USING YOUR VISUAL SUPER POWERS

Written by Torbjörn Ryber in September 2013

With the intent of inspiring YOU to use visual techniques when solving problems.

CONTENT

Introduction

I want it all, and I want it now

-Queen: I want it all

The Problem

This line of text from Queen nicely sums up how people want their problems solved nowadays. For us it is fairly uninteresting with excuses of various kinds. We have a problem we want to be solved. In Internet time, we want that to happen right now.

We build IT support and develop work processes in order to solve a problem for someone. If we do not solve the problem, the system does not work! This is true regardless of what happens to be in the specification or what is said at meetings. It’s impossible to know everything from the beginning and it's usually not even possible to know everything when the project is over and the system is delivered in a first version. There might not even be a best way to solve a problem but several equally good!

Scott Barber talks about that focus of today's performance testing 2.0 is the user's perception of performance. If the user feels that it is slow, there is a problem regardless of whether the response time is one or ten seconds. A ten-second response time may not be experienced as long if the user while waiting receives valuable information to read. The conclusion is the same for all tests, if the user perceives something as a problem - then we'll have to do our best to solve it.

A professional problem solver ensures that the real problem is resolved and does what it takes to go all the way to the finish.

Jerry Weinberg’s definition that *Quality is value to someone* still holds true in my point of view and should be the guideline for all the work we do.

The Solution

A modern IT project is very much about communication between the participants and here visualization is a fantastic tool. The pictures showing up on the wall showing workload, performance and urgent problems is a good example of this, whether we call it Kanban or Scrum. Fifteen minutes spent every morning in front of the board reporting progress and jointly working out how to solve problems makes collaboration easier.

The basic methods of Lean contain lots of graphic elements - value stream mapping is done visually and in collaborative problem solving we use the A3 method, also called the Lean Canvas. We solve many problems common in front of a whiteboard by visually describing problems and solutions, which then form the basis for our work. As a problem solver you have to constantly look for information and to structure and visualize it.

Words are fantastic and part of the fun of reading books is that we create images in our heads from what we read. The problem if we are to agree on something with others based only on text, our own images will differ from the images others have in their heads. In the dialogue with customers as well as within a development project, it is therefore very powerful to combine words with pictures and show everyone how we have interpreted the words. When creating visual models we have to explicitly describe what we think and have this critically reviewed by others.

This Paper

My goal is to show how visual techniques can be used in IT-projects to facilitate collaboration and enhance problem solving. I do not have any aim to give complete solutions or being academically correct but rather to inspire you. The paper has testing focus but most of the methods used are not specific for testing.

To make this an enjoyable read I use material from real life. At least most of it is true. Let´s just say it´s a stew of some real stories. Any characters used have traits from several people I worked with over the years. All of the actual problem solving comes from real examples. Much of the story is built around a single project that did exist. It is definitely more of an experience report than an academic study even though I make connections to great sources of information that you can dive into in order to learn more. If you find something you recognize in the story we may have worked together!

Visual Problem Solving in Four Steps

If you can't explain it simply, you don't understand it well enough.

- Albert Einstein

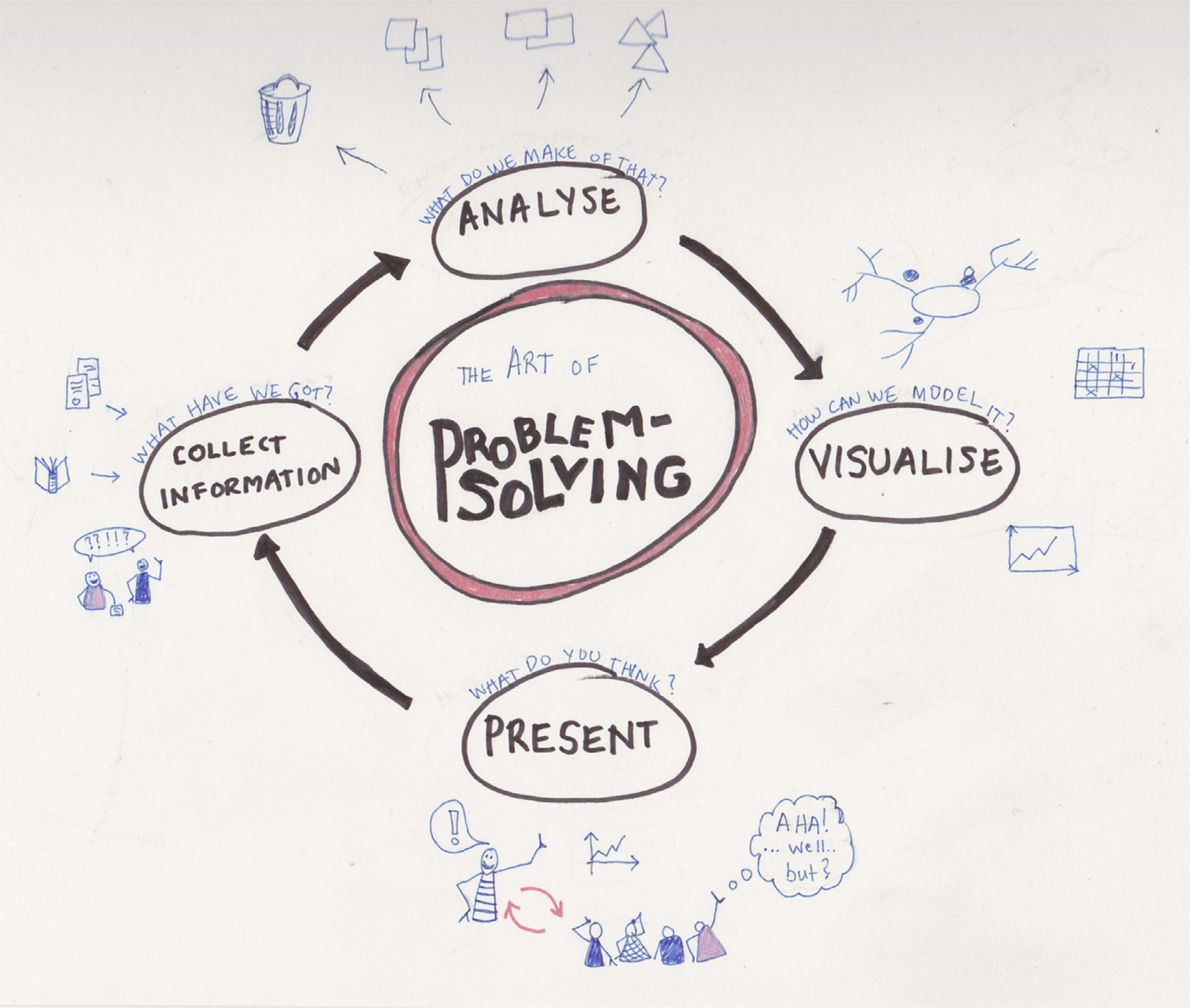
If you cannot model the problem, you have not really understood it

- Torbjörn Ryber

All models are by definition simplifications of reality. We create them in order to understand something better but must always keep in mind that they are biased and incomplete representations of reality. That said, here's my model of visual problem solving:

1. Collect all the information you can find
   a. Read documents, there usually are some, sometimes many – although content quality will vary
   b. Talk to the right people – you will know who they are when you find them
   c. If there is tool or piece of software – explore it!
2. Analyse what you found
   a. Try to find a place where you can spread it all out to get an overview
   b. Sort, group and throw away
3. Visualise
   a. What kind of model or drawing may fit here?
   b. It may be a good idea to use several different models to understand better
4. Present
   a. Make it big – stick it on the wall, use a projector or a white board
   b. Explain how you have interpreted the information and ask for input
   c. This means that you start collecting more information so you are back on step one again…

Continue going in cycles until you all agree on at least the important things



Picture: the art of problem solving

Here are some excellent books on visual problem solving that have inspired me.

1. Dan Roam has written two must read's called *The Back of the Napkin* and *Blah-Blah-Blah.* Besides having cool titles they are excellent sources for information on tools for visual problem solvers. Get to know the blah-blah meter to identify the message.
2. Becky Agerbeck is a great GF, graphic facilitator that is. Her book is called *The Graphic Facilitators Guide* and focus on the role of visually describing what is discussed in a meeting or presentation while keeping out of the discussion. Nice tips on creating your visual library.
3. Mike Rohde's *The Sketchnote Handbook* describes the power of combining words and pictures in your note taking. Try analyzing the next presentation you attend – is the message clear enough to sketchnote?
4. Then read David Sibbet's *Visual Meetings* and you have a pretty awesome start on getting information to becoming a great visual problem solver. Then, according to Malcolm Gladwell's book Outliers, You just need to practice 10 000 hours to become the best!

But I Can't Draw!

Sure you can! It is not about creating art, it is about content described visually in a clear and consistent manner. With a few simple components in your visual library you can create some great stuff.

Guidelines

- Be consistent in your pictures, lines, color, size and text format
- Make it easy to read. White space, write legibly, avoid spaghetti connectors. It will get messy – redraw until it looks neat
- Learn a few basic shapes and icons and practice drawing them quickly and consistently
- Content is really king, not the art. Try to get the essentials out and don't try to put everything in the drawing

Text

Practice writing using different styles and size. Make sure what you write is legible and that you use different formats consistently. Lower case is usually faster to write and easier to read than upper case.

Practice writing the alphabet in upper and lower case. Write quickly to spot your weaknesses and slowly to practice specific letters. What type of your own writing is reasonable fast and easy to read for others? Even at a distance?

Exercise

UPPER CASE

……

……

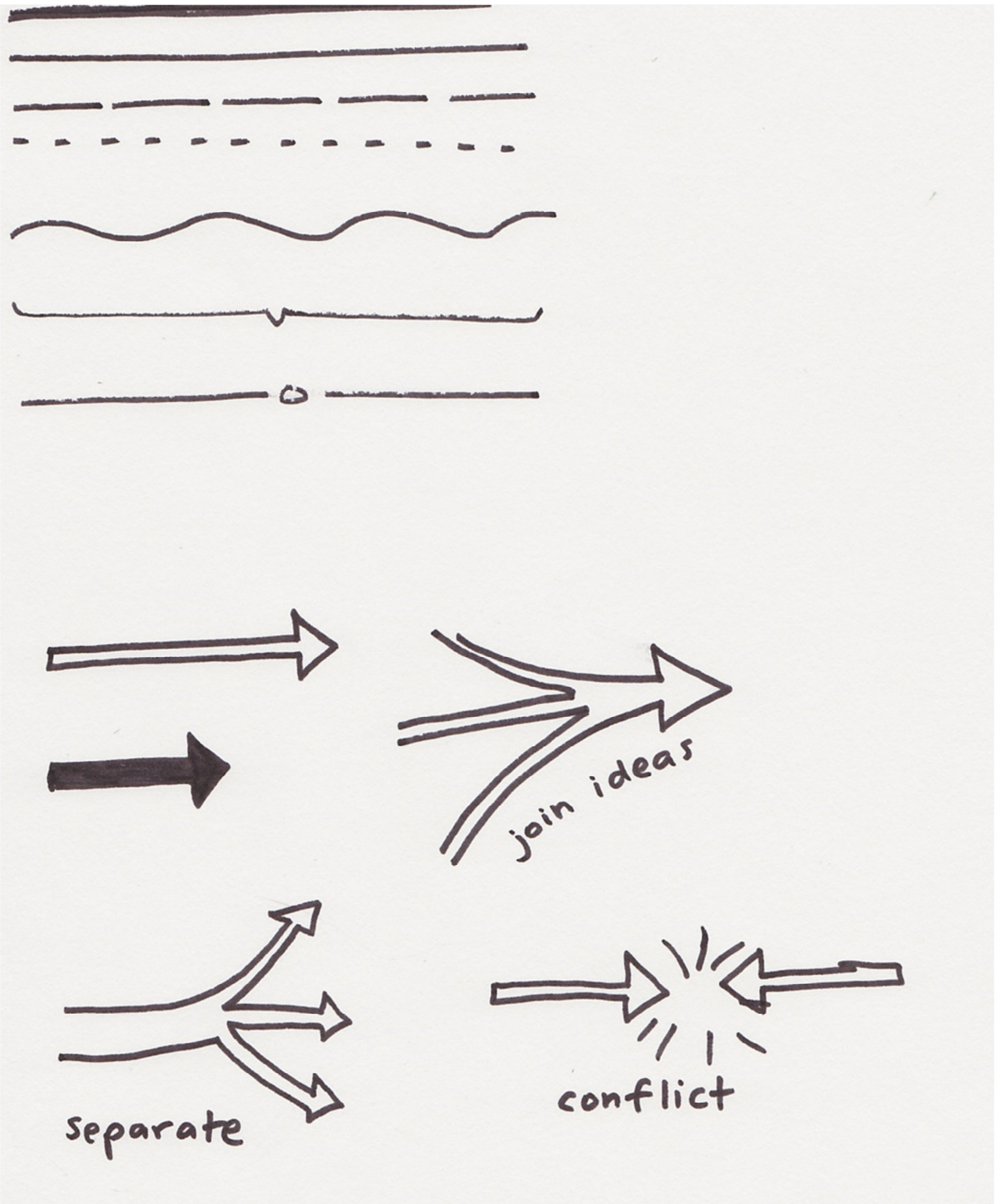
lower case

……

………

Separators and Connectors

Use thick or thin lines to group or divide you drawings. The thicker and more colorful – the more distinct is the division or connection. Thin, dotted or undulating line is more vague.



Exercise

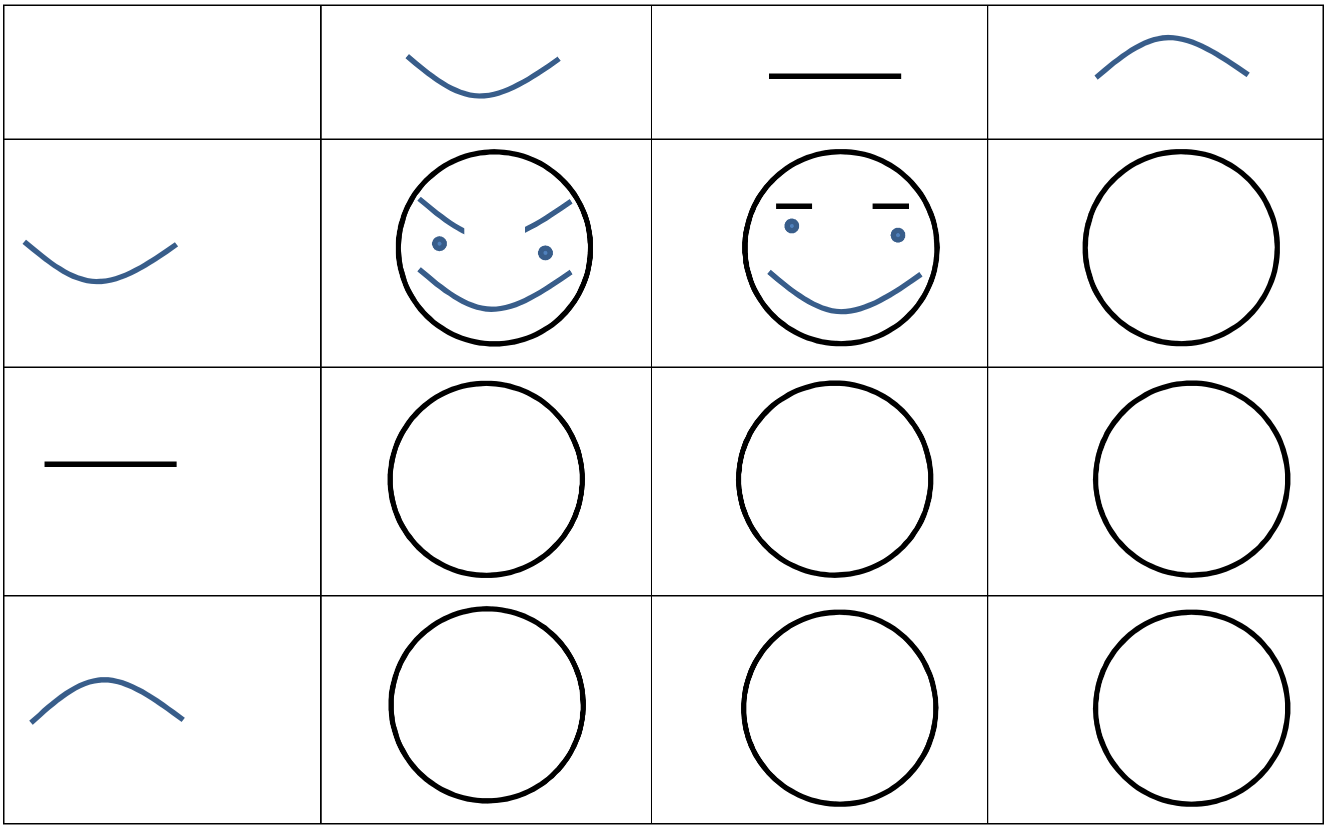
Practise drawing some variations of arrows and separators. They are abundant in problem and solution pictures.

People

I consider drawing people very important since users always are a central part of the problem. Start using smileys or stickmen and if you feel comfortable draw star-people or box-people to make the visuals more alive and fun.

Exercise

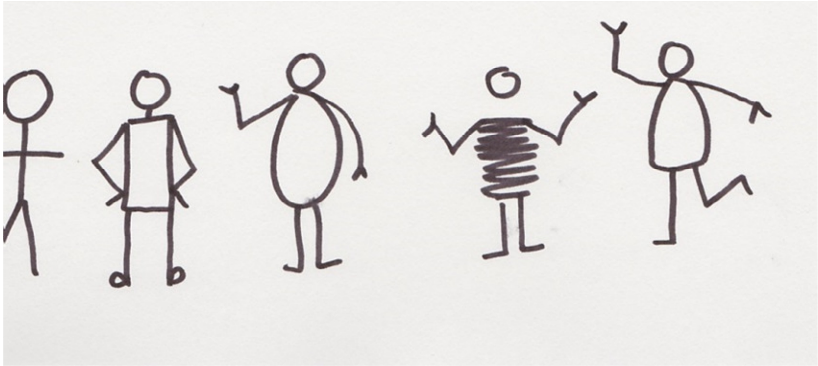
Faces: continue drawing the faces below. What moods are they in? The exercise comes from Rohde: the Sketchnote Handbook.



Decoration: add noses, beards, hair, hats, glasses or whatever you feel like to make the faces more alive.

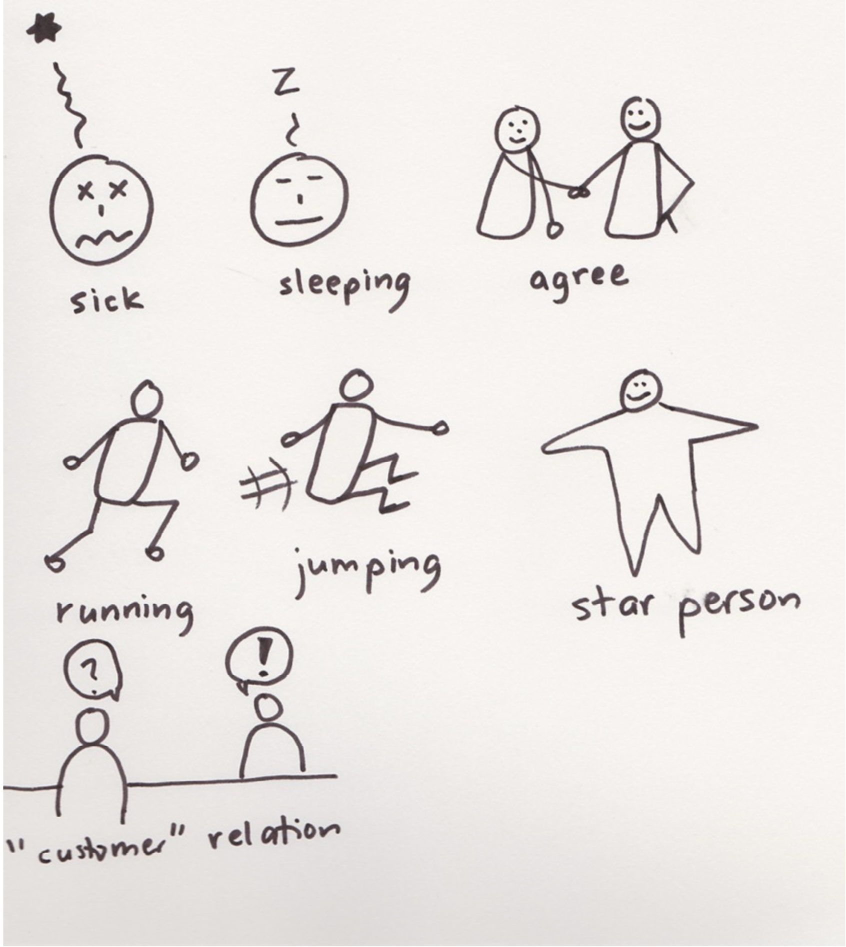
People: Try drawing the different simple people below. My tip is to select a type you like and practice a lot during the boring non-visual meetings you are forced to attend. When you feel ready – step up and draw them on the white board.

Stick **Box** **Bean** **Squiggle** **Tobbe's own**



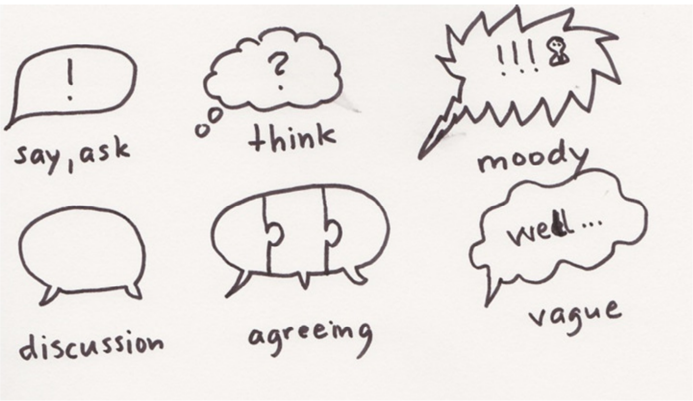
Variations

Combine people and add attributes to create meaning. Don´t sweat too much over details, you can always add text to clarify.



Bubbles

Combine thinking and talking bubbles to show communication and moods very clearly. Get inspired by reading comic strips.



Exercise

Agneta and AnniFrid decided they would go dancing coming Saturday. Agneta loved the idea but AnniFrid was more reluctant. Visualize their discussion.

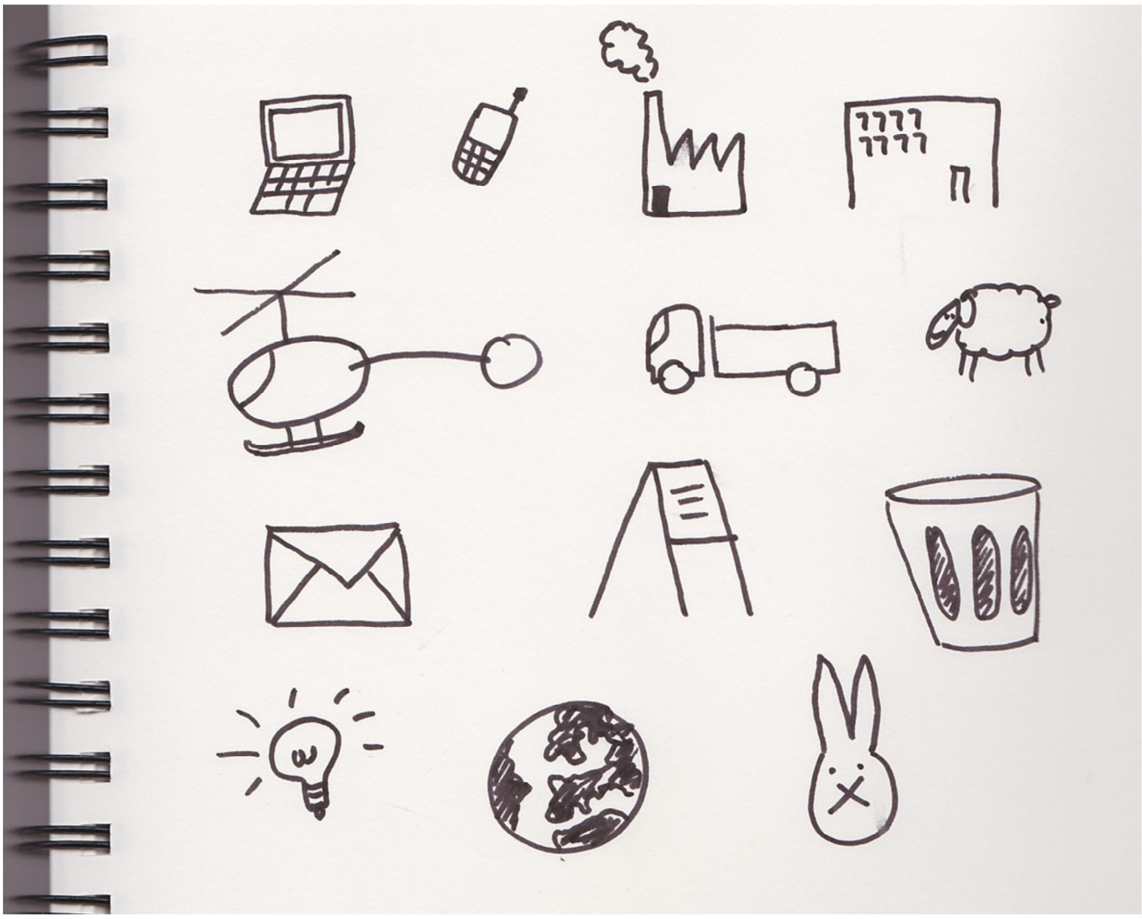
Icons

Some icons are more valuable than others. For me in IT common icons are tool, system, idea, document and e-mail. What are yours?

Draw icons for some of the words below. Show the person next to you. How easy are the icons to understand? Information, question, answer, tool, idea, global, time, database, document, key, start, stop or any word of your choice.

Things

Most representations of things can be drawn by adding simple elements together. Start with a line, a dot, a circle, a triangle and a square. Practice writing the things that are most common in your line of business. Practice a lot and you will make an impression! This is a page from my notebook where I practice drawing simple objects. I sometimes invent my own but often use my reference literature as a start.



Exercise

Draw nine things connected to your work. If you have to write a description you probably have not found the essential part that makes a thing easy to recognize. Use the five basic elements and add some detail where it makes sense.

Draw nine things in your home

Sources

Mike Rohde’s and Brandy Agerbeck’s books contain lots of ideas where to start. Another great source are the *bikablo dictionaries* produced by Neuland. They also show you how to make pictures look even better using colors and shading.

The NEW Process AND Where Testing Fits

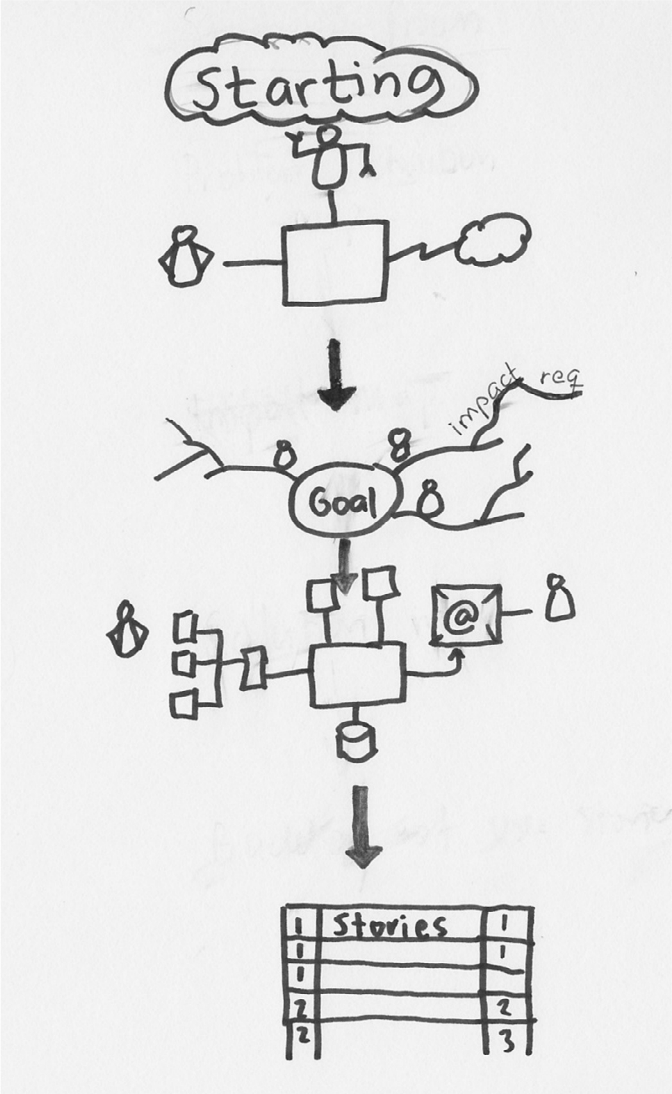
My role as tester is often about general problem solving. By working with visual test design from an early stage and make it visible to others. I analyze and challenge what we think we know.

Regardless if the model is called an impact map or a state graph, it helps to increase consensus and clarify doubts. The key is that models help us clarify ambiguities throughout the project lifetime.

In the Beginning There Was Darkness

Visualizing the problems is shining the light on shadowy areas to see what there really is.

Starting a project I use the following first steps:



Problem definition

Understanding and visualizing the problem. Where are we?

Impact Map

Getting to know what the users really want to have. Overall goal, users and what they need

Solution Map

Visually describing the solution

Describing our test strategy

To Do List

Creating a list of things to do. The product backlog

Defining the problem clearly from the start and creating the impact map helps us focus on the users' real needs. This serves as a basis for all our testing, especially scenarios and acceptance testing.

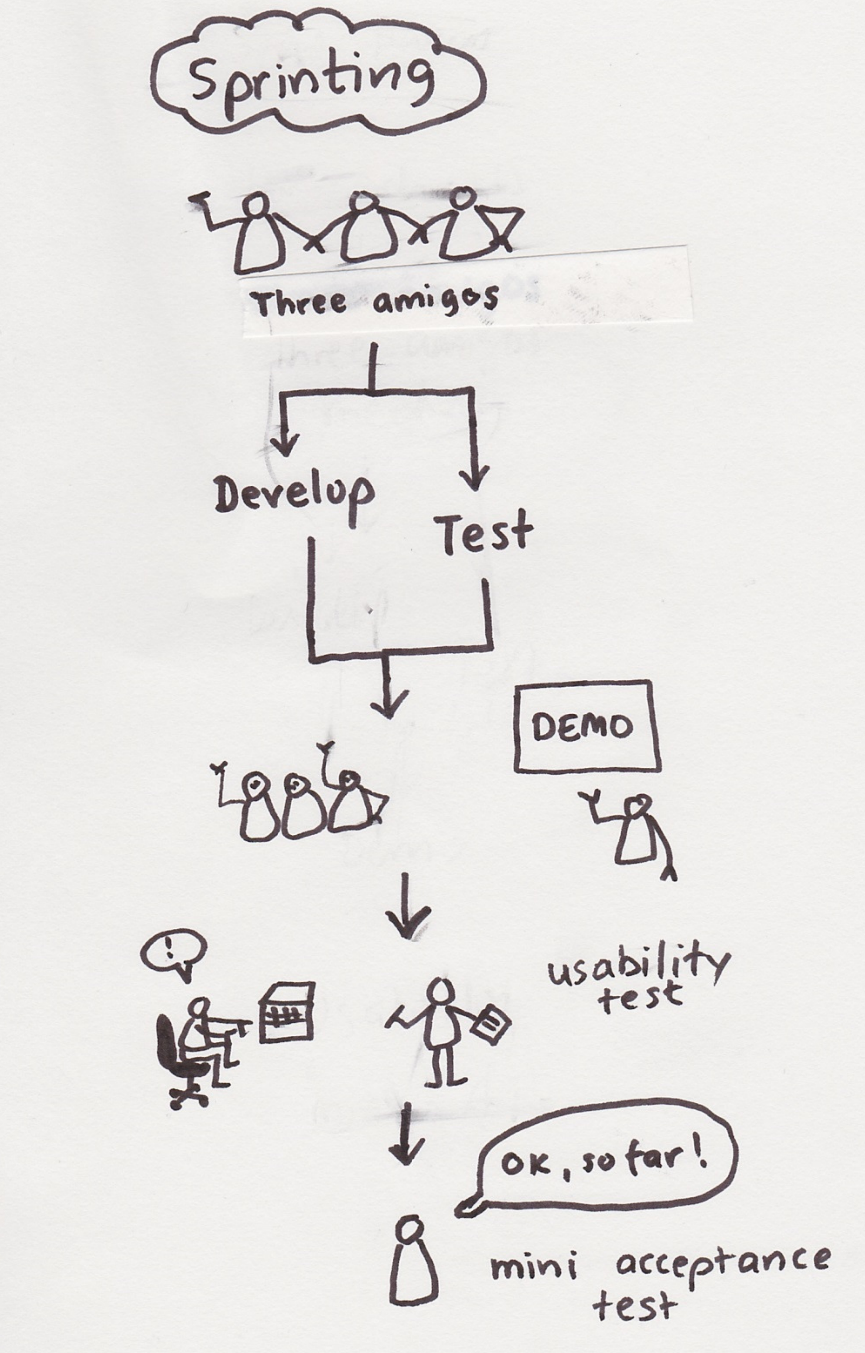
The solution map helps us plan the test environment and visually mapping our testing effort on the map. This is import input for our test design on where and what to test.

The story list is general used in the project to plan in what order to do things and serves as input for what we should focus on and in what order.

So is this really the tester’s job? Well, my point is that these four models are a great way to start a project. If someone else creates them, your job is to analyze and clarify anything that is missing or wrong. If they don´t exist I usually create them in order to structure my own work. Pretty often, but not always, the models are accepted as input to the project as a whole. If not, they still help me identify problems and guide my testing effort.

Now What? - The Sprint Process

Since all projects I worked in during the last five years use sprints I will use that to explain what happens after the planning part is done.



The Sprint Process

Digging deep into each item to do in order to agree. Visualize in test models if possible.

Coding and testing each item until we are done

Showing the client what we have done. Check functionality for each item in the plan.

Usability testing of all new and radically changed parts to get instant feedback

Let the client really feel the system. You must inform them what is not yet ready to test!

At the start of each sprint we have a planning meeting where we select what items on the product backlog to build in this sprint. In order to really understand what to build we started to have what we call *Three Amigos* meetings. They turned out to be really effective and the final acceptance of what we built turned out to be much better than when we did not have these meetings.

Three amigos

The purpose of the meeting is to discuss in detail what a user story really means. Developers like to have them in order to create a *given –when – then* chart that they can later automate. This table is then sometimes called the acceptance criteria for a story. The most extreme form of Continuous Deployment fanatics would want to automatically put the new code in production when these tests have passed. As a tester I think this is a bit dangerous…

Given	When	Then
Given that a user enters a veterinarian code	The button next is pressed	Show the Motivation for veterinarian page
Given that a user enters an invalid code	The button next is pressed	Show error message and stay on the entry page

Now a tester would want to enter all possible variations here but that would not be advisable at this moment of time at least. The main purpose is to identify important typical cases that we need to agree on how to handle. We can then add any number of variations we think are important to test but they will not necessarily be part of the acceptance criteria.

My problem with this approach is that we focus too much on automated checks and the developers need and it is not visual enough. Here I like to create some test design models that could also be tables but in another format and I often use state graphs or scenario descriptions to make it more visual.

Gojko Adzic has written about these things in *Specification by Example* and *Bridging the Gap.* Read about Continuous Integration and Continuous Delivery in the books with the same titles, written by Duvall et al. and Humble- Farley.

Develop and Test

We have divided testing into two separate parts. The first is fully automated tests that are run mainly to check that the basic stuff works, this is done partly by the developers themselves, partly by technical testers that focus on automation. With fully automated I mean that whenever a piece of code or test is checked in – all of the code is rebuilt, the database built and loaded with test data and then the automated tests are run for instant feedback. That is mainly what we use the given-when-then format for.

Now I dig into test design depending on what we are building. I show my models to the rest of the project continuously and ask for their feedback. A lot of the time we clear things out before I get to test things hands on.

Basically my idea is to build trust, credibility and mutual understanding with the developers. I try not to sound like I give any orders on what to fix but rather discuss what I found and ask for their opinion. Most of the time they take the initiative and say, we have to fix this.

The Demo

The demo meeting in my opinion is more of a project meeting activity with a fairly shallow presentation of what we have done so far. It is certainly good to have but does not give us nearly as much feedback as needed. So we decided to add a couple of more activities.

Usability Testing

Whenever we have a new graphical interface or a radically changed one, we do a simple yet effective usability test. This can be right after the demo or during the first days of the following sprint when we have some slack. The good thing about having this type of test regularly is that we get feedback from real users in real situations so issues on usability are not just regarded as the testers' subjective ideas than can be ignored. The client's opinions are rightly valued as more important to handle.

Mini-Acceptance Test

In order to avoid big catastrophes in the end of a project we decided to let out customers do acceptance testing continuously. It is very important to communicate what is not yet ready to test in order not to make them feel like nothing works. During a demo we can avoid doing things that we know are not really done although we claim they are but when we let the users free they can and often will spot our shortcomings.

Anyhow, the result from these mini acceptance tests is that customers realize that some of things they have asked for does not really work the way they figured they would. So often there are changes in the requirements as well. But when the final acceptance test is done at the end of the project there are rarely any objections to what is delivered and we have some very happy users. And that was the point of it all, right!

DEFINING THE PROBLEM

A curious thing about a lot of problem solving is that we don´t really seem to bother about understanding the problem before we have a solution ready in our minds. This refers not only to vendors of standard systems but to building IT-systems in general. To avoid having future users hating our solutions we are usually better off if we try to understand what their problem really is before we try to solve it.

HOW IT ALL STARTED

- It´s a mess sighed Agneta. We just don´t seem to ever finish this project. I really need this system to help me administrating the work with the journal.
- Well said developers AnniFrid and Benny with one voice, in IT we have these testers that are said to be really great. Maybe they can help us out.

They all agreed that this would be a great idea and proceeded to contact Wolf, the test department manager.

- Yeah, I got just the right guy for you. You can have Bjorn, our kick-ass consultant, he´ll test your system until it's great. He can practically do anything, big-shot guy!
- Eh, may I interrupt, says Bjorn – feeling a little uncomfortable with the direction of the discussion. I am happy to listen to your story and see how I can help. Don´t you have an important meeting to go to Wolf, he says.
- He´s fun but he talks too much, Bjorn says to Agneta with a smile.

Wolf decides that it's time to leave before any more jokes are made at his expense.

- Upper management Thursday video meeting in conference room VIP 2, he mutters while leaving the room.
- Now tell me about your problem Agneta.

A two hour interview follows where Bjorn takes a lot of notes and Agneta promises to give him access to all documentation that has been produced so far.

Bjorn reads all the documents, takes some more notes and decides to print all of it out in order to support the Swedish paper industry. Then he schedules another meeting with Agneta.

- OK, he says. I have read through all of the documents in the project map and I have some questions, rather a lot actually.
- Which of all these documents are still valid as of right now he says, pointing at all of the papers spread out on the big table in conference room the Seagull. Cause I found a lot of stuff that seem to be contradictions to what you told in our last meeting. Also the dates for the last updates for some of the documents are several years old. How long has this project being going on for?
- Well, it started and then stopped and then started again…a couple of times. But we never seem to be able to finish. But let´s get back to the documentation.

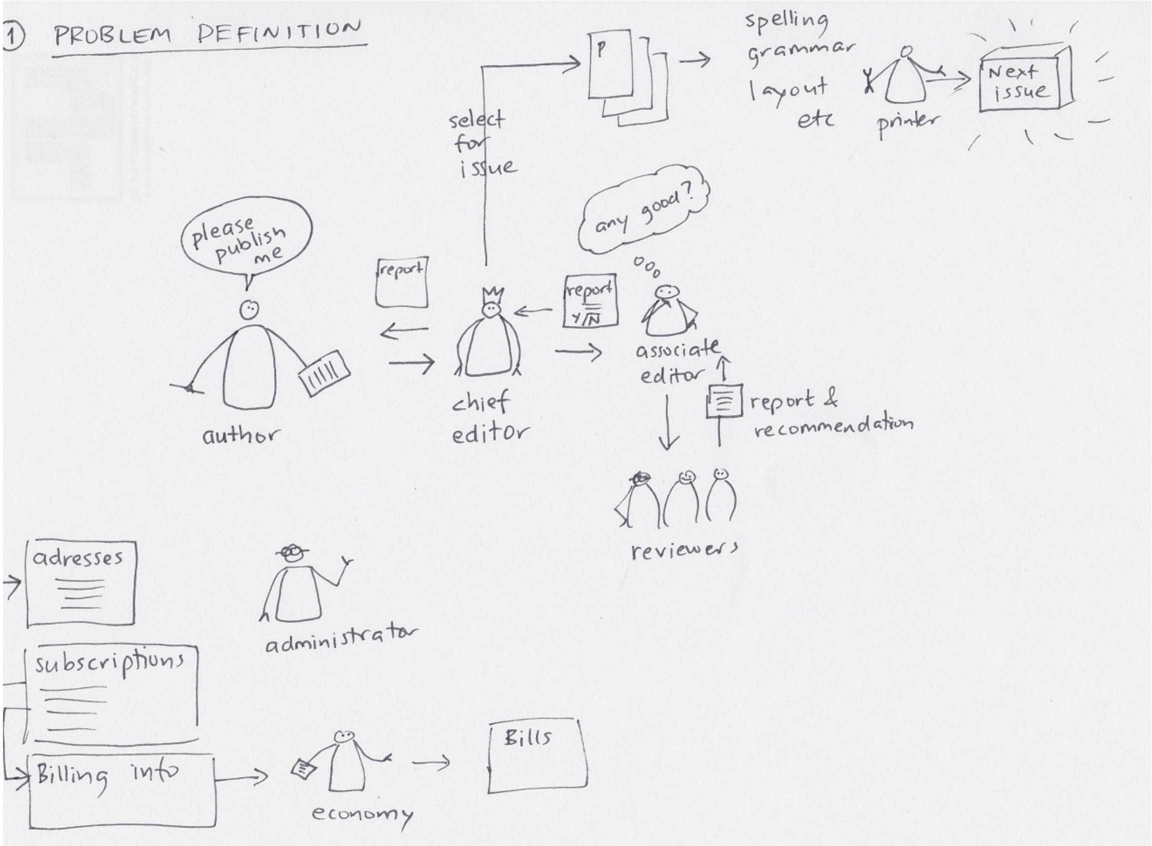
After an hour of looking through the papers they end up with three documents that seem to be the most recent and most likely to be at least partly true.

- Let’s analyze what we have so far. Three documents and my notes from the first meeting. Let me try to visualize the problem.

He grabs his pens and walks up to the flip chart. To small he figures and decides to use the whiteboard. He turns to Agneta who’s sipping a fresh cup of tea with honey and smiles encouragingly.

- Let´s begin where it all starts. An author has written a paper and wants it published in the journal.

He draws a simple stick man and a square with some text in. Then an arrow that connects to the chief editor in the middle. Using simple and easy to understand graphics helps the common understanding without intimidating anyone with special technical language or symbols that only a few people get. After an hour of drawing and discussing they have the first rough map of the problem. He takes a photo with his phone camera.



Picture: Problem map first sketch

This is a really god start. We have identified the main users involved and realize that there are some pretty distinct areas. One is the management of submissions and reviews, another creating the next issue and the third to handle subscriptions.

Exercise

Create a visual problem definition for the following:

In order to sell a medical product in Sweden it has to be authorized by the Swedish medical board (LV). In some situations a physician or a veterinarian wants their patient to have a medical product that is not authorized. The way around this is to motivate why the patient should be allowed to buy an existing but not authorized product and get a temporary license to buy. Today´s solution is that applications are faxed but tomorrow's solution is that all communication is done electronically. The law says that only pharmacists can apply for a license. This means that the motivation is first sent to a pharmacy, then the pharmacy creates the application based on the motivation already written. The Swedish medical board then decides to grant or deny permission for a temporary license or to ask for additional information before they can decide. Either the physician or the pharmacy can be asked for additional information but not both at the same time – at least that is our hypothesis. When a temporary license has been granted – the pharmacy can order the product and sell it to the patient. If a license has been denied – the physician can appeal by going to court but this is said to be outside the scope of the project, at least for now.

At this point of time you are all thinking solutions but try to refrain for a while. Just visualize the problem. Decide what parts to draw and what parts to omit. Don´t worry too much, you can always redraw.

Business Impact mapping

To really get into the needs of what the goals really are, Bjorn creates a business impact map, also known as effect map. This helps us focus on what we need before we start solving the problem. This map usually contains a lot of detail so the manual option here is good as a start but the best choice is usually a tool of your choice. There are plenty of free mind-map tools, which is essential if someone else is going to update the map in the future. The creation of the impact map follows he same four basic steps as the creation of the problem map.

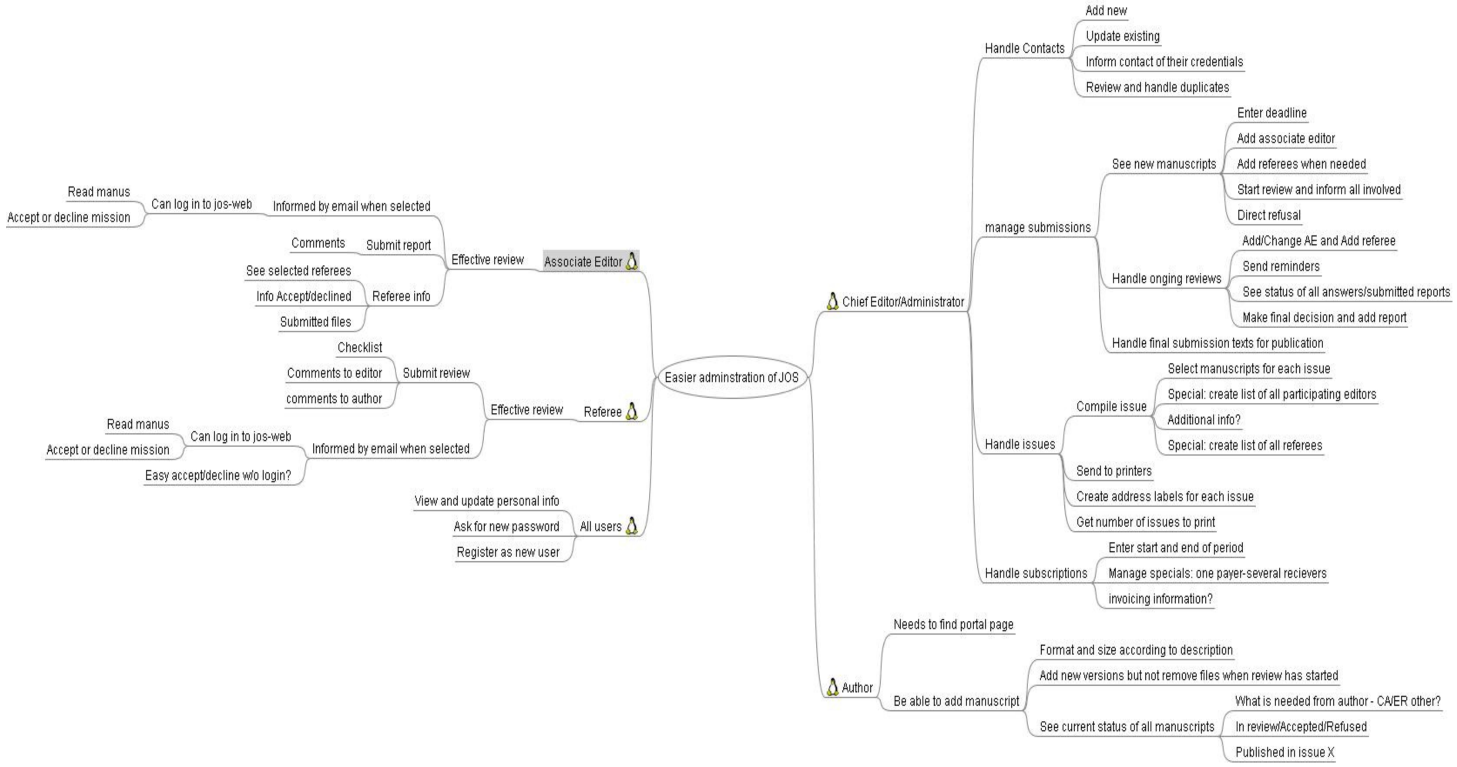
The business impact map was originally created in Sweden by Ingrid Domingues (Ottersten) and Mijo Balic as a means to understand better what their clients really wanted when creating web sites. It has started to spread in the agile community largely thanks to Gojko Adzic. I have used the impact map in several projects in order to support both requirements and testing. The key success factors are

- Visibility: You get the whole map on one single page
- True needs are identified: first needs – then solution
- There is a connection between need – requirements and the solution

We have to be honest to ourselves. A lot of the time we – IT-people that develop systems – are eager to solve problems but we hardly ever understand what the problems really are. I am not going to rant too much about that fact right now though but prefer to offer a step on the way to make the users of our systems happier.

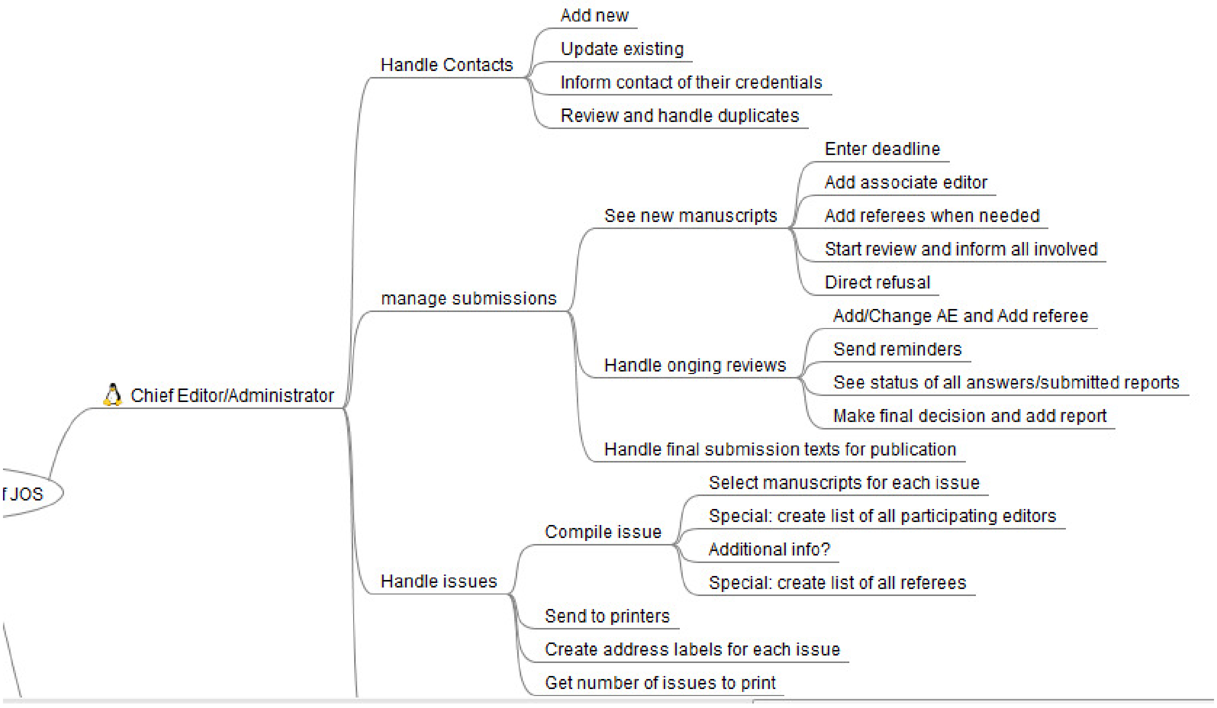
Bjorn refines the map over the problem area consisting of all people and parts involved and also puts together all of the needs of the different people in a business impact map. He does all of the work on paper since this seem to stimulate the creative process. A side effect is that people seem to be more interested in his little hand drawn stick men than the clip art diagrams he used to put together. Suddenly during discussions often someone grabs a pen and start correcting his drawings. He meets with Agneta several times over the week and in the end they have a map that they agree on. Bjorn has invited Agneta to another meeting to show her the latest version of the business impact map!

- Hi Agneta, I have tried to put together the complete picture of what you want the system to handle. Let´s have a look and see what needs to be changed.
- I think you have the target groups alright and the main goal is to make the administration of the journal easier.


Easier adminstration of JOS
Associate Editor
Effective review
Informed by email when selected
Can log in to jos-web
Read manus
Accept or decline mission
Submit report
Comments
Referee info
See selected referees
Info Accept/declined
Submitted files
Referee
Effective review
Submit review
Checklist
Comments to editor
comments to author
Informed by email when selected
Can log in to jos-web
Read manus
Accept or decline mission
Easy accept/decline w/o login?
All users
View and update personal info
Ask for new password
Register as new user
Chief Editor/Administrator
Handle Contacts
Add new
Update existing
Inform contact of their credentials
Review and handle duplicates
manage submissions
See new manuscripts
Enter deadline
Add associate editor
Add referees when needed
Start review and inform all involved
Direct refusal
Handle onging reviews
Add/Change AE and Add referee
Send reminders
See status of all answers/submitted reports
Make final decision and add report
Handle final submission texts for publication
Handle issues
Compile issue
Select manuscripts for each issue
Special: create list of all participating editors
Additional info?
Special: create list of all referees
Send to printers
Create address labels for each issue
Get number of issues to print
Handle subscriptions
Enter start and end of period
Manage specials: one payer-several recievers
invoicing information?
Author
Needs to find portal page
Be able to add manuscript
Format and size according to description
Add new versions but not remove files when review has started
See current status of all manuscripts
What is needed from author - CA/ER other?
In review/Accepted/Refused
Published in issue X

Picture: The full business impact map

- Let´s zoom in on the administrator’s tasks. Here I have noted four different needs. However the management of subscriptions you told me has low priority right now so we´ll skip that for now. The first need concerns how to handle all of the addresses of the registered contacts. The administrator needs to be able to add new contacts but we have not yet specified how this is going to be done.
- Yes, and I need to be able to search and analyze the current list of all contact in order to remove or update invalid ones. But you missed my button!
- Eh, what button are you talking about?
- The button where I print out a list of all contacts to send to the distributors.
- OK, I understand. That´s a button you have in the existing system. I suggest that the real need you have is that all subscribers must get the next issue. How we get those addresses to the distributor is part of the solution. There may be a better way than you printing them out manually. I will make a note of that under *Handle Issues – Create address labels for each issue* to make sure that it is taken care of.



f JOS
Chief Editor/Administrator
Handle Contacts
Add new
Update existing
Inform contact of their credentials
Review and handle duplicates
manage submissions
See new manuscripts
Enter deadline
Add associate editor
Add referees when needed
Start review and inform all involved
Direct refusal
Handle onging reviews
Add/Change AE and Add referee
Send reminders
See status of all answers/submitted reports
Make final decision and add report
Handle final submission texts for publication
Handle issues
Compile issue
Select manuscripts for each issue
Special: create list of all participating editors
Additional info?
Special: create list of all referees
Send to printers
Create address labels for each issue
Get number of issues to print

Picture: the administrators needs

In the end we will have one single model covering all of the needs. If it grows large we need to print it out big. A3 may be large enough for smaller projects.

Next step is to prioritize. This can be done on any level you want from focus group to single requirement. After that is done we can use the map to create user stories or some other format we prefer. It maps pretty nicely to the user story format.

Example: User Story: As an administrator I need to be able to add new contacts in order to keep the contact list updated.

Now we may also realise that the effect maybe should be called *Keep contact list updated* instead of *Handle contacts*. You chose, the important thing is that the message is clear.

EXERCISE:

Create an impact map for the license application system.

Patients are sick and in need of help. They want to be able to buy what the doctor prescribes. Most applications start when a human visits their physician. The requirements for the patients are a bit vague since they are not really users of a future planned IT-system. Think of what a patient might want or need in order to get well. There are some variations of licenses: a medical clinic can apply for a general license that enables them to buy a product to several patient during two years at a time. A veterinarian can apply for a license for either a single animal or a group of animals like all cows at a farm.

A physician has to enter their personal id in order to get write the motivation – that has to do with security. The code is checked against the entries in an existing system called PRELIST. They have to enter a motivation and select a valid product that is listed as being available on the market but not authorized. This system is called

MEDPROD. They need to be able to attach documents to least the most a their motivation. They can but do not have to enter a specific pharmacy where to send the motivation. They need to somehow be able to get current status of the application, especially the end result and if they need to enter additional information. Entering additional information means the motivation will not have to pass through the pharmacy again.

The pharmacy needs to know what motivations are addressed to them directly. This is their backlog. They also need to be able to search all existing motivations that are in the systems in order to create applications on non-addressed motivations. They need to know the status of ongoing application – especially if they need to enter additional information and when the final decision is made so they can order the product and sell it to the patient. Important fact is that in order to sell something, a valid prescription is also needed. That is currently handled by another system called PRESCRIPTS.

The Swedish medical board needs to know the status of all ongoing applications, especially those that it’s their turn to handle. This means new applications and new additions to ongoing applications. They need to be able to read all of the information in the application and the motivation. They need to enter their decision in the system so that the pharmacy, patient and physician can be notified. They also need some help in knowing what they need to do and possibility to handle applications internally. That means assigning to a specific person or group, prioritizing internally. Also they need to be able to search earlier applications in order to re-use similar or same reasoning when making their decisions.

There is a general requirement that the system protects sensitive information and stops unauthorized use. Since the number of users that will apply only a few times a year is large – 30 000 physicians and 5000 pharmacists. The system must be intuitive and stopping users from at least the most obvious mistakes. Number of applications each year is 100 000, number of required additions today is 30 000. The performance must handle this amount of traffic.

Visualising the Solution

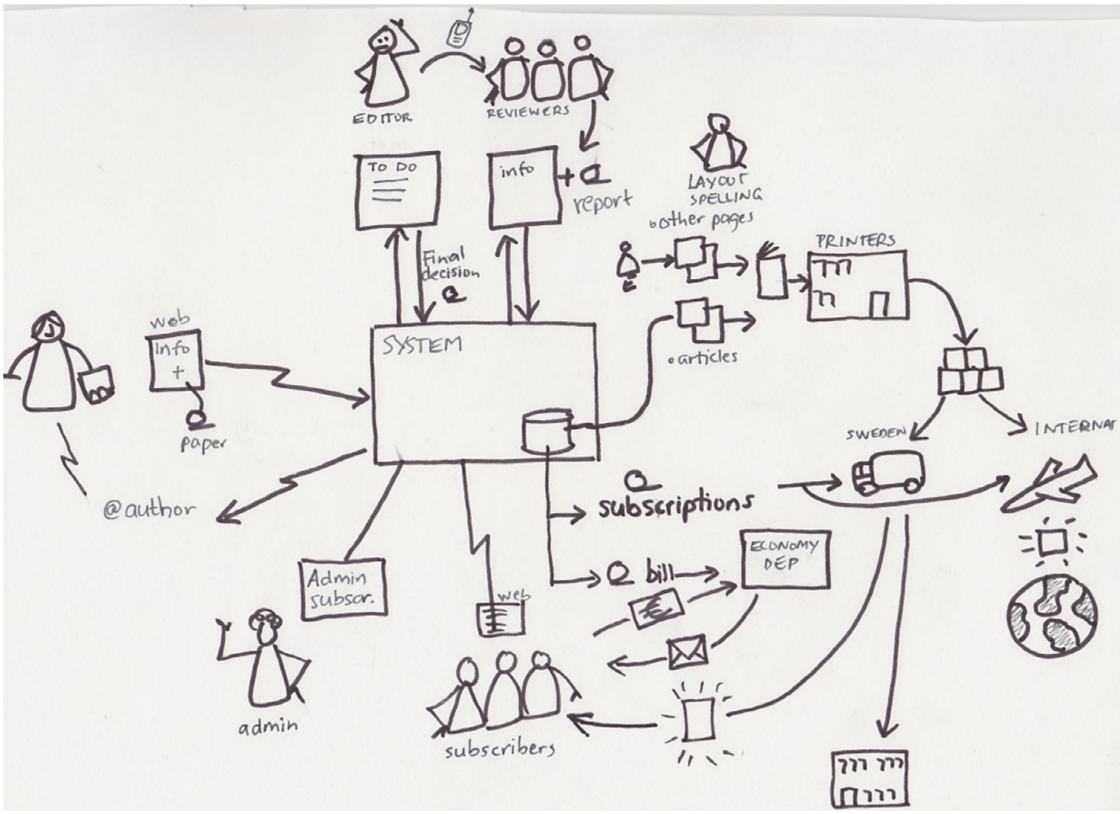
The textual description of the suggested process is:

- A person that wants to submit a manuscript
  - uses the link on xxx.yyy.se to get to the registration page. Enter First Name, Surname and e-mail
  - A confirmation mail with a link to confirm registration is sent to the e-mail registered
  - When the confirmation link is clicked the user is registered correctly and an e-mail with user id and password is sent out
- The person can now login to the system
  - And enter the manuscript that they want to have published
- The chief editor
  - is now informed that a new manuscript has arrived and is ready to be reviewed
  - enter a deadline for the review process (normally 8 weeks from start date)
  - selects an associate editor from the contact registered in the contacts list
  - selects five reviewers from the contact list. If necessary adds new contacts
  - a number of e-mails are automatically sent
    - the person submitting the manuscript gets a thank you note for submitting and info on what will happen next
    - the associate editor gets a mail describing that he/she are selected and what they will have to do and what reviewers are selected
    - the five reviewers get a mail describing that they are selected as reviewers, what the work is about as well as user id and password for JOS.
- The associate editor can now
  - Read the manuscript
  - accept or decline the task. Their choice is registered in JOS.
  - If they decline the chief editor should be informed

…etc

In close cooperation with the developers Benny and AnniFrid, Bjorn creates the solution map. We know more in detail what information needs to be sent to whom. And since the project already has a version of a solution, some things are already in place.

This serves as a basis for selecting test areas, what points to interact with the code, what types of tests to use where and the boundaries for what we test. I also connect this to test environment and test databases to get an overview.



Picture: Solution map.

Exercise:

A) Simple: Suggest test boundaries and test types for the map of the journal system above. Add them to the map.

B) Harder: Create a rough system map for the license management system based on the following design decisions.

1. The physician will have four pages to select from. The first is a login age. Then there are three choices: enter new, status existing and add information to existing.
2. The pharmacy will have a main page with a to do list where they can also search, a page to make new applications and one to add information to an existing one.
3. All webpages communicate via web services. All of the information entered through the web pages is saved in a database.
4. The Swedish medical board has a system where they look at all information in the applications and make their decisions based on that information.
5. We integrate with their system via a file area but are not responsible for usability or function of their system
6. How can the testing effort be visually and textually entered into the system map to get a good overview? This will then guide your test design.

Managing the Example Project

- But how do we get this thing finished Bjorn, says Agneta during one of their regular coffee breaks. It´s been going on forever and we just can't seem to get things done.
- Well, I think one of the problems is that you try to do everything at once and therefore manage to finish nothing. I suggest that we list all the things you want done in a spreadsheet and then sort by priority.
- So all the priory ones first then?
- Hmmm, that might not be the best solution. I think we should first try to get a couple of developers and then go through this together the four of us.

Bjorn ends up being named Scrum master since he suggested they use that method to get working quickly. It turns out that the appointed project manager is mostly administrative and does not really want to get his hands dirty actually trying to run the project. He does however book the needed resources and then to let the other project members manage themselves which turns out to be a pretty good idea.

- Welcome to the restart of the journal project. Bjorn greets the web developer AnniFrid and the back-end developer Benny. So let me brief you on what we have got so far.

The four of them spend a full day at a nearby conference place going through the problem definition, the impact map, solution map and possible ways of working together. They also enjoy a good lunch and find out that they have children the same age that are really into ponies.

- Looking at the system map and the impact map, I think the best solution is to build sprints by role says AnniFrid. If we start by building all of the functions that are needed for the author to submit a paper we have something we can easily demo. We have made some estimations for the user stories you listed and think at least I can fit that into my schedule.
- Yes, Benny agrees. That gives me a chance to make sure everything is connected. There is some code already so I think we can manage that amount of work in two weeks. Is that OK with you Agneta?
- Sounds great if I actually can get something in just two weeks.
- What about the testing Bjorn?
- Well I can surely test from the web pages you create. I will then need access to the database to check that data is updated as planned. I will probably have some questions regarding that but until I can have the administrator page up and running that will have to be the solution for checking the results. Besides your suggestion to build by role fits nicely with my suggested test area listing.
- Ok, then that's our plan they say in unison.

The outcome from a day well spent is the following decisions:

1. Bjorn will put all needs into a spreadsheet and Agneta will review them
2. The developers will suggest in what order to build what is left in order to optimize their effort

3. Starting next week we will have weekly meetings where we discuss what has been done and decide what to do next. We will go through all items so we agree on what they mean
4. Every other week we will have a demo of what’s new
5. After each demo.
6. Agneta will do a mini acceptance test supported by Bjorn in his role as test manager
7. Each time we build a completely new part of the system we will also do some fast and frugal acceptance testing in order to catch bigger mistakes quickly
8. Other than that. When we feel the need to communicate, let’s e-mail or better yet, meet in person
9. Bjorn will create a Scrum board with columns to do, developing, test and done

To do Spreadsheet – AKA product backlog

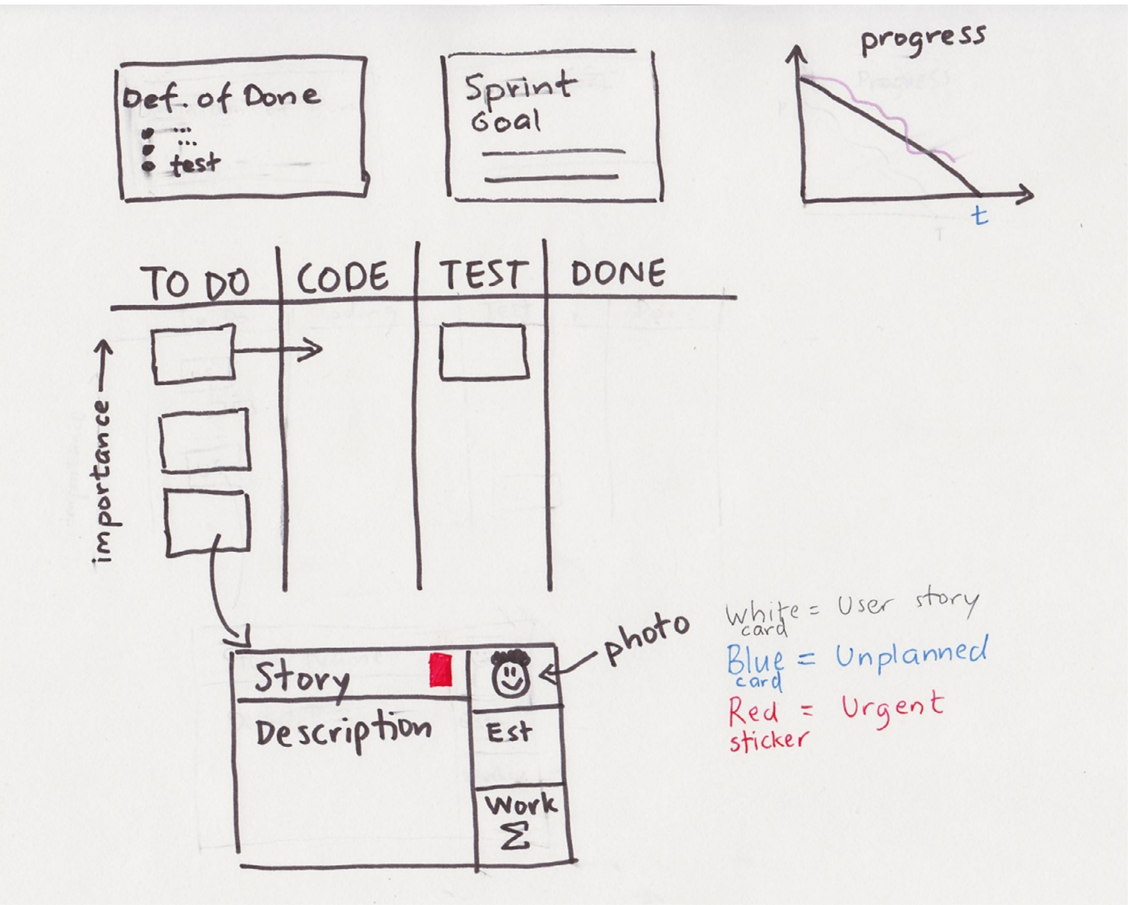
This list is updated daily and discussed at each weekly meeting. Sprint planning is done on priority and available hours.

Sprint	Description	Estimate	Priority	Responsible	Status
1	Submit manuscript including info needed. Front-end	40	1	AnniFrid	Ongoing
1	Submit manuscript including info needed. Back-end	40	1	Benny	Testing
1	Update DB w new fields	5	1	DBA	Done
2	View new submission	8	1	AnniFrid	
2	Sort and search submission	8	2		
99	Stuff not so important		3		

Scrum Board

I have used a number of different boards. It is the nature of scrum to create your own board design. This a fairly accurate description of how we do it:

1. A sign with the definition of done ready for everyone to see – it is easy to forget the starting goals when time is starting to run out.
2. A sprint goal – basically what we will demo
3. The progress graph is updated every day at the daily scrum
4. Stories are usually the basic element – if they are too big we divide them in tasks of recommended max estimate 8 hours.
5. Importance of what to do is from top to bottom – at least that is the idea
6. Every task has the photo of the person working on it for everyone to see
7. Work so far is added daily as well as the estimate of remaining work day
8. Unplanned work is put on a blue note and specific test tasks on green paper
9. Problems are tagged with a red sticker



Picture: scrum board

Reporting

Daily test reporting is done on the daily scrum. I also write a weekly report, a sprint report and a final report. In order to get them done and to have people actually read them I keep them short and informative.

Weekly

I use a weekly test report containing

- What have we done since last report
- What are we doing now and the coming days
- What problems do we have

I also use a simple dashboard that I always keep updated. When asked for the current status of testing I can get an updated report in a few minutes. Sometimes we put this table up on the wall depending on the need for information sharing.

Test dashboard						
Text	**Preparations**	**Func**	**UX**	**Effort planned**	**Quality**	**Comments**
Migration of data	Done	Ongoing		Large	☹	
Submissions	Done	Done		Normal	☺	
Editor tasks	Started			Low	😐	
Reviewing						

Sprint Test Report

I use the same dashboard but change the reporting points to

- The testing effort this sprint: result and time
- Problems we need to take care of so the next sprint can work better for us
- Any other comments that we see fit

Final TEST Report

So should test give a recommendation or not? Well I don’t think that the testers are the gate keepers of quality or that is our decision to decide if we should go live or not. But I do believe that after testing the application we should have an opinion and not only report facts of what we have done. I we find serious bugs all the way into the last day of testing we should inform what we feel about that and how ready the system is to release. The decision is made on a different level anyhow.

I usually sum up important happenings and the total effort and budget. Other than that I may have to supply information required by others because of a template or process.

Use a dashboard over the testing effort per month or per sprint.

Test Status per sprint						
Test area	Priority	1	2	3	4	Comments
Migration of data	High	🟥 ☹	🟨	🟩	🟩	
Submissions	Normal	🟩 ☺	🟩	🟩	🟩	
Editor tasks	Low		🟨	🟩	🟩	
Reviewing		🟨 😐	🟥	🟨	🟩	

Introduction to Test Design

So now when I have spent so much time on impact mapping, solution map and managing the project do we get to dig into some core test design? Yes of course we will. The aim of this paper is not to make a complete listing of techniques but to make examples some of the more visual ones.

Choosing Techniques

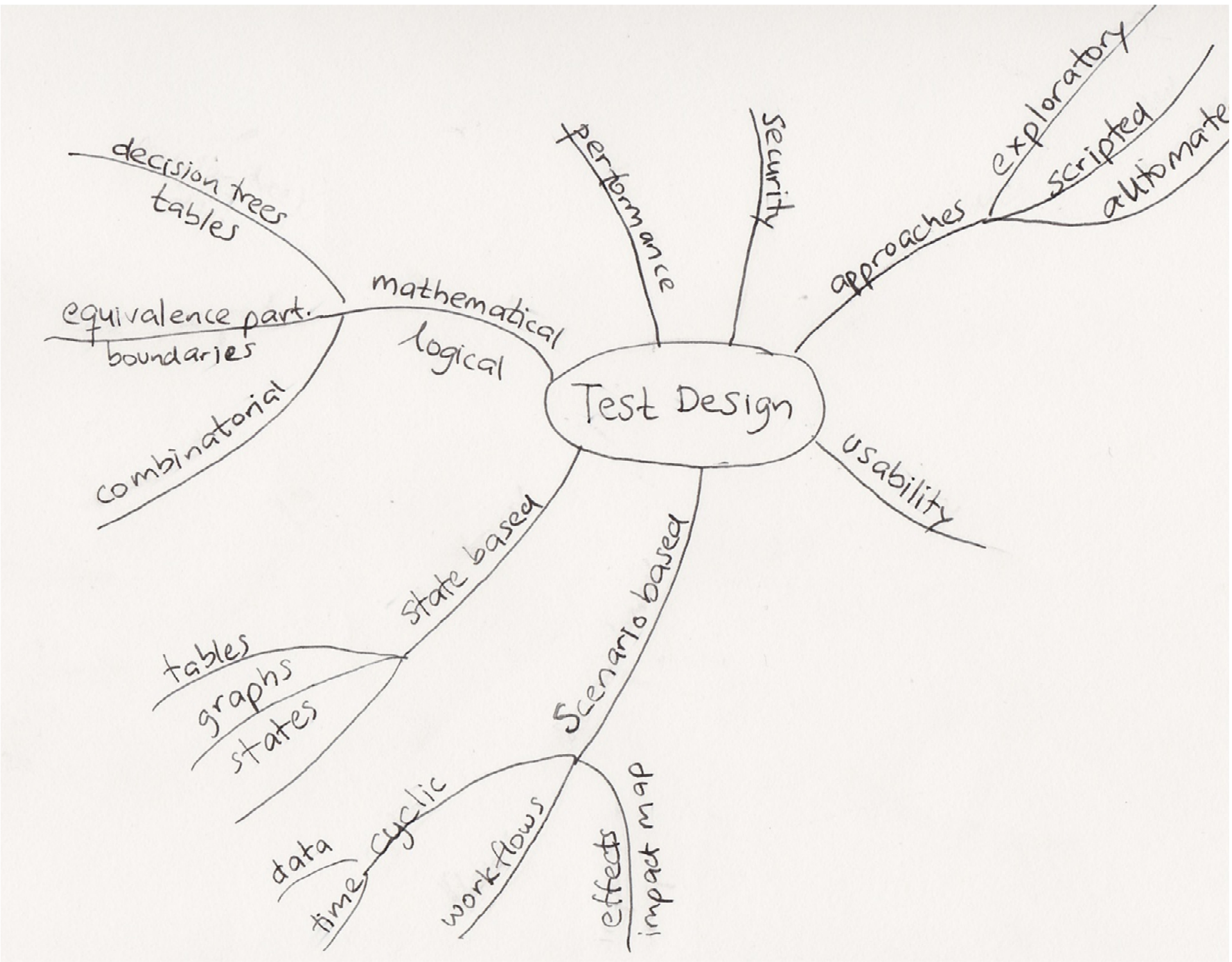
There are plenty of useful techniques that focus on different aspects of testing. Rather than trying to list as many as possible here I will instead discuss the thought process and give you some examples. Many of the attempts to list the techniques end up with mostly the mathematically and logical. Then it is popular to add what is called experience-based testing and exploratory testing. Experienced based testing is a rather sloppy way of lumping lots of techniques together and ET latter is an approach and not a technique.

Certainly there are a lot of situations where the mathematical techniques are useful. Working in banks for eight years there have been plenty of rules and formulas to validate. Equivalence partitioning, combinatorial coverage and decision trees/tables are some of the most useful ones.

State-based testing which is often very close to the actual implementation is a technique I find powerful a lot of the time. It is also easy to make very visual.

Scenario testing is useful when focusing on the system as a whole either in business processes, usability or the effects listed in the impact map.

Performance testing is always important since it will disturb the users experience if using the system feels like one of those horrible dreams where you run as fast as you can but do not seem to move at all. How much testing you have to do depends on number of users, environment and how much data there is. Most performance tests I have seen are rather performance measurements often with some trimming. For a pharmaceutical application with a load of more than ten million requests per day it is extremely important to performance test. For the example applications in this paper with three simultaneous users and a five hundred request a day we will normally chose to have a more limited approach.

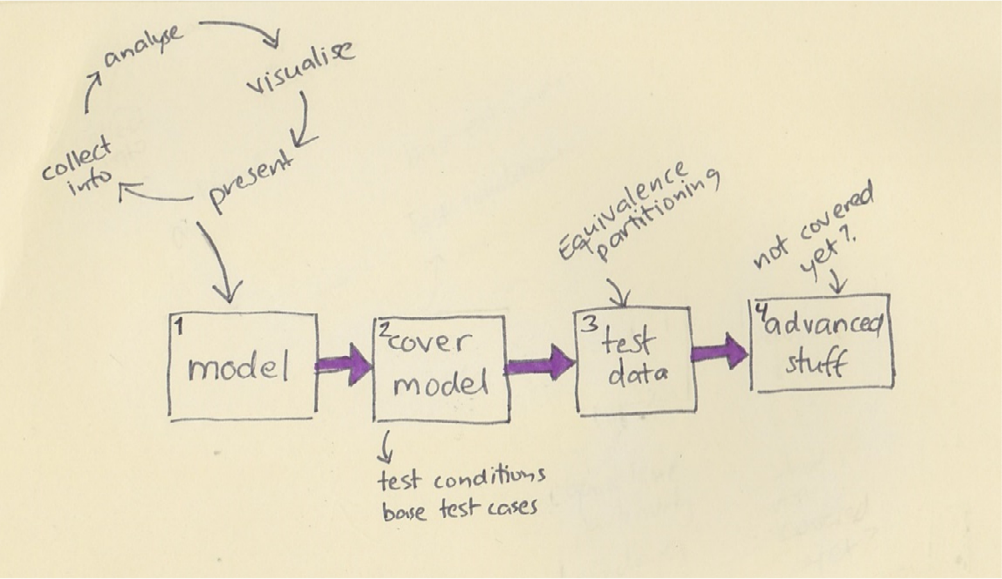


Picture: Test design techniques and approaches

THE OVERARCHING PROCESS

When I wrote the book Essential Software Test Design I made a simple general model of the steps. I think that it still works fairly well to explain the four steps of test design.

1. The first step is to create the model. It is actually the four steps previously mentioned in visual problem solving shown as one single step. Gather data, analyze, visualize and present for others. Repeat until you agree.
2. Using the agreed upon model we now have to cover it. Covering means that any boxes or arrows in a graph or a cell in a table is touched by why I call a base test case. Some like to call it covering the test conditions.
3. If we cover a specific test condition there are usually many variations of test data, many more that are possible to cover. We will have to do our best to find a limited set that still has god coverage. This is where techniques like equivalence partitioning and combinatorial testing comes in to support other techniques.
4. Then I realized that the nicely structured approach of the techniques needs to be supported by thinking that would not fit into a formula or number of steps so I added advanced testing as a last step. This really means that we have to think one more time about what we have not covered yet using the selected technique.



Picture: The four steps of the generic test design process

State Transition Testing

State based testing is often closer to the real implementation than flows. I find it much more powerful than flow graphs when describing complex systems. For me testing very often includes checking what is in the database and here states come naturally. The chapter *State based testing* in my book covers that in detail.

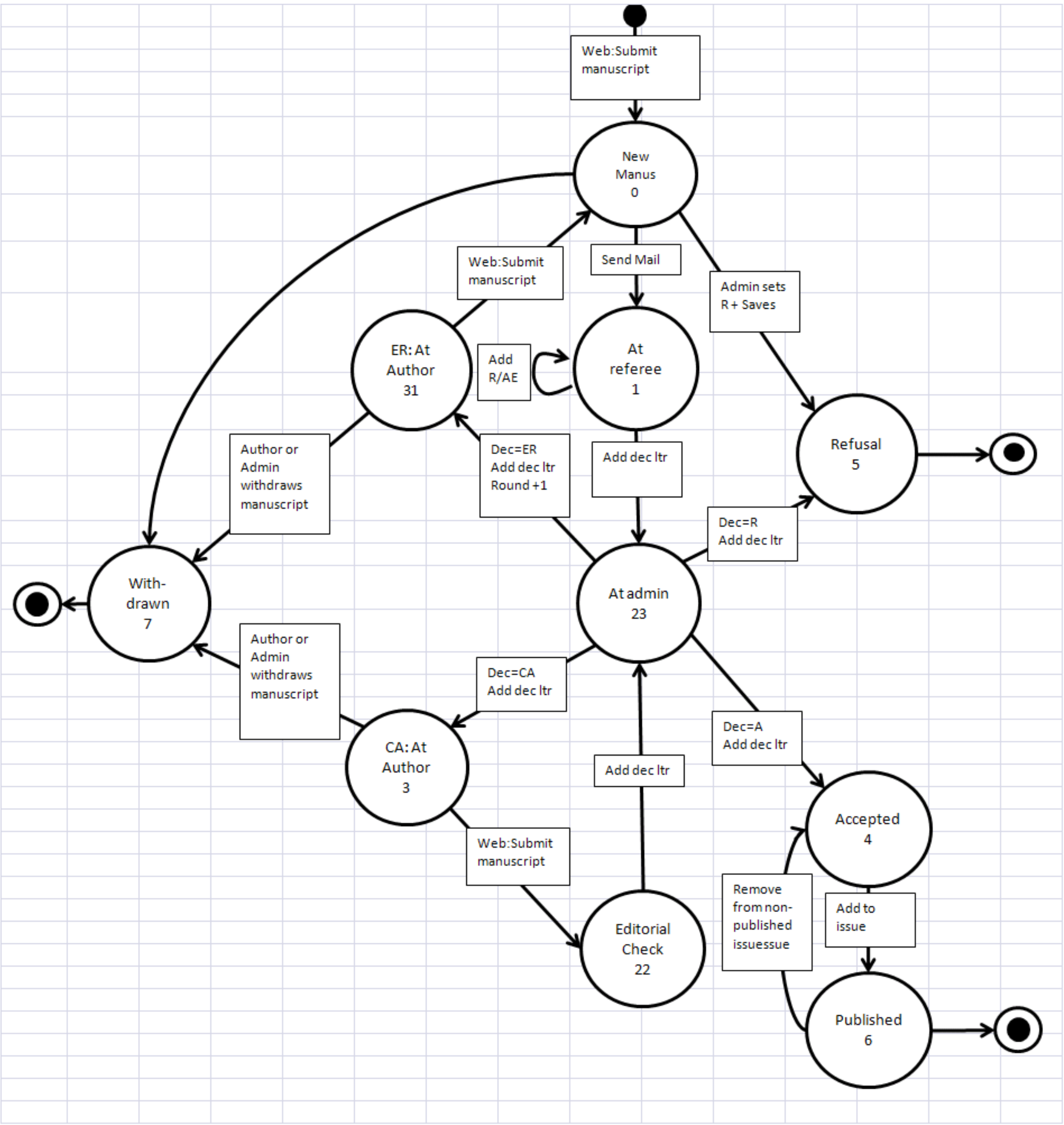
I think the easiest way to explain state graphs is to show you an example.

The state graph

For the journal system one of the key components is the submission. For a large part of the application everything revolves around what states the submission are in and what we have to do in order to go to the next state until we have reached an end point.

Coding wise, the developers somehow need to know what to show the user on a specific screen. This is usually done by a select statement from a database. Similarly when I test something I think along the lines – if I press the button *Submit paper*, then I should do a select statement where status is *New Manus.* And if I then go to the administrator's page the newly submitted manuscript should show up with the right status. If the chief editor choses to refuse the submission by clicking that button in the application I need to do another select to see that the status has changed and on the administrator page it should no longer be possible to edit the submission.

In this particular project we had a lot of discussion on when to show submissions and when to send out e-mails. I created a state graph and presented it for the team. We then had an excellent discussion around what would help the developer's code and we ended up with the graph below. In fact, this graph ended up as one of the more central documents both for coding and testing.



Picture: State graph of submission states

How to read the graph: When a manus is in the state *New Manus* and an assistant referee is added, the status will change to *At referee* and an e-mail is sent to the author saying that we started the review. We can then add reviewers or change assistant editor without the state being changed. During the review process the reviewers and associate editor add decision letters when they are done reviewing. The state the changes to *At admin* where it stays until a new decision letter is added and a decision made to *Refuse, Accept* or to ask for some smaller changes called *Conditional Acceptance* or a lot of changes called *Encouraged Refusal.* Withdrawn means that the author does not want to continue the process. The arrow back from *Accepted* to *Published* means that the editor can change their mind on what papers to include in the next issue.

The graph look fairly easy but it took many hours of discussion and redesign before we finally agreed on this final version.

Covering the graph with tests

The next step is to cover the graph with tests. This means that we make sure that every state is covered. Then cover every transition and t last combinations of transitions until we have complete flows from start to end. More combinations give means more detailed testing.

The third part of the generic model is adding test data. In our example this can mean adding more reviewers, some that accept and some that do not. Of those that accept some actually review and some don't. You get the picture, I think.

The fourth part which I mysteriously called advanced testing is everything you can imagine outside the neatly structured process. Like endless loops reviewing, changing editors and reviewers in mid-process or adding and deleting attachments. Some would like to call this experience based testing or brain storming and would be a fairly accurate yet somewhat vague description of what you actually do.

State Descriptions

Depending on what type of system you are modeling, the states have different attributes. You have to decide what attributes to include the model, every additional attribute may double the number of possible states. Modeling a mechanical device like an ATM-machine could have a state description when the machine is open and ready for a customer described by the following attributes:

1. Show on screen: Welcome, please enter your card in the slot
2. Card slot: active, meaning that if someone puts a card in, the machine will feed it in so it can read it
3. Numerical Keyboard: off, no input will be registered at this time
4. Extra buttons for fast selection: off, no input at this time

Example journal system

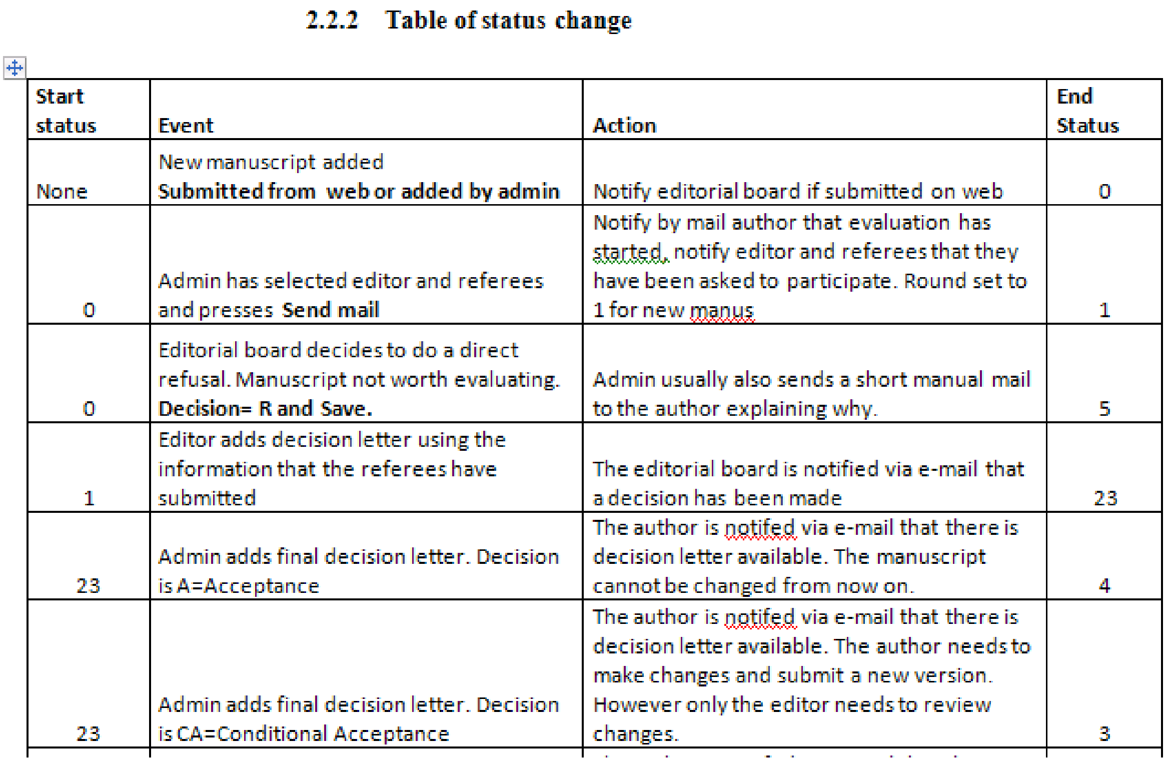
The journal system is a bit different. Here the attributes are places the submission can be seen and edited and by whom. When a paper is submitted – does it show up in the administrator's inbox? When a review has started, does that show correctly in the administrator's screen, on the reviewer's screen, for the author? What attributes can different people read or edit during different stages of the review process?

	Submitted	***Under review***	***Accepted***	***Published***
Author page	Shows submitted. Cannot edit submission info	Shows review. Cannot edit submission info	Shows accepted. Cannot edit submitted info. Is allowed to add new files with changes	Shows Published. Can only read submission overview and no files.
Admin page	Shows submitted	Shows under review. Can edit review info	Can select for issue. Cannot change info	Cannot change any data. Can remove from system.
Ass. Editor page	Nothing yet	Shows under review. Can edit review info	Shows as accepted. Cannot change anything	Shows as accepted. Cannot change anything
Reviewer page	Nothing yet	Shows under review. Can edit	Shows as accepted. Cannot change	Shows as accepted. Cannot change

		review info until own review is submitted then cannot edit	anything	anything
Database	Check tables X,Y,Z for each state	Check tables X,Y,Z for each state	Check tables X,Y,Z for each state	Check tables X,Y,Z for each state

STATE CHANGE TABLE

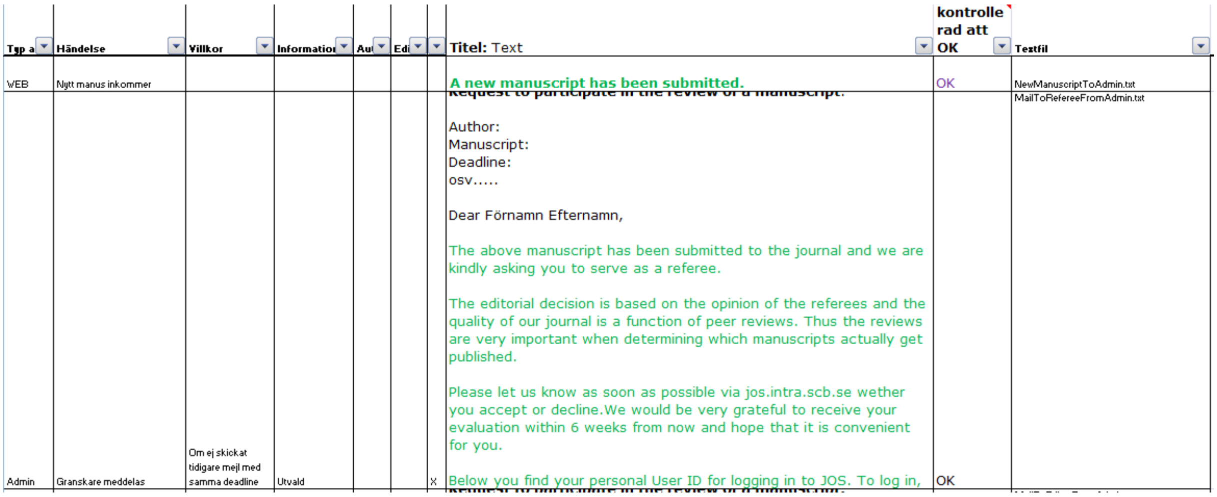
I can be very useful to create a table where each state transition is described in detail. What are the events that trigger a change of state and what is the response from the system.



2.2.2 Table of status change

Start status	Event	Action	End Status
None	New manuscript added **Submitted from web or added by admin**	Notify editorial board if submitted on web	0
0	Admin has selected editor and referees and presses **Send mail**	Notify by mail author that evaluation has started, notify editor and referees that they have been asked to participate. Round set to 1 for new manus	1
0	Editorial board decides to do a direct refusal. Manuscript not worth evaluating. **Decision= R and Save.**	Admin usually also sends a short manual mail to the author explaining why.	5
1	Editor adds decision letter using the information that the referees have submitted	The editorial board is notified via e-mail that a decision has been made	23
23	Admin adds final decision letter. Decision is A=Acceptance	The author is notifed via e-mail that there is decision letter available. The manuscript cannot be changed from now on.	4
23	Admin adds final decision letter. Decision is CA=Conditional Acceptance	The author is notifed via e-mail that there is decision letter available. The author needs to make changes and submit a new version. However only the editor needs to review changes.	3

The rules for when to send e-mails turned out to be fairly complex and under constant change. We added another table with detailed rules for sending out e-mail.



Typ a	Händelse	Villkor	Informatio	Au	Edi		**Titel:** Text	**kontrollerad att OK**	**Textfil**
WEB	Nytt manus inkommer						**A new manuscript has been submitted.**	OK	NewManuscriptToAdmin.txt
Admin	Granskare meddelas	Om ej skickat tidigare mejl med samma deadline	Utvald			X	**Request to participate in the review of a manuscript:** Author: Manuscript: Deadline: osv..... Dear Förnamn Efternamn, The above manuscript has been submitted to the journal and we are kindly asking you to serve as a referee. The editorial decision is based on the opinion of the referees and the quality of our journal is a function of peer reviews. Thus the reviews are very important when determining which manuscripts actually get published. Please let us know as soon as possible via jos.intra.scb.se wether you accept or decline.We would be very grateful to receive your evaluation within 6 weeks from now and hope that it is convenient for you. Below you find your personal User ID for logging in to JOS. To log in,	OK	MailToRefereeFromAdmin.txt

Exercise

Create a state graph, a state change table and a state attribute/test check list for the license application system previously described. List some ideas of how you would do to test using the models you created.

Scenario Testing

Start with the impact map to find out what different users want to achieve. Start with the easy scenarios where everything works out fine without any problems. Gradually increase complexity to cover special needs, error handling and combinations but always try to finish the flow from start to end.

I always try to keep instructions short and to the point and rather use overviews in the form of bullet lists or tables. So many things will change during the project that if you write detailed instructions they will be wrong by the time you start testing and you will start hating everyone for making changes since you have to do a lot of rework.

When executing these tests, the simple instructions and exploratory testing will help you find a lot of possible problems. Use the test design models as an execution log as well, that will save you some time and give you an excellent overview of what you have done and where the problems are.

Workflows

Think of this as thin red threads starting at one of the natural starting point ending up in one of the final states. Then start messing with the threads so they get messy and not go as straight as they can.

Example: Journal System

Start with the simple ones and gradually increase complexity

1. Submit a paper, assign an assistant editor and reviewers. Accept the submission.
2. Submit a paper, refuse it at once
3. Submit a paper, ask for rework, submit again, ask for additional information, finally accept paper
4. Use the above three scenarios and make variations in what files you attach, how many reviewers you use, what information you enter in different pages. Do an overall check to see if things look OK for all different users

Cyclic Testing

Create an overview table for the application you are testing. Make sure you have a fairly large amount of data in the system. Look in different parts of the system as well as in the database.

Add several submissions for the same author and let them have different statuses. Use an overview table like the one here. I often do the testing using a printed version of the table and then log the results directly on paper. Add digitally later to keep a record if you feel that is needed.

If you look at the different screens using different search criteria: what shows up? Make some changes and search again. You may have to change dates manually in order to get production like tests.

Exercise Scenario Testing

a) Use the state table for the journal system described previously and adapt it.

b) Create some scenario tests for the license application system using the variations in this chapter. Use tables and lists. There is no point in making detailed descriptions, by the time you get to test they will be wrong anyhow.

Usability testing

Today, more than ever before, we really need to focus on the user experience, UX. In the heart of all the testing we do must always be the question – will this solve the user's problem, will they understand and like this application? The knowledge for this is generally rather poor in most of the projects I have worked in. This means that user interaction designers or usability experts are a rare sight. So what can we do about that?

When I started working in IT back in the mid-nineties the impression I got from the more experienced testers was hat usability was reserved for the experts and not included in my tasks as a test manager or tester. Today I have a different point of view. Since I am convinced that testing 2.0 always have the user in focus that means that impact mapping to understand what the user needs, coupled with usability is a core task in testing. The good news is that we can with relatively simple tools and actions do some really powerful testing.

I advocate an easy approach that is not perfect but is a lot of bang for the buck. I have used this approach on several applications and am astonished how much information we get from relatively basic testing. Another advantage is that when I write the report on the testing – the problems mentioned come from observation of real users doing real tasks in the system we are building. Getting this really good feedback means that the goal to solve the identified problem is accepted by all team members. Compare this to the endless discussion that may arise when you as a tester try to convince developers that since YOU think it is hard to use it should be changed! I have often heard the comment that everyone has their ideas around usability so what makes you the one to listen to?

The General Process

I have tried to make this as easy as possible in order to actually getting it done.

Preparations

Write a couple of pages on how the usability testing will be done and what we will get from them. Distribute this information in advance. Make sure that the message is clear: *"We are not testing how good you are but what problems you have with our design so we can make it easier for you and your colleagues to use the application."*

Ask for the participation of three to five users per area. Five gives really good feedback but it can be really hard to get this many users to participate and the budget to execute it. Having only one or two users is a bit dangerous. You may get someone that really hates all new things including computers, the new system they are forced to learn and use and whatever you show sucks in their opinion. You think I rant about this – it has happened for real more than once! Three is the magic number - a good compromise – it's doable and gives a lot of good feedback.

Create a list of specific things you want the user to perform. This should be inspired mostly from the impact map and the high level requirements and focus on real situation that are likely to happen. Creating the user guide is a tool for critical analysis of the application as well as great inspiration for tests. Make sure time for writing the guide is booked on documentation and not on testing since managers often wonder why testing is so expensive. Also add the situations where you think there may be a problem in situations that are less likely but still possible. Don´t try to use to complicated strange tests that you invented but users consider so unlikely that they don’t care about them. Start and end with tests that you are fairly sure that the user will succeed with – this is important to make them feel good about their work. Prepare the system with production like data and pick specific data for your tests. Make sure you run the same tests yourself in advance so you know that they are possible to execute and then reset the database. It is essential to be prepared well and act professionally in order to build trust with the users. Since I am stressing this you can understand that there have been occasions when preparations were insufficient and the testing went down really bad. If you mess u – there may not be another chance to try again.

If users are to receive training or user guidelines prior to real usage then make sure they have the same preconditions for the usability testing.

The Test Session

Start by repeating the goal of the usability test and explain what will happen during the coming hour. Explain that we need to stay focused and not be disturbed by phones, e-mail or curios colleagues. You will give one task at a time to solve and they think aloud while solving it. This helps you understand their though process.

Try to have only one person giving instructions and asking questions. It is a great opportunity for developers, designers and requirements people to be observers of how the users interact compared to what was designed for. Use only one observer at a time to avoid making the user feel uncomfortable. Piling up the whole project in the back of the room and trying to have them NOT discussing what they see during the session is generally a bad idea. We never use any recording tools anymore since this seems to be a disturbing factor for many and we got all the information we needed during the session.

Ask the user to solve one small problem at a time and take notes on your observations. If you have an additional observer you can compare notes afterwards to get an even better report. If a user gets stuck, start by giving them clues but not direct instructions. – Are there any other places you can look for information, how did you do that on the last test, where do you normally look for help when you get stuck? Don´t tell them exactly what do until they have tried on their own for a while. Finish the session by asking them the open-ended question what their general thought are and if they want to stress some things they found extra hard or easy to do.

A session should take a maximum of one and a half hours including a short intro and summing up in the end. It is very intense for the users as well as the observers. After the session is over, try to analyze and sum up as soon as you can since you do have information in your head that was not written down. As soon as you start the next session you will start mixing things up if you don't flush it out in written form.

Analyze and Report

When you have finished all the sessions for an area you need to sum up all your observations. Severity of problems found is your subjective impression paired with the actual problems written down. This depends on how many users had the problem and how hard it was to solve the given task. If you already see possible solutions, write them down as well.

Deciding What to Fix

Now is the moment of truth. Gather the project and present your report. Describe what type of problems there were and how severe you think they are. Suggest possible solutions. The project team will now decide what to fix and in what order. Some things may be really easy to handle and will therefore be given high priority, others may be really hard to solve and will be handled only if we find the problems really important to solve. Remember your task as a tester is not to always get your personal opinion executed but to inform others so you can decide together what to do. I have found that having this more humble approach usually gets more problems fixed.

After fixes are done you need to redo at least some of the usability tests. Probably a lot less testing than the first time if the interface hasn't changed completely.

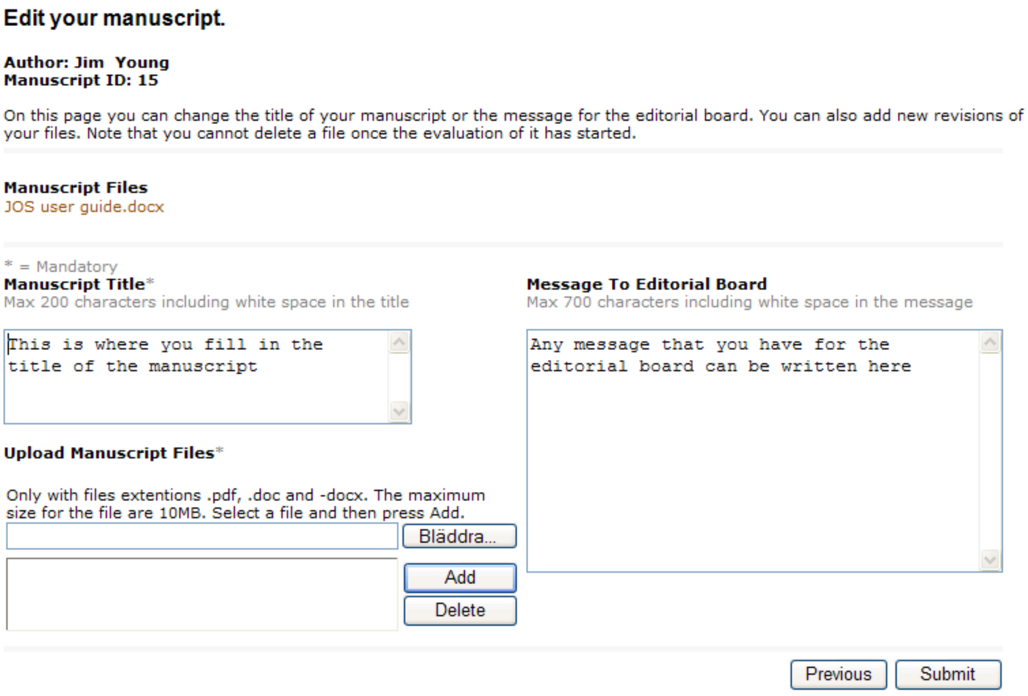
Example Journal system

For the journal system we built we had the following approach. The chief editor was part of the project and tested during the whole time so she decided we did not need any additional usability tests on the chief editor tasks. We created tests for the author, the associate editor and the reviewer.

Now this is a limited world since all of the people involved are researchers. This means one person can be author of one paper, associate editor of another and reviewer of a third. So we ended up doing several sessions for different roles with a few selected people.

We did not only find usability problems but also requirements that needed to be added or changed.

This is the interface we started with



Edit your manuscript.

Author: Jim Young
Manuscript ID: 15

On this page you can change the title of your manuscript or the message for the editorial board. You can also add new revisions of your files. Note that you cannot delete a file once the evaluation of it has started.

Manuscript Files
JOS user guide.docx

* = Mandatory
Manuscript Title*
Max 200 characters including white space in the title

This is where you fill in the title of the manuscript

Message To Editorial Board
Max 700 characters including white space in the message

Any message that you have for the editorial board can be written here

Upload Manuscript Files*

Only with files extentions .pdf, .doc and -docx. The maximum size for the file are 10MB. Select a file and then press Add.

Our findings in short

1. The application interface is rather messy and it is hard to find a good workflow
2. Users want to be able to register on their own directly on the web and select their own passwords
3. The error handing is inconsequent and hard to understand

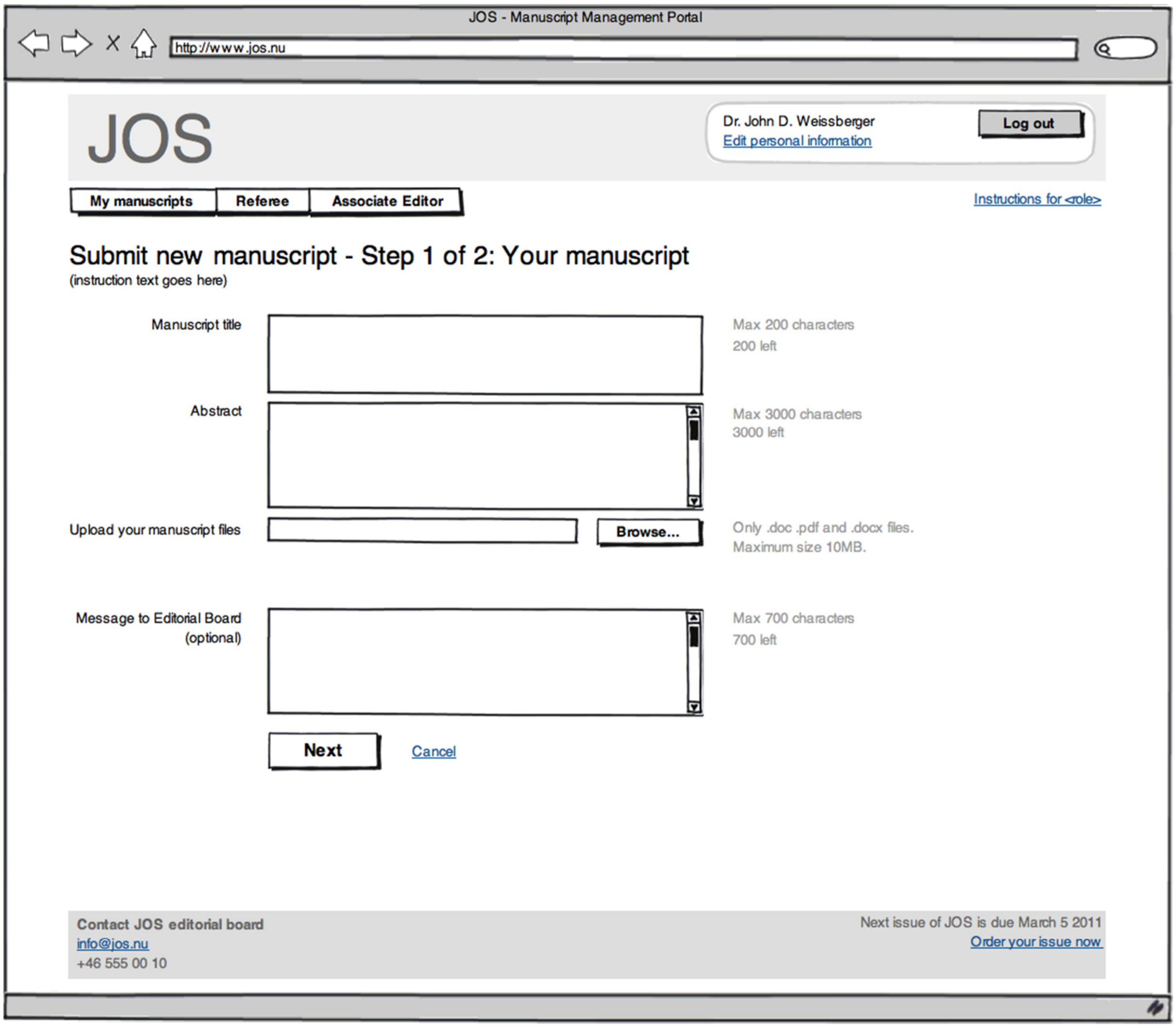
At this point we decided to bring in an interface designer to help out. We managed to get 40 hour in the budget for this.

Decided actions

Together with the interface designer we came up with the following list.

1. Increased focus on each page on the next step in order to get an easy to understand workflow
    a. Highlight buttons with size and color
    b. Hide information that does not need to be seen by everyone
2. Create a workflow that supports the users way of working
    a. Restructure we pages
    b. Better error handling
3. Cleaner design
    a. Better overview
    b. Easier forms
    c. Calmer and more modern graphics
    d. Larger fonts and narrower columns for text

And here is the new design



Submit new manuscript - Step 1 of 2: Your manuscript

(instruction text goes here)

Manuscript title — Max 200 characters, 200 left

Abstract — Max 3000 characters, 3000 left

Upload your manuscript files — Browse... — Only .doc .pdf and .docx files. Maximum size 10MB.

Message to Editorial Board (optional) — Max 700 characters, 700 left

Contact JOS editorial board
info@jos.nu
+46 555 00 10

Next issue of JOS is due March 5 2011
Order your issue now

EXERCISE

Suggest usability tests for the license application. One page per role you want to do usability tests for. Write down your suggested approach for the usability tests. Keep it short.

Discuss pros and cons with different approaches.

SUMMING THIS UP

Dear reader, I am so happy you made it this far. I hope that this has inspired you to try out more visual techniques to become better problem solvers. I had fun writing it but I did take many hours more than I originally planned. Maybe it will evolve into a book in the future!

I think that visualizing problems will help you solving them as well as reaching consensus regardless if your role is tester, business analyst or developer. As a bonus it has made my work more fun.

I would love to get your feedback on what you read on torbjorn@ryber.se

Now, just do it!