World's First 1,5 Year Professional Test Education

by Martin Nilsson from House of Test
Martin.nilsson@houseoftest.se
Revision # 5
2015-06-10

This article is about the world's first 1,5 yearlong professional test education that is fully based on a Context Driven Test-perspective. Software testing is very complex and requires a high level of skills in a broad set of areas, even more so than in most professions. Not only do the tester need to understand the technology he or she is working with, the tester also needs to understand the context can comprise of the customer, the business needs, the co-workers, the processes, the approaches to software development, the methodologies of testing and on top of that the tester must be able to communicate the information found in different ways to different stakeholders.

My name is Martin Nilsson and currently I am developing and teaching together with my colleagues at House of Test, Erik Brickarp and Maria Kedemo, a professional test education. It is in the form of a vocational university in Sweden called Yrkeshögskola and is hosted by Hermods AB [1]. At the time of writing of this article the students are not through the first year of the education. Because of that, and because we built the foundation of the core of testing during the first months, the focus of this paper is on the first half year of the education. The students started their education in the middle of 2014 and will graduate at the end of 2015.

I have been in the testing profession since 2007 when I began my testing career by accident at Ericsson in Sweden. After spending almost five years there I became a consultant at House of Test and has since then been through several different assignments from Maersk to Qlik where I also coordinated around a hundred testers. My latest assignment has been to develop and teach the test education that this paper is about.

Since we are in total three teachers for two classes the experiences between us can vary. This paper represents first and foremost my own viewpoints, results and conclusions.

TABLE OF CONTENT

Background

Yrkeshögskola

Yrkeshögskola is a form of a vocational university that is focused on providing workers with specific practical skills that different industries require. To strengthen the ties to the industry there is a board for each education with representatives from the industry that can confirm that the education is providing workers with the skills needed. In Sweden there are a couple of different varieties of software testing educations held by different companies in different cities meaning that the educations differ between them. The education that this paper concerns has two classes, one in Malmö, which I and Maria Kedemo have been teaching, and one in Örebro, which is taught by our colleague Erik Brickarp.

Previous Iterations of the education

This particular software test education has existed for 7-8 years but last year the whole setup for the education was remade based on a Context Driven Test-approach and mindset and was shortened from two years to one and a half. The previous iteration was heavily influenced by ISTQB and "Factory School"-thinking [2]. The reason for the complete overhaul of the education was that it produced testers with very, very low testing skills.

My company, House of Test, started an incubator program a couple of years back. We handpicked four students from a previous iteration of the education that we believed were the shining stars of that year's class. This was four people who were industrious and they showed a commitment and a glowing interest for testing. It turned out though that without being given a template to follow they were pretty much stuck, they lacked an approach to handle a situation without a guide to follow. One example was when they were given a problem to solve for our company: They were tasked with creating a test plan to test if the company could make a switch from MS Office to Google Docs. This turned out to be too difficult for them and at that moment it dawned on us the level of education that had been given to them. Today they do awesome work at their respective workplaces but our feeling is that they had lost two years attending the education.

Henrik Andersson, my boss, was on the board of the education and when the education was going to change he wrote a proposal to remake the education after a CDT model. The proposal went through without a comment and was fully supported by the industry representatives. When I and Erik were handed the assignment to develop and teach the new courses I started to dig into details about what the students had actually been taught previous years and what I learned stunned me. During two years of education, with the exception with roughly half a year of different internships, the students had not been testing any software in class. There was only one exception; the class that our Incubators were in got to test for two weeks after complaining about the lack of testing in the professional hands-on education that they were attending. So needlessly to say, we wanted to change things.

Other Test Educations

Though this is the first education of this kind that is outspoken taught from a CDT perspective there have been educations in Sweden built on a similar mindset [3]. There are also other educations such as the Black Box Software Testing (BBST) courses but they are intended for another type of attendees. There are of course also test certifications that promise an understanding about test after only three days of powerpointing.

A Context Driven Test Approach

The Inspiration

The inspiration to how we developed this education comes in big parts from the work of Gerald Weinberg, Michael Bolton, Cem Kaner, James and Jon Bach. But the influences also come from a wide spectrum of contributions to both the testing community as well as about problem solving in general.

The Challenge

Software can today be found in lots of places; in watches, lightbulbs, cars, regular pc-programs and a multitude of different other places. The technologies we see today get increasingly complex when more people are using more software in more places, more often and everything gets more connected. And with the "Internet of Things" we will most likely not see a slowdown in objects around us that contains software.

The challenge we faced when starting to develop this education was to cover all the different contexts that our students might find themselves working in. It might require a radically different mindset and approach if a tester works in an upstart company developing web applications compared to working in a project running over ten years in the Swedish space industry. We also had to be mindful of the tools and technologies we teach because what is hot and used in the industry today can be obsolete before the students finish their education.

This means that to prepare our students for the largest number of contexts they can find themselves working in we needed to find the smallest common dominator in software testing, the core of testing. And we needed to create a plan for teaching this to the students who themselves provided a complexity. The students are in different ages with different backgrounds, some are directly from high school, some have been studying at universities and some have been working for several years in a variety of different jobs.

The core idea

The core ideas behind the approach to educate the students can be summarized as following:

- What is test? The understanding of what a test is and what a tester do.
- Thinking like a tester; to be able to think laterally, define a problem and be able to redefine it.
- Learning to learn; the ability to learn about new contexts and new technologies.
- Throwing everything you got at a problem; to build up an arsenal of tools and skills than can be used for testing and problem solving in different contexts.
- The ability to provide value in form of information to different stakeholders in different ways.

The practical implementation

We defined three "legs" that the teaching was standing on: The theory, the practical testing and other exercises. The theory was taught traditionally in front of a white board together with home assignments. The first books the students got to read were books by Gerald Weinberg: "Perfect Software – And other illusions about testing", "Are your lights on?" and "A general introduction to systems thinking".

The general approach for the practical testing that we pushed our students to use can be summarized as following:

1) Learn about the product. Typically touring the product to understand the product and the context.
2) Analyze what quality attributes might be most important for the customer.
3) Do a risk analysis.
4) Create test charters based on the biggest risks for the most important quality attributes.
5) Execute Exploratory Test sessions based on the charters.
6) Summarize the testing conducted in, for example, a low tech dashboard or in a report based on the quality attributes.

The other exercises did not concern testing directly but were aimed at practicing problem solving and cooperation. One example was an exercise were different teams were to build the best airplanes they could (according to a set of requirements). After ten minutes I simply failed the whole class because the time was up and no one had demonstrated a working airplane for me. After that exercise the students never forgot to ask for the time they had available for a task and hopefully they will ask for the time available for testing in their professional work life so that they can provide better estimates about what testing can be managed in that time.

We also encouraged the students to write and to engaged in social media. When a test conference was going on I let the Twitter feed roll during breaks. I also made sure that every student has a Linked-in profile that is up to date so that when we had guest speakers visiting the class they were ready to add them to their networks.

Course 1 – Introduction to Test

The first month there was a lot of focus on definitions, showing a general picture of what kind of testing there is and on getting the students toes wet with hands on testing. The main points covered in this course were the following:

- What is test?
- What does Quality mean?
- The Problem of not testing everything.
- The complexity of knowing what to test.

- How do you know you are done?
- Using Oracles.

Though we did explain for the students that there exist different definitions of Test we focused on the two following:

"Questioning a product in order to evaluate it". – James Bach

"Testing is an Empirical technical investigation done to provide stakeholders, information about quality of a product or a service." - Cem Kaner

We kept pushing these definitions to the students in their exercises by asking them what questions they have asked about the product and what information they could present to a potential stakeholder.

The students to go start testing the first day and while not have gone through the entire concept of Exploratory Testing we did task the students to document their testing using the style of ET Session Notes from day one. It turned out that ET Session notes-style of documentation was something that really helped the students to focus their work and to document it and they did some really interesting tests. For samples of the charters they created for their first day homework see Appendix A – A sample of the students first independent testing.

In addition to the testing we wanted to create a habit for the students to visualize what they are learning, both when studying but also when investigating the product they are about to test and thus trigger different ideas about the product see Appendix B – Visualizations of a product.

The examination assignment consisted of one theoretical and one practical part. For a sample for the assignment see Appendix C – Examination Assignment sample from course "Introduction to Test".

Course 2 – Thinking Like a Tester

In this course we focused a lot on problem solving, problem definitions, lateral and critical thinking. Several exercises were about getting creative describing an object or an abstract image to train the students to challenge their initial assumption on what they are actually looking at, see Appendix D – Answers to exercise about what a brick can be used for. The following points were covered in this course:

- Test polarities
- Heuristics
- Multiple choices
- Modelling
- Complexity and hidden complexity
- Investigation and analysis
- Questioning and evaluation

This education and course would not be worth much if the students were not thinking critically about what we teach them so we encouraged them to question what we say and present to them. On the examination assignment we had the following question:

"Critically examine this claim: "Session Based Test Management is the best way to organize testing". Formulate three critical questions about the claim."

Lateral thinking can be tricky to teach, but one particular exercise we asked the students to perform over and over again was to use a rule of three that is inspired by Gerald Weinberg's:

"If I can't think of at least three different interpretations of what I received, I haven't thought enough about what it might mean".

We asked the students to ask three question from other points of view; "Three ways that a user can misuse this program", "Three ways that this function can fail", "Three ways the hardware can cause problems" and so on.

For a sample for the examination assignment see Appendix E - Examination sample from course "Thinking like a Tester"

Course 3 – Project Contexts

This theoretical course gave three weeks of introduction to different ways that a development project can be organized. The emphasis was on traditional waterfalls methods and on agile approaches. The following points were covered:

- Scrum
- XP
- Lean
- Outsourcing and distributed teams
- Waterfall models
- RUP

The idea was to prepare the students of various contexts so that no matter where they end up working they have an understanding of the pros and cons in their context and have them thinking about how they can be as effective as possible in a given situation. For example; in a waterfall type of project setup there is an opportunity to read in deep into documentation such as requirements, but you might be lacking a daily and natural contact with developers or architects. On the other side of the spectrum you have in a typical Agile context access to developers and other resources in your team but might lack the time to read deeper into documentation because there might be a fast pace of deliveries.

We were careful to avoid putting forward any way of working as a best practice, though we might be biased towards certain ways of working, and focused on how and why certain approaches aroused. Waterfall models are for example shunned by a lot of people in development (and people might have good reasons for it) but the model was not created to be evil, someone was trying to solve a problem and had a reasoning about how to solve that problem. It was also important for the students to understand that certain ways of organizing a

project might be because of the specific context surrounding that project. For example if you are going to launch a rocket into space you might be forced to working in a more waterfall type of project setup were a tester has to study requirements while the software and hardware is being developed. In comparison working in a web-world might benefit from an Agile approach and might be hurt by a waterfall type of setup.

Course 4 – Programming for Testers

The course setup assumed that the students had a rudimentary skills in programing (this was a requirement to be accepted to the education). The course began with an introduction to how the fundamentals of a computer works from the level of a transistor, to binary and hex, to how different programming languages work in practice. The idea was that if the tester understands the reason why the number 65537 might be interesting to use when testing then the tester can do better and more informed testing instead of seeing the number as a magic one.

We also included test automation. In the examination assignment we gave a specific task for the students to automate. While we did recommend Selenium we did give the students the option to use another tool such as Sikuli.

For the actual programming the chosen language was Java. The reasons for choosing Java were the following:

- It is fairly easy to put together a tool that can help the testing.
- Knowing Java makes it fairly easy to migrate to C# or similar languages as well as to various scripting languages but going from a scripting language to Java might be harder.
- It is a language used in a lot of development projects and it might be good to understand how the developers are coding and the challenges they face.
- It is widely used and spread and there is a lot of information and help to be found online.

To become good at programming you simply have to put the hours into actual coding. To make our students want to do that and continue once the course is over we designed an examination task so that every student had to code a test data generation tool. The tool was supposed to create different strings with different characters, generate random e-mail addresses and produce a counterstring according to the description presented by James Bach.

The idea was that the final result would be so useful to the students that they would use their own tool in the future and therefore hopefully continue developing their tool. We had hoped to be able to continue building tools during the rest of the education specific for different tasks. For a description of the tool see Appendix G – Creating a Test Tool.

Course 5 – Test methodology

Though we, the teachers, might have a stronger bias towards CDT approach to testing we needed to prepare our students for working in a context were other approaches and methodologies are used and favored. We also wanted to give the students a better understanding and a critical view of different methodologies and different testing "truths" that can be found in the industry. The following points were covered in this course:

- Context Driven Testing
- Agile Testing

- ISTQB
- TMap
- Myths within testing

The course was influenced by "Four schools of Test" by Bret Pettichord [2]. Not only does the presentation give a basic understanding on why certain approaches are favored by different people, but it also give a short history lesson in the short history of our trade. We strived for our students to understand how the development in software testing has led up to the point we are today so that they can make reasonable judgements about the development of the future. Though CDT is considered by us, the teachers, to be the cutting edge approach to- and mentality about testing today, it might have changed tomorrow if a new approach or idea emerges. Therefore it was also important for the students to question what we teach about modern testing and not become immune to new ideas when the industry changes.

It was also important for the students to have seen them the at least the major test methodologies because they might end up working in an environment that favors one or another. We do not want our students to fail at future interviews because they have never heard or read about an approach favored by a company. The education did however not pay for any certification of any kind as it was not requested by the companies represented in the board of the education.

The remaining and ongoing courses

The remaining and ongoing courses at the time of this writing are the following:

- Test Strategy
- Test Design and Test Techniques
- Bug Reporting
- Test Tools

After the last course the students are too take a two month internship at a company before graduating from the education.

The Results

First of all: the students had a great time! Most students did not know much about being a software tester, and several just signed up by chance or as a last option. But after the first half year the class was having the lowest dropout rate of all the educations that the Hermods was hosting in the Malmö region. Out of 32 starting one had dropped out because he had other interests and one dropped out due to illness. One piece of feedback from a student was that for the first time in his life he had fun coming to school. The students are having fun!

The coming year the education is expanding to also have another class in Gothenburg which will make a total of three classes starting yearly.

An important result regarding the approach for testing described above was that the students were able to provide value in form of information from day one about a product being tested. At early stages of their education, a couple of months in, the information was of course of low quality due to lack of experience; the touring was shallow and mixed up with testing, the analysis of risks and quality attributes was a hit or miss. But all my students could basically present the following:

"After touring the product (learning about the products and it´s context), doing an analysis of risks and what quality attributes that I/we believe are most important: this is what I/we have chosen to test, these are the results and this is what might threaten the by customer perceived quality".

Despite lacking deeper knowledge in test techniques or tools (covered later in the education) the students could provide information from day one, and that information could potentially be worth a lot for the right stakeholder.

The programming course had troubles with its initial setup. Amongst other things precious time were lost over Christmas and New Year's holiday, so the coming year it will be re planned to not lose that time. Also it seemed that Java was a little bit too difficult to teach in the time we had for the level the students were at. Though they had formally taken programming courses some had taken them over a decade ago and not programmed since. With students having difficulties with the programming lead to difficulties to create more tools at later courses. The hurdles were still a little to big for the students, especially for those who still had not yet passed the course.

We also face the same issues as other educations; students were not always very active in class but were doing other stuff instead of paying attention. Some students believed they could skip classes and still make it by studying for exam the night before. The student coming directly from high school did not always know how to study properly by taking notes, repeat previous classes and coming prepared to school.

But a more difficult challenge was that the lack of experience from the software testing industry that made it hard for the students to take in all our lessons and it was difficult for them to understand what might be important to test and to report. This often resulted in a shallow kind of testing.

From an educational point of view, with thirty students, it was a struggle against time to give the students individually a deeper and more in-depth feedback on their testing and their study.

Conclusions and the future

What we see today is that Software Test educations that previously was a subject covered by shorter courses or educations are today spanning up to two years. We also see that when people previously stumbled into software testing by chance, there are today students who have been preparing for the profession from start, they are testers from start. We also believe that there is a trend in these kinds of educations and we are slowly seeing more of them (at least in Sweden).

Exploratory Testing and Session Based Test Management suited the students very well. In the future they might face contexts were they need to use another type of approach and documentation but I do not see that transition to be of any problem, they should be able to switch to another way of documenting their test if needed. ET was also very useful from day one in the education and despite the students lacking experience from testing, the approach and the documentation gave the students a framework to do very interesting testing in their first home assignment. Therefore I consider claims that ET is for experienced testers to be false, testers at any level can gain from it.

Teaching testing is a huge area and can't be fully covered during one and a half year, even less so in a three day PowerPoint course. Even though the time frame sounded long it was a struggle to fit everything into the classes and therefore it is important to show the students how to continue their learning once they are out working in the software industry.

There are a lot of niches to be found in software testing so there is a place for most people. Don´t have deeper technical skills? Throw your language skills at the product, specialize at end user testing were the users are expected to lack technical skills or find another specialization were you have an advantage.

Programming in Java was in this education a little bit too difficult for the students due to the limited time we had to teach. Perhaps a better structure and more time might solve that problem for the next class but we are also considering switching to Python to gain speed in the creation of helpful tools.

The lack of experience from the software development industry did create hurdles for the students when learning. I am expecting that the lessons from this education will start to be apparent for the students a year or two into their testing careers meaning it might not be until then that we will see the real results of this education. So keep an eye out for my students around and make a note that around 2017 they should be up to full speed by then!

More information about the education

For some more information about what is going on then check out Erik's blog [5], House of Tests blog [6] and you can have a sneak peak at what is happening in the classroom on Twitter via the hashtag #YHTest.

Acknowledgements

First of all a big thanks to the students of YHTest who has given it all in this education, without you my work as a teacher would have been much more difficult!

A shout out to the people in ISST for providing tips and suggestions when I needed help! A special nod to James Christie, Martin Jansson and Michael Bolton!

Big thanks for everyone at House of Test who always jumped at the opportunity to give a helping hand!

And an applause for my fellow teachers Maria and Erik who I know are putting in all they got to give the students the best possible education a tester can get.

To all of you who tool your time and came to visit my students and share your experiences and wisdoms! We greatly appreciate it [4]!

To Henrik Andersson who did paid the path for us to finally teach some students real testing!

A high five to the Incubators! We are proud of your accomplishments! And thanks for letting us test the first embryos to creating the CDT approach for this education!

And lastly a bunch of roses to my wife who had to put up with me working late evenings and weekends!

References

(1) The YHTest education home page: http://www.ecutbildning.se/Utbildningar/Yrkesutbildningar/Mjukvarutestare/

(2) "Four schools of Test" by Bret Pettichord - http://www.testingeducation.org/conference/wtst_pettichord_FSofST2.pdfs

(3) Rikard Edgrens presentation "Trying to Teach Testing Skills and Judgment": http://www.thetesteye.com/papers/TryingToTeachTestingSkillsAndJudgment.pdf

(4) Credit from Maria to all who have met and talked to our students: https://mkedemo.wordpress.com/2015/04/27/guest-lecturing-at-the-vocational-education-program-software-tester-in-malmo/

(5) Erick Brickarps Blogg: http://erik.brickarp.se/

(6) House of Tests blog: http://www.houseoftest.se/2015/04/guest-lecturing-vocational-education-program-software-tester-malmo-2/

(7)

Appendix A – A sample of the students first independent testing

The first homework the students got was to pick a software themselves and perform a 1.5hour long test sessions. At this stage they had only been introduced to documentation in form of ET session notes and the definitions of test, they were in other words completely new to test. The idea was give back their test documentation at the end of the education to show them just how far they had come. But a lot of student surprised by creating very interesting charters which shows that just giving the frame for documenting the test in ET Session note-style is very powerful. Below are some interesting samples of the charters the students created (translated from Swedish).

Sample Charter 1: To compare how a beginner at excel works with the program compared to someone who is used to the program.

The student created a problem to solve using Excel and documented how he solved the problem compared to his relative who had not used the program. The idea came from frustration in miscommunication between the student and his relative when trying to explain how to use Excel.

Sample Charter 2: Write a paragraph of text in three different languages and run through Google Translate and investigate if the original meaning of the text was translated or got lost.

A student gifted with several languages created this charter and drew up different state diagrams were he translated texts back and forth to see if the original message was retained in different languages.

Sample Charter 3: Test the Swedish shortcuts in Mozilla Firefox

The student wanted to see if all the shortcuts that existed for an English keyboard layout were translated to the Swedish equivalents.

Sample Charter 4: Learning session about Mozilla Firefox and checking if there are any obvious issues that can bother beginner users.

The student had not used Mozilla Firefox before and thought it might be a good opportunity to learn about the browser and to see if there are any issues that can bother a beginner uses such as himself.

Appendix B – Visualizations of a product

One early exercise was about letting the students visualize a product under test in different ways. They were encouraged to be as outrageous and creative as possible. The idea is to generate new ways of looking at the product, to be able to redefine the context of the testing. The software to test was a simple homepage based on the template found here: http://www.tooplate.com/view/2018-notebook. Here are three different visualizations of the product, which was a web software, which spawned completely different ideas about what to test:

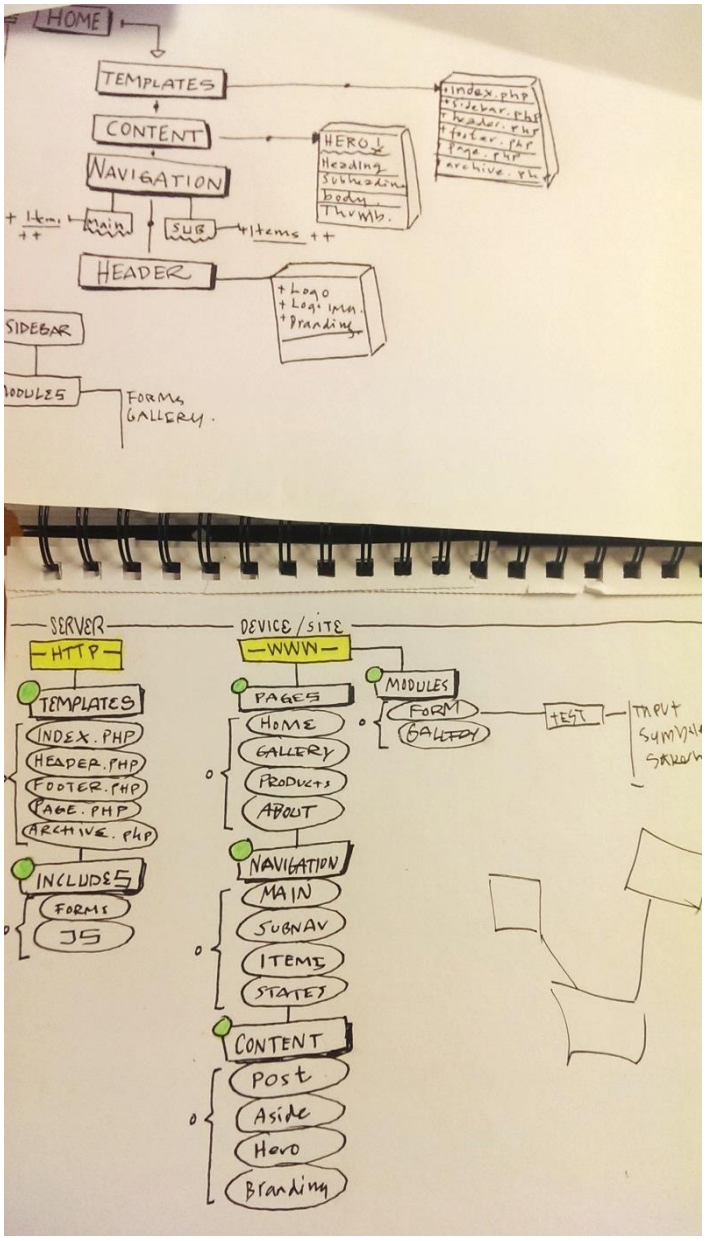


Figure 1 - A student who has been working with web design created this visualization. His visualization focused a lot on the technical hierarchy of the software.

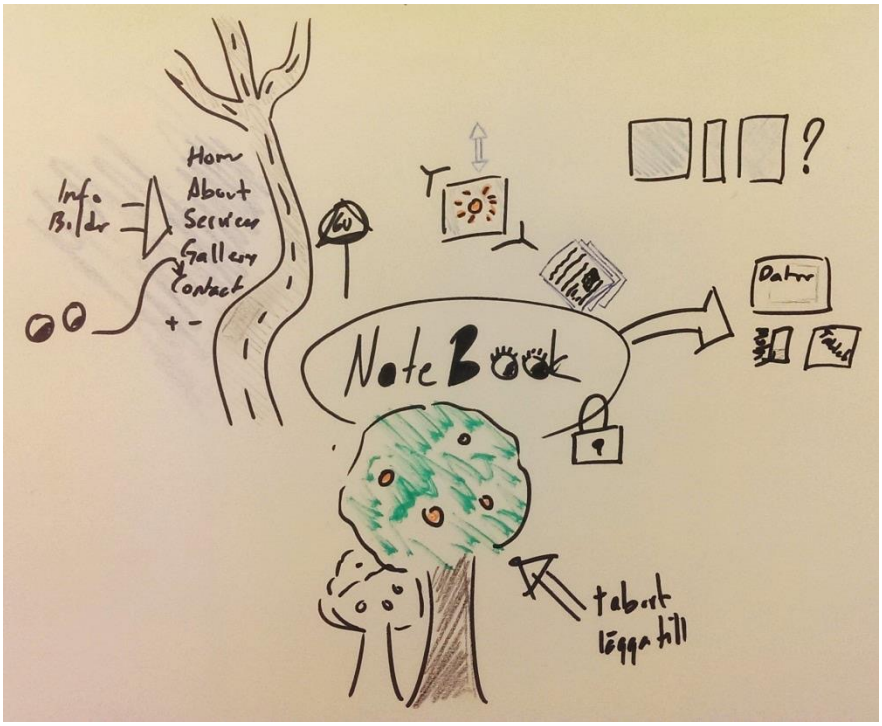


Figure 2 - The idea behind the visualization was to see the product as a flow that can be seen of different monitor setups as well as on different devices. Illustration created by Christopher Grönvall.

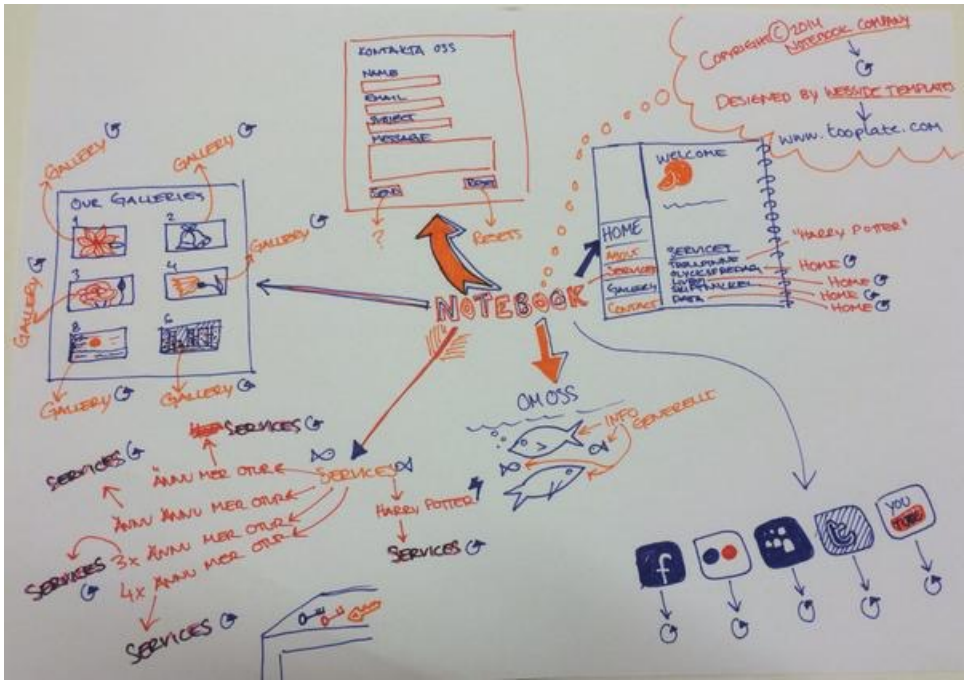


Figure 3 - A visualization focused on breaking down the most important features of the product

Appendix C – Examination Assignment sample from course "Introduction to Test"

The first examination consisted of an assignment for the students to do at home. The first part consisted of theoretical question and the second part was practical were they were to test an open source program we had selected. We generally wanted to avoid examinations which were focused on one "right" answer so we wanted to ask the as open ended questions as possible. But when we did want to ask specific questions with a specific answer we usually asked the students to provide an example which put the answer in context. For example we had the following question on the theoretical part*:*

"How do you know when you are done testing? Please provide an example!"

For the practical testing part we asked the students to provide us with the following information about their testing:

- *The planning (for example list of your charters and if you have created any visualizations).*
- *A short description on what you have focused your testing on.*
- *A summary of the information you have uncovered during the testing (for example bugs, questions and answers to developers, potential improvements).*
- *Provide the two, in your opinion, most important quality characteristics for the product and motivate why you have picked them.*
- *A list of oracles you have used.*

Appendix D – Answers to exercise about what a brick can be used for

The students got a task to find different usages for a brick. The idea behind the exercise is to challenge oneself to view what is in front of you from a different perspective. The analogy to test is that what you have in front of you might not be perceived by you as it is by a potential stakeholder or customer. The answers below are a translated sample of the ideas the students came up with and generally they first came up with the obvious answers before getting completely crazy:

Building:

- Build a house
- Crash a window
- Shield yourself from the outside world
- Build a factory that supports the communist regime that oppresses the people

Tools:

- Use as a hammer
- Use for digging

Mathematics:

- Count the holes in the brick
- Count the brick
- Measure the brick/use it as measurement
- Use as weight
- Role-play and pretend the brick is Einstein

Religious:

- Worship the brick as a god
- Write the words of god in the brick
- Create a cult around the brick (Brickthulu, a Cthulu -ripoff)

Exercise:

- Curl the brick
- Bench it
- Throw it
- Crush the brick and use the powder in a protein shake (lots of fibers)

Culture and music:

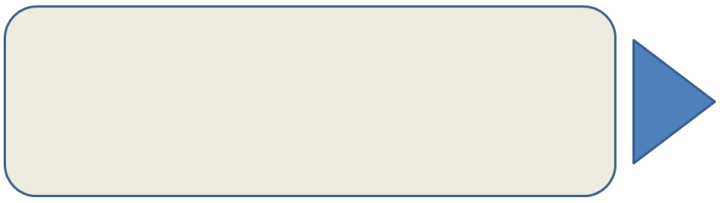
- Create a band name called "Brick" and use the brick on the album cover
- Use as the world's most quiet groupie
- Use as plectrum
- Silence critique

Appendix E - Examination sample from course "Thinking like a Tester"

This examination assignment consisted of three parts: A theoretical part, an abstract part and a practical part. In the theoretical part we again tried to ask as open ended questions as we could. For example we asked the students the following:

"What is a "heuristic"? Explain and give an example of a mnemonic within testing. The example should contain explanations for the different letters. Optional is to provide an example were the mnemonic could be used."

In the abstract part we wanted to make the students to think outside the box. We presented them with the following image:



One of the questions we asked them regarding the image was:

"Think of three different types of software the figure below could be representing. For example "This is a software gamepad for mobile phones used in games. (the triangle to move forward and the rectangle to jump)""

In the practical part we used the same setup as in the first course but this time we required them to choose two mnemonics to use in their testing, and we required them to provide us with the most interesting tests they did (in their own opinion) and the most interesting tests they did not perform (and to go as out of bounds crazy as they could imagine).

Appendix F – Student developed Heuristic

During our work with Heuristics in the course "Thinking Like a Tester" one student started writing on one himself. Here is Björn Paulson's first attempt:

Customer ROUTINE

The Routine to make sure you keep track on how the software fits the customer.

Requirements - What practical requirements do we have on our software to work on customer hardware or software? Are there any other requirements in the customer's context that needs to be considered?

Outlook - How does the future looks for our customer? Can we expect that the customer will require certain updates or changes? Or new features?

Usability - Is the product easy to use for the customer? Does it need to be simplified for the client?

Troubleshooting - If problems occur for the customer, how do we handle them? Do we have a bug reporting system in place? Do we have the customer support that can help?

Improvements - Can the product be improved to fit the customer's needs better? If the customers requires changes to the product, how easy is it to implement them?

Needs - What are the actual customer needs?

Expectations - What is the customer expecting and does that match the actual needs? What they expect and what they need to solve their problem might not always be the same thing.

Appendix G – Creating a Test Tool

In the programing course we wanted to create an incentative for the students to continue programming after the course was over. What we came up with was an examination assignment where the students were to create a data generation tool for use in testing. The tool was supposed to contain the following:

- *counter string* <length>
  Generates a counter string with a set length
  Support length from 2-1000
  Counter strings are explained here: http://www.satisfice.com/blog/archives/22
  Example: "counter string 5" generates "2*4*6"

- *email* <length>
  Generate a random email with a set length
  Support length from 6-255
  Learn more about valid email characters: http://en.wikipedia.org/wiki/Email_address
  Format: <rnd 1+ characters>@<rnd 1+ characters >.<rnd 2-3 characters >
  Example: "email 10" generates e.g. agzh@sy.st

- *string* <characters types> <length>
  Generates a random string (text) with a set length and type of characters
  Support length from 1-1000
  Character types is a list of allowed characters
  u = upper-case (A-Z)
  l = lower-case (a-z)
  i = international (e.g. å Ö ä ü é)
  s = special characters (e.g. ! $? _ | >)
  p = space
  d = digits (0-9)
  Example: "string udp 8" generates e.g. "A9 JZKF88 "

- *phone number* <country code> <country>
  Generates a valid phone number for a specific country
  Country code is wither y or n (yes or no). If yes, add e.g. +46 for a Swedish number.
  Country, choose 5 countries you support, use Google to learn about phone number rules.
  The phone numbers must be randomly generated but it's okey to e.g. only generate numbers from a certain area (e.g. 070<7 random digits> in Sweden)
  Example: "phone number y norway" generates e.g. "+4721492250"

- At least two more commands of your choice
  Example: "xss" to return an XSS string from a predefined list or a help command