You Know When It's Not There: How Trust Enables and Enhances Collaboration

by Ellen Gottesdiener

Great products are built by teams that collaborate in a healthy and transparent manner, and *trust* is a key ingredient of effective collaboration.

On a software project, trust is needed between business and technical people. Businesspeople must trust technical staff with their time and money to build the right thing as quickly and cost effectively as possible. Technical staff need to trust customers to define their needs, renegotiate agreements when needs change, and interact throughout the project. Sponsors must trust product "owners" on the team to represent the business and user needs. Developers have to trust analysts to get the right requirements as quickly and clearly as possible. Testers have to trust developers to provide clean-enough code to begin system or integration testing. Project leads must trust team members to keep their word on delivery commitments — and so on.

The good news is that trust isn't "a feeling" that "just happens." Rather, it is the result of specific actions team leaders can take in a systematic way. Here I explore practical techniques that help you build and sustain trust on your teams.

WHAT IS TRUST?

It seems almost silly to define a word as simple as *trust*, but like a lot of basic concepts, it gains clarity and depth as you think it through. Trust is about a relationship in which people rely on each other. Trust implies an interpersonal dynamic — in other words, your belief that the trusted person will look out for your best interests in a specific area. You have faith that the trusted party is honest and sincere, will fulfill his commitments, and will act in a way that's congruent with his words.

Congruence is a crucial element. If a business partner tells me she will keep me informed about changing requirements but later insists on changes I never knew about, this is not congruent behavior. If the team architect says in a planning meeting that he will present a demonstration of a portion of the application in two weeks but then shows different software without informing me of the change, there is no consistency between his words and his actions. I am likely to deem both people untrustworthy.

You know trust. You know when it's not there.

THE HIGH PRICE OF LOW TRUST

Lack of trust is harmful to teams, organizations, and the bottom line. Low trust levels result in project churn, team turnover, low morale, and unhappy customers. Evidence of low trust can be found in hidden agendas, rumors, gossip, whining, and "subversive" stakeholder behavior.

In studying failed projects, Johann Rost explored the darker side of failure and found patterns of subversive behavior by team members [12]. These behaviors included lying, withholding or delaying access to information, providing misleading or vague answers, and supplying poisoning ideas. Such subversive behavior can be attributed to personal goals conflicting with organizational goals, power plays, attempts at retribution, rivalry, and more. The bottom line is that these projects suffered from lack of trust.

THREE TYPES OF TRUST

Organization development experts Dennis and Michelle Reina identified three types of trust that are important in organizations [11]:

1. Contractual trust
2. Communication trust
3. Competence trust

With *contractual trust*, the entire team consistently understands goals, and everyone shares a common understanding of roles and responsibilities. Boundaries are clear. Team members have mutual interests; you will look out for my best interests, and I will look out for yours, because we succeed together. If things change,

we renegotiate. Individuals fulfill specific responsibilities, but we are jointly responsible in the end.

Communication trust involves honest and frequent communication. Teams engage in truth-telling, and with good purpose. Team members admit their mistakes, maintain confidentiality, and seek and provide feedback. Their actions are congruent with their words, and vice versa.

When *competence trust* exists, team members respect one another's ability to fulfill their mutual responsibilities. Because they succeed together, team members rely on mutual learning and development and actively seek each other's input to achieve objectives. They help each other. They honor agreements. They respect one another's skills and knowledge.

WHEN TO BUILD TRUST

The time to build trust is at the very beginning of a project, release, or iteration, when you decide what to build — the process Frederick Brooks called "the hardest single part of building a software system" [1]. This is when the team charters the work, defines its scope, and begins to elicit requirements. In my experience, a team's ability to explicitly establish the three types of trust — contractual, communication, and competence — at the start is an early predictor of project success.

Product requirements are emergent and dynamic. Effective requirements — the right requirements delivered at the right time, developed efficiently — require intense human interaction. Trust is essential. Trust sets the stage for effective requirements, and developing effective requirements builds trust.

HOW TO BUILD TRUST

Now let's look at key actions you can take to build trust within your teams.

Define the Product's Vision and Scope

Although IT may coordinate or facilitate this activity, it must be business-driven and involve all the project stakeholders. The high-level *product vision statement* tells team members what they need to build, for whom they need to build it, and how it's different from what exists today.

The vision statement sets the stage for contractual trust by establishing shared goals for business and technical staff. To get this right, you must have the right people (stakeholders) participate in defining the vision and the requirements needed to fulfill it.

A statement of scope clarifies the "contract" about what to build, at least for this iteration or release. Clear boundaries are necessary for contractual trust. Capers Jones estimates that 80% of projects are at risk for "scope creep" [7, 8]. Changes to requirements are normal and useful in many projects, but unconstrained or uncontrolled change — scope creep — is not healthy.

If the team can't agree on the product's vision and scope, you are not ready to plan and build the product. The most successful outcome is to cancel the project.

Involve the Stakeholders

Stakeholder involvement is probably the most important element of a successful software project. No matter what method you're using to develop the product requirements, this is the point in the project when the needs and interests of all stakeholders converge. To build trust, you need to involve product owners, developers, subject matter experts, business analysts, project managers, and specialists in testing, QA, auditing, regulatory issues, training, sales and marketing, and more.

To elicit requirements, you identify stakeholders and other sources for requirements information, plan a strategy to involve the stakeholders effectively, and then collaborate to explore, discover, and acquire sufficient requirements to begin development. During project startup, you need to make smart choices about whom to involve and how to engage them. This practice builds both contractual and communication trust.

It's best to use a combination of elicitation techniques, such as prototyping, facilitated workshops, focus groups, document analysis, and research-based approaches (such as surveys and competitive analysis). Using multiple techniques helps the team build competence with requirements and taps the competence of various stakeholders. This approach also improves the quality of the team's requirements-related conversations and documentation, building communication trust.

Start by holding a scoping workshop to identify your stakeholders based on the product vision. Next, define the product scope and a stakeholder involvement strategy [5]. Understanding these players, the ways they will be involved in the project, and the ways you can tap their expertise touches all three types of trust: contractual, communication, and competence.

A special word to agile teams: in the agile approach, you typically acquire requirements primarily through the product owner and then revisit requirements continually. If you're using this approach, the product owner may make decisions about priorities for iterations and releases. Still, you may need to use varying techniques

to access other types of stakeholders to more completely advise and inform your understanding of requirements.

Keep a Glossary

The *glossary* defines the meaning of the business terms in your product domain. These terms are the basis for all project and product communications and product requirements. A glossary provides a shared vocabulary — a fundamental building block of trust as well as the key to effective requirements. Project teams need to speak the same language, thereby building competence and communication trust.

Consider how many times you've participated in discussions in which people were using the same term differently or using different terms to mean the same thing. This wastes time and causes conflict and confusion. One shared glossary of business terms, which evolves throughout the project, should be the single point of meaning for all stakeholders.

Set Criteria for Prioritizing Requirements

Not all requirements are created equal, and not all requirements are understood equally well. Teams need an explicit, agreed-upon method for evaluating and filtering requirements so that they can make smart, justifiable choices. This practice both requires and builds contractual trust.

Some teams rely on ranking schemes, such as "high, medium, low" or "must, should, could, won't" (i.e., MoSCoW prioritization). In such schemes, the meaning of the ranking is not explicitly defined. These techniques might be sufficient for lower-risk or simpler projects, but for larger, more complex projects, a simple ranking scheme is insufficient. In those cases, you should clearly and precisely define what each ranking means in terms of business value.

For example, on one data warehouse project I worked on, we needed to rank certain data or data groups according to their priority. In our first requirements workshop, we defined our "must" ranking: it meant that if the data or data grouping was not available to view or report on, the company would be violating an external regulation (thereby putting the business operations at risk). We also established definitions for the other two priority levels. This practice allowed us to quickly rank specific data during our workshops.

Your prioritization criteria amount to a contract for prioritizing requirements, and they establish boundaries (and thus contractual trust) for the work that needs to be completed at a particular point in time. The criteria also provide a basis for healthy team communication about competing needs.

Tailor your prioritization criteria to your project. In my example, the objective was to avoid regulatory violations and fines. On your project, the criteria might include losing market share, risking a large amount of money, increasing operational efficiencies, minimizing or maximizing organizational impact, achieving a fast return on an investment, and so on. Obtain agreement among the business and technical stakeholders before applying the ranking scheme to your project requirements.

Make Decisions Transparently

Defining prioritization criteria sets the stage for, but is not a substitute for, explicit decision making. How many times have you walked out of a meeting unsure whether a decision had been made? Or heard people dissing the decision at the water cooler? Or realized that you disagreed with the decision but had never been given a chance to voice your opinion? In these kinds of cases, decision making is opaque.

Transparent decision making is essential for communication trust. Software projects involve a myriad of decisions, and participatory decisions *work* [10]. Healthy teams use a transparent, participatory decision-making process for high-stakes decisions.

Requirements work is about continually deciding which requirements will be built to achieve business value. You must decide what is in scope and what is out of scope; which requirements you will build for the next iteration; which user community you are targeting — that is, which users' needs will be favored; and how much training and self-help you will build into the product. In addition to product requirements, you must define what the length of iterations or releases will be, who will do what work, how to engage stakeholders, what tools to use, and so on.

To make these important decisions, teams need *decision rules* and a protocol for decision making. I call such a collaboration pattern Decide How to Decide [3]. This pattern provides a repeatable mechanism teams can use to reach closure by explicitly defining what the decision rule is and how the team will participate.

I use a checking-in tool called a "gradient of agreement" that I first learned about from Sam Kaner [9]. With time and experience, I have modified the gradient and checking-in process. I demonstrate it in workshops and encourage teams to use it for all their participatory decisions. When participatory decisions are used and the decision maker explicitly consults with stakeholders

who will implement the decision, you improve both the sustainability and the quality of your decisions.

Transparent decision making is a powerful way to build trust early in a project. It builds and sustains all three types of trust: contractual (you have a contract for reaching closure), communication (decision making is explicit and everyone owns the process and outcome), and competence (decisions are made with input from knowledgeable stakeholders). This approach also builds competence in decision makers.

Play Around

Building prototypes and creating multiple interweaving requirements models at the start of each iteration, release, or requirements phase will increase team competence and provide for rich communication. In his book *Serious Play*, author Michael Schrage reveals the role of "play" in product innovation [13]. Playing around with models and prototypes unleashes ideas, confirms needs, and feeds successive revisions of products, enhancing both competence in the domain and communication among business and technical stakeholders.

Work the Wall

In this technique, you post the team's work — plans, charts, requirements and design diagrams, test status, and more. It's about making the team's work transparent and visible. Alistair Cockburn refers to wall work as "information radiators" [2], and Ron Jeffries discusses the use of big visible charts [6].

Wall work ensures that communication is open and visible, thus building communication trust. It also allows team members to learn about the work of others, building competence trust.

No More Meetings

Typical meetings are a waste of time, money, and energy. They are poorly planned (if at all) and inadequately monitored, have vague goals, do not reach closure, and might even have the wrong attendees.

Meetings like these are true trust-busters. They violate all three types of trust: contractual (the purpose is unclear), communication (the meeting processes are ineffective), and competence (the needed participants aren't there, and communication is one-way and results in little mutual reliance and learning).

My advice? Don't have typical meetings. Instead, build trust by communicating in these ways:

- Daily stand-ups (e.g., daily Scrums) to obtain status, make work commitments, and uncover potential obstacles
- Demos or prototypes to share and get feedback on a slice of product functionality
- Reviews or inspections to detect errors or defects in a work product
- Facilitated workshops to produce work products such as plans, requirements, designs, and retrospective findings (see sidebar on page 10)

These types of group sessions have three things in common:

1. Protocols for team interaction, which serve as a contract for the process and the outcome
2. Process guidance (via a facilitator, moderator, project leader, or ScrumMaster) to ensure healthy communication
3. Prework for strong contracting and for building team competence (except in the case of daily stand-ups, which do not require prework)

These elements exploit the positive power of group work and provide a framework for attaining the glorious group situation in which, as Aristotle said, "The whole is more than the sum of its parts; the part is more than a fraction of the whole."

Hold Retrospectives

Retrospectives are about team self-reflection on product and process for a predefined time frame (such as an iteration, a release, a milestone, or a project). The team inspects real project data and engages in thorough reflection in order to learn and adapt.

Retrospectives are a powerful mechanism for kick-starting and sustaining trust. Indeed, as agile teams know, the retrospective ritual is integral to the agile process. You are not "doing agile" unless you're doing retrospectives at the end of each iteration.

Retrospectives facilitate transparent communication and joint accountability (which is part of contractual trust). These sessions also enable and build individual and team competence.

THE TIME IS NOW

For many project teams, the hardest part of software is the soft stuff. They struggle because they lack an essential element of healthy collaboration: trust. Leaders of successful teams know that they need to build three

types of trust: contractual trust, communication trust, and competence trust.

What can you do today to build and sustain trust on your team?

BUILDING TRUST THROUGH FACILITATED WORKSHOPS

With facilitated workshops, I contract for the event beforehand using a planning team to balance the needs of all stakeholders [4]. To guide the planning, the team and I follow a planning and design framework called "The 6 Ps":

1. Purpose
2. Participants
3. Principles
4. Products
5. Place
6. Process

In this way, we build contractual and communication trust before people get together in the same room at the same time.

When you plan a workshop, seek ways to build trust. For example, to ensure transparent communication, actively try to identify hidden agendas people may have. Additionally, delineate the decisions and the decision-making process you'll use in the workshop.

During the workshop, the facilitator helps the team adhere to the ground rules and stay focused on the workshop's purpose. Through a well-planned process, the facilitator helps the team become competent in the requirements, design, plan, or other work products.

You also build trust in workshops by letting conflict surface. Conflict is normal when humans interact and should be viewed as supplying energy and opportunities. Pushing down or ignoring conflict or disagreements promotes distrust. Group competence grows when you explicitly address conflicts. At the same time, the group learns to deal with conflicts, such as competing priorities, in an appropriate manner.

After a workshop, you need to ensure continuity in communication and competence for nonparticipating stakeholders. You do this by sharing outcomes. I recommend conducting a show-and-tell to allow sponsors and key stakeholders to learn from the team, thus increasing management competence. The show-and-tell also builds transparent communication up and down; it builds trust with senior management. If the group's work is on the wall, it is transparent as well.

Note: If you cannot hold these gatherings at the same time in the same place because members are physically distributed, you should define and agree to special protocols for these types of gatherings.

REFERENCES

1. Brooks, Frederick P. "No Silver Bullet: Essence and Accidents of Software Engineering." In *Proceedings of the IFIP Tenth World Computing Conference*, IFIP, 1986, pp. 1069-1076.

2. Cockburn, Alistair. *Agile Software Development.* Addison-Wesley Professional, 2002.

3. Gottesdiener, Ellen. "Decide How to Decide." *Software Development,* Vol. 9, No. 1, January 2001, pp. 65-70 (http://ebgconsulting.com/articles.php#people).

4. Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley Professional, 2002.

5. Gottesdiener, Ellen. *The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements*. GOAL/QPC, 2005.

6. Jeffries, Ron. "Big Visible Charts." Xprogramming.com, 20 October 2004 (www.xprogramming.com/xpmag/BigVisibleCharts.htm).

7. Jones, Capers. *Patterns of Software Systems Failure and Success.* International Thomson Computer Press, 1995.

8. Jones, Capers. "Strategies for Managing Requirements Creep." *IEEE Computer*, Vol. 29, No. 6, June 1996, pp. 92-94.

9. Kaner, Sam. "Participatory Decision-Making: Tools for Reaching Closure." Tutorial Proceedings, International Association of Facilitators, Tulsa, Oklahoma, USA, January 1997.

10. Nutt, Paul C. "Leverage, Resistance and Success of Implementation Approaches." *Journal of Management Studies,* Vol. 35, No. 2, March 1998.

11. Reina, Dennis S., and Michelle L. Reina. *Trust and Betrayal in the Workplace: Building Effective Relationships in Your Organization*. 2nd ed. Berrett-Koehler, 2006.

12. Rost, Johann. "Political Reasons for Failed Software Projects." *IEEE Software*, Vol. 21, No. 6, November/December 2004, pp. 104, 102-103.

13. Schrage, Michael D. *Serious Play: How the World's Best Companies Simulate to Innovate*. Harvard Business School Press, 1999.

Ellen Gottesdiener, Principal Consultant of EBG Consulting, helps teams to collaboratively explore requirements, shape their development processes, and plan and review their work. Her book Requirements by Collaboration: Workshops for Defining Needs *describes how to use multiple models to elicit requirements in collaborative workshops. Ms. Gottesdiener helps agile teams to define their product and release roadmaps and elicit just enough requirements to achieve iteration and product goals. Her most recent book,* The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements, *is becoming the go-to industry guide for requirements good practices for business owners and analysts. She is a Certified Professional Facilitator (CPF) and a member of IEEE, IIBA, IAF, ACM, and DAMA. Ms. Gottesdiener can be reached at EBG Consulting, Inc., 1424 Ironwood Drive West, Carmel, IN 46033-8722, USA; Tel: +1 317 844 3747; Fax: +1 317 844 7374; E-mail: ellen@ebgconsulting.com; Web site: www.ebgconsulting.com.*