of the tool. However, it was made very clear to participants that other questions could not be answered, lest bias be introduced into the experiment.

After subjects completed the last task, they assessed their overall experience with the tool by completing a post-experiment questionnaire (Appendix 13). This questionnaire included questions about usability of the tool as well as questions about what data and visual representations are appropriate for tools to support trust. Upon completion of the questionnaire, I debriefed the participant in a short interview period, asking the latter to provide rationale for some of their responses. The purpose of this debriefing session was to collect additional qualitative information about participants' insights, trust rankings, and overall experience.

5.2.3 Pilot Experiment

A pilot run of the experiment was conducted on April 2, 2012 with 1 human subject with the purpose of diagnosing and correcting any remaining usability issues before deploying to the subject pool, as well as to test whether the results for trust and attributions matched the experiment hypotheses.

Based on the usability results, I modified the average reply time visualization to take up less space in the interface relative to the other visualizations. I also modified the contact list to group the contacts by the projects to which they belong. My human subject made this suggestion because he said it matched his model of organizing his own contacts by project or

affiliations (e.g., “friends” and “co-workers” on a buddy list). The “select location” drop down list was removed for the purposes of the experiment, because using a single location for each participant is sufficient. In practice, the “select location” feature would presumably be useful, since users of the tool could be located anywhere in the world.

The subject’s trust scores and attribution scores in each condition matched the research hypotheses. Their trust score was higher in the situational conditions than in the dispositional condition, and their attribution score was more situational in the situational conditions than in the dispositional condition. The total time of the experiment was 74 minutes, 16 minutes short of the expected running time.

After the appropriate modifications were made to Theseus’ interface, human subjects were recruited and subsequently scheduled. The experiments began shortly after, and concluded on August 3rd, 2012.

5.2.4 Full Experiment Results and Analysis

Data analysis was conducted using quantitative and qualitative methods. All statistical tests were conducted in R 2.15.1 using the R-Studio software (http://www.rstudio.com). Results for the constructs *usability*, *trust*, and *insight* are presented in the following subsections:

- **Usability** 5.2.4.1
- **Trust** 5.2.4.2
- **Insight** 5.2.4.3

5.2.4.1 Usability

Our results indicate that participants found this version of Theseus usable. Table 5-4 shows the median, mean, and standard deviation for individual items in the usability questionnaire as well as the categories. The scale shown in Table 5-4 comes from the original survey instrument (Lewis, 1993), where scores range from 1 (strongly agree) to 5 (strongly

disagree) with lower being better. Table 5-6 aggregates these scores into a total overall usability score.

Table 5-4. Usability Statistics. Lower scores indicate higher usability.

Question	Median Rating	Mean Rating	Standard Deviation
Usefulness (1=strongly agree; 5=strongly disagree)	**1.77**	**1.8**	**0.30**
Overall, I am satisfied with how easy it is to use this system.	2.0	1.8	0.53
It was simple to use this system.	2.0	1.6	0.54
I could effectively complete the tasks and scenarios using this system.	2.0	1.7	0.51
I was able to efficiently complete the tasks and scenarios using this system.	2.0	1.8	0.44
I was able to complete the tasks and scenarios quickly using this system.	2.0	1.7	0.51
I felt comfortable using this system.	2.0	1.8	0.48
It was easy to learn to use this system.	2.0	1.6	0.50
I believe I could become productive quickly using this system.	2.0	1.8	0.68
My interaction with the system is clear and understandable.	2.0	1.8	0.38
Interacting with the system does not require a lot of my mental effort.	2.0	2.1	1.02
Overall, I am satisfied with this system.	2.0	2.0	0.60
Information Quality (1=strongly agree; 5=strongly disagree)	**2.07**	**2.0**	**0.39**
The system gave error messages that clearly told me how to fix problems.	3.0	2.5	0.82
Whenever I made a mistake using the system, I could recover easily and quickly.	2.0	2.1	0.92
The information (such as on-line help, on screen messages and other documentation) provided with this system was clear.	2.0	1.8	0.78
It was easy to find the information I needed.	2.0	1.8	0.49
The information provided for the system was easy to understand.	2.0	1.9	0.69
The information was effective in helping me complete the tasks and scenarios.	2.0	1.9	0.52
The organization of information on the system screens was clear.	2.0	2.0	0.77

Interface Quality (1=strongly agree; 5=strongly disagree)	**2.33**	**2.4**	**0.73**
The interface of this system was pleasant.	2.0	2.2	0.88
I liked using the interface of this system.	2.0	2.1	0.85
The system has all the functions and capabilities I expect it to have.	3.0	2.9	0.96
Intention to Use (1=strongly agree; 5=strongly disagree)	**2.0**	**2.6**	**0.97**
Assuming I have access to the system, I intend to use it.	2.0	2.6	0.96
Given that I have access to the system, I predict that I would use it.	2.0	2.6	1.0

One point of concern with these results is that the numbers of questionnaire items in each category are not equal. For instance, there are 11 items referring to usefulness, while only3 items refer to the interface quality. As such, Theseus' overall usability score could be unfairly biased toward usefulness, over interface quality.

With this issue in mind, an exploratory factor analysis using all Likert scale questions except those in "Intention to Use" (21 in total) was conducted to confirm the groupings (Lewis, 1993) used above. Exploratory factor analysis (EFA) is a technique within factor analysis that researchers use to uncover relationships between measured variables and identify a set of latent constructs among them. Researchers also commonly use EFA to develop scales for questionnaires. In this analysis, oblique rotation was used because it permits factors to be correlated. For instance, usefulness of the system could be related to the quality of the interface. Results of the EFA suggest an alternate grouping of the Likert items based on *interface quality*, *learnability*, and *efficiency*.

Table 5-5. Obliquely rotated component loadings for 21 survey items.

Component	1	2	3
Satisfied with Theseus	0.58	-0.18	0.10
Effective Info	0.65	0.05	0.32
Clear Organization	0.60	0.31	-0.13
Pleasant Interface	0.77	0.36	-0.08
Like Interface	0.83	0.21	0.07
Satisfied with Ease	0.55	0.70	0.09
Simple to Use	0.13	0.97	0.20
Comfortable to Use	0.56	0.57	0.16
Easy to Learn	0.15	0.64	0.21
Information Effective	0.13	0.19	0.82
Efficient Use	0.22	0.17	0.77
Complete Quickly	0.15	0.04	0.60
Become Productive	0.15		0.03
Clear Interaction	0.39	0.17	-0.22
Mentally Demanding	-0.06	0.14	0.06
Clearly Fix Problems	-0.13	-0.30	0.31
Recover Easily	-0.03	0.08	0.17
Help is Clear	0.27	0.03	0.05
Easy to Find Info	0.48	0.28	0.35
Tool Info Easy	0.21	0.27	0.32
Tool Has all Functions	0.40	-0.02	0.22
SS Loadings	3.83	2.84	
Proportion Variance	0.18	0.14	0.11
Cumulative Variance	0.18	0.32	0.43

Ten items loaded onto Factor 1. It is clear from Table 5-5 that the items relate to being satisfied with the overall interface and to what extent one can be productive with it. This factor loads onto the reported level of overall satisfaction with the system, the effectiveness of the information presented, the clarity of the organization of the information, whether participants thought the interface was pleasant, whether participants liked the interface, whether subjects thought they could become productive with the interface, whether participants found their interaction with the system clear, whether they found the help and documentation clear, whether it was easy to find information they needed, and whether the system had all the functionalities they expected it to have.

Five items loaded onto Factor 2. Table 5-5 shows the items relate to the learnability of Theseus. This factor loads onto the reported level of satisfaction with how easy the tool is to use, the simplicity of using the tool, whether subjects are comfortable using the system, the ease with which the system can be learned, and whether using the system requires a lot of mental effort.

Six items loaded onto Factor 3. The items relate to the efficiency with which participants reported they could use the tool. This factor loads onto whether subjects could effectively complete tasks with the tool, how efficiently they could complete the tasks, how quickly they could complete the tasks, whether the tool clearly told them how to fix errors, how easily participants could recover from errors, and whether the information provided for the system was easy to understand.

The results of the factor analysis indicate different groupings for the usability items. However, the groupings both produce the same overall usability score. As such, the original groupings were deemed sufficient to characterize the usability of the tool. Performing the factor analysis is beneficial because it shows two additional dimensions of Theseus' usability: learnability (Mean=1.76; SD=0.42) and efficiency (Mean=1.95; SD=0.39). The interface quality was also good (Mean=2.02; SD=0.41).

The **overall usability score** for each participant was calculated by summing all 21 responses to items from the questionnaire. Table 5-6 below reports the descriptive statistics for overall usability scores. Lower scores indicate greater usability. **Potential scores range from 21 (highest usability) to 105 (lowest usability), with a neutral score or midpoint of 63.**

Table 5-6. Descriptive Statistics for 'Overall Usability' Scores.

Min.	29.00
1st Quartile	41.50
Median	46.00
Mean	45.52
3rd Quartile	50.25
Max	56.00
Std. dev	7.05

The data show that usability scores ranged from 29 to 56, with most scores between 41.50 and 50.25. The median was 46 and the mean was 45.52, both below the midpoint score of 63. The maximum score out of N=40 subjects was 56, but even this score was 7 points below the midpoint and thus still on the usable side. Taken together with the responses to the usability Likert items and subject feedback, the results suggest that Theseus is highly usable.

One of the biggest concerns revealed from subject feedback is that, despite efforts to make it easier to compare developers using the drag-n-drop metaphor, Theseus still needs a better mechanism to show multiple developers at once. This result was reflected in answers to the question about the user interface being "pleasant to use" as well as "liking the user interface." In the video logs subjects reported various problems during Task 2 concerning the existing mechanism in the interface, a table that displays each developer in a new row. When more than 3 developers are displayed, a scrollbar appears to allow scrolling through contacts. Subjects found the scrolling unpleasant, especially because information about the fourth and fifth developer was cut off at the bottom of the screen. Some subjects, especially three of the software developers interviewed (P-29, P-32, P-34) indicated the drag and drop functionality helped, but they would like to see a more "polished" way to arrange contacts.

Several subjects also indicated that it would be useful to compare sections of the project pie charts more easily across subjects. It was mentally demanding to hover over one developer's pie, leave that pie piece, hover over the other developer's pie, and then mentally

compare the work items. This is reflected in answers to the question about mentally demanding operations with Theseus.

Another interesting result has to do with the "intention to use" portion of the questionnaire. In general, software developers indicated they would not use the tool because at their company, all team members are located in the office. There are no teams at any other locations or in other time zones. Despite this, one subject indicated that in his last job where everyone was a telecommuter, "Theseus would have been invaluable" (P-29). Other participants said that they would not likely use the tool because they believed the tool to be more useful for a project manager who needs to make sure the members of their team are available and responsive on the project.

Another concern was that some subjects felt the tool was missing important information. This concern was reflected in the "Interface Quality" category for answers to the question about the system "having all of the capabilities one expects it to have." For example, some subjects indicated that a metric for work items that would be useful to see was the estimated time to completion for the work item. Their rationale was that a developer may have many work items open and unresolved, but if those work items are small changes, the developer is not as busy as if all of the work items were large changes.

Similarly, several subjects indicated that a temporal view of work item assignments and resolutions would be useful for determining each contact's individual behavior and productivity. A developer who regularly delivers on time would be more easily identifiable in such a visualization, they posited.

Summary

Theseus' usability was assessed based on data collected from questionnaires, video log footage, and interviews that took place after the experiment concluded. The usability score statistics show that overall, the tool is on the usable side. An exploratory factor analysis

of categories in the usability questionnaire suggested different groupings for the categories, but the usability score calculated from those alternate groupings was not significantly different than the original. Analysis of video indicated that some subjects became frustrated with certain aspects of the user interface, such as scrolling to compare developers. Interviews with subjects indicated that they believed certain information was missing from the tool, and these concerns were reflected in their responses to the questionnaires.

Whereas this sub-section presented results on the *usability* of Theseus, the next two sub-sections assess its *utility*. Each differs in its approach. The first sub-section quantitatively analyzes whether Theseus influences perceived trustworthiness; the second sub-section takes a qualitative approach toward measuring insights generated with the tool.

5.2.4.2 Trust

This section presents the quantitative analysis of inter-personal trust scores and attribution scores. Each sub-section presents a graph illustrating the standard deviation of the scores in each of the four conditions. The statistical null hypotheses for inter-personal trust scores and attribution scores appear alongside each respective graph. The *statistical* hypothesis asks whether there is a main overall significant effect, i.e., whether any significant difference exists. This is what the result of an ANOVA, for instance, would indicate. The *research* hypotheses, in contrast, claim where the significant differences are. Answering the research questions require performing post-hoc comparison tests.

Inter-Personal Trust Scores

In each condition, participants filled out a questionnaire that asked them about the developer they just evaluated. Their answers produced an inter-personal trust score that ranged from 15 (low trust) to 75 (high trust), with a neutral score or midpoint of 45. Using the presence or absence of Theseus as the experimental control yields the following statistical null hypothesis:

> **H_o (statistical null hypothesis):** Using Theseus does not result in any difference in specific inter-personal trust compared with not using Theseus.

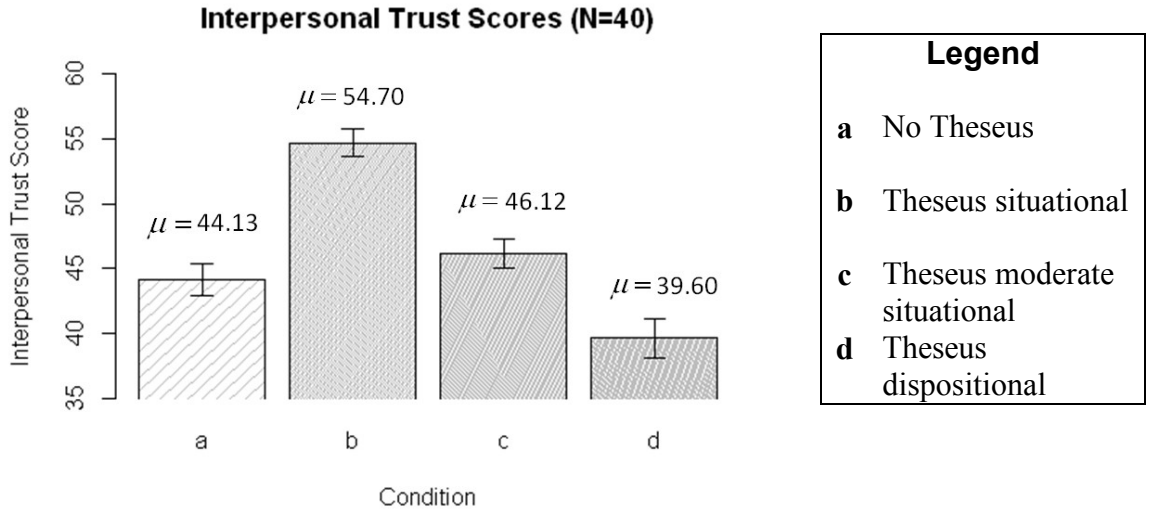


Figure 5-7. Standard Deviation of Inter-personal Trust Scores.

The statistical null hypothesis addresses inter-personal trust scores with Theseus and without Theseus. Because this experiment follows a within-subjects design, a repeated measures test is appropriate. In addition, the data follows the normal distribution. As such, a one-way repeated measures ANOVA was conducted to compare the effect of Theseus on trust scores calculated from results of the specific inter-personal trust questionnaires in *No Theseus (a)*, *Theseus situational (b)*, *Theseus moderate situational (c)*, and *Theseus dispositional (d)* conditions. These conditions are shown in Figure 5-7 on the horizontal axis as *a*, *b*, *c*, and *d*, respectively.

The analysis shows there was a significant effect of Theseus on inter-personal trust score [$F(3, 117) = 27.03$, $p<0.001$, partial $\eta^2=0.41$]. Because the result is significant, post-hoc comparison tests were conducted to verify the research hypotheses:

- ***HYPOTHESIS 1-1:*** *Specific inter-personal trust scores are higher in Theseus situational (b) than No Theseus (a).*
- ***HYPOTHESIS 1-2:*** *Specific inter-personal trust scores are higher in Theseus situational (b) than Theseus moderate situational (c).*

- ***HYPOTHESIS 1-3:*** *Specific inter-personal trust scores are lower in Theseus dispositional (d) than both Theseus situational (b) and Theseus moderate situational (c)*

H1-1

There is significant support for H1-1. Post-hoc comparisons using the Tukey HSD test[1] indicated that the mean score for the *Theseus situational (b)* condition (Mean=54.70, SD=6.52) was significantly higher than the mean score in the *No Theseus (a)* condition (Mean=44.13, SD=7.93, p=0).

H1-2

The comparison shows significant support for H1-2. The mean score for the *Theseus moderate situational (c)* condition (Mean=46.12, SD=6.99) was not significantly different from the mean in the *No Theseus (a)* condition (Mean=44.13, SD=7.93, p=0.696). Because the mean in the *Theseus situational (b)* condition is higher than in the *No Theseus (a)* condition, we can infer that the mean in the *Theseus situational (b)* condition is significantly higher than in the *Theseus moderate situational (c)* condition.

H1-3

The results provide partial support for H1-3. The mean score for the *Theseus dispositional (d)* condition (Mean=39.60, SD=9.71) was not significantly different than the mean in the *No Theseus (a)* condition (Mean=44.13, SD=7.93, p=0.081), but it approached significance at the $p<0.1$ level. Because the mean is significantly higher in the *Theseus situational (b)* condition than in the *No Theseus (a)* condition, we can infer that indeed, the

[1] The Tukey HSD test is typically used in conjunction with ANOVA to find means that are significantly different from each other. It is essentially a t-test, except that it corrects for experiment-wise error rate. When a statistical test makes multiple comparisons, the probability of making a type I error (rejecting a true null hypothesis) increases. The Tukey test corrects for that, and is thus more suitable for multiple comparisons than conducting a number of t-tests would be. The Tukey test also has good control on type II errors (failing to reject a false null hypothesis) when the sample size is equal. In this experiment, the group size is N=40 for all conditions. Therefore, a Tukey test is appropriate.

mean in the *Theseus dispositional (d)* condition is significantly lower than in the *Theseus situational (b)* condition. However, because there is only weak support for a difference in means between the *No Theseus (a)* and *Theseus dispositional (d)* conditions, we cannot conclude that the mean in the *Theseus dispositional (d)* condition is significantly lower than in the *Theseus moderate situational(c)* condition.

Summary

Taken together, the results suggest that the presence of Theseus really does have a significant effect on users' perceived trustworthiness toward globally distributed developers. The ANOVA showed that we can reject the statistical hypothesis H_0, which stated there was no difference in mean between any of the four conditions. Specifically, the post-hoc comparisons show that there is support for our research hypothesis H1-1; Theseus significantly increases inter-personal trust compared with using no tool at all.

Whereas this sub-section presented the quantitative analysis of inter-personal trust scores, the next sub-section presents the results of the analysis of the attribution scores. Again, results for the overall statistical effect appear before the results of the post-hoc comparisons and answers to the research hypotheses.

Attribution Scores

In each condition, participants ranked a list of explanations for why a distributed team member failed to deliver their code on time, from most likely to least likely. Their answers produced an attribution score that ranged from -7 (highly situational) to 7 (highly dispositional), with a neutral score or midpoint of 0. Using the presence or absence of Theseus as the experimental control yields the following statistical null hypothesis:

> **H_0 (statistical null hypothesis):** Using Theseus does not result in any difference in types of attributions compared with not using Theseus.

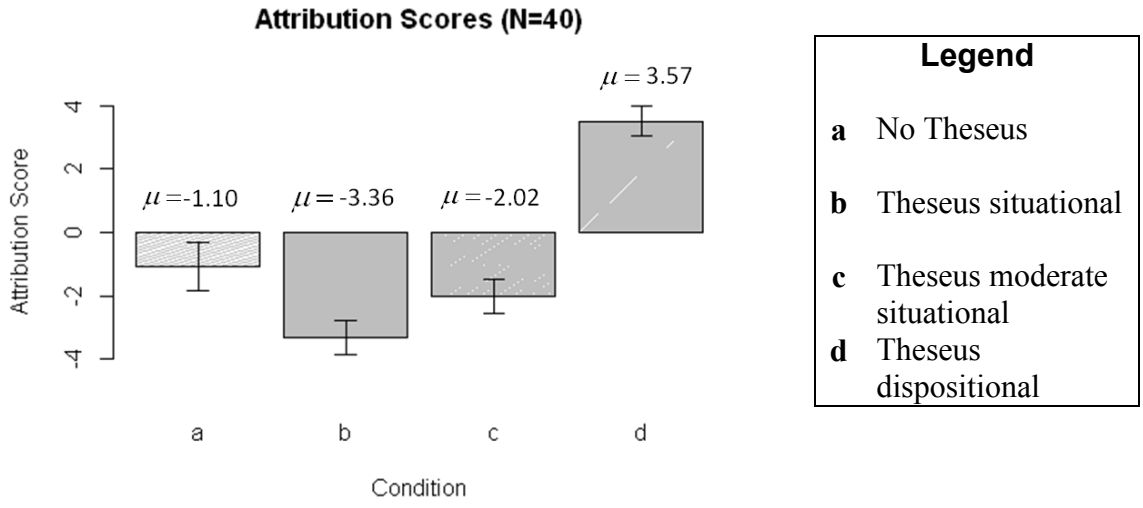


Figure 5-8. Standard Deviation of Attribution Scores.

The statistical null hypothesis has to do with attribution types with Theseus and without Theseus. Because this experiment follows a within-subjects design, a repeated measures test is appropriate. In addition, the data follows the normal distribution. As such, a one-way repeated measures ANOVA was conducted to compare the effect of Theseus on attribution types in *no Theseus*, *Theseus situational*, *Theseus moderate situational*, and *Theseus dispositional* conditions. These conditions are shown in Figure 5-8 as *a*, *b*, *c*, and *d*, respectively.

The data analysis indicates there is a significant effect of Theseus on attribution type for the four conditions [$F(3, 117) = 25.96$, $p<0.001$, partial $\eta^2=0.40$]. In order to verify the research hypotheses, we performed post-hoc comparisons to determine between which conditions the effect exists.

- ***HYPOTHESIS 2-1:*** *Attributions are more situational in Theseus situational (b) than No Theseus (a).*
- ***HYPOTHESIS 2-2:*** *Attributions are more situational in Theseus situational (b) than Theseus moderate situational (c).*
- ***HYPOTHESIS 2-3:*** *Attributions are more dispositional in Theseus dispositional (d) than both Theseus situational (b) and Theseus moderate situational (c).*

Again, to determine where the significance exists, we conducted the Tukey HSD test for the same reasons as above.

H2-1

There is statistical support for H2-1. Post-hoc comparisons using the Tukey HSD test indicated that the mean score for the *Theseus situational (b)* condition (Mean=-3.36, SD =3.43) was marginally lower (more situational) than the mean in the *No Theseus (a)* condition (Mean=-1.1, SD=4.82, p=0.052). However, the result is not strongly significant.

H2-2

There is also support for H2-2. The mean score for the *Theseus moderate situational (c)* condition (Mean=-2.02, SD=3.40) was not significantly lower than in the *No Theseus (a)* condition (Mean=-1.1, SD=4.82, p=0.704). Taken together with the fact that the *Theseus situational (b)* mean was marginally different from the *No Theseus (a)* mean, we can infer that there is also a marginally significant difference between mean attribution scores in *Theseus situational (b)* and *Theseus moderate situational (c)* conditions.

H2-3

The analysis shows significant support for H2-3. The mean score for the *Theseus dispositional (d)* condition (Mean=3.57, SD=2.99) was significantly higher (more dispositional) than the mean in the *Theseus situational (b)* condition (Mean=-2.02, SD=3.40, p=0).The mean score for the *Theseus dispositional (d)* condition (Mean=3.57, SD=2.99) was also significantly higher (more dispositional) than in the *No Theseus (a)* condition (Mean=-1.1, SD=4.82, p=0). Because there is no significant difference in means between *Theseus moderate situational (c)* and *No Theseus (a)* conditions on attributions (by H2-2), we can infer that the mean is significantly more dispositional in the *Theseus dispositional (d)* condition compared with the Theseus *moderate situational (c)* condition.

Summary

In sum, the data analysis suggests that the presence of Theseus does have a significant effect on the types of attributions distributed team members make about their colleagues. The results of the ANOVA suggest that we can reject the statistical null hypothesis, H_0, which states there is no difference between attribution scores in any of the four conditions. The post-hoc comparisons show marginally significant evidence to support our research hypothesis (H2-1), that using Theseus results in more situational attributions compared with using no tool.

While this sub-section presented a quantitative analysis of trust scores with and without Theseus, the next sub-section takes a qualitative approach toward assessing Theseus's utility. It presents the results from the insight-based portion of the experiment, which complement the quantitative results.

5.2.4.3 Insight

Following the insight-based evaluation (Saraiya et al., 2005), several insight characteristics were measured for each participant. Recall that insights are individual observations about data by the participant, units of discovery. During each task with Theseus, participants were instructed to think aloud their observations about the developers in the interface. *Moare Recorder*, a commercial usability testing software package was used to capture these observations, as well as document usability problems. After all experiments were complete, I analyzed each participant's recording and tabulated all insights for each task. Because the insight-based evaluation is more qualitative than quantitative; a comparison of general tendencies is appropriate when presenting the results.

Number of Insights

By watching the video logs of participants, I counted the number of total insights about the developers in each task for each participant. Because every participant completed each task, there are insights common across participants. Figure 5-9 shows that the total number of insights was highest for Task 2 and lowest for Task 3c. A possible explanation for this is that in Task 2, developers compared four different developers, compared to one developer in each of the sub-tasks of Task 3. As such, they observed more information.

An interesting explanation for the relative difference in number of insights between Task 3a and Task 3b could be that subjects' confidence toward making dispositional attributions was higher in Task 3b. Research has shown that subjects become more confident making dispositional attributions with the passage of time (Frank and Gilovich, 1989). In Task 3c, subjects felt most comfortable giving dispositional attributions, needing fewer insights to do so.

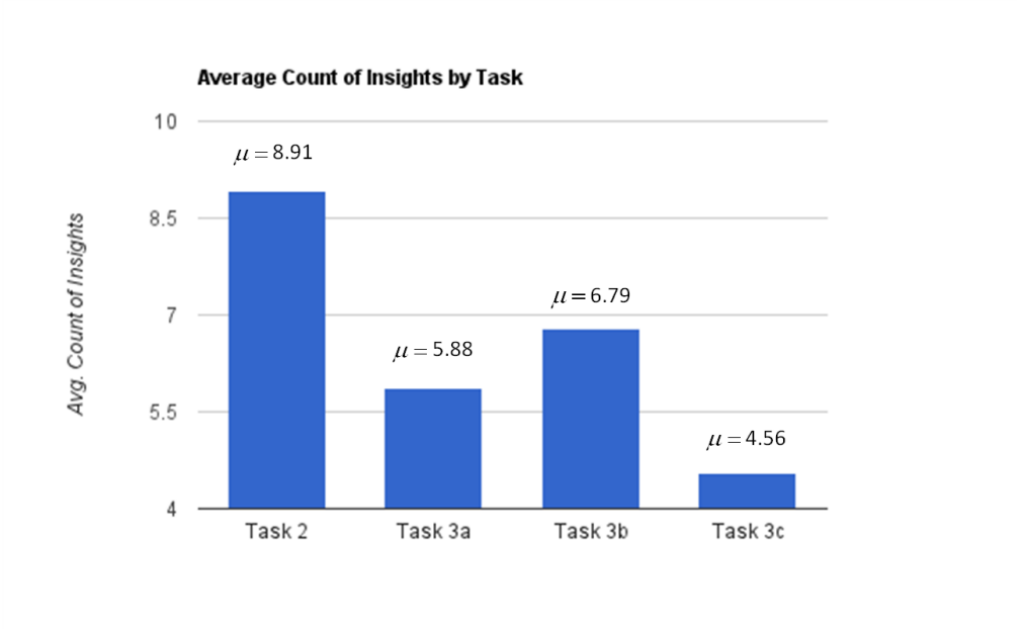


Figure 5-9. Average number of insights per task.

Table 5-7 below shows six example representative insights for each task. Although different participants articulated insights in different ways, the essence of each insight was the same, i.e. it had the same overall meaning.

Table 5-7. Example insights made by participants in each task.

Task 2	Task 3a	Task 3b	Task 3c
Jim seems quite busy in 3 projects	Barney is mostly on my project, he's very involved in it and is in charge of some critical work items. He's still very responsive despite all that.	Our time of day overlap is virtually non-existent; there's one golden hour.	The response time is unusually long, he's a little bit funky for being in my same location.
It appears that Jim spends most of the time in my project, seems engaged in the project	He's all the way in Israel, but he seems to respond quite quickly.	He is probably e-mailing during other times in the day	He seems like a very "9-5" type of person.
His response seems about average, he's pretty organized and I see his rhythm	Barney is on the ball, he responds to e-mails on Saturday and Sunday	He doesn't appear to be in the office much.	He doesn't seem like he's willing to allocate time for this project.
Lloyd is least available, he's spread	The linear reply scale shows that he is	He is involved in many projects and is	He's not on many projects, and given

out and kind of erratic. He's also on multiple projects	faster to respond than most people	surely distracted	his low work item count, I don't think he's working on too many tasks
Albert seems disciplined; he has a stable rhythm	Barney is working on many critical items but they are still open; he's probably not very good at resolving them.	His response time is crazy long, even considering the time zone, not a good sign.	I can't trust this developer to reply on time if the situation was urgent.
There's no overlap in our time to meet	Barney spends most of the time in my project so I think he will be responsive to me.	He's an e-mail-a-holic, works outside normal work hours, so I believe he's getting things done.	He is not working on any urgent tasks, yet his availability is poor.

Average Time to First Insight

The average time first insight was recorded for each task. A shorter time to first insight suggests that the subject is able to get immersed into the tool and the data more quickly. Shorter times are also indicative of fast learning times of the user interface. The data show that in general, average time to first insight is shortest for Task 3c and longest for Task 3a. In general, average time to first insight decreased with subsequent tasks.

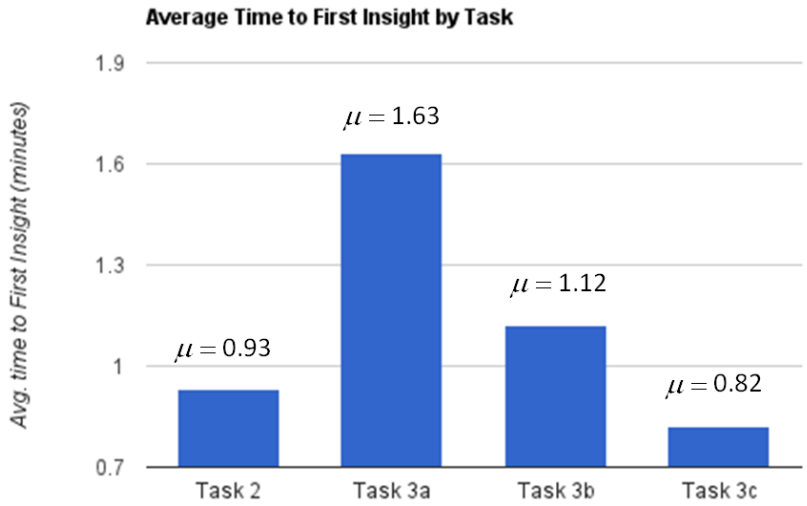


Figure 5-10. Average time (in minutes) to first insight for each task.

One possible explanation is that users gradually became more proficient using the tool after the completion of each task, and thus could find the relevant information sooner. Because the tasks in 3a, 3b, and 3c are all similar except for the developer being evaluated, it is likely that participants were able to apply knowledge of the interface gained in Task 3a to Tasks 3b and 3c.

Average Total Time

This characteristic is a measure of the average total time participants took with each task until they felt they had finished. Lower times either suggest a more efficient tool, or that subjects felt they could not gain much insight. The average total time was lowest for Task 3c and highest for Task 3a. Again, because Task 3c is at the end of a series of similar tasks, it makes sense that participants were familiar enough with the tool at that point to complete the task

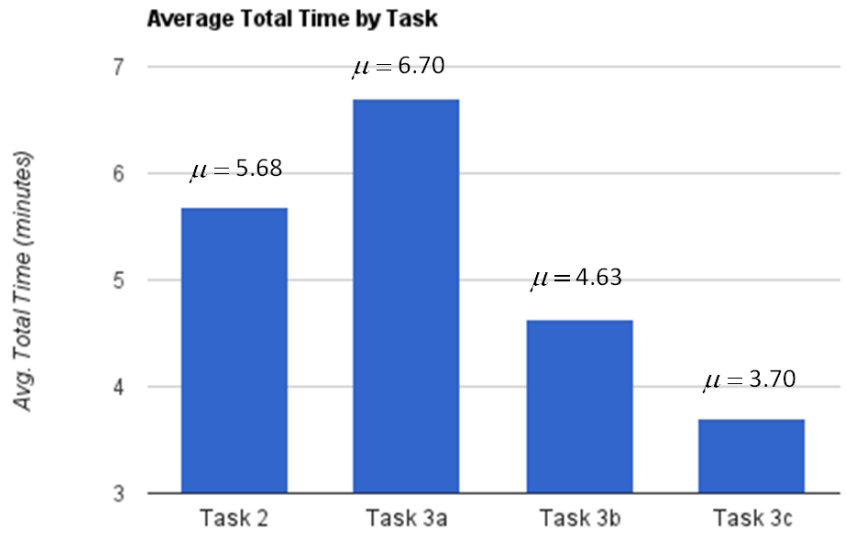


Figure 5-11. The average total time (in minutes) per task.

more quickly than 3b or a. An explanation for the long time on Task 3a is that it was the first task in which subjects were asked to do the attribution ranking. It is curious that Task 2 did not take as long as Task 3a, since subjects were asked to evaluate four developers instead of one. One possible reason for this could be that once the four developers were loaded into the interface, it was easy to judge the developers relative to one another, resulting in more insights in a shorter amount of time. Indeed, Figure 5-9 shows that the most insights were generated in Task 2.

Incorrect Insights

Incorrect insights are observations that are not true. They come from incorrect interpretations of the visualizations in the user interface. Incorrect insights were generally rare, but they did occur. Table 5-8 lists the total number of incorrect insights per task.

Table 5-8. Total Incorrect Insights.

Task	Incorrect Insights
Task 2	9 (out of 303)
Task 3a	6 (out of 200)
Task 3b	8 (out of 231)
Task 3c	7 (out of 155)

One example pertains to the linear scale e-mail responsiveness visualization, in which a blue dot on a linear scale, by distance from the center, indicates how much faster (to the left) or how much slower (to the right), a developer is than the average response time. Three participants incorrectly interpreted a fast response as a slow response, and vice a versa.

Insights Over Time

Tracking insights over the course of each task enables the examination of how insights accumulate over time. The graph below shows insight curves for the actual insight counts. In essence, the graph shows the rate of insight generation using Theseus.

Figure 5-12 illustrates the average accumulation of insight over time. Recall that during the course of each task, participants were asked to comment on individual observations that make about the developers shown in the interface.

Figure 5-12 shows several interesting trends. First, participants took longest to reach the first insight in Task 3a, the first task in which they evaluate a single developer. In Task 3b, participants steadily gained up to 8 insights before they could not gain any more. This leveling off happened earlier in Task 3c.

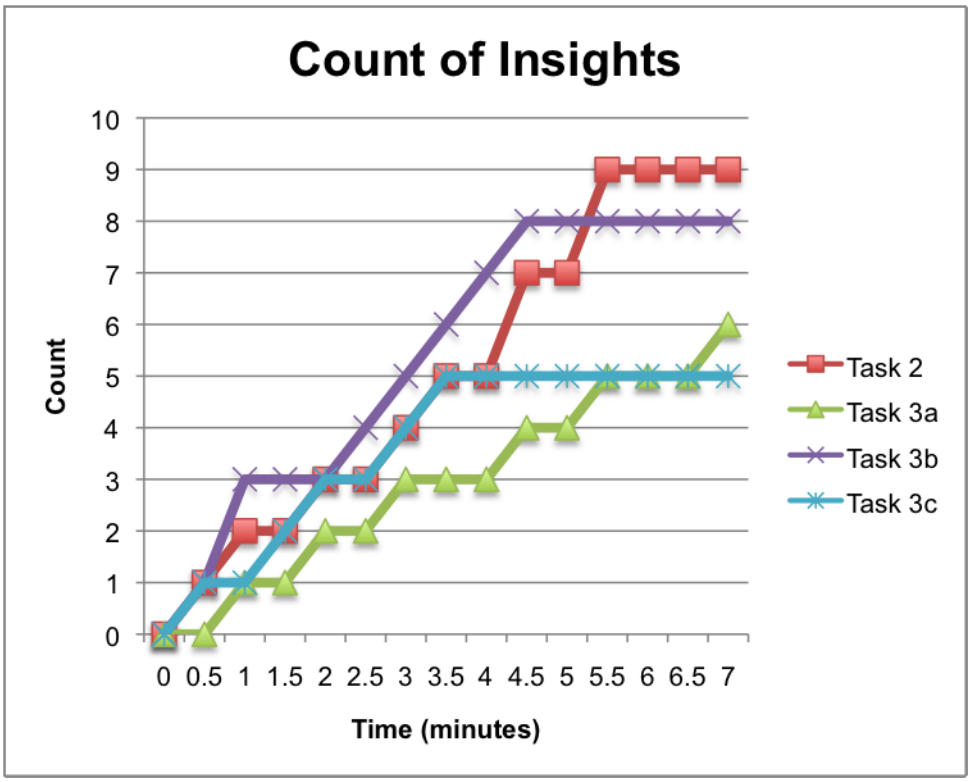


Figure 5-12. Average number of insights, over time, for tasks 2, 3a, 3b, and 3c.

Task 2 shows a step-like curve, suggesting an initial period of loading multiple developers into the visualization, followed by a small number of insights for each developer before ending in a plateau. The curve for Task 2 naturally rises above the other curves because subjects were asked to compare multiple developers at once. As a consequence, the most insights were generated in this task.

5.2.4.4 Participant Demographics

It is natural to wonder whether subjects with professional software development experience made more insights than others. Out of the 40 subjects in the experiment, 12 were professional software developers. However, an ANOVA analysis showed that there was no significant difference in number of insights between professional software developers and others. Due to the limited number of professional software developers recruited, we can only report general trends. On average, professional software developers made faster first insights than their student counterparts. They also finished the tasks more quickly on average than the other participants and made fewer incorrect insights.

Similarly, no significant differences in trust scores or attribution scores were found between developers and non-developers.

5.2.4.5 Experiment Summary

The overarching goal of this experiment was to demonstrate that Theseus is useful: it has utility and it is usable. While the first part of the evaluation (phase 1) focused exclusively on the usability of Theseus, this experiment focused on the utility of the tool. As such, it was designed to measure the effect of Theseus on distributed team members' sense of perceived trustworthiness toward others. Three dependent variables were evaluated: trust, usability, and insights.

The experiment clearly shows that Theseus has utility and is usable. A usability score calculated from responses to the post-experiment questionnaire showed that subjects found the tool to be usable. Overall, subjects indicated they would use the tool if it were available as well as rely on the information it provides, but pointed out that some important information was missing, such as a temporal view of the work items resolved by other developers on the team.

In addition, results from a one-way repeated measures ANOVA analysis confirmed the research hypothesis that Theseus results in higher perceived trustworthiness compared with not using the tool at all. Moreover, the results indicated that Theseus has a significant effect on the types of attributions made. There was significant evidence to show that Theseus results in more situational attributions than without the tool; the evidence was not especially strong. Table 5-9 provides a summarized view.

Measuring subjects' insights revealed they quickly became immersed in the data about their distributed team members. Participants agreed they thought they could become relatively productive with the interface in only a short time. Moreover, they made very few incorrect insights using the interface. After making 5-10 insights, the number of insights for each participant began to plateau. On average, software developers reached insights more quickly and made fewer incorrect observations than other participants.

Table 5-9. Experiment findings for perceived trustworthiness.

Trust Measure	Research Hypothesis	Significant Effects	Notes
Specific Inter-personal Trust	***H1:*** Using Theseus results in higher inter-personal trust compared with not using Theseus.	Information Condition***	Significant effect of Theseus on trust score
	H1-1.	(B,A)***	Inter-personal trust higher in *Theseus situational (b)* compared with *No Theseus (a)*
	H1-2.	(C,B)***	Inter-personal trust higher in *Theseus situational (b)* than *Theseus moderate situational (c)*
	H1-3	(D,B)***; (D,C)$^{+}$	Inter-personal trust lower in *Theseus dispositional (d)* than *Theseus situational (b)* but not *Theseus moderate situational (c)*
Attribution Type	***H2:*** Using Theseus results in more situational attributions compared with not using Theseus.	Information Condition***	Significant effect of Theseus on attribution type

	H2-1	$(B,A)^{+}$	Attribution more situational in *Theseus situational (b)* than *No Theseus (a)*
	H2-2	$(B,C)^{+}$	Attribution more situational in *Theseus situational (b)* than *Theseus moderate situational (c)*
	H2-3	$(D,B)^{***}$; $(D,C)^{***}$	Attribution more dispositional in *Theseus dispositional (d)* than *Theseus situational (b) and Theseus moderate situational (c)*

$^{***}p<0.001$; $^{**}p<0.01$; $^{*}p<0.05$; $^{+}p<0.1$

5.2.5 Discussion of Results

This section discusses the results of the experiment and addresses an unexpected finding that emerged: while interpersonal trust scores were higher with Theseus than without it, there was only marginal evidence that attributions were more situational with Theseus than without it. After addressing this result and providing a potential interpretation, we discuss some of the important limitations of this experiment. We then return to the central research question of this work: can a tool usefully engender perceived trustworthiness among distributed team members? We situate the results of this experiment in the context of trust and globally distributed software development.

5.2.5.1 Unexpected Results

The chief unexpected outcome of the experiment was Theseus only results in marginally more situational attributions compared with using no tool. A possible explanation is that although Theseus does not significantly change the kinds of attributions developers give, it helps them make more precise attributions. The figure below helps to illustrate this point:

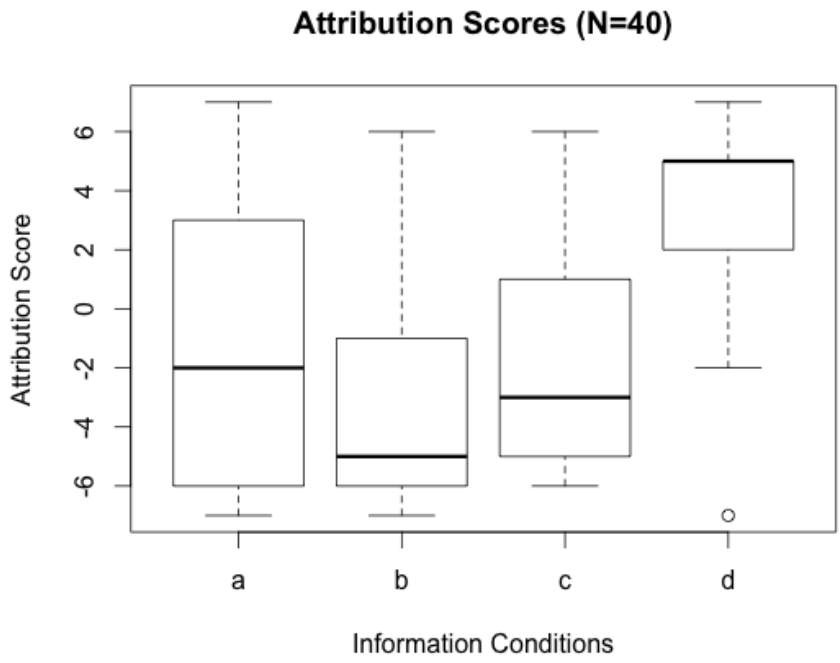


Figure 5-13. Box plots of attribution scores for each experimental condition.

We see that the variance in the *No Theseus (a)* condition is large enough as to not make attributions in the *Theseus situational (b)* and *Theseus moderate situational (c)* conditions significantly different. The variance in these other conditions was much smaller. This reduction in variance relative to the *No Theseus (a)* condition suggests that subjects were more certain of the appropriate attribution for the situation when they were using the tool.

This result is of interest because it shows a general trend of subjects settling on an appropriate attribution, whether it be situational or dispositional. In the *No Theseus (a)* condition, the average attribution score was -1.1, which although not significantly different from a mean score of 0 [t(39)=-1.44, p=0.1568], is slightly situational to begin with. In other words, subjects were already prepared to assume a situational reason for the failure of their remote colleague to deliver on time. Because of this starting bias, the tool failed to significantly push subjects any further in the situational direction during the *Theseus situational (b)* condition. However, the tool successfully pushed subjects toward the

dispositional direction in the *Theseus dispositional (d)* condition. The statistical analyses showed that subjects gave significantly higher dispositional attributions using Theseus than without using it.

A logical next question is whether the tool can breed mistrust, perhaps incorrectly founded. A valid concern is that, in the absence of certain information, participants' perceptions of their distributed colleagues may be overly influenced by the graphical representations, resulting in unfair comparison across their colleagues. Actually, the results show that this is not the case. While subjects gave more dispositional attributions using Theseus than without it, the dispositional attributions were appropriate considering the experimental condition. A tool that breeds mistrust would result in dispositional attributions when they were not warranted.

5.2.5.2 Limitations

This experiment has several limitations that should be addressed. The first has to do with the small number (12) of professional software developers recruited for the study. The remaining 28 subjects were a combination of graduates and undergraduates with software development experience from internships and graduate research. Ideally, all 40 subjects would have been professional developers. However, this limitation was handled in an appropriate manner. Although it could not be demonstrated quantitatively that there were no significant differences in the results between actual software developers and graduate students, the qualitative results indicated the only differences were in number of insights and time to first insight using the tool.

Another concern is that the team dynamics of distributed collaborations are very difficult to replicate in the laboratory. This limits the applicability of the results to real-world situations. The design of this experiment reduces this threat somewhat by presenting participants with a realistic scenario regarding a developer's failure to deliver a task on time,

a recognizable and possibly familiar situation grounded in data observed in studies of globally distributed teams. The significance of the results of the quantitative analysis add confidence to the generalizability of the study.

When considering the results of this experiment, it is important to understand to which aspects of trust they apply. First, this dissertation talked about *trust as a process* that takes time to develop, because it is based on people's shared experiences working together (Al-Ani, Trainer et al. 2012). This idea of trust as a dynamic process of observation, surprise, and adaptation comes out of the interview data as per our qualitative coding. For example, if trust exists, when a breakdown does occur, it will not likely to be a big deal. "I trust them, something happened outside of their control, something came up. They'll handle it." The trust level is not likely to go down. If the trust doesn't exist, when a breakdown occurs, it may be "They always do this...I'll have to check up on them more in the future."

A limitation of the experiment is that it doesn't measure the idea of trust as a process that takes time--because it cannot by design. In a laboratory experiment it is only practical to look at a few tasks. A longitudinal study might look at several failures that occur over weeks, months, and years; what trust looks like at each point, and how it changes or adjusts in response. Recognizing that trust may take time, the experiment attempted to address his limitation by measuring trust at several points throughout the experiment. Indeed, subjects reported that they felt their level of trust changing from moment to moment as they used the interface to learn about their team members. In the real world, trust will surely take longer to develop.

Second, this work has talked about trust as a perceived quality of someone, i.e. perceived trustworthiness. The lack of information in distributed teams leads to a tendency to over-emphasize personal characteristics when a failure occurs, leading to fundamental attribution error. In the experiment, when a subject makes a fundamental attribution error in

response to a peer who does not deliver on time, the subject is really saying, "This person doesn't seem trustworthy at this moment." The subject is *not* saying, "I can't trust them in the future." This kind of statement is reflective of trust as a process, and is something not measured in this experiment. Besides the specific inter-personal trust scores, attributions are a way to measure trust; if the other party seems trustworthy, the subject is more likely to give a situational attribution. If the other party does not seem trustworthy, the subject is more likely to give a dispositional attribution. By exposing information about people's availability and responsiveness, Theseus helps people build these expectations, upon which trust rests.

5.2.5.3 "Tools to Support Trust"

The results of this experiment provide an alternate way to think about the development of trust in the context of globally distributed software development. Traditional views of swift trust report that trust can exist simply by having pre-conceived notions about new collaborators to a project (Meyerson et al., 1996). This experiment showed that trust can also be viewed as an information problem. Visualizing information from the design space presented in this dissertation can engender perceived trustworthiness among distributed developers.

While the design space can assist in the design and development of new tools to support trust, it is interesting to wonder whether it would be useful to apply the design space to existing collaboration tools, such as e-mail, instant messenger clients, address books, and source-code repositories like Git. In other words, we could ask the question, do existing tools also support trust? And in what ways? Using the design space presented in this dissertation, those questions can be answered. Instead of using the "tool" as the unit of analysis, it might be interesting to ask which visualizations are effective as well as which ones are ineffective.

If true, we would be able to take "good" visualizations from different tools and apply them to new designs. A single visualization that can address multiple traces and multiple

trust factors may be more desirable than multiple visualizations that address the same set of factors. For example, the time zone overlap visualization may be able to be extended to encode more traces, such as calendars and e-mail messages, using different visual representations such as a color gradient. This approach effectively takes visual representations from both the SeeSoft and Awarenex tools. In this way, discussing the visualization as the unit of analysis may provide more insight to design than considering each tool in turn. The design space provides a conceptual mechanism and language by which existing tools can be analyzed and discussed by designers and implementers.

Chapter 6. Summary and Conclusions

This chapter summarizes the dissertation and presents clear statements about answers to its central research questions. It also discusses promising directions for future research, based on the results reported here. This chapter also explicitly addresses Research Question 2. Taking a retrospective look at both the design space presented in Chapter 2 and the results of the human subjects experiment, it presents principles for the design of visual interfaces to support trust.

6.1 Summary

Chapter one presented the idea of trust's role in everyday collaborations in distributed teams. The chapter looked at several different scenarios grounded in real-world observations of distributed teams where expectations about other members of a distributed team are unknown or incomplete. The insight of this dissertation is that these scenarios are related in a fundamental way.

Chapter two described the overarching path through the problem space that was illustrated in Chapter one. The solution this dissertation provides is to identify information that reduces uncertainty and leads to the formation of expectations upon which trust rests, and then to summarize this information in a visual interface of a software prototype that developers find useful. In distributed teams, trust is fragile and easily broken. In addition, it is slow and takes time. By exposing certain information about one's colleagues sooner rather than later, and summarizing this information in visual interfaces, this dissertation proposes that a tool can support the development of trust between globally distributed team members. This chapter formulated the thesis statement and research questions that guided this dissertation. It also summarized the contribution of this work.

Chapter three presented background literature and earlier work into the areas of trust, traces, and visualization. It examined the origins of trust as well as what particular

collaborative traces have been used by development tools in the awareness literature. It reviewed visualization tools that help developers understand each other's activities and changes. In addition, this chapter foreshadowed the impact of visualizing collaborative traces on perceived trustworthiness by reviewing results from an early pilot study conducted with Ariadne, the predecessor to the Theseus tool. The results of this experiment motivated the design space presented in this dissertation.

This chapter also presented a model of the design space for designing visualizations to support trust, which consisted of three matrices:

1. Trust Factors x Collaborative Traces
2. Collaborative Traces x Visual Representations
3. Visual Representations x Trust Factors

Chapter four outlined the design of the Theseus tool and early prototype of its visual interface. Theseus provides the requisite data for visualizations in the interface by mining collaborative traces from existing development repositories and generating data that cannot be easily acquired in sufficient quantity and quality.

Chapter five presented two phases of evaluation of the Theseus prototype. The first phase used three common usability inspection methods to demonstrate usability: Heuristic Evaluation, Cognitive Dimensions of Notations, and Cognitive Walkthrough. Based on the results of these inspection methods, help and documentation as well as drag and drop functionality were added to the interface.

In the second phase of the evaluation, the tool was evaluated first for utility and second for usability. This was accomplished by measuring changes in perceived trustworthiness as well as the quality of insights made by subjects. The result from this phase of the evaluation suggested that the tool had a significant effect on inter-personal trust and attribution types. While the tool resulted in higher trust compared with no tool at all, it

resulted in only marginally more situational attributions. The usability questionnaires indicated high usability of the tool, and an analysis of insights showed that users were able to quickly immerse themselves in the data and generate correct insights fairly quickly.

Taken together with the design space for tools to support trust, the results of the evaluation suggest a number of design principles for building and analyzing systems that can shape distributed developers' sense of trust toward each other:

6.2 Design Principles

- User-centricity: Remember that the developer is the most important aspect. Tailor data presentation to the user. For example, when showing average reply time of a remote developer, show the average reply time to the user of Theseus in question. This metric is likely to be more meaningful for the end-user.
- Comparison: By far, the biggest usability concern for Theseus was comparing multiple developers at the same time. The tool must have a clear interaction mechanism for viewing multiple developers at once. The mechanism should make efficient space of the monitor so users do not have to keep scrolling down.
    - Compare developers at the same site: On average, developers agree that when comparing reply times, one should consider the reply time of all developers at a particular site, not at all locations. Time zones and other site related issues are likely to affect the response time.
- Work Overlap: knowing a time zone is important, but knowing someone's actual working hours can be more useful to developers. Some people don't follow the 9-5 "standard."
- Temporality: Showing collaborative traces over time helps the user notice patterns in the visualized developer's activity, such as their working rhythms.

- Less is more: 1 visualization that can do the job of 2 is always preferred over 2 visualizations. Use the least amount of different visualizations possible.
- Fair Reply: In general, response time of one day within 3 time zones and 36 hours between more is a good baseline. Time critical messages should be answered in 5 hours regardless.
- Task: The most useful visualizations integrate with common development tasks, such as sending e-mail and reading pull requests.
- Similarity: Developers that are related (e.g., same project or team) should share similar position and size in the visualization. Developers who perceive themselves as similar tend to build trust more quickly.
- Size/Insight Balance: the size of a visualization should only be as big as it needs to be to effectively convey the meaning. Simple linear scales should not take up the same amount of white space as a line-based visualization.

6.3 Limitations of this Research

This work has demonstrated, through laboratory experiment, that a software tool can increase one's perceived trustworthiness towards others on a distributed team. However, the work presented here provides but one solution path through the problem space of trust and distributed development teams. There are several limitations of this work that should be kept in mind. The first is that understanding trust requires understanding many factors: whether a shared history of working together exists, each party's propensity to trust their peers, what the task is, each party's role in the organization, power dynamics, the gender of each party involved, and other variables. Each factor restricts the extent to which the results of this work can generalize to different situations. We cannot assume that similar results will show up in other situations. For example, one party may trust another person, i.e. have an

expectation, of them to produce production quality, bug-free code. They may *not*, however, trust that same person to babysit their child or borrow their car.

We should be careful not to reduce trustworthiness to a matter of collaborative traces. Perceived trustworthiness is strongly psychologically motivated as well. Propensity to trust is a general personality trait that conveys a general expectation of how trusting one is. It is proposed to be a stable within-party factor that is influenced by the trustor's social, cultural, and personal experiences as well as their personality type. Thus, propensity to trust differs from one individual to the next. Previous research has shown that it has a significant effect on people's sense of trust toward others (Mayer et al. 1995; Jarvenpaa and Leidner, 1999).

Nevertheless, interviews with distributed teams (Al-Ani and Redmiles, 2009; Al-Ani et al. 20011) and results from the experiment presented in this dissertation suggest that it is also common for trustors to have varying levels of trust toward different trustees. As such, it has been our assumption that propensity to trust alone is not sufficient. In order to address this variance, we believe it is crucial to better understand characteristics and activities of the trustee. Visualizations using data provided by collaborative traces can be designed to be an interface to such information.

6.4 Directions for Future Work

This dissertation illuminates several potential areas of study for future research. First, there are questions this work did not directly address, but are significant for understanding the trust formation process in distributed software development teams. For example, the controlled experiment presented in Chapter 5 assumed that distributed team members have no information about their distributed colleagues. Emerging research on the next generation of coordination tools shows that team members are likely to at least have partial information "fragments" about their colleagues (Fritz and Murphy, 2010). Moreover, this dissertation did not address levels of perceived trustworthiness over multiple interactions with the same team

members. Software development projects are likely to last many months, even years. As team members build up relationships and working histories with their team mates, they are likely to be able to better assess their perceived trustworthiness.

There are also follow-on studies to address questions raised by this study. Interviews with participants suggested that the tool may be more useful for managers of distributed teams. As such, it would be appropriate to examine the impact of role on perceived trustworthiness of distributed team members. It may be the case that the wealth of experience managers typically possess give them better faculties to assess whether a team mate is really unresponsive or just overloaded with assignments. This might significantly impact the type of attributions they give, biasing them toward the more situational side. As a consequence their levels of trust may be higher than an inexperienced or younger developer on their first project.

In addition, a concern that runs deeps through this work is the short duration of the study. As mentioned earlier, trust takes time to develop. As such, the ideas in this work would be genuinely enriched by longitudinal studies. Examples to point to for inspiration include work by Cramton and Hinds (2009) and Orr and Scott (2007). In these works, the authors studied and documented "instantiation surprises" and "institutional exceptions," respectively, in large-scale global projects over several years. Assuming sufficient resources, similar studies could be conducted in which the effect of collaboration tools on perceived trustworthiness is examined.

Finally, a key point of focus of this work is the proposition that a design space can inform the design and development of tools that can support trust in distributed teams. This study began to look at only a few dimensions from this design space, showing with a controlled experiment that they can significantly impact levels of perceived trustworthiness. But there is still an opportunity to push this further, and study both the design and

development of tools using the design space presented here, in both short and long-term collaborative relationships.

Bibliography

Ackerman, M. S. and Malone, T. W. (1990): 'Answer Garden: a tool for growing organizational memory', ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems, April 1990, pp. 31-39.

Al-Ani, B., Trainer, E., Ripley, R., Sarma, A., van der Hoek, A., Redmiles, D.F. (2008): 'Continuous Coordination within the Context of Cooperative and Human Aspects of Software Engineering', *Cooperative and Human Aspects of Software Engineering (CHASE)Workshop, ICSE 2008*, May 2008, pp. 1-4.

Al-Ani, B. and D.F. Redmiles (2009): 'Trust in Distributed Teams: Support through Continuous Coordination', *IEEE Software*, vol. 26, no. 6, November 2009, pp. 35-40.

Al-Ani, B., Trainer, E., Redmiles, D.F., and Simmons, E. (2012): 'Trust and Surprise in Distributed Teams: Towards an Understanding of Expectations and Adaptations', The 4th ACM International Conference on Intercultural Collaboration (ICIC 2012, Bengaluru, India), pp. 97-106.

Allen, T.J. (1984): *Managing the Flow of Technology*, MIT Press, Cambridge, MA.

Begel, A., Khoo, Y.P., and Zimmermann, T. (2010): 'Codebook: discovering and exploiting relationships in software repositories', *32nd ACM/IEEE International Conference on Software Engineering*, May 2010, pp. 125-134.

Begole, J., Tang, J. C., Smith, R. B., and Yankelovich, N. (2002): 'Work rhythms: analyzing visualizations of awareness histories of distributed groups', ACM Conference on Computer Supported Cooperative Work (CSCW 2002), November 2002, pp. 334-343.

Bertin, J. (1983): *Semiology of Graphics*, University of Wisconsin Press, Madison, WI.

Biehl, J. T., Czerwinski, M., Smith, G., and Robertson, G. G. (2007): 'FASTDash: a visual dashboard for fostering awareness in software teams', SIGCHI Conference on Human Factors in Computing Systems, April 2007, pp. 1313-1322.

Bird, C., Nagappan, N., Devanbu, P., Gall, H., and Murphy, B. (2009): 'Does Distributed Development Affect Software Quality?: An Empirical Case Study of Windows Vista', *Communications of the ACM*, vol. 52, no. 8, August 2009, pp. 85-93.

Blackwell, A F, Green, T.R.G (2000), 'A cognitive Dimensions Questionnaire Optimised for Users', in A.F. Blackwell & E. Bilotta (Eds.) Proceedings of the twelfth Annual Meeting of the Psychology of Programming Interest Group, pp. 137-152.

Bos, N., Olson, J., Gergle, D., Olson, G., and Wright, Z. (2002): 'Effects of four computer-mediated communications channels on trust development', *SIGCHI Conference on Human Factors in Computing Systems: Changing Our World, Changing Ourselves*, April 2002, pp. 135-140.

Buchan, N.R., Croson, R.T.A., and Solnick, S. (2008): 'Trust and Gender: An Examination of Behavior and Beliefs in the Investment Game', *Journal of Economic Behavior and Organization*, vol. 68, pp. 466-476.

Butler, J.K. (1991): 'Toward understanding and measuring conditions of trust: Evolution of a conditions of trust inventory', *Journal of Management*, vol. 17, pp. 643-663.

Butler, J. K. and R. S. Cantrell (1994): 'Communication Factors and Trust: An Exploratory Study', *Psychological Reports*, vol. 74, no. 1, pp. 33–34.

Chaudhuri, A., and Gangadharan, L. (2003): 'Gender Differences in Trust and Reciprocity', Department of Economics – Working Paper Series 875, The University of Melbourne.

Chin, J. P., Diehl, V. A., and Norman, K. (1988): 'Development of an instrument measuring user satisfaction of the human-computer interface', In Proceedings of CHI '88 Conference on Human Factors in Computing Systems (CHI 1998), pp. 213-218.

Chin, W. W., and Newsted, P. R. (1999): Structural Equation Modeling analysis with Small Samples Using Partial Least Squares', In Rick Hoyle (Ed.), *Statistical Strategies for Small Sample Research*, Sage Publications, pp. 307-341.

Clore, G. L. (1994): 'Why emotions are never unconscious', In P. Ekman and R. J. Davidson (Eds.), *The nature of emotion: Fundamental questions* (pp. 285-290). New York: Oxford University Press.

Cramton, C. (2001): 'The mutual knowledge problem and its consequences for dispersed collaboration.' *Organization Science*, 12, 346-371.

Cramton, C. D. (2002): 'Attribution in distributed work groups', In P. J. Hinds, S. Kiesler (Eds.). *Distributed Work*, MIT Press (2002), pp. 191-212.

Cramton, C.D., and Hinds, P. (2009): 'The dialectical dynamics of nested structuration in globally distributed teams', *Academy of Management Best Paper Proceedings*, (2009).

Cubranic, D. and Murphy, G. C. (2003): 'Hipikat: recommending pertinent software development artifacts', International Conference on Software Engineering (ICSE 2003), May 2003, pp. 408-418.

DeLine, R., Czerwinski, M., and Robertson, G. (2005): 'Easing Program Comprehension by Sharing Navigation Data', IEEE Symposium on Visual Languages and Human-Centric Computing, September 2005, pp. 241-248.

Dewan, P. and R. Hegde. (2007): 'Semi-Synchronous Conflict Detection and Resolution in Asynchronous Software Development', European Conference on Computer-Supported Cooperative Work (ECSCW), September 2007, pp. 159-178.

Eick, S. G., Steffen, J. L., and Sumner, E. E. (1992): 'Seesoft-A Tool for Visualizing Line Oriented Software Statistics', IEEE Transactions on Software Engineering, vol. 18, no. 11, November 1992, pp. 957-968.

Ellis, J. B., Wahid, S., Danis, C., and Kellogg, W. A. (2007): 'Task and social visualization in software development: evaluation of a prototype', ACM SIGCHI Conference on Human Factors in Computing Systems, April-May 2007, pp. 577-586.

Ferrin, D., Bligh, M., and Kohles, J. (2007): 'Can I Trust You to Trust Me? A Theory of Trust, Monitoring, and Cooperation in Interpersonal and Intergroup Relationships', *Group and Organization Management*, vol. 32, no. 4, August 2007, pp. 465-500.

Fitzpatrick, G., Marshall, P., and Phillips, A. (2006): 'CVS integration with notification and chat: lightweight software team collaboration', ACM Conference on Computer Supported Cooperative Work (CSCW), November 2006, pp. 49-58.

Frank, M. G., and Gilovich, T. (1989): 'Effect of memory perspective on retrospective causal attributions', *Journal of Personality and Social Psychology*, vol. 57, no. 3, pp. 399–403.

Fritz, T., and Murphy, G.C. (2010): 'Using information fragments to answer the questions developers ask', 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10), ACM, New York, NY, USA, pp. 175-184.

Furst, S., Blackburn, R., and Rosen, B. (1999): 'Virtual team effectiveness: A proposed research agenda', *Information Systems Journal*, vol. 9 (1999), pp. 249-269.

German, D., Hindle, A., and Jordan N. (2004): 'Visualizing the evolution of software using softChange', International Conference on Software Engineering and Knowledge Engineering, 2004, pp. 336–341.

Gil, Y. and D. Artz. (2006) 'Towards content trust of web resources', 15th *International World Wide Web Conferenc*e (WWW), May 2006, pp. 565-574.

Green, T. (1989): 'Cognitive Dimensions of Notations', *British Computer Society HCI Specialist Group on People and Computers*, pp. 43-50.

Handy, C. (1995): 'Trust and the Virtual Organization', *Harvard Business Review***,** May-June 1995, pp. 40-50.

Herbsleb, J. D. and Grinter, R. E. (1999): 'Splitting the organization and integrating the code: Conway's law revisited', International Conference on Software Engineering (ICSE), May 1999, pp. 85-95.

Herbsleb, J.D., Mockus, A., and Roberts, J.A. (2006). 'Collaboration in Software Engineering Projects: A Theory of Coordination', International Conference on Information Systems, Milwaukee, WI.

Hertzum, M., Andersen, V., Andersen, H.H.K., and Hansen, C.B. (2002): 'Trust in Information Sources: Seeking Information from People, Documents, and Virtual Agents', *Interacting with Computers*, vol. 14, no. 5, pp. 575-599.

Hertzum, M. (2002): 'The Importance of Trust in Software Engineers' Assessment and Choice of Information Sources', *Information and Organization*, vol. 12, no. 1, pp. 1-18.

Holton, J. A. (2001): 'Building Trust and Collaboration in a Virtual Team', *Team Performance Management*, vol. 7, no. 3-4 (2001), pp. 36-47.

Hupfer, S., Cheng, L., Ross, S., and Patterson, J. (2004): 'Introducing collaboration into an application development environment', ACM Conference on Computer Supported Cooperative Work, November 2004, pp. 21-24.

Iacono, C., and Weisband, S. (1997): 'Developing Trust in Virtual Teams', Hawaii International Conference on System Sciences (Wailea, HI, USA), January 1997, pp. 412-420.

Jarvenpaa, S. L., Knoll, K., and Leidner, D. E. (1998): 'Is Anybody Out There?: Antecedents of Trust in Global Virtual Teams', *Journal Manage. Info. Systems*, vol. 14, no. 4 (1998), pp. 29-64.

Jarvenpaa, S. L. and Leidner, D. E. (1999): 'Communication and Trust in Global Virtual Teams', *Organization Science*, vol. 10, no. 6, June 1999, pp. 791-815.

Jeanquart-Barone, S. (1993): 'Trust Differences Between Supervisors and Sub-ordinates: Examining the Role of Race and Gender', *Sex Roles*, vol. 19, nos. 1-2, pp. 1-10.

Kalman, Y.M. and Rafaeli, S. (2005): 'Email Chronemics: Unobtrusive Profiling of Response Times', Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS), January 2005, pp. 108b.

Karagiannis, T. and Vojnovic, M. (2009): 'Behavioral profiles for advanced email features', In Proceedings of the 18th international conference on World wide web (WWW '09), pp. 711-720.

Kersten, M. and Murphy, G. C. (2006): 'Using task context to improve programmer productivity', ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), November 2006, pp 1-11.

Lee, J.D. and See, K.A. (2004): 'Trust in Automation: Designing for Appropriate Reliance,' *Human Factors*, vol. 46, pp. 50-80.

Levin, D. Z. and R. Cross (2004): 'The Strength of Weak Ties You Can Trust: The Mediating Role of Trust in Effective Knowledge Transfer', *Management Science*, vol. 50, no. 11, pp. 1477–90.

Lewis, J.W. (1993): 'IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use', *Technical Report 54.76*, pp. 1-39.

Lohse, G.L., Biolsi, K., Walker, N., and Rueter, H.H. (1994): 'A Classification of Visual Representations', *Communications of the ACM*, vol. 37, no. 12, December 1994, pp. 36-49.

Mackinlay, J. D. (1986): 'Automating the Design of Graphical Presentations of Relational Information', *ACM Transactions on Graphics*, vol. 5, April 1986, pp. 110-141.

Maddux, W.W. and Brewer, M.B. (2005): 'Gender Differences in the Relational and Collective Bases for Trust', *Group Processes and Intergroup Relations*, vol. 8, no. 2, pp. 159-171.

Maletic, J.I., Marcus, A., and Collard, M.L. (2002): 'A Task Oriented View of Software Visualization', In *Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT '02)*, 2002, pp. 32-40.

Mattox, D., Smith, K., and Seligman, L. (1998): 'Software Agents for Data Management', In Thuraisingham, B. *Handbook of Data Management*, CRC Press: New York. pp. 703-722.

Maybury, M., D'Amore, R. and House, D. (2000): 'Automated Discovery and Mapping of Expertise', In Ackerman, M., Cohen, A., Pipek, V. and Wulf, V. (eds.). *Beyond Knowledge Management: Sharing Expertise*, Cambridge: MIT Press.

Mayer, R. C., Davis, J. H., and Schoorman, F. D. (1995): 'An integration model of organizational trust', *Academy of Management Review,* vol. 20, no. 3, pp. 709–734.

McAllister, D.J. (1995): 'Affect- and cognition-based trust as foundations for interpersonal cooperation in organizations', *Academy of Management Journal*, vol. 38, no. 1, 1995, pp. 24–59.

McDonald, D. W. and Ackerman, M. S. (2000): 'Expertise recommender: a flexible recommendation system and architecture', ACM Conference on Computer Supported Cooperative Work (CSCW), December 2000, pp. 231-240.

Meyerson, D., Weick, K.E., and Kramer, R.M. (1996): 'Swift Trust and Temporary Groups,' In *Trust in Organizations: Frontiers of Theory and Research,* Kramer (Eds.), R.M., and Tyler, T.R. Sage Publications, Thousand Oaks, CA, pp. 166-195.

Mockus, A. and Herbsleb, J. D. (2002): 'Expertise browser: a quantitative approach to identifying expertise', International Conference on Software Engineering (ICSE), May 2002, pp. 503-512.

Moody, D.L. (2010): 'The Physics of Notations: A Scientific Approach to Designing Visual Notations in Software Engineering', *IEEE Transactions on Software Engineering*, vol. 35, no. 6, November 2009, pp. 756-779.

Nardi, B. A., Whittaker, S., and Schwarz, H. (2002): 'NetWORKers and their Activity in Intensional Networks', Computer-Supported Cooperative Work, vol. 11, no. 1-2, April 2002, pp. 205-242.

Nielsen, J.K. (1993): 'Heuristic Evaluation', *Usability Inspection Methods*, pp. 203-232, Wiley, New York, NY.

Nooteboom, B.,Berger, H., and Noorderhaven, N.G. (1997): 'Effects of trust and governance on relational risk', *Academy of Management Journal*, vol. 40, no. 2, pp. 308-338.

Olson, G.M. and J.S. Olson (2000): 'Distance Matters', Human- *Computer Interaction*, vol. 15, no. 2-3, 2000, pp. 139-178.

O'Reilly, C., Bustard, D., and Morrow, P. (2005): 'The war room command console: shared visualizations for inclusive team coordination', ACM Symposium on Software Visualization, May 2005, pp. 57-65.

Orr, R.J. and Scott, W.R. (2007): 'Institutional Exceptions on Global Projects: A Process Model', *Journal of International Business Studies* vol. 39, no. 4, pp. 562-588.

Piccoli, G., and B. Ives (2003): 'Trust and the unintended effects of behavior control in virtual teams', *MIS Quarterly*, vol. 27, no. 3, pp. 365–376.

Pyysiäinen, J. (2003): 'Building Trust in Global Inter-Organizational Software Development Projects: Problems and Practices', *International Workshop on Global Development (GSD) at ICSE 2003*, May 2003, pp. 69-74.

Robinson, M. D., and Clore, G. L. (2002): 'Belief and Feeling: Evidence for an Accessibility Model of Emotional Self-Report', *Psychological Bulletin*, vol. 128, no. 6, pp. 934-960.

Rosen, B., Furst, S., and Blackburn, R. (2006): 'Training for Virtual Teams: An Investigation of Current Practices and Future Needs', *Human Resources Management*, vol. 45, no. 2, pp. 229-247.

Ross, W., and J. LaCroix (1996): 'Multiple meanings of trust in negotiation theory and research: A literature review and integrative model', *The International Journal of Conflict Management,* vol. 7, pp. 314-360.

Sabherwal, R. The role of trust in outsourced IS development projects', *Communications of the ACM,* vol. 42, no. 2 (1999), pp. 80-86.

Sarma, A., Noroozi, Z., and van der Hoek, A. (2003): 'Palantír: raising awareness among configuration management workspaces', International Conference on Software Engineering (ICSE), May 2003, pp. 444–454.

Sarma, A., Maccherone, L., Wagstrom, P., and Herbsleb, J. (2009): 'Tesseract: Interactive visual exploration of socio-technical relationships in software development', In *Proceedings of the 31st International Conference on Software Engineering* (ICSE '09). May 2009, pp. 23-33.

Schummer, T., and Haake, J.M. (2001): 'Supporting distributed software development by modes of collaboration', European Conference on Computer Supported Cooperative Work (ECSCW), September 2001, pp. 79–98.

Shneiderman, B. (2003): *The Craft of Information Visualization: Readings and Reflections*, Morgan Kaufmann, San Francisco, CA.

Spence, R. (2001): *Information Visualization: Design for Interaction*, Prentice Hall, Upper Saddle River, NJ.

Storey, M. D., Cubranic, D., and German, D. M. (2005): 'On the use of visualization to support awareness of human activities in software development: a survey and a framework', *ACM Symposium on Software Visualization*, May 2005, pp. 192-202.

Sun, Y.L., Yu, W., Han, Z., and Liu, K.J.R. (2006): 'Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks', *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, February 2006, pp. 305-317.

Szóstek, A. M. and Markopoulos, P. (2006): 'Factors defining face-to-face interruptions in the office environment', Conference on Human Factors in Computing Systems, April 2006, pp. 1379-1384.

Tee, K., Greenberg, S., and Gutwin, C. (2006): 'Providing artifact awareness to a distributed group through screen sharing', ACM Conference on Computer-Supported Cooperative Work (CSCW), November 2006, pp. 99-108.

Trainer, E., Quirk, S., de Souza, C. R. B., Redmiles, D.F. (2005): 'Bridging the Gap between Technical and Social Dependencies with Ariadne', *OOPSLA Workshop on Eclipse Technology Exchange*, pp. 26-30.

Trainer, E., Quirk, S., de Souza, C.R.B., and Redmiles, D.F. (2008): 'Analyzing a Socio-Technical Visualization Tool Using Usability Inspection Methods', *IEEE Symposium on Visual Languages and Human Centric Computing*, September 2008, pp. 78-81.

Trainer, E. and D.F. Redmiles (2009): 'A Survey of Visualization Tools that Promote Awareness of Software Development Activities', *ISR Technical Report #UCI-ISR-09-5*, December 2009, pp. 1-55.

Trainer, E., Al-Ani, B., Redmiles, D.F. (2011): 'Impact of Collaborative Traces on Trustworthiness', *Cooperative and Human Aspects of Software Engineering (CHASE) Workshop, ICSE 2011*, May 2011, pp. 40-47.

Trainer, E. and Redmiles, D.F. (2012): 'Foundations for the Design of Visualizations that Support Trust in Distributed Teams', In Proceedings of the 2012 ACM International Conference on Advanced Visual Interfaces (AVI), pp. 34-41.

Tufte, E. (1990): *Envisioning Information*, Graphics Press, Cheshire, CT.

Tufte, E. (2006): *Beautiful Evidence*, Graphics Press, Cheshire, CT.

Tyler, J. R. and Tang, J. C. (2003): 'When Can I Expect an Email Response? A Study of Rhythms in Email Usage', *Proceedings from ECSCW '03: European Conference on Computer-Supported Cooperative Work*, pp. 239- 258.

Venkatesh, V., and Davis, F. D. (2000): 'A theoretical extension of the technology acceptance model: Four longitudinal field studies', *Management Science*, vol. 46, no. 2, pp. 186–204.

Walther, J. B., and Bunz, U. (2005): 'The rules of virtual groups: Trust, liking, and performance in computer-mediated communication', *Journal of Communication*, vol. 55, pp. 828-846

Wharton, C. W., Reiman, J., Lewis, C., and Polson, P. (1994): 'The cognitive walkthrough method: A practitioner's guide', *Usability Inspection Methods*, pp. 105-140, Wiley, New York, NY.

Wilson, J.M., Straus, S.G., and McEvily, W.J. (2006): 'All in due time: The development of trust in computer-mediated and face-to-face groups', *Organizational Behavior and Human Decision Processes*, vol. 99, no. 1, pp. 16-33.

Ye, Y., Yamamoto, Y., and Nakakoji, K. (2007): 'A socio-technical framework for supporting programmers', *ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, September 2007, pp. 351-360.

Zolin, R., Hinds, P.J., Fruchter, R. and Levitt, R.E. (2004): 'Interpersonal Trust in Cross-Functional Global Teams: A longitudinal Study', *Information and Organization*, vol. 14, no. 1, pp. 1-26.

Appendix 1. Experiment Protocol Summary

Approx. Timing	Event
0:00	Subject arrives. He is seated at a table in front of a laptop computer.
0:01	Review study information sheet with subject and ask him to give verbal consent.
0:04	Administer demographics questionnaire.
0:07	Collect demographics questionnaire; Administer baseline trust pre-questionnaire.
0:10	Collect trust pre-questionnaire; Give subject Task 1 instructions sheet; Tell subject to begin.
0:14	Collect subject's ranked attribution list; Administer interpersonal trust questionnaire.
0:17	Collect trust questionnaire; Give subject 5 minute break.
0:22	Move subject to new room; Administer training on Theseus.
0:28	Give subject Task 2 instructions sheet; Tell subject to begin.
0:38	Collect Task 2 instructions sheet.
0:39	Hand out Task 3a instructions; tell subject to begin.
0:49	Collect subject's ranked attribution list; Administer interpersonal trust questionnaire.
0:52	Collect trust questionnaire.
0:53	Hand out Task 3b instructions; tell subject to begin.
0:63	Collect subject's ranked attribution list; Administer interpersonal trust questionnaire.
0:66	Collect trust questionnaire
0:67	Hand out Task 3c instructions; tell subject to begin.
0:77	Collect subject's ranked attribution list; Administer interpersonal trust questionnaire.
0:80	Collect trust questionnaire.
0:81	Administer post-experiment questionnaire.
When Subject finishes post-questionnaire (around 0:87)	Collect post-experiment questionnaire; Address any questions; Pay and dismiss subject.

Appendix 2. Pre-Experiment Questionnaires

What year were you born? _____

In what country were you born? ☐ USA Other _____

Is English your first language?

☐ Yes ☐ No. My first language is: _____

Which languages do you speak? _____

_____

_____

_____

Which languages do you speak at home? _____

_____

_____

What is your race or ethnic origin?

☐ American Indian or Alaska Native
☐ Asian
☐ Black or African American
☐ Hispanic or Latino
☐ Native Hawaiian or Pacific Islander
☐ Caucasian
☐ Other ethnic or racial category: _____

What is the highest academic degree you have received?

- ☐ Less than high school diploma
- ☐ High school diploma or equivalent
- ☐ Undergraduate degree (for example, B.A., B.S., etc.)
- ☐ Graduate degree (Masters or Doctorate)

What was your field/major? _____

What is your current occupation? _____

How many years of work experience do you have? _____

How many of those years are in software development? _____

How many of those years in *distributed* software development (i.e., your collaborator/s worked in a different location from you)? _____

Pre-Experiment Trust Questionnaire

Instructions: For each question, fill in the circle (O) above the response that matches your level of agreement.

1. People motivate me to do my best work.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. In dealing with strangers, one is better off to be cautious until they have provided evidence that they are trustworthy.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. People are usually polite to me.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. Parents can usually be relied upon to keep their promises.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. Most elected public officials are really sincere in their campaign promises.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I enjoy working with others.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. Parents and teachers are likely to say what they believe themselves and not just what they think is good for the child to hear.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. Most salesmen are honest in describing their products.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. Most of my colleagues share the same interests as me.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. A large share of accident claims filed against insurance companies are phony.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. The courts give fair and unbiased treatment to everyone.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. I feel comfortable socializing with my colleagues.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. Most people would be horrified if they knew how much news that the public hears and sees is distorted.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. Most people answer public opinion polls honestly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. I find new social networking technologies like Facebook and LinkedIn useful.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

16. Most major national sports are fixed in one way or another.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

17. People tend to get along very well with me.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

18. It is safe to believe that in spite of what people say most people are primarily interested in their own welfare.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

19. Even though we have reports in newspapers, radio, and TV, it is hard to get objective accounts of public events.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

20. Most students in school would *not* cheat, even if they were sure of getting away with it.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

21. The future seems very promising.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

22. Most experts can be relied upon to tell the truth about the limits of their knowledge.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

23. Most idealists are sincere and usually practice what they preach.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

24. Most repairmen will not overcharge even if they think you are ignorant of their specialty.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

25. People can solve most of their problems if they invest the necessary effort.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

26. Hypocrisy is on the increase in our society.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

27. Most parents can be relied upon to carry out their threats of punishments.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

28. Most people can be counted on to do what they say they will do.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 3. Task 1

Instructions: You have to come into the office this weekend to work on the "MIRTH" project. **Victor Ward**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule.

Given what you know about how people behave, which explanation do you think most likely describes why Victor was unable to deliver on time?

A. He is less willing to devote whatever time necessary in order to finish his work.

B. He may have had to implement new features with last minute notice.

C. He probably had too many tasks in other projects.

D. He might not be a very reliable person when it comes to meeting project deadlines.

E. He could have had to integrate more changes into the build than expected.

F. He might not be agreeable to spending time outside work on this particular project.

G. He probably is not as dependable as others when getting his work in ahead of the due date.

H. He may have got distracted by work in other projects in which he is involved.

<continued on next page>

Below, order each of the explanations from *most likely* (no. 1) to *least likely* (no. 8). Write the letter corresponding to the explanation next to its appropriate rank. For example, if you feel explanation B is most likely, write 'B' in the space next to the number '1.'

Rank the explanations from most to least likely.

1._____
2._____
3._____
4._____
5._____
6._____
7._____
8._____

Most likely

Least likely

Appendix 4. Post-Task Questionnaire: Task 1

Instructions: Please consider **Victor Ward**, the developer in the last task. Recognizing that you may not have all the information necessary to answer each question, fill in the circle (O) above the response that matches your level of agreement.

1. If we decided to meet to discuss work, I would be certain he would be there.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. I could rely on him to e-mail an important message for me if I didn't have time to do it myself.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. I could expect him to tell me the truth.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. I would feel a sense of loss if one of us was transferred and we could no longer work together.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. If he was going to do a job for me and it didn't arrive on time, I would guess there was a good reason for the delay.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I would be comfortable giving him a task or problem which was critical to the project, even if I could not monitor him.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. I can rely on him to respond to my messages in a reasonable amount of time.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. If he promised to do me a favor, he would follow through.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. If he couldn't get together with me as planned, I would believe his excuse that something important had come up.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. If he gave me a compliment, I would question if he really meant what was said.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. He would never intentionally misrepresent my point of view to others.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. If he didn't think I handled a certain situation very well, he would not criticize me in front of other people.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. If I told him what things I worry about he would not think my concerns were silly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. If he unexpectedly laughed at something I did or said, I would wonder if he was being critical and unkind.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. I would expect him to play fair.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 5. Theseus Instructions

Instructions: For the remainder of the experiment, you will be using a novel software prototype called Theseus, which is a web application. Theseus provides a suite of visualizations that show availability information about distributed team members. In its current implementation, Theseus integrates with your Google contacts and work item repositories.
In order to familiarize yourself with the tool for today's study, follow these steps.

Initializing Theseus:

1. Click on the button labeled "AUTHORIZE THESEUS" (Figure 1). A new window will appear asking you to authenticate with Google. Click on "Grant access." (Figure 2a) This step gives Theseus access to your Google contacts.

Figure 1. Push button to initialize Theseus.

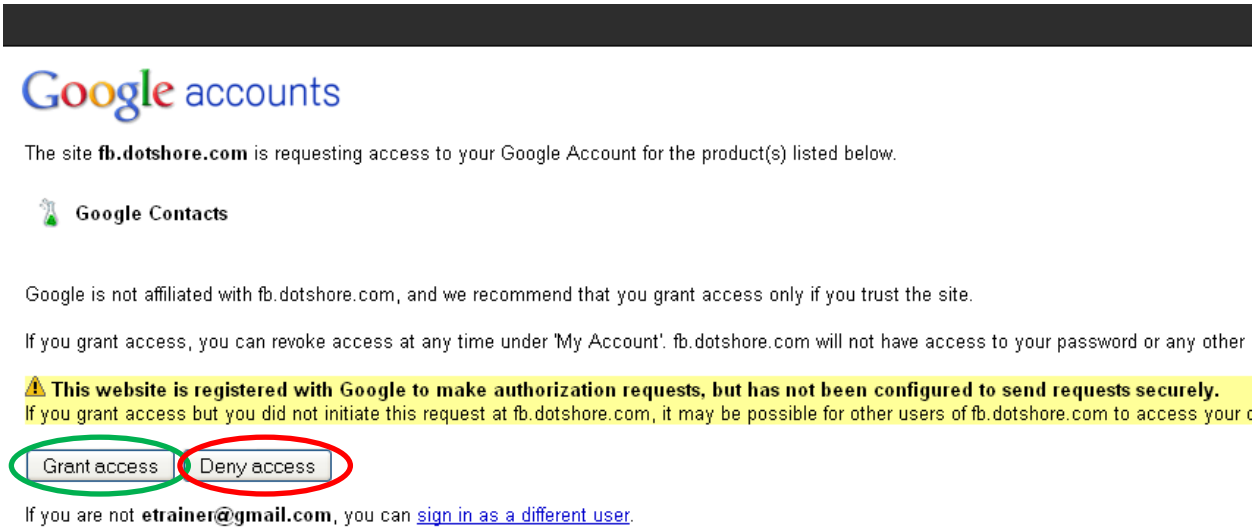


Figure 2. Authorization window, with options to authenticate (a, green circle) or not (b, red circle).

2. If you do not have a Google account or do not wish to expose your contacts, click the "Deny access" button (Figure 2b). Your login credentials will not be used and Theseus will generate a set of contacts instead.

Figure 3. Familiar *loading* animated .gif with accompanying text shows user that Theseus is loading user and project data.

3. The authorization window will close and Theseus will indicate that it is loading contact information and fetching project data (Figure 3).

Viewing Contacts:

4. Once the contact list becomes visible (left), you can select contacts to view information about them (Figure 4a). The "Contact List" uses a tree view to group contacts by the projects in which they participate. Once a contact is selected, the visualizations will populate on the right hand side of the contact list (Figure 4b).

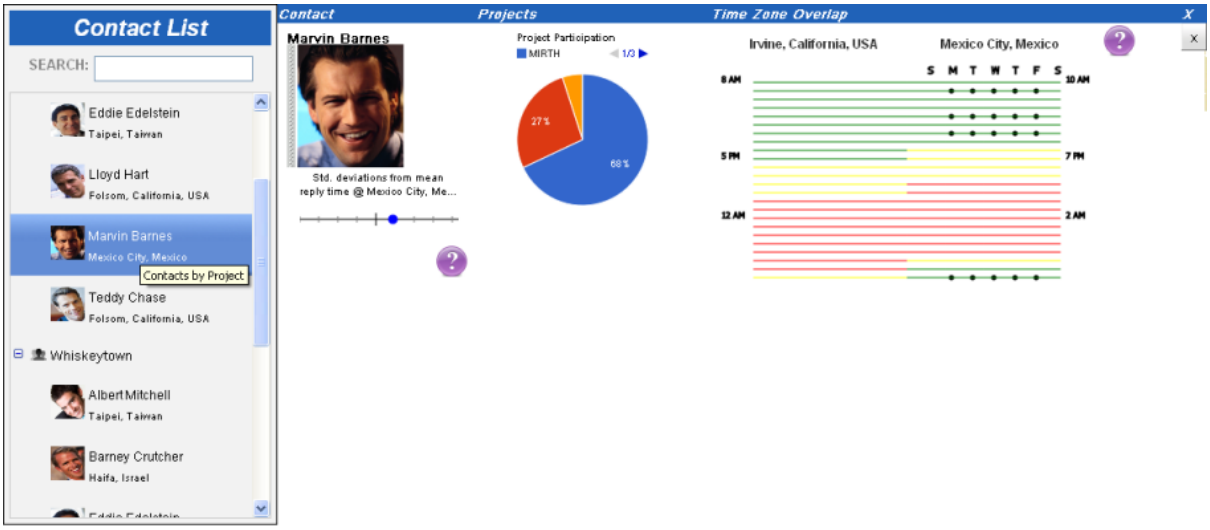


Figure 4. Theseus main display. To the left is the contact list (a), to the right is the visualization display panel (b), and to the right of each visualization are instructions (c), as clickable ? buttons.

5. You can click the purple question marks to the right of visualizations for instructions on their use (Figure 4c). See the end section of this document for the same instructions.
6. To view multiple contacts at once, select more contacts (repeat step 4).

Deleting Contacts:

7. (Individual) To remove a contact from the table, click the “x” button on the in the rightmost column of the same row (Figure 4d).
8. (All) To clear all contacts from the table, click the “X” column heading at top right.

Organizing Contacts:

9. If you run out of vertical space searching for contacts, a scroll bar will appear to the right so you can scroll down the table of added contacts. Additionally, you can drag and drop contacts to re-arrange them in the table to make comparisons easier (Figure 5). **Make sure to always drag a contact up, not down, to the position you like. Dragging a contact downward in the table will cause the drag and drop operation to fail.**

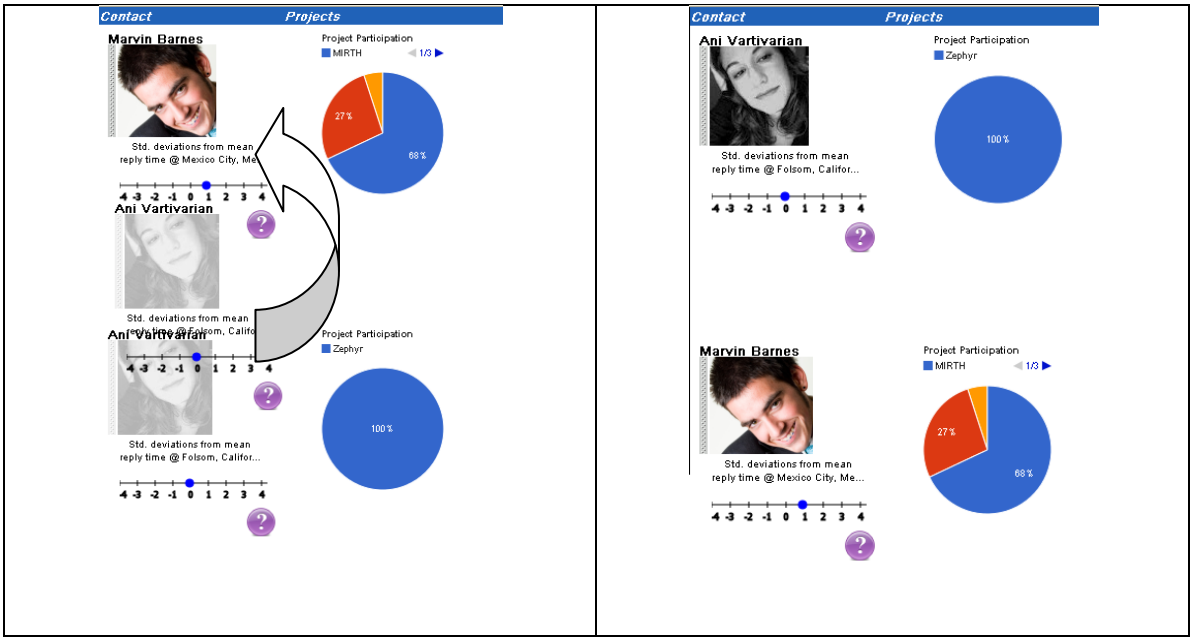
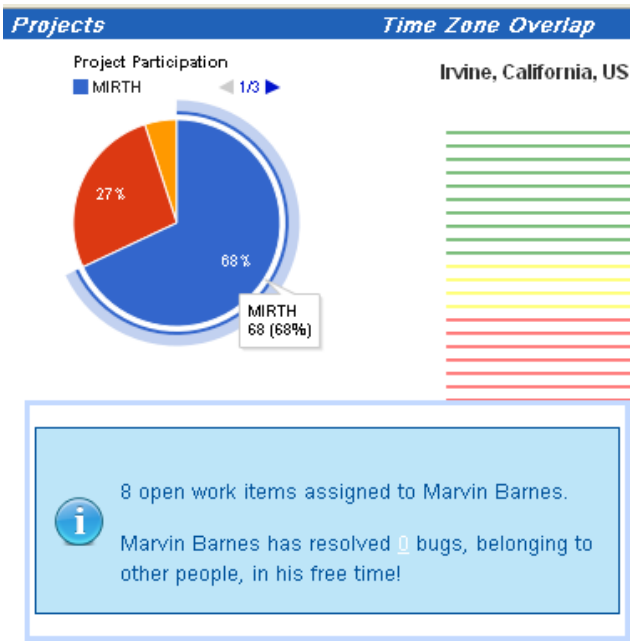


Figure 5. Display during (left) and after (right) drag n drop. Select a contact by the texturized grey strip to the left of the image and drag it up.

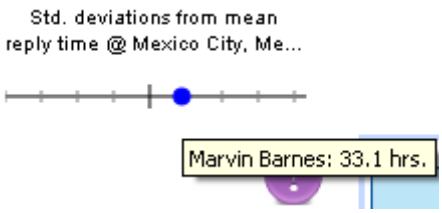
Interpreting the Visualizations:

Theseus has 3 main visualizations:

1. The *Projects* visualization uses a pie chart to show a contact's activity broken down by the projects in which they participate. The percentages are calculated by the number of work items fixed in the project. Clicking on a slice of the pie displays the number of open work items that contact has *in the project* as well as *overall* work items that contact has resolved that were not originally assigned to them. This indicates "extra" work the contact is undertaking.

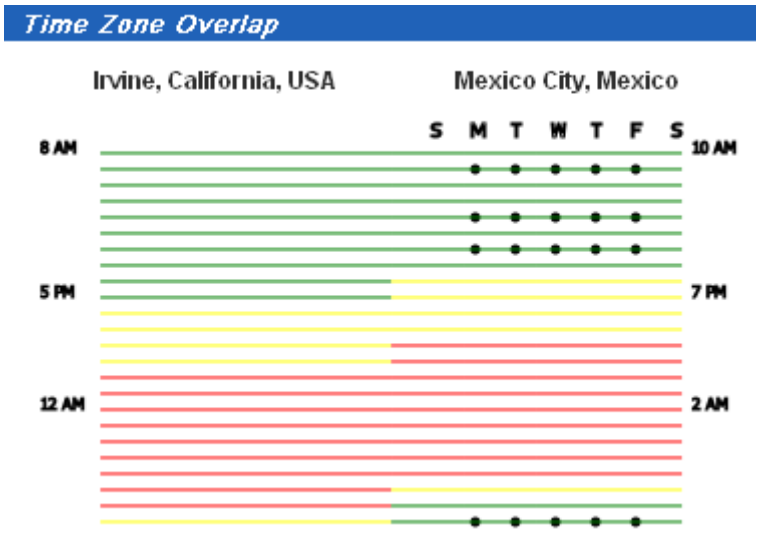


2. The *E-mail Responsiveness* visualization shows the contact's standard score of their average reply time to e-mails using a linear scale. A standard score ranges from -4 to 4 (with 0 being average), and indicates how many standard deviations a data point is above (pos. values) or below (neg. values) the mean. A faster response time appears in this visualization to the left as a negative value, while a slower response will have a negative value, and appears to the right. The contact's average reply time to email (in hours) is shown on top of the linear scale.



3. The *Time Zone Overlap* visualization shows overlap in the time of day between your current location (Irvine), and the contact's location. It is

composed of two separate parts: the time in the user's location (Irvine) is displayed to the left while the time in the contact's location appears to the right. The seven days of the week are shown with a one letter abbreviation on top of the visualization.
Each line indicates an hour of the day starting at 8 am at the user's current location. A green line indicates normal work hours, 8 am - 5 pm. A yellow line shows the time of the day where people are awake, but not necessarily at work, 7 am and 6 pm - 9 pm. The red line indicates sleeping hours, 10 pm - 6 am. A black circle on a line indicates that a contact is checking and sending e-mails during that hour.



Appendix 6. Task 2

Instructions: From the following list of developers, use Theseus to identify who you believe to be most available to reply to e-mail. Identify who is least so. As you use the tool, think aloud, mention individual observations about each individual, and justify your selections.

A. Jim Roberts

B. Bob Rice

C. Lloyd Hart

D. Albert Mitchell

Below, put the most available contact in the first slot and put the least available contact in the second slot. For example, if you feel 'Bob Rice is most available, write 'B' in the space next to the text 'Most.'

Most_____

Select the most available and least available developer.

Least _____

Appendix 7. Task 3a

Instructions: You have to come into the office this weekend to work on the "MIRTH" project. **Barney Crutcher**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule.

Use Theseus to determine which explanation most likely describes why Barney was unable to deliver on time:

A. He is less willing to devote whatever time necessary in order to finish his work.

B. He may have had to implement new features with last minute notice.

C. He probably had too many tasks in other projects.

D. He might not be a very reliable person when it comes to meeting project deadlines.

E. He could have had to integrate more changes into the build than expected.

F. He might not be agreeable to spending time outside work on this particular project.

G. He probably is not as dependable as others when getting his work in ahead of the due date.

H. He may have got distracted by work in other projects in which he is involved.

<continued on next page>

As you use Theseus, think aloud, verbalize your observations about each individual, and justify your selections.

Below, order each of the explanations from *most likely* (no. 1) to *least likely* (no. 8). Write the letter corresponding to the explanation next to its appropriate rank. For example, if you feel explanation B is most likely, write 'B' in the space next to the number '1.'

Rank the explanations from most to least likely.

1._____

2._____

3._____

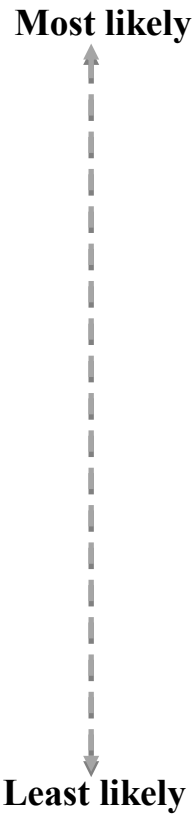
4._____

5._____

6._____

7._____

8._____



Appendix 8. Post-Task Questionnaire: Task 3a

Instructions: Please consider **Barney Crutcher**, the developer in the last task. Recognizing that you may not have all the information necessary to answer each question, fill in the circle (O) above the response that matches your level of agreement.

1. If we decided to meet to discuss work, I would be certain he would be there.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. I could rely on him to e-mail an important message for me if I didn't have time to do it myself.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. I could expect him to tell me the truth.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. I would feel a sense of loss if one of us was transferred and we could no longer work together.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. If he was going to do a job for me and it didn't arrive on time, I would guess there was a good reason for the delay.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I would be comfortable giving him a task or problem which was critical to the project, even if I could not monitor him.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. I can rely on him to respond to my messages in a reasonable amount of time.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. If he promised to do me a favor, he would follow through.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. If he couldn't get together with me as planned, I would believe his excuse that something important had come up.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. If he gave me a compliment, I would question if he really meant what was said.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. He would never intentionally misrepresent my point of view to others.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. If he didn't think I handled a certain situation very well, he would not criticize me in front of other people.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. If I told him what things I worry about he would not think my concerns were silly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. If he unexpectedly laughed at something I did or said, I would wonder if he was being critical and unkind.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. I would expect him to play fair.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 9. Task 3b

Instructions: You have to come into the office this weekend to work on the "MIRTH" project. **Eddie Edelstein**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule.

Use Theseus to determine which explanation most likely describes why Eddie was unable to deliver on time:

A. He is less willing to devote whatever time necessary in order to finish his work.

B. He may have had to implement new features with last minute notice.

C. He probably had too many tasks in other projects.

D. He might not be a very reliable person when it comes to meeting project deadlines.

E. He could have had to integrate more changes into the build than expected.

F. He might not be agreeable to spending time outside work on this particular project.

G. He probably is not as dependable as others when getting his work in ahead of the due date.

H. He may have got distracted by work in other projects in which he is involved.

<continued on next page>

As you use Theseus, think aloud, verbalize your observations about each individual, and justify your selections.

Below, order each of the explanations from *most likely* (no. 1) to *least likely* (no. 8). Write the letter corresponding to the explanation next to its appropriate rank. For example, if you feel explanation B is most likely, write 'B' in the space next to the number '1.'

Rank the explanations from most to least likely.

1._____

2._____

3._____

4._____

5._____

6._____

7._____

8._____

Appendix 10. Post-Task Questionnaire: Task 3b

Instructions: Please consider **Eddie Edelstein**, the developer in the last task. Recognizing that you may not have all the information necessary to answer each question, fill in the circle (O) above the response that matches your level of agreement.

1. If we decided to meet to discuss work, I would be certain he would be there.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. I could rely on him to e-mail an important message for me if I didn't have time to do it myself.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. I could expect him to tell me the truth.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. I would feel a sense of loss if one of us was transferred and we could no longer work together.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. If he was going to do a job for me and it didn't arrive on time, I would guess there was a good reason for the delay.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I would be comfortable giving him a task or problem which was critical to the project, even if I could not monitor him.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. I can rely on him to respond to my messages in a reasonable amount of time.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. If he promised to do me a favor, he would follow through.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. If he couldn't get together with me as planned, I would believe his excuse that something important had come up.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. If he gave me a compliment, I would question if he really meant what was said.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. He would never intentionally misrepresent my point of view to others.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. If he didn't think I handled a certain situation very well, he would not criticize me in front of other people.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. If I told him what things I worry about he would not think my concerns were silly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. If he unexpectedly laughed at something I did or said, I would wonder if he was being critical and unkind.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. I would expect him to play fair.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 11. Task 3c

Instructions: You have to come into the office this weekend to work on the "MIRTH" project. **Teddy Chase**, a software engineer on your team, failed to check in his source-code changes on time, and has not been responsive over e-mail. As a result, you are not able to integrate your new changes into the build, and the project has slipped a week behind schedule.

Use Theseus to determine which explanation most likely describes why Teddy was unable to deliver on time:

A. He is less willing to devote whatever time necessary in order to finish his work.

B. He may have had to implement new features with last minute notice.

C. He probably had too many tasks in other projects.

D. He might not be a very reliable person when it comes to meeting project deadlines.

E. He could have had to integrate more changes into the build than expected.

F. He might not be agreeable to spending time outside work on this particular project.

G. He probably is not as dependable as others when getting his work in ahead of the due date.

H. He may have got distracted by work in other projects in which he is involved.

<continued on next page>

As you use Theseus, think aloud, verbalize your observations about each individual, and justify your selections.

Below, order each of the explanations from *most likely* (no. 1) to *least likely* (no. 8). Write the letter corresponding to the explanation next to its appropriate rank. For example, if you feel explanation B is most likely, write 'B' in the space next to the number '1.'

Rank the explanations from most to least likely.

1._____

2._____

3._____

4._____

5._____

6._____

7._____

8._____

Appendix 12. Post-Task Questionnaire: Task 3c

Instructions: Please consider **Teddy Chase**, the developer in the last task. Recognizing that you may not have all the information necessary to answer each question, fill in the circle (O) above the response that matches your level of agreement.

1. If we decided to meet to discuss work, I would be certain he would be there.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. I could rely on him to e-mail an important message for me if I didn't have time to do it myself.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. I could expect him to tell me the truth.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. I would feel a sense of loss if one of us was transferred and we could no longer work together.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. If he was going to do a job for me and it didn't arrive on time, I would guess there was a good reason for the delay.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I would be comfortable giving him a task or problem which was critical to the project, even if I could not monitor him.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. I can rely on him to respond to my messages in a reasonable amount of time.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. If he promised to do me a favor, he would follow through.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. If he couldn't get together with me as planned, I would believe his excuse that something important had come up.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. If he gave me a compliment, I would question if he really meant what was said.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. He would never intentionally misrepresent my point of view to others.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. If he didn't think I handled a certain situation very well, he would not criticize me in front of other people.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. If I told him what things I worry about he would not think my concerns were silly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. If he unexpectedly laughed at something I did or said, I would wonder if he was being critical and unkind.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. I would expect him to play fair.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 13. Post-Experiment Questionnaire

Instructions: Please consider your use of the Theseus tool today. For each question, fill in the circle (O) above the response that matches your level of agreement.

1. Overall, I am satisfied with how easy it is to use this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

2. It was simple to use this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

3. Using new interfaces is easy for me.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

4. I could effectively complete the tasks and scenarios using this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

5. I was able to efficiently complete the tasks and scenarios using this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

6. I was able to complete the tasks and scenarios quickly using this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

7. I felt comfortable using this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

8. People often tell me that I am skilled with technology.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

9. It was easy to learn to use this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

10. I believe I could become productive quickly using this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

11. The system gave error messages that clearly told me how to fix problems.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

12. Whenever I made a mistake using the system, I could recover easily and quickly.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

13. The information (such as on-line help, on screen messages and other documentation) provided with this system was clear.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

14. It was easy to find the information I needed.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

15. The information provided for the system was easy to understand.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

16. I am usually the first of my family and friends to adopt new technologies.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

17. The information was effective in helping me complete the tasks and scenarios.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

18. The organization of information on the system screens was clear.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

19. The interface of this system was pleasant.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

20. I liked using the interface of this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

21. The system has all the functions and capabilities I expect it to have.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

22. The tool displayed information that was useful in considering others' personal availability and responsiveness to messages.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

23. The interface used visual representations I understood.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

24. The interface used information from data sources I understood.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

25. The tool used relevant information sources from which to make judgments about personal availability and response rate to messages.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

26. Personal availability and one's time to reply are important aspects of collaboration.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

27. The interface helped me compare developers' physical proximity.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

28. The visualizations gave me a good sense of developers' responsiveness.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

29. The interface used appropriate visual representations for the data displayed.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

30. I would rely on the information provided by Theseus.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

31. The individual visualizations in Theseus provide accurate information.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

32. I enjoy using new technologies.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

33. Overall, the tool provides accurate information.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

34. I was motivated to perform tasks in this study.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

35. It is reasonable to expect an e-mail reply within the same day, no matter what location the responder is in.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

36. When comparing responsiveness between a group of developers, it is best to compare developers at the same site rather than developers among all sites.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

37. Assuming I have access to the system, I intend to use it.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

38. My interaction with the system is clear and understandable.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

39. Interacting with the system does not require a lot of my mental effort.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

40. Given that I have access to the system, I predict that I would use it.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

41. Overall I am satisfied with this system.

O	O	O	O	O
strongly agree	agree	neutral	disagree	strongly disagree

Appendix 14. Questionnaire Scoring Keys

1	2	3	4	5
strongly agree	agree	neutral	disagree	strongly disagree

Baseline Trust Scoring Key:

1. For the following items, use the recorded response as the score: Items **2, 10, 13, 16, 18, 19, and 26**.
2. For the remaining items, take the recorded response and covert it. If a 1, score a 5; if a 2, score it a 4; if a 3, keep it at 3; if a 4, score it a 2, and 5, score it as a 1. Do this for the following items: Items **4, 5, 7, 8, 11, 14, 20, 21, 22, 23, 24, 27, and 28**.
3. Add up the points for each item. This total is the score.
4. Higher scores indicate greater Interpersonal Trust. Scores can range from 20 (low trust) to 100 (highest trust), with a neutral score or midpoint of 60.

Task Specific-Interpersonal Trust Scoring Key:

1. For the following items, use the recorded response as the score: Items **14**.
2. For the remaining items, take the recorded response and convert it. If a 1, score a 5; if a 2, score it a 4; if a 3, keep it at 3; if a 4, score it a 2, and 5, score it as a 1. Do this for the following items: Items **1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,** and **15.**
3. Add up the points for each item. This total is the score.
4. Higher scores indicate greater specific interpersonal trust. Scores can range from 15 (low trust) to 75 (highest trust), with a neutral score or midpoint of 45.

Post-Experiment Trust Scoring Key:

Usability

Overall (Items **1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 37, 38, 39, 40,** and **41**)

System Usefulness (Items **1, 2, 4, 5, 6, 7, 9, 10, 38, 39,** and **41**)

Information Quality (Items **11, 12, 13, 14, 15, 17, 18**)

Interface Quality (Items **19, 20,** and **21**)

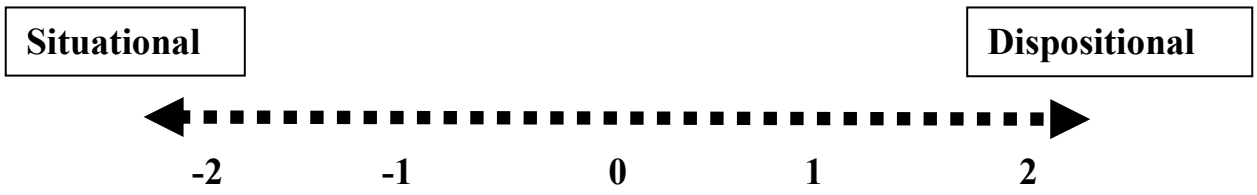
Intention to Use (Items **37** and **40**)

1. Average the scores from the appropriate items to obtain the scale and subscale scores.
2. Lower scores indicate greater usability. Overall score can range from 23(high usability) to 115(low usability), with a neutral score or midpoint of 69.

Model (22-31, 33)

1. Use the recorded response as the score. Lower scores indicate greater value of the design space.

Attribution List Scoring Key



Attribution	Point Value
He might not be a very reliable person when it comes to meeting project deadlines. He probably is not as dependable as others when getting his work in ahead of the due date.	2
He is less willing to devote whatever time necessary in order to finish his work. He might not be agreeable to spending time outside work on this particular project.	1
He probably had too many tasks in other projects. He may have got distracted by work in other projects in which he is involved.	-1
He may have had to implement new features with last minute notice. He could have had to integrate more changes into the build than expected.	-2

4 X (Point value of no. 1 item on scale) +

3 X (Point value of no. 2 item on scale) +

2 X (Point value of no. 3 item on sale) +

1 X (Point value of no. 4 item on scale) +

= TOTAL SCORE

Lower scores indicate situational attributions. Higher scores indicate dispositional attributions. Scores can range from -7 (highly situational) to 7 (highly dispositional), with a neutral score or midpoint of 0.