The Art of Visualization

Selena Delesie

Delesie Solutions

Consulting Software Tester and Agile Coach

selena@delesiesolutions.com

ABSTRACT

Good testers use their experiences and understanding of a product to apply oracles and heuristics that guide testing in uncovering problems and to communicate with a variety of stakeholders. Great testers go beyond. They are skilled in employing different techniques to create visual representations of software requirements, solutions, and problems. They recognize that the simplicity of a visual aid can disseminate information more accurately than the spoken and written equivalent. Great testers are talented in using such visual representations when collaborating and communicating with a variety of stakeholders so participants arrive at a shared understanding.

1. INTRODUCTION

> *"Genius… is the capacity to see ten things where the ordinary man sees one, and where the man of talent sees two or three, plus the ability to register that multiple perception in the material of his art."* Ezra Pound

Soon after human beings discovered that internal 'mental pictures' could be externalised, the first marks evolved into pictures in cave paintings. As civilizations developed, pictures developed into symbols, then alphabets and scripts (such as Chinese characters and hieroglyphics), and then letters themselves. Letters and the written word have since become the defacto standard for communicating information.

Today we are in an 'information explosion', based on an assumption that the spoken and written word are the only correct vehicle for learning, analyzing and disseminating information. We limit our brains in expression and understanding because we rely predominantly on words, sentences, numbers, and logic as the building blocks in society.[1]

[1] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 37-38.

2. COMMUNICATION

Communication is a two-way street, and involves the exchange and flow of information and ideas from one person to another and back again. It includes content, spoken and written mediums used, and context (the way a message is delivered, e.g. body language, emotions). It is effective when the receiver understands the exact information the sender had intended to transmit. Feedback is used as a mechanism to help confirm this, so to understand what the other party was trying to communicate.

2.1 VERBAL COMMUNICATION

Suppose you are in a requirements definition meeting in which the product manager is communicating the needs of a new software project, and a number of different stakeholders are present. When a requirement is identified, do you ask clarifying questions to obtain all the information you need to fully understand it?

When you do not ask clarifying questions to understand the requirements being communicated, you are choosing not to provide feedback, and thus, not engage in the communication. Even if you did provide feedback for every comment or question you had, you would find it is nearly impossible to understand every aspect of a requirement exactly as the speaker intended through verbal communication alone.

2.2 WRITTEN COMMUNICATION

Written documentation is often created to fill the gap people perceive in verbal communication; often to the extreme. Written documentation also has a lot of disadvantages, yet is the predominant medium in business communication. It is surprising when you consider that the written word:

- ***Obscures key words.*** Key words are spread throughout written material, which makes it difficult to make associations between key concepts.
- ***Wastes our time.*** We spend time reading unnecessary words, re-reading unnecessary words, and searching for key words.

- ***Is difficult to remember.*** Text tends to be a single colour (which is boring), has endless similar-looking lists (which puts the brain into a semi-hypnotic trance), and contains more information than we need to understand key concepts (which makes it hard to remember content).
- ***Does not stimulate brain creativity.*** Linear format of content prevents brain from making associations which counteracts creativity, understanding, and memory.[2]

The result is that we spend a lot of time not being able to concentrate, becoming bored and frustrated, writing more content to try to understand the content we are reading, and misunderstanding the intent of the writer. In business terms, this means higher costs and longer timeframes to develop software, poor software quality, unsatisfied customers, and less revenue.

So why do we spend so much time putting requirements, software designs, and decisions into lengthy, uninspiring, misunderstood documents?

2.3 COMMUNICATION SURVEY

The Ambysoft 2008 Agile Principles and Practices survey explored the effectiveness of communication strategies amongst agile team members and stakeholders. Nearly 340 individuals responded to the questionnaire, providing feedback on how effective different methods of communication were in their experience. Results are summarized in Figure 1. Notice that detailed documentation was perceived to be very ineffective both within the team, and when communicating with stakeholders. In both cases, face to face communication, and face to face communication while using a whiteboard, was perceived to be the most effective, followed by the use of overview diagrams.

While this survey was answered predominantly by programmers on agile teams, I consider the results to be applicable for testers as well, given my own experiences in using these different communication strategies.

As testers we must consider which communication strategies will be most effective in relaying different information, such as software problems, test strategies, risks, and test results. For example:

- Sending emails to programmers to describe software problems will be much less effective than talking about the problems with them in person, or creating diagrams to enhance the verbal description.
- Relying on detailed requirements documents to understand software intent and uses is less effective than discussing requirements with business stakeholders and programmers, and modeling requirements in different ways.

Figure 1. Effectiveness of communication strategies on agile development teams.[3]

Answers rated on a range of -5 (very ineffective) to +5 (very effective).

Communication Strategy	Within Team	With Stakeholders
Face to face (F2F)	4.25	4.06
F2F at Whiteboard	4.24	3.46
Overview diagrams	2.54	1.89
Online chat	2.10	0.15
Overview documentation	1.84	1.86
Teleconference calls	1.42	1.51
Videoconferencing	1.34	1.62
Email	1.08	1.32
Detailed Documentation	-0.34	0.16

2.4 COMMUNICATION MEDIUM

In an attempt to improve communication it is important to select an appropriate communication medium for a particular situation. There are a myriad available:

- Wiki's and word processing tools can be used for collaborative writing ventures.
- There are similar tools available for collaborative modeling.
- A variety of discussion tools such as email, instant messaging, chat rooms, and mailing lists are helpful for communicating text messages and attachments.
- Documented artifacts can be managed in version control tools.
- Video can be used for enable conversations and meetings between people in different locations.

[2] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 49-50.

[3] Scott W. Ambler, "Communication on Agile Software Projects", Agile Modeling, 2001-2009, 1 June 2010, <http://www.agilemodeling.com/essays/communication.htm#Figure1>.

- Models that are simple to create enable shared understanding and learning.[4]

3. MODELING

> *"Creative thinking may mean simply the realisation that there is no particular virtue in doing things the way they have always been done."* Rufolf Flesch

Tony Buzan did an experiment some years ago across a variety of schools, professions, nationalities, and languages, which evaluated different skills people use when writing notes. He discovered that 95% of participants employed only three elements in their notes:[5]

- *Linear patterning:* Writing in straight lines, full grammar structure, chronological sequence, or historical sequence.
- *Symbols:* Including letters, words, numbers, and non-alpha-numeric symbols.
- *Analysis:* Content quality found to be adversely affected by linear nature of content presentation.

There are many other skills that can be used to activate the most highly evolved area of the human brain, the cerebral cortex. Elements that are not often used in typical communications, particularly in the business world, are: *visual rhythm, patterns, colours, images, visualization, dimension, spatial awareness, completeness, and association*.[6]

As we make use of a larger range of these skills, the brain will in turn improve comprehension, thinking capacity, memory, learning, creativity, and confidence. In business terms, this means lower costs and shorter timeframes to develop software, higher software quality, satisfied customers, and increased revenue.

3.1 WHY USE VISUAL MODELS?

Modeling enhances verbal and written communication, and is a useful technique to help people arrive at a shared understanding. Other reasons to consider using models include:.[7]

- ***A lot of information*** beyond written and verbal communication is needed even if the requirements or software design are fully known. Models can efficiently represent these in different ways, from different perspectives, providing richer detail and fuller comprehension.
- ***Customers*** typically do not know exactly what they want, and are unable to communicate all they know they want. Models can be used to confirm we understood what the customer identified, and to show the customer basic representations of how the software will work to gain their approval early on.
- ***People make errors***. Models are useful for brainstorming root causes for problems, and making decisions for solutions and designs.
- We typically ***build from existing software***. Models are helpful for representing existing software systems in simple, easy to understand terms, which accelerates work for new features.
- ***Projects are subject to change***, which influences earlier decisions. Simple modeling techniques that represent decisions, systems, and user requests are easier to understand and modify when changes occur than are lengthy documentation.

3.2 MODELS ENCOURAGE COLLABORATION

Collaborating with other people in the creation of models is incredibly effective for a number of reasons. In group brainstorming situations that use a variety of elements, and are supportive and inclusive, creative thought takes hold to clear assumptions, explore new ideas, capture and develop insight, and build off ideas through association with other ideas. This leads to more comprehensive analysis and problem solving, improved decision-making, and more creative software solutions.[8]

3.3 MODELS IMPROVE COMMUNICATION

In using models to enhance communication, I have found simple diagrams are effective for most people. Many stakeholders, or even programmers and testers, do not understand complex modeling techniques such as UML. This means that software-based modeling tools are likely to hinder communication and collaboration with stakeholders, not foster it. Simple modeling techniques that are easy to learn and understand are inclusive to more stakeholders. Employing simple tools like pen, paper, whiteboards, and index cards may seem almost juvenile, but are instead

---

[4] Scott W. Ambler, "Communication on Agile Software Projects", Agile Modeling, 2001-2009, 1 June 2010, <http://www.agilemodeling.com/essays/communication.htm>.

[5] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 45.

[6] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 46.

[7] Hans Van Vilet, Software Engineering Principles and Practice (West Sussex: John Wiley & Sons Ltd., 2008) 354.

[8] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 173.

welcoming. They increase communication and collaboration, reduce complexity, and reduce time invested on a particular topic.

In my experience, simple flowcharts, informal diagrams, and user stories to be effective in communications with senior management, while artifacts like data flow diagrams, flowcharts, and use cases to be more effective with technical colleagues. In most situations, discussing these artifacts while they are being created, or in a draft form, has proven more effective than sending a lengthy document with the artifacts contained within them.

4. REAL-LIFE APPLICATION OF MODELS

I worked on a voice recognition project a few years ago, which had a feature called "Call by Name". It allowed a user to say a name that was in their smart phone's address book out loud and have the phone call the appropriate number.

The requirements were specified similarly to:

> "Feature: Call By Name. Users will be able to place phone calls by speaking "Call By Name" and then speaking the name of the person they wish to call that is in their smart phone address book. The call will then be placed to that person's phone number. When several options are found, they will be provided to the user to confirm which person to call."

The feature seemed straightforward to senior management and product management, so no other information was provided. The programmers knew there were details missing, but took creative liberty in implementing it as they believed was appropriate. The feature was implemented before testers were involved in the project.

Note that voice recognition technology is complex and has mediocre accuracy rates. A lot of logic is subsequently built into software to improve voice recognition software user experiences. This was not apparent to management, customer support, or beta customers though, and these stakeholders did not understand why something that seemed so simple could have so many problems.

The testers had a lot to learn when they were finally added to the project. To quickly understand how the product worked, and was intended to work, testers used different modeling techniques to express their understanding, capture information from other stakeholders, and share different test artifacts.

4.1 USER STORIES

User stories are typically used in Agile Development methodologies. They are software requirements written in the everyday language of the user, are typically limited to the size of a 3x5 inch index card, and are written by the customer or business-facing person representing the customer. While they are most effective when used with Acceptance Tests (which identify more specific details of the requirements), they are still useful for understanding why functionality is needed, and what it needs to do.

The use of User Stories could have improved the definition of the Call By Name feature requirements, by highlighting smaller pieces of required functionality and how they would be used by the user. *Agile Development, and subsequently User Stories, were not familiar to the organization at that time.* As a tester, understanding how to write and interpret User Stories improves communication and collaboration with customer-facing stakeholders because you are communicating in terms of the user needs, not the technical needs. Some examples of what User Stories might look like for the Call By Name feature are shown in Figure 2.

Figure 2. User Stories: Call By Name Feature

4.2 FLOWCHARTS

A flowchart is a diagram that represents an algorithm or process, which can give a step-by-step solution to a given problem. Data is represented in boxes, and arrows connect them representing flow / direction of flow of data. I have often seen flowcharts used by software testers to learn, understand, discover problems in, and use as guidelines in testing, software. They are also useful in analyzing, designing, documenting or managing a process or program in various fields.[9]

Upon joining the project, one tester decided to model the Call By Name feature to understand how the internal software logic worked from start to end of the process. They were able to raise concerns about missing requirements, incomplete design logic, and poor user experiences under particular conditions. As a result, stakeholders collaborated to correct the software inadequacies, and more efficiently than when no such model existed. The Flowchart is shown in Figure 3.

Figure 3. Flowchart: Call By Name Feature

[9] "Flowchart", Wikipedia.org, June 2010, 22 June 2010, <http://en.wikipedia.org/wiki/Flowchart#cite_note-SSEV-0>.

4.3 SEQUENCE DIAGRAMS

Sequence Diagrams are a type of UML (Unified Modeling Language) model. They are a type of interaction diagram used to model the time ordering in which sequences of messages are exchanged between two or more objects, and may contain conditions that must hold for particular messages to be sent.[10]

After a colleague had created a flowchart to understand how the Call By Name feature worked, a team member still had some questions about the timing of events in the software logic. Being somewhat familiar with Sequence Diagrams, this tester spoke with the programmer who developed the feature to get the information he felt he was missing. In collaboration with the programmer, a Sequence Diagram was created which highlighted some timing concerns that warranted further review and subsequent software improvements. The Sequence Diagram is shown in Figure 4.

Figure 4. Sequence Diagram: Call By Name Feature

4.4 MIND MAPS

Mind maps are external expressions of associative thought processes that proceed from or connect to a central idea, such that the main themes of a particular idea radiate from the central image as branches, and topics of lesser importance branch out in succession from there.[11]

One tester decided to use Mind Maps to quickly draft a test strategy for testing the Call By Name feature; the nodes for which became the exploratory test charters during test sessions. The Mind Map is shown in Figure 5.

Figure 5. Mind Map: Call By Name Feature

5. CONCLUSION

Whether you are interacting with someone about requirements, software design, or a problem to be solved, you need to understand your audience so they are properly engaged. Effective communication involves information being shared like a two-way street: both parties give a message, listen, and provide feedback. Selecting the right mode of communication for the specific people involved, the specific topic being discussed, and the unique context of that topic is an important success factor.

Understanding the position, responsibility, and knowledge-base of the other person(s) will help determine the type and depth of details to include. Providing too much or too little detail may cause the other person(s) to become disengaged. In my experience, the higher the person's position, the less details you will include.

Determining the simplest and most dynamic means of communication that will work for your situation will increase creativity, collaboration, and engagement of involved parties. If a conversation and a diagram capturing what was discussed is the simplest thing that works, why would you create a lengthy document instead? In another situation, perhaps a lengthy document is required to satisfy legal requirements for a project. Be ready to pick the simplest approach that is most likely to succeed in each situation, and be prepared to change your approach to meet the needs of each new situation.

After looking at how both verbal and written communication are often inadequate for fully communicating and understanding information, we can see how using visual models can enhance and bring people to a shared understanding and agreement.

Unfortunately, many people believe that lengthy official documents with approval processes, formal scheduled meetings, and formal presentations are indicators of professionalism. These are often indicators of a lack of trust in delivering, and a need for someone to have more control.

A mind shift may be necessary at all levels of the organization to agree that less documentation but more modeling, and less complex tools but more simple inclusive tools for collaboration, are more appropriate for successful software delivery.

Start communicating the benefits of improved agility and software understanding (less rework, faster time to delivery, less cost, fewer defects, improved quality), and the downsides of continuing to use formal, heavy-documentation approaches (more rework, slower time to delivery, more defects, more cost, less quality)[12].

Otherwise, simplify and enhance your own communications, and start modeling.

---

[10] Hans Van Vilet, Software Engineering Principles and Practice (West Sussex: John Wiley & Sons Ltd., 2008) 283.

[11] Tony Buzan, The Mind Map Book (London: BBC Books, 1993) 59.

[12] Scott W. Ambler, "Overcoming Requirements Modeling Challenges", Agile Modeling, 2001-2009, 1 June 2010, <http://www.agilemodeling.com/essays/requirementsChallenges.htm>.

APPENDIX A: TYPES OF MODELS

There are many types of modeling techniques at your disposal to enhance communication and understanding of software products. While I recommend selecting the simplest tool possible to represent an aspect of software or a problem, it is helpful as a tester to be aware of the different ones available. You never know when you may encounter a programmer who uses several advanced types of UML models to represent different aspects of the software he is developing. In this case, knowing how to read, interpret, and even create these models will improve your ability to collaborate, and improve your reputation.

Some Recommended Resources:

Ambler, Scott W. Agile Modeling. Wiley, 2002.

Ambler, Scott W. The Elements of UML 2.0 Style. Cambridge University Press, 2005.

Ambler, Scott W. The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press, 2004.

Buzan, Tony. The Mind Map Book. BBC Books, 1993.

Cockburn, Alistair. Writing Effective Use Cases. Addison-Wesley Professional, 2000.

Cohn, Mike. User Stories Applied: For Agile Software Development. Addison-Wesley Professional, 2004.

Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2003.

Van Vilet, Hans. Software Engineering Principles and Practice. John Wiley & Sons Ltd., 2008

BIBLIOGRAPHY

Ambler, Scott W. Agile Modeling. 2001-2009. <http://www.agilemodeling.com>.

Buzan, Tony. The Mind Map Book. London: BBC Books, 1993.

Northey, Margot and Joan McKibbin. Impact: A Guide to Business Communication: Sixth Edition. Toronto: Pearson Education Canada, Inc., 2005.

Van Vilet, Hans. Software Engineering Principles and Practice. West Sussex: John Wiley & Sons Ltd., 2008.

“Flowchart.” Wikipedia.org. June 2010. 22 June 2010. <http://en.wikipedia.org/wiki/Flowchart#cite_note-SSEV-0>.