How To Communicate Effectively In IT Projects

By Krzysztof Rakowski

Published on June 27th, 2014 in Business, Communication, Workflow with 23 Comments

One of the most important factors in the success or failure of any IT project is communication. Communicating effectively can be quite difficult, especially when a project involves many people with different backgrounds, experience, skills, responsibilities and levels of authority. The problem compounds when the people involved belong to different organizations with different working guidelines.

Effective communication happens when a message is delivered whose content has the same meaning for the recipient as it does for the sender, thus inciting the desired action.

Why Communicate Effectively?

Consider a few scenarios. You will probably recall similar situations from your own experience.

- A team leader has to keep an eye on the status of a project. All tasks are stored in an issue-tracking system. Unfortunately, the tasks haven't been named very descriptively. For example, a bug in the contact form is described as "Something is wrong with the form," and a need for a database backup is described as "Please help! URGENT!" The team leader would have to open each ticket every time to recall what it's about. Of course, any sane person would change the descriptions immediately to "Contact form does not validate" and "Back up db0234@host1 database."
- A developer receives an email whose subject line reads "Just one question." He can't tell

at a glance that the email is about a bug in the search engine and should be forwarded to his colleague. He has to spend time opening the email and digesting its contents in order to decide to forward the message.

- A project manager organizes one or two hour-long meetings every week to discuss the progress of a project with the whole team. Each person speaks about their part for a few minutes and then sits bored for the rest of the meeting. From time to time, someone brings up a bigger issue that matters only to them and the project manager. In short, considering the hourly wages of the employees, a lot of money is being wasted on these counterproductive meetings.
- A developer is trying to concentrate on a complicated problem but is constantly distracted by phone calls or by colleagues who walk in to ask about non-urgent matters.

As we can see, **effective communication is critical**. Without it, many problems arise: lost time (which means lost money), bad code, inefficient development, delays and products that don't meet expectations. Ultimately, the reputation of the company and the client's trust are at risk.

The people involved may feel that their time is not being respected, which leads to frustration. (Image credit[1])

In this article, I will share some observations from an IT project I've been involved in for almost three years. As the development team leader, I work with about 30 people from different professions: developers, testers, administrators, designers, usability experts, project managers and people on the client's side. Working in such an environment, I've identified the main obstacles to effective communication.

I've also been involved in devising techniques to overcome those obstacles. Most of the problems and countermeasures we'll discuss apply to team environments, but freelancers might see value in them with their clients and partners.

Effective communication saves money, **time and effort**, and it happens when you do the following:

- make a message's subject easily identifiable (by "message," I mean not only email, but any form of communication);
- make a message's content quickly understood;
- be explicit in your message;
- manage effectively;
- involve only those resources (people, tools, etc.) that are needed to complete a task.

Proper communication leads to the following outcomes:

- the pace of a project is sustained;
- team leaders maintain control of the project's progress;
- people with different responsibilities and levels of involvement are better engaged in the project;
- people feel their time is respected and well used.

Effective communication in IT projects can be encapsulated by three words: explicitness, traceability and readability.

Email

Email is the primary means of internal and external communication at most companies. Surprisingly, many people still don't know how to use it properly.

The subject line is the first thing that a recipient notices. It should be brief and should explain the contents of the email. The recipient might want to refer to the correspondence in future, perhaps weeks or months later. Therefore, the subject line should clearly identify the project (including the customer, depending on the organization) and the subject matter.

Of course, not every subject fits neatly into a project — in such cases, take extra care to **make the subject line clear**. Consider that, while you might be working on only one project for a given customer and "ACME Corp: new images" sounds like a good subject line to you, your coworkers in the marketing department might be working on many projects for the same customer, all of which involve "new images."

Here are some examples of good subject lines:

- ACME Corp. | HR Portal | draft of functional documentation, v. 0.1
- ACME product page — questions after the meeting with marketing dept. on March 5th
- Please, send your report — deadline: March 10th

Nicknames for clients and projects and the separator symbol should be agreed on by everyone involved in the project, because they enable recipients to sort their inbox according to filtering rules (especially for managers, who get hundreds of emails an hour).

Here are some real-life examples of bad subject lines:

- ACME
- Question
- Request
- New images
- We're going for lunch at 1 pm

That last one is from a follow-up email that contained important documentation — true story!

A clear subject line quickly tells the recipient the contents of the message and whether they need to respond in any way. For this reason, avoid changing the topic of conversation during an email thread ("BTW, about that other thing, did you…" is a dead giveaway). Either change the subject line or send a separate email.

The "To" and "Cc" fields are useful ways to indicate who is the addressee of a message and who just needs to be informed without taking any action. By default, the person in the "To" field should read and probably respond, while the person in the "Cc" field would do enough to just read the message. Many **managers want to stay informed on matters** that they are not directly responsible for, and they'll configure their filtering rules accordingly, browsing those messages from time to time. Don't "Cc" someone if you expect a prompt response.

One last rule, perhaps a lifesaver, is to do everything in writing. People tend to forget about arrangements made by phone or in meetings. Perhaps a form of communication other than email would be appropriate in these situations. We'll discuss that next.

After a call or meeting, write everything down and send it to everyone involved. This way, you won't miss

anything. (Image credit[2])

Issue-Tracking

When you're managing — whether it's one large project or many small projects — compile all issues in one place to enable all team members to track their progress and know what needs to be done. Many teams make the mistake of assigning tasks and documenting important details in email. That might be easy to track when you have only one or two tasks to complete. In all other cases, it is the road to failure.

Issue-tracking systems — including Redmine[3], Mantis BT[4], Bugzilla[5], Jira[6] and many more[7] — enable clients, developers and managers to work together in well-structured process. Every issue — be it a bug report, feature request or question — becomes easy to track, with information about the person responsible, a full history and often even time-tracking and deadline reminders.

Every company employs a workflow that make sense for its business. Still, some rules apply to all situations, and here are ones that I've laid down after analyzing thousands of issues over the last few years:

- Title each issue as descriptively as possible. Remember that most people — whether developers, project managers or testers — deal with dozens of issues every day. They should get at least some sense of an issue's subject from its title. Thus, avoid titles like "Something's wrong" or "A typo." Rather, use self-explanatory titles, like "API throws NullPointerException when no attributes provided" or "Typo in the telephone number on contact page." Issues with such titles will be processed more quickly because they are easier to manage.
- Use attributes accordingly. Many trackers let you set special attributes on each issue, such as status, priority and category. Use them! The issues will be easier to sort, delegate and review.
- Most trackers allow you to set relationships between issues. For example, you could

mark an issue as being blocked by another issue or being a duplicate or just being similar. Relationships make it easier to assign tasks and find solutions.

- Write about only one thing per issue. For example, if you find many bugs in an application, treat every bug as a separate issue. This way, tasks can be assigned to different people, who can work on them simultaneously. Different issues demand different people: designers, front-end coders, programmers, etc. When everything is in one task, managing and completing it takes much longer.
- Provide as much information as possible. Link to the buggy web page; write all of the steps needed to replicate an issue; attach screenshots. I worked with someone who attached a Word document with a few words and screenshots, titling the issue "Everything is in the attachment" — a big no-no.
- Write down in the comments section everything that happens with an issue. If someone explains something to you by phone or if a task is changed, document it. Don't write, "We spoke about that issue on the phone, so now you know what the problem is." Imagine that another developer will be assigned to this task in future and that anything not written down will be lost. Keep your project's bus factor[8] as high as possible.

To sum up, keep everything in the issue tracker. Give each task its own issue. And describe the issue as best you can.

Meetings

Some people consider meetings to be a corporate nightmare. Many hours are wasted every week on meetings that contain no content and that don't end with any useful decisions. "Status meetings" are the most popular kind of this: A dozen or so people gather to talk about different, often unrelated, projects. In fact, most of them are bored and not paying attention, waking up only when they need to talk for a few minutes about their situation.

A better way to stay in touch with team members is to speak with them frequently in person. When a team is working on a project, organize short morning meetings (5 to 15 minutes), like the "stand-up meetings" popular in agile development. This way, every member of the team will always know the status of the project. One member (usually the leader) could report to the boss, freeing up everyone else. Another good way to collect information about the status of a project is daily one-on-one communication. This will take up some time of the team leader, but it yields great results and saves time overall.

If you really do need a meeting with certain people, follow these practices:

- Prepare an agenda. This way, invitees can prepare themselves for the meeting and — if necessary — submit their opinion, send someone else to fill in or just avoid the meeting if their attendance is unnecessary.
- Stick to the agenda. Moderate the discussion and discourage talk of things that could be handled by a smaller group.

- Don’t make decisions “together” — that won’t work. Instead, the small team that is responsible for a task should propose a solution, which should be discussed briefly and then decided on by the person in charge.

One last remark about meetings. Many people in IT do work that is creative in nature. Programmers, for instance, need time to build their concentration, and breaking this process is very destructive. Not all questions need to be answered immediately in person or by phone; some could be sent by email or instant messaging, thus protecting the creative process.

Other Guidelines

Let’s move on to guidelines that are useful not just in IT projects, but in life in general.

Having a **common vocabulary with others** is essential to communication, in both professional and personal life. This means using the same names for activities, projects, customers and so on. Misunderstandings arise when two people are talking about the same thing but don’t understand each other because they are using different terminology. Formally establishing a common vocabulary using a tool such as a wiki or whiteboard would be useful.

Managing the knowledge that is generated in a project is important, too. D**ocumentation often gets outdated** because no one cares to update it when the project’s requirements change. Important decisions and guidelines are lost in email and other communication. This often leads to misunderstanding or, worse, conflict with the client. Therefore, confirm every new version of documentation with everyone involved. When updating the issue-tracking system, remember to record details of phone conversations and email, noting who took what action or made which decision or provided what information. In short, log of every action and piece of information regarding an issue.

When working on a complicated project, I often find it useful to post important tasks on colorful sticky notes on a whiteboard (in addition to the issue-tracking system). Some are marked “waiting,” others “in progress,” yet others “done.” The color of the magnet affixed to each note indicates which team member is assigned to the task. **People tend to ignore email notifications** from issue-tracking systems; marking a task as complete on a whiteboard requires someone to stand up and reposition the note. Everybody can see the status of tasks and the progress of the project at a glance, which is the big advantage of this method. I also tend to write down other essential information, such as the time when the production environment was last updated or the number of bugs left to be resolved, broken down by team member.

Sticky notes on a whiteboard; such dashboards are a great place for daily stand-up meetings, too. (Image credit[9])

If you develop software, then you probably use tools to manage your source code, such as Git, Mercurial or SVN. To facilitate your team's work, implement some sort of workflow such as a branching model. Remember also to describe your commits informatively.

How To Introduce These Strategies

Forcing everyone to work according to specified rules is hard. Some people are reluctant because they feel that crafting their message more carefully would take a lot of time. Others don't see the benefit or are just unable to formulate concise thoughts.

I suggest two ways of working with such people. First, show them that the amount of work necessary to better communicate is much smaller than the time wasted otherwise. For example, if a ticket in the issue-tracking system has a meaningless subject line and its contents are hard to process, then the time wasted will accumulate with every person who needs to work on it. A simple countermeasure is to **invest a few minutes to prepare the issue better**, so that time is not wasted afterwards. Other ways to enforce the rules are by implementing validation mechanisms (for example, in the issue-tracker) or by rejecting communication that doesn't conform to the standard. It might sound brutal, but it's effective.

Conclusion

The rules of communication can be summarized in a few points:

- Communicate clearly. Load the most important information at the beginning of your message (in the subject line, title and first sentence). Learn how to encapsulate a message in one sentence.
- Involve only the people who are necessary to move something forward. Don't waste anyone else's time.
- Keep everything in writing and easy to access.
- Read everything you write to make sure it's easy to understand.
- Break down big problems into small tasks that are easy to digest.

I hope you find these insights useful to your work. If you have any thoughts, please share them in the comments section below, or catch me on Twitter[10].

(al, il)

Front page image credits: Jon Ashcroft[11]

FOOTNOTES

1 https://www.flickr.com/photos/34653106@N00/66393869/

2 http://www.flickr.com/photos/littlemad/8002274002/in/pool-smashingconf

3 http://www.redmine.org

4 http://www.mantisbt.org/

5 http://www.bugzilla.org/

6 https://www.atlassian.com/software/jira

7 http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

8 https://en.wikipedia.org/wiki/Bus_factor

9 https://www.flickr.com/photos/23392528@N02/13009862125/

10 http://twitter.com/krzrak

11 http://www.flickr.com/photos/theilluminated/5138288695/in/photostream/

Krzysztof Rakowski

Krzysztof Rakowski (@krzrak) has over twelve years of experience in designing and developing web applications in different technologies. Currently, he leads a team of Python wizards in the one of the top Polish interactive agencies - K2 Internet.