Impact of Collaborative Traces on Trustworthiness
Erik H. Trainer, Ban Al-Ani, David F. Redmiles
University of California, Irvine
Department of Informatics
Irvine, CA 92697-3425
+1 (949) 824-2901
{etrainer, balani, redmiles}@ics.uci.edu

ABSTRACT
We investigated how trust among software developers would be affected by providing them with visualizations of collaborative traces. We define collaborative traces to be representations of the past and current activity of a group of developers manipulating software development artifacts. In this paper, we report two main findings. First, we report the results of our controlled experiment in which collaborative traces were visualized. Second, we present an overview of tools which aim to represent collaborative software engineering traces. Our experiment provides evidence that collaborative traces can support the development of several factors of trust identified in our field study. However, we also identified some shortcomings of our current visualizations, gaining insights into future improvements. From our review of tools that represent collaborative traces, we observed that such representations can drive the design of tools that aim to support trust. We also present a table of tools; the table can be used to guide discussion and the design of tools that promote trust in software development.

Categories and Subject Descriptors

General Terms

Keywords
Trust, distributed teams, collaboration, dependency, awareness, cooperation, coordination, tools.

1. INTRODUCTION
Our research was motivated by the following research question: Can collaborative traces influence the sense of trust in virtual teams? Collaborative traces are representations of past and current activity of a group of developers manipulating software development artifacts: for example, project meta-data such as the distribution of source-code authorship, work item and module assignments, and source-code call-graphs annotated with authorship, as well as events such as real-time source-code changes in progress, source-code check-ins, bug fixes, and developer-developer correspondence such as postings from mailing lists and discussion forums. Collaborative traces lie hidden in project repositories and activity logs. They are typically not easily accessible without tool support.

Ariadne is a tool we developed to support virtual teams (VTs) by identifying and visualizing interdependencies between team members. The typical team involves many individuals occupying a variety of roles, performing activities over many months or years working in various locations across multiple time zones. Over time, many interdependencies arise between them. For example, in our previous work, we observed that a program’s source-code call-graph implies interdependencies between the developers implementing that code [11]. Developers are aware of these interdependencies, but only to a limited extent. Ariadne provides developers an enhanced awareness of interdependencies using a visual approach. Interdependencies are one example of collaborative traces; they describe relationships between developers based on the source-code they implement.

Trust is often an issue for virtual teams because members may not have had experience working with one another and are typically self-directed [37]. They also have to adapt to the diversity in culture and geographic backgrounds associated with such teams. We sought to investigate the impact of collaborative traces Ariadne provides on developers’ sense of team members’ trustworthiness and conducted a pilot laboratory experiment to this end.

Our initial findings provide evidence that visualizing collaborative traces can influence individuals’ sense of trust toward other developers. Ariadne’s visualization of developer interdependencies led participants of our pilot study to gain a sense of who could be considered trustworthy. However, each of the participants remarked they would feel more comfortable trusting developers if they could see more traces than currently visualized in the interface.

In the next section, we define the concept of trust in more detail and report on some of our initial findings of forces that influence trust in distributed teams. We then describe the experiment conducted using Ariadne visualizations and introduce collaborative traces that can be used to illustrate factors that promote trustworthiness. We discuss implications for tools that promote trust and conclude.
2. TRUST IN PERSPECTIVE

Highly productive development environments are realized by human and cooperative aspects of work, such as the beliefs among individuals that their peers will consistently meet expectations and self-coordinate their work to ensure the smooth coordination of development activities. For developers this may mean believing that code producers will notify them when the APIs they use change or that their peers will send them messages warning of check-in conflicts to the repository. For team leads and project managers this may mean relying on a developer’s ability to commit bug-free code on time.

2.1 An Understanding of Trust

There are many definitions of trust which have been proposed within different contexts [8][39][21]. Our synthesis of these definitions led us to conclude that trust can be considered the belief that the trustee will meet the expectations of the trustor. Expectations that form as a result of actions and behaviors, in turn, influence the level of trust as we have defined it.

It is useful to think about these expectations from two perspectives of trust: 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 be vulnerable.

Affective trust refers to trust that arises from emotional ties among group members and reflects beliefs about reciprocated care and concerns [39]. For example, a senior software developer who has implemented the majority of a software module or set of modules and who rarely introduces bugs will gain another programmer’s cognitive trust, while a programmer who frequently meets deadlines, is quick to respond to other developers’ questions, and sends e-mails to coordinate check-ins will gain another programmer’s affective-based trust.

Trust plays a crucial role in innovation, efficiency, and effectiveness, as team members will be less likely to cross-check each other’s work [8][39][21]. It can promote open exchange and information sharing because people are more likely to collaborate with individuals they trust [21]. Distributed team members are often characterized by a diversity that can lead to innovation in collaborations [26][28][33][18], yet trust must exist for team members to feel that they can openly express innovative ideas. Diverse teams typically need more time to establish trust but can eventually outperform more homogenous teams [18].

2.2 Trust Forces: Initial Field Study

In a previous study of leadership, communication and task allocation in a Fortune 500 company in the United States, we identified four initial forces that appear to impact trust: team size, type of project, team diversity, and leader characteristics, with time as a common thread running through all these forces [2][3].

Our data indicated that good leadership and an adequate time allocation can act as two forces that positively influence trust in a virtual team. Conversely, a large team size, high team diversity, and a challenging project deliverable can have negative influences on trust within a virtual 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. The key to maintaining equilibrium, at least, is to reach a state such that the trust level remains positive. If the level of trust within a team sinks below a certain level, it can imply that the team is not performing to the best of its ability e.g., limited effectiveness and innovation.

Figure 1 models the impact of the forces indicated by our initial field study findings. The diagram illustrates that trust is the end result of the net forces acting on it. The trust threshold represents the minimum level of trust necessary for virtual developers to collaborate effectively as a team. However, we recognize that forces we have identified were dictated, and thus limited, by the scope of the field study. The limitations of the study stem from the limited number of participants interviewed and that trust was not the primary focus of the study. Our review of the literature in other fields of study led us to surmise that there are many other forces that can influence trust, and this assumption motivated our proposal to further investigate these forces.

3. TRUST TOOL SUPPORT: ARIADNE

Based on our current understanding of trust, we anticipate that trust leads to greater effectiveness and increased collaboration; thus the lack of collaboration can mean that trust is lacking. One way to detect a lack of collaboration is through lack of interdependency emerging over a period of time.

Our previous work includes the development and evaluation of tools that support coordination activities amongst VT members [3][27]. Such tools can help overcome issues related to communication, context, and awareness that are missing in distributed teams. They give developers visual information at an appropriate time. Their general purpose is to raise awareness among users to better organize and understand their respective tasks and self-coordinate their activities to avoid situations where their work obstructs or interferes with others’ activities. Ariadne is one such tool. Its purpose is to support a VT by identifying and visualizing interdependencies amongst team members.

Trust can also be managed by managing developers’ expectations of others in their team. We anticipate that visualizing interdependencies can lead to an understanding of other team members’ workloads and their ability to meet commitments. Establishing realistic expectations of VT members’ abilities can increase the likelihood of their meeting these expectations and increasing trust in these teams.

Visualizing interdependencies of previous projects can also provide collaborative traces. In this instance, it can indicate the likelihood of future collaborations and whether team members can play an integral role in the development process. We sought to explore these possibilities through a controlled experiment by utilizing Ariadne’s visualizations to represent collaborative traces in previous projects.

3.1 Visualization Approach

Ariadne is a visual tool that infers dependencies between virtual teams based on the modules they author (shown in Figure 2). Ariadne visualizations allow developers and managers to identify relevant collaborative traces central to their coordination needs. Essentially, Ariadne calculates a sociogram representing
dependencies between developers through the modules with which they work [38].

![Image](Figure 2. Source-code on the horizontal axis with dependencies represented by brackets.)

To take advantage of available screen real estate, the visualization lays out dependency information in a type of table-based fashion placing the most numerous data items along the longest screen dimension. Called code units occupy the x-axis and authors occupy the y-axis, with both ordered alphabetically by default. Users can reorder the axes based upon queries against all the data and its associated meta-data. We draw connections from a dependent author to the code unit they are dependent upon and back to the author responsible for that code unit (see Figure 2) and repeat for each set of socio-technical dependency information in the project. This visualization technique preserves the ease of identifying connections in common social network diagrams. Developers can readily determine inbound and outbound connections by glancing at the y-axis.

3.2 Trust Experiment

We conducted a pilot experiment to investigate a central research question: can visualizations of distributed collaborative traces influence a developer’s sense of team members’ trustworthiness?

We chose to utilize the visualizations created by Ariadne to explore whether these visualizations can play a positive role in managing the expectations of participants based on the role developers represented in these visualization played. We also found that both cognitive and affective trust was influenced by the visualizations.

We recruited six UCI Informatics graduate students by sending a department-wide e-mail announcement that stated we were conducting a study of virtual teams and software development with the stipulation that participants should have programming experience, professional or otherwise. Five males and one female responded to our call for subjects. Two participants had three to five years experience developing software in a professional organization. Participants ranged from 25 to 34 years of age. The experiment was conducted over a period of one week, with the average session lasting 35 minutes.

Each participant was presented with two scenarios and corresponding visualizations displayed on a series of three monitors (Figure 3). The purpose of the first scenario was to explore the impact of Ariadne when a developer has no prior knowledge of others they need to collaborate with. The second scenario was designed to allow us to explore whether Ariadne can manage a developer’s expectations of others within their own team.

![Image](Figure 3. Visualization configuration for pilot experiment.)

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, namely: Pierre, Toulouse, France, Ajith, Bangalore, India, Evan, Victoria, Canada, Gregor, Frankfurt, Germany, Gavin, Dublin, Ireland, Van Ha Noi, Vietnam, Koji, Osaka, Japan, Lars, Uppsala, Sweden.

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.

When participants made their choice they typically chose members they felt had the most interdependencies with others in the team as well as individuals who were located in the same time zone. When the participants were asked how they made their choice; they indicated that the visualization and the representation of interdependencies in these visualizations led them to trust the central figure (e.g., Figure 4) because “they know a lot of the code based on the incoming connections to them”.

When asked which developer would be less likely to meet commitments i.e. who engendered less affective trust, participants typically chose those who they felt were contributing less to the team because it meant “we don’t know what he’s done” (Figure
5). Their identification of central and non-central figures was made easier through Ariadne’s visualization.

![Figure 4. Central developer “evan” has incoming dependencies from every developer in the project.](image)

Ariadne can motivate communication on large, diverse teams by revealing both social and technical connections. Knowing that a person worked on a particular module or module series can help build trust in that person. Knowing they’ve worked with a colleague can also help. The visualization may also help to reduce the time team members expend monitoring each other, backing up or duplicating work, and documenting problems. Such defensive behaviors often occur in teams that lack affective or cognitive trust.

4. COLLABORATIVE TRACES

Software design and development activities typically produce a “project memory,” [10] of archival artifacts such as source-code, e-mail lists, design documentation, problem reports and change histories of all this information that developers use as a means toward coordinating the smooth flow of work. As illustrated in the pilot experiment described previously, Ariadne uses source-code dependency call-graphs annotated with authorship information at the line level for each artifact (e.g. package, class, method) fetched from the project’s Configuration Management (CM) repository.

A host of visualization tools have been created and empirically evaluated. They have been found to be useful for different technical and social (socio-technical) aspects of information gathering activities. In earlier work [36] we looked at 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, and recommending the right people and the right artifacts at the right time to match the task at-hand, determining opportunistic times to contact developers (and the latter’s willingness, in turn) for help, all of which change over the course of development.

In summary, the general purpose of these tools is to raise awareness amongst software developers, not only to better organize and understand their respective tasks, but also to self-coordinate their activities to avoid situations in which their work threatens to obstruct or interfere with the activities of others. Such tools allow individuals to monitor each other, especially when they share dependent tasks. Research has found that not providing such awareness can increase the chance of behavioral invisibility and the possibility of reducing trust [39].

An important property of these tools observed in our own research and others (e.g., [34]) is that they use, what we term, collaborative traces. Collaborative traces are representations of past and

3.3 Ariadne and the Trust Forces

Social network awareness provided by Ariadne adds to the information in normal system networks to increase familiarity among team members. Although this familiarity is based on team members knowing the parts of the larger system they’re responsible for, it’s a start at increased familiarity, which can help ameliorate negative trust caused by project size and team diversity. Ariadne also supports leadership by providing an interactive analysis tool that allows managers to adjust team members’ responsibilities. For instance, when a few team members have many responsibilities, a manager can redistribute those responsibilities.

In the second scenario the participants were presented with two new Ariadne visualizations which represented the interdependencies in the team the participants were currently assigned to, and a different list of developers. These developers and their locations included: You, Seattle, Washington, Jean-Luc, Montreal, Canada, Pradeep, Calcutta, India, Alberto, Lisbon, Portugal, Ming, Shanghai, China. 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, based on the Ariadne visualizations, 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.

The data gathered from this scenario regarding questions pertaining cognitive and affective trust was inconclusive. When asked about whose work participants would cross-check for accuracy, as well as who they would rely on to not break their code, many participants replied they would need more information. Many participants remarked they would need to see developers’ code or test case success rates. Some respondents asked to see the developers’ activity in other ongoing projects when their contributions to the current project (i.e. not many incoming dependencies) seemed small.

However, some participants stated that they were less likely to risk relying on certain team member based on their inactivity because they “might break my code because they are inactive and not collaborating with me. Since they are not collaborating they might not have the same practices as me and more is left up in the air”. Whereas others stated that they would trust others with their code because “If someone breaks it, it probably wasn’t that great to begin with.”
Table 1. An overview of tools and the collaborative traces they use.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Ex. Tools</th>
<th>Collaborative Traces</th>
<th>Input Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand Change Management and Evolution</td>
<td>ELVIN [16], Command Console [25], softCHANGE [17], SeeSoft [14], Tesseract [29]</td>
<td>Change-sets/commit logs and associated meta-data (e.g. authorship, size, descriptions, and time), file revisions, source-code call-graphs, source-code authorship, bug reports, e-mails</td>
<td>CM Repository, E-mail Database, Mailing List, Bug Tracker</td>
</tr>
<tr>
<td>Understand Developers' Activities</td>
<td>FASTDash [7], SHO [15], SeeSoft, Tesseract, Jazz [19]</td>
<td>Changesets/commit logs and meta-data, file revisions, source-code call-graphs, source-code authorship, work items, bug reports, ChangesInProgress, e-mails</td>
<td>CM Repository, E-mail Database, Mailing List, Bug Tracker, Workspace</td>
</tr>
<tr>
<td>Find Relevant People and Artifacts</td>
<td>ExpertiseRecommender [24], Answer Garden [1], Mylar [22], Team Tracks [12], Tesseract</td>
<td>Change sets/commit logs, source-code call graphs, source-code authorship, bug reports, developer navigation patterns, developer profiles, e-mails</td>
<td>CM Repository, E-mail Database, Bug Tracker, Employee Directory, Expertise Database, Organizational Chart</td>
</tr>
<tr>
<td>Determine Availability</td>
<td>CommunityBar [35], Jazz, Awareness [6]</td>
<td>Idle time, artifacts in current workspace, location coordinates, meetings in progress</td>
<td>GPS, Keyboard Input, Video, Shared Calendar, Workspace</td>
</tr>
</tbody>
</table>

current activity of a group of developers manipulating software development artifacts: for example, project meta-data such as the distribution of source-code authorship, work item and module assignments, and source-code call-graphs annotated with authorship, as well as events such as real-time source-code changes in progress, source-code check-ins, bug fixes, and developer-developer correspondence. Collaborative traces lie hidden in project repositories and activity logs. They are not easily accessible without tool support. Tools can derive collaborative traces from development environments (e.g. Eclipse and Jazz workspaces), personal exchanges (e.g. e-mail and instant messaging), and project repositories (e.g. employee directories, organizational charts, source-code repositories, work item databases, and mailing lists). Table 1 lists some of the tools that have been developed to raise awareness and the collaborative traces they use. Note that this list is not exhaustive. It simply serves to illustrate examples of collaborative traces.

4.1 Discussion: Collaborative Traces that can Support the Development of Trust

Given our understanding of forces that influence trust from our field study, our pilot experiment, and the observation that collaborative traces of developer activity drive many of the tools that support awareness, we wish to close the loop by mapping collaborative traces to the factors that influence trust. In other words, we wish to identify collaborative traces for engendering trust in virtual teams. Table 2 lists example factors that influence trust we have found by conducting initial fieldwork and a literature review on trust. We have begun to fill out the collaborative traces that may provide information about these factors and whether they positively or negatively affect trust, consistent with our “net force” model presented earlier (see the “valence” column in the table). This list is not exhaustive, but rather ongoing work which we are refining incrementally.

Visualizing only a small subset of collaborative traces will not tell the whole story. Ariadne’s visualization could actually breed distrust, perhaps incorrectly founded. For example, participants in the pilot experiment used interdependencies from a call-graph 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. In the absence of other information about the developers in the project, participants’ perceptions of them may have been overly influenced by Ariadne’s graphical representation of interdependencies. It is likely that these “untrustworthy” developers were involved in other software development activities, which interdependencies alone cannot convey.

Collaborative traces that provide information about factors known to influence perceptions of trustworthy may fill in these gaps. Change-sets and authorship can reveal information about developers’ expertise and their general development activity. 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. Seeing that a known-to-be trustworthy colleague frequently uses source-code written by another developer can be an indicator of the latter’s reputation. Similarly, the fact that a developer has authored over 80% of a key module can be an indicator of positive reputation. Developers who 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. We hope to empirically verify these claims in our future work on collaborative traces.

We anticipate that the transparency into activities provided by collaborative traces such as these can help others formulate expectations regarding both cognitive and affective trust. However, we also foresee various challenges of this work.
Table 2. A summary overview of collaborative traces which can engender trust in distributed teams.

<table>
<thead>
<tr>
<th>Trust Factors</th>
<th>Valence</th>
<th>Collaborative Traces</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>+</td>
<td>Developer profiles, employee directories</td>
<td>[20]</td>
</tr>
<tr>
<td>Frequent initiations</td>
<td>+</td>
<td>Chat threads, e-mails, mailing list postings, instant messages, discussion forum postings, work item statuses</td>
<td>[21][20][9]</td>
</tr>
<tr>
<td>and responses</td>
<td></td>
<td>Change sets/commit logs, developers’ interdependencies, e-mail messages, work item statuses, mailing list postings, calendars</td>
<td>[21]</td>
</tr>
<tr>
<td>General activity</td>
<td>+</td>
<td>Change sets/commit logs, developers’ interdependencies, work item statuses</td>
<td>[40]</td>
</tr>
<tr>
<td>Frequent updates of</td>
<td>+</td>
<td>E-mail messages, mailing list postings, source-code call-graphs, work item statuses</td>
<td>[21]</td>
</tr>
<tr>
<td>project progress</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reputation</td>
<td>-</td>
<td>Change sets/commit logs, developers’ interdependencies, work item assignments, source-code authorship, developer profiles</td>
<td>[26]</td>
</tr>
<tr>
<td>Vague responsibilities</td>
<td>-</td>
<td>Work item assignments, employee directories, project specifications, organizational charts</td>
<td>[8][39]</td>
</tr>
<tr>
<td>Use of multiple</td>
<td>+</td>
<td>E-mails, chat threads, mailing list postings</td>
<td></td>
</tr>
<tr>
<td>communication media</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expertise</td>
<td>+</td>
<td>Change sets/commit logs, source-code authorship, developer profiles, work item assignments and descriptions</td>
<td>[4]</td>
</tr>
</tbody>
</table>

4.1.1 Managing Collaborative Traces Over Time

There may be many, many collaborative traces generated over the lifetime of a software project. This presents a challenge for identifying the right traces at the right time. We anticipate that gaining one’s cognitive or affective trust may take time. For example, collaborative traces over the entire history of a project might be needed to determine one’s reliability and whether or not they typically wait until the last minute to deliver source-code that others depend on to get their own work done. In contrast, someone who commits source-code to the repository daily without bugs may gain another developer’s cognitive trust within a few weeks. Grouping collaborative traces into finer grained temporal segments that correspond to important development events such as “daily commits”, “weekly builds”, and “project releases” may help. In other instances, team leaders and project managers may choose to trust others simply by nature of their role or years of experience in the organization.

4.1.2 Different Views for Different Users

A related question is how to visually represent collaborative traces for different users of the tools such as project managers and researchers, and developers. For example, the Ariadne visualization uses abstract representations of interdependencies rather than the source-code itself. This view may be good for project managers or team leaders who want a high-level view of the coordination efforts of their development teams in order to gauge the status of contributions. Yet developers may be more interested in the details of the contributions themselves: what the source-code looks like and overall what work was done at a finer level. Instead of a view of the whole project, they may be interested in the source-code relevant only to the development task at hand, to see whether the work represented by the collaborative trace matches what was specified in a work item description for example. Moreover, whether the visualization exists on a large, shared display or is integrated with the development environment should be a concern for researchers interested in visualizing collaborative traces for trust.

4.1.3 Anticipating Threats to the Value System

Choosing particular renderings of collaborative traces for trust will likely present challenges to the value system of a development organization. Tools that visualize collaborative traces create a transparency not usually afforded by typical development tools like Configuration Management systems.

Researchers exploring visualizations of collaborative activity (e.g., [5]) have noted that the renderings of those visualizations can create opportunities for gaming existing incentive systems. In the Ariadne visualization, developers like ‘gregor’ can make themselves look good to others by adding a line or two to a core software module used by many developers on the team, such as a library, even if the contribution is functionally trivial (e.g., a comment or initialization of a field). Providing finer grained visual abstractions, such as a zoomable window that displays a readable view of the calling or called source-code in question whenever an individual dependency bracket is selected may help limit these misrepresentations.

4.1.4 Propensity to Trust

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 [21][23].

Nevertheless, we have noted in our interviews with distributed team members 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. Collaborative traces provide an interface to such information.

5. CONCLUSION

Trust is a critical component of successful and productive collaborations across many disciplines. Indeed, participants in our field study of distributed teams cited trust as a key concern. In response we analyzed our field data to find a set of four forces that act to influence the development of trust.

We conducted a pilot experiment with Ariadne to investigate how trust might be supported with visualizations of collaborative activity. Ariadne visualizes dependencies between developers based on the artifacts they use. Dependencies are one way to measure the extent of collaboration taking place. A lack of collaboration can imply a lack of trust. We found evidence that the tool can engender trustworthiness toward developers whose code is highly depended upon by others. Yet our experiment had shortcomings with respect to reliability: it was tested with a small number of subjects who were provided with limited information about a fictitious team. In terms of the study’s results, participants desired to see information we did not happen to show.

We observed that additional information can be provided by collaborative traces, representations of past and current activity of a group of developers manipulating software development artifacts. As such, we have begun to identify collaborative traces corresponding to known factors that influence trust, which represents a starting point for researchers interested in engendering trust with tool support. However, researchers should identify appropriate renderings of the traces for different users that stay true to value systems of development teams. In our future work, we wish to explore these issues and improve upon our pilot study.

6. ACKNOWLEDGMENTS

This work is supported by the U.S. NSF under grants 0943262 and 0808783. We thank pilot study participants and colleagues for their advice, especially Matt Bietz and Steve Abrams. We thank Cleidson de Souza and Stephen Quirk for early work on Ariadne.

7. REFERENCES

Evidence of the Benefits of Workspace Awareness in
Sarma, A., Redmiles, D., van der Hoek, A.