Unusual Testing
Lessons Learned from being a casualty simulation victim

Nathalie Rooseboom de Vries van Delft
Nathalie.van.delft@capgemini.com

Abstract

Hobbies can be an inspiration for many analogies in software and system testing, but it can also be the other way around. I've been a so called casualty simulation victim for a couple of years now, playing a patient in hospitals, a victim who needs help from a first aider (both in First Aid lessons and ambulance training) and at disaster re-enactments. I used my knowledge from the Software Testing process for the benefit of being better and more structured in my casualty simulation situations. In return I got a whole bunch of tips and lessons learned that I could use within my job as software tester. Many lessons are particularly useful for software testing, but there are also lessons that are beneficial for all other disciplines in software and system development.

Biography

Nathalie Rooseboom de Vries van Delft is Community of Practice leader Testing, CTO office advisor and Managing Consultant at Capgemini Netherlands, responsible for thought leadership and testing competence development. She fulfills the roles of test manager and advisor with various clients. She speaks on national and international test events on regular basis, writes in specialist publications and participates in the Dutch Standardization Body (NEN) workgroup for Software and System development. She is very passionate about (software) testing in general, but the subjects Data Warehouse Testing, E2E-testing, Standardization, Ethics/Philosophy and Test Architecture (Framework) are most favorite.

Nathalie is also known online as 'FunTESTic'. www.funtestic.nl.

A relation between software testing and casualty simulation might be not an obvious one. But when you think a little bit further, you can see the similarities. Both are testing and although the object under test is different, the process is very similar in both disciplines. In this paper I have used a basic software testing process which is also used in casualty simulation and have extracted the lessons learned from the latter so that you might use them or be inspired to change things in your work in software testing.

1. The background of Casualty Simulation

The craft of impersonating victims has existed since the second World War. In England a gentleman called Eric Claxton started the organization 'Casualties Union' in 1942. A comprehensive history of the founding of the organization can be found on the website of the British casualties union but in short it came down to the fact the nurses and other first responders where so shocked by the (war) traumas presented to them that they weren't able to deliver help adequately. Claxton figured that if the helpers could practice with fake victims with fake traumas they could get used to it and would be able to offer the help when confronted with the gruesome reality.

Because the British initiative proved to be successful also a Dutch version of the union was started, called LOTUS (Dutch abbreviation for Education for Impersonating Casualty Simulation Victims). Every LOTUS, as every casualty simulation victim is called, has a certificate 'First Aid' and follows a two year course, which is finished by a practical exam. In contrast to what many think in software testing, domain knowledge is crucial for executing a scenario successfully and a casualty simulation victim is taught this domain knowledge, sometimes with specialties, extensively. A casualty simulation victim is obligated to go to a minimal number of lessons each year to keep the LOTUS certificate (and do a practical competence test). Practice is just as important as knowledge.

2. A familiar testing process applied in an unfamiliar environment…

Within my casualty simulation work I use a familiar testing process. I specifically say A testing process here and not THE testing process, because there are of course more processes, but this process works best for me in the casualty simulation situations. I use the process as is shown in figure 1.

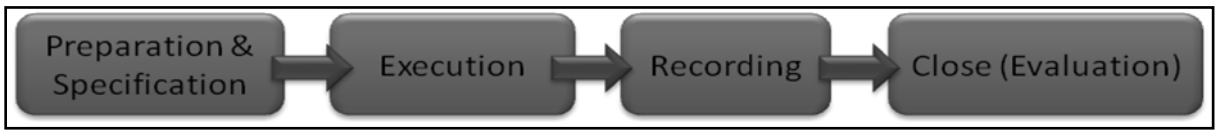

Figure 1, a familiar testing process…

2.1 Preparation & Specification

Just like in software testing the preparation and specification phase is the phase where you set your goals. You can perhaps imagine that a First Aid lesson has a whole other goal, namely the education of a person, than a drill in a hospital or a huge disaster re-enactment, where the medical staff is supposed to give the right medical treatment.

The specification gives the story in rough lines. As casualty simulation victim you get a request for a drill, exam or lesson. In this request is described what the event is that happened, in case of a drill, or what the theme is in case of a lesson. In a complete request there is also a description of the trauma. After that you decide how you're going to moulage this (moulage is the term to put on the make-up). You can do extra research to how the trauma is supposed to look and what the behavior is of a patient that suffers from that particular trauma. I use medical databases like PubMed for this purpose. Finally you decide what your story is. A good first responder will always ask what happened, so you'll have to have a story that fits (except when you play a confused victim of course).

In my work as software tester this helped me to realize detailed scenarios for end-to-end testing. In my previous project I tested purchase and validation systems. By making a very detailed story about what type of customer I could be, as close to real life as possible, I can also test the system as a real life person would use the system.

***TIP**: In your organization it might be beneficial to visit the marketing department. The department ought to have customer profiles that you can use to create 'mindsets' that you can then use to make your stories to test.*

Preparing moulage for disaster re-enactment Livex 2009.

For a casualty simulation victim it is really important that the environment or scene is correct. The environment has to tell the story to the first responder so that he, or she, can determine a diagnosis with help of this environment. An unconscious person next to an exposed power cable can give a whole other diagnosis as the same person next to a bottle of chemicals, or even without attributes the diagnosis is different again. Also in software testing it is important to have your environment in order and to have it very clear what this environment is when testing is done or used in production. A specific defect can have a whole other meaning in one specific environment than in another environment.

2.2 Execution

After the preparation and specification phase the execution phase starts. For a Casualty Simulation Victim, this means playing your role and continue playing that role. A first aider (or even doctor) knows that it's a drill, but if the casualty simulation victim really plays his, or her, role well and with conviction the first responder will react more naturally. I even have an anecdote about where the first responder forgot that it was a drill…

Every week I play the role of a woman in labour distress in an Amsterdam Hospital. In this hospital the delivery room trains with weekly scenarios where an emergency situation emerges during a normal labour. The obstetrical team has to solve the problem with the means they have available. The situation is made as real as possible, we use a real delivery room, use real medication (not really injecting it ☺) and have to wait real-time waits for results. I have all the IV's and monitoring devices attached (with tape) to my body. Although the team KNOWS it's a drill they DO react as in real life. At one time there was this assistant gynecologist who tried to pull off my specialized pants (which simulate certain clinical picture in

obstetrics) to be able to do an internal exam. Luckily my costume was well secured to my body and the training leader was well in time to stop him.

My experience is that the more real life you make the scenario, the more detailed and accurate the information obtained from the scenarios will be.

***TIP**: Sometimes it helps to re-create the environment that your end-users will have when using the system, including time-margins and noises. Environmental factors can have an impact on usage of a system that isn't always taken into consideration testing in your comfy office…*

Photo: Jorit Schlenter, Livex 2009

At the execution phase the factor FUN is an important one. The more fun you have during the execution, the better you will do your job. Mostly you play misery, fear and pain, but if you feel really miserable doing this, your role – strangely enough- will not be convincing. If you have fun playing the role the results are accordingly so. You will be much more convincing!

Still, playing a casualty simulation victim isn't without any danger. The first rule we learn in our casualty simulation lessons and in the LOTUS learning book is: "Mind danger and take care of your own safety". I can give an example of a casualty simulation victim where the first aider started the reanimation procedure and broke a rib. That should never have happened. The same goes for the 'Heimlich' maneuver when playing a choking scenario, I always place my hands between my body and the hands of the first aider and tell them: "Ok, stop now and show me gently where you would place your hands and how you would make the maneuver when using it forcefully".

If a real emergency occurs we know the stop word "NO PLAY". This is an internationally acknowledged sign that the drill has to be stopped immediately. In a large drill re-enactment there was a case of real hypothermia victims at a certain point. Because we had casualty simulation victims playing hypothermia too, it was impossible for the first responders to spot the real victims. For a certain time the drill was stopped by calling the NO PLAY situation. The real hypothermia victims could now be localized very fast and helped. It works the other way around too. When you play an unconscious victim and you really fall unconscious, the first aider can ask you 'No Play?'. You are obligated to react on this question; when you don't react the first responder will know you are a real victim and will act on this, by calling 911 for real for example.

I once used the ‘no play’ rule in my work as software tester, which a introduced to the participants of the project in an earlier stage with the term ‘no test’. During an end-to-end test I ran into an issue that would also would be highly likely to occur in the (current) production systems and had high priority. I used the ‘no play’ situation and by doing so the organization knew it was a serious situation and acted upon this accordingly. This resulted in a joined effort where the issue was solved within hours (mind: this was including pinpointing, risk analysis, data adjustment, re-testing (including scoped regression test) and roll-out on production systems) . The organization knew that with this ‘no test situation’ it was a real serious situation and also gave commitment for acute assistance from people in their departments. It’s of course very important that these kinds of agreements are clearly communicated throughout the whole organization AND that it is not to be used lightly. A ‘no test’ situation decision should never be taken ‘alone’ but only after analysis so that you can do this with a certain amount of certainty AND only then in collaboration with management.

Exploded firework in hand, moulage from lesson ‘firework trauma’

2.3 Recording

Also in casualty simulation work there is a recording phase. This is somewhat different than in software testing though, as you cannot bring any notebooks or take notes when playing a role. So you just have to remember everything. Luckily there’s a nuance which narrows ‘everything’ a bit down; you have to remember everything with the goal in mind. I do this by using keywords. Because I have prepared the scenario very well, I only have to add the bits and pieces to the scenario I have memorized. With huge drills there are photographers, but they only take atmospheric images and one cannot rely on those pictures to reconstruct the problem or learning from them.

At specific exercises or drills there is a team of observers. For example in my hospital drills there is an observer for each discipline that participates in the drill, examining the medical skills of each discipline, including the timeline. At First Aid exams the observation is done by the examiner. In rare occasions video is used. I find this a very powerful tool, because you can rerun the tape over and over again and details can be highlighted even more.

External observation is very powerful. This is also the case in software testing. This is one of the reasons that where techniques like “pair testing” are used or where one tests in teams are so successful and give relatively good results. I’m also a huge fan of usability labs, where a user can be filmed during the usage of the software. Sometimes a very small remark can give a lot of information that would have gotten lost in an unobserved test. I also find observation during regular processes very useful for the development of (specifically end-to-end) test scenarios . By observing the user in his regular working process performing

his tasks you can prevent design of tests where your user later remarks “Oh, but we don’t ever do this in real life”.

I also found that observing a user adds a bonus. The user feels that interest is taken in his or her job. This results in an increased involvement from the business in the (testing)project, contributes to the team bonding and helps with the acceptation of the product just because the attitude is more positive as a whole.

***TIP**: You can use your web-cam as a recording device during your execution to film your responses and exclamations during tests. It will sometimes give you unique insights and a whole different perspective on your execution.*

2.4 Close [Evaluation]

In software testing, the close is probably the most neglected part of the process. The focus mostly lies on fixing (as fast as possible) defects, re-test and delivery of the product. In casualty simulation the evaluation takes up the same amount of time (generally) as the whole execution and there is a lot of information and lessons learned in the process.

After the drill is finished the evaluation is done. This evaluation takes up at least thirty minutes. The evaluation is divided in two parts. First the observation team does an internal evaluation, without the participants. Of course the participants mostly discuss vigorously in the hallway about the drill. The second part is done with observers and participants. They always start with the good points and what went right. After that the learning points are discussed. The evaluation is always done with respect and understanding to each other’s knowledge, abilities and perception.

Drill in hospital, re-enactment of labour distress; pre-eclamptia (seizure due to high blood pressure)
Photo: Joost van de Broek, Volkskrant

What I noticed particularly is that during these evaluations one never points a finger to a specific party or participant, never blames a person. Maybe this is because in these drills they are focused on the positive feedback. One always gives feedback and suggestions in a constructive manner. One simply doesn’t seem to be busy with who is responsible for the mistake or error, but one makes the observation that something went wrong and after that directly focuses on how to cope with that and how to solve it.

I notice myself that the 'finger pointing' is a difficult point within my job as a software tester. One seems to think that by figuring out who is to blame, who is the guilty one, that one can find the solution there. This results in a vicious circle. Nobody is willing to work together anymore when one gets the blame and one will take a defensive attitude where he/she will try to shift the blame to another person. Meetings get a very tense character. It is very important to focus on the positive things when starting these kinds of meetings. After that, one should very factually sum up the problem(s) without any judgment and one should emphasize that a solution is to be found. Every attempt of 'finger pointing' should be resolutely stopped.

***TIP**: In meetings where everybody seems to speak at once and there's no structure or willingness to hear each other out, you can use the 'speaking attribute' (also sometimes referred to as 'talking stick'). It's a small object that gives the holder the 'speaking right'; everybody who doesn't hold it MUST be quiet. It seems a childish solution, but in my experience it apparently helps to have a physical object making it visible who speaks.*

Another lesson from these evaluations is 'letting it land'. In most cases somebody says something and, without considering it thoughtfully, very quickly the next point is addressed. How many times did it happen that you say something to somebody and later on that person says he doesn't remember you saying that? It's important that you verify that what you say has 'landed' at the recipient. You can do this by asking the recipient to summarize what you said or let them give an active reaction. You can insert a pause between points and give the participants in the meeting some time to think about what is said and to react. In my opinion this is done way too little and valuable time is lost: firstly because people forget what you have said and secondly because you have to keep repeating your message.

3. Four Gems: Added value, jargon, checklisting and SpeakUp!

Doing casualty simulation drills gives a huge amount of information, tools, thoughts and small, quick solutions you can use in software testing. I will share four of these jewels in the next part of this paper.

3.1 Added Value

Added value is something we Dutch know from the fishing business. One fishes for sole, but one catches all other kinds of fishes you can eat too. During one of my casualty simulation drills in the hospital I got severe blood loss on the 'first line' of help. The bleeding was of such a serious nature that I had to be transferred to the 'second line'. During this drill (in the evaluation) a participant noticed that finding the arteries at the second line locations would be very difficult because pressure would have been almost gone. In some cases the anesthetist had to be called in to set an IV. Nowadays the rooms in the first line have an IV set in their stock and the nurses there have learned how to apply an IV. Valuable time is won because one can administer fluids even before transfer to the second line and the survival ratio of patients has gone up.

In software testing you can also have added value, but most of the time this isn't noticed, is considered non-relevant or not in scope. By actively noticing these gems and returning the added value information to the organization, you can be an added value yourself to this organization you test for. It's still up to your stakeholder to do something with it, but not mentioned at all, is indeed a missed opportunity for certain.

***TIP**: As testers we use the system more extensively than a normal user would and we can notice Repetitive strain injury-invoking (RSI) movements (for example clicking on a sequence of buttons) much faster. You can use this knowledge in your advice to your stakeholders.*

'model picture' during one of my moulage lessons, Photo: Erik Jonker, 2009.

3.2 Jargon

From one of my larger drills that I participated in (disaster re-enactment of the Dutch Flooding Disaster of 1953, called FloodEx) to practice the dispatch of incoming foreign help units, I got a DVD with the evaluation. It's worth its weight in gold ten times over. A number of quotes from this DVD are:

"Another major issue is 'language', 'terminology' and 'jargon'. Not all participants speak English and not all those who claim they do – for example the Dutch- speak it in such a way that the English would necessarily recognize it"

"…when conversational language was not an issue, the use of jargon and acronyms were. There was a tendency to assume that the foreign units understood a jargon that the Dutch believed to be universal but that was in fact very much their own" and

"One cannot expect the Dutch civilians being saved, to understand 'Latvian' or 'Polish'…".

All three these quotes are connected to communication. Also in software testing we know this issue. As software testers we use a jargon that is perceived by us as normal language that is understood by everybody who we talk with. We simply don't think about it that a non-tester doesn't understand our language at all. The same also counts the other way around. An accountant who has to explain what he means, uses his jargon unintentionally, but it's very difficult for us testers to understand. It's not a requirement to understand each other's language to be able to do a good testing job, but it IS very useful to have a translator present who is really into both jargons and can take some distance from the matter involved to avoid misunderstandings and wrong assumptions. This was also one of the tips from FloodEx; by using translators, valuable information was not lost.

3.3 Checklisting

In the business of first aid checklists are used extensively . Particularly in stressful situations a checklist provides a grip so that (fatal) mistakes can be prevented.

In a certain hospital the side on which an operation was to be done was switched. So instead of operating on the right side, one operated on the left side and vice versa. In this hospital the checklist 'TOP' (Time out Protocol) was introduced. It's a list with questions that is used during transfers of the patient from one department to another, specifically before a patient is going under anesthesia. The list is run through in the presence of the conscious patient. Running though this checklist only takes a minute. That minute investment saved a lot of misery in the future. When the hospital implemented this TOP checklist only one switch of sides on which to operate, occurred in two years. Investigation showed that in that case the checklist wasn't used.

Also in the hospital where I drill, they use checklists. They have checklists for every acute situation. During these situations these checklists are run through, even when time is critical. Forgetting something can be a lot more fatal then an extra minute that it costs to run through the checklist.

In software testing I once in a while see a checklist passing by, but in my opinion the added value of a checklist is highly underestimated and is used way too little.

It seems that using a checklist isn't 'high tech' enough and is perceived as inferior material. Specifically during transfers from one team to another team a simple checklist with items to discuss can have a priceless value, one doesn't have to remember every item, which probably cannot be remembered anyway. The checklist should be a standard (help) tool in the whole software development lifecycle.

The time-out protocol from the hospital (taken from the book of "M. Heres, Simulation Training..")…

Name of patient
Date of birth

Medical indication
Are there allergies?
Which medication
Is there maternal disease? Time of pregnancy
Duration of pregnancy

A translation to a time-out protocol for software testing [example]…

Name of system
Version number of system
Date of release
Known issues
Specific workarounds
Are there known difficulties in the design or build?
Time till next release

3.4 Speak Up!

The fourth and also last lesson in this article is ‘Speak Up!’. For us Dutch it’s very common to speak our minds if we think something is fishy. We are proud of this ‘feature’ that seems to be typically Dutch. But even in our country it isn’t customary for everybody to say everything to everybody, specifically in hierarchal organizations or structures, and this is even more the case in countries where hierarchy is more embedded in the cultural background.

A notorious example in history is the plane crash in Tenerife in 1977. A Pan-Am aircraft landed on a KLM aircraft on the runway. There were 583 casualties to mourn. This disaster might have been prevented. Investigation showed that the pilot, a highly decorated pilot veteran was irritated by the long waits at the airport and was anxious to leave. Beside him was his co-pilot, who respected his boss very much, saw an error to be made but didn’t say anything because he was confident the pilot knew what he was doing. If the co-pilot had used ‘Speak Up!’ on that particular moment the pilot could have noticed the error and it probably wouldn’t have occurred at all.

SpeakUp! is a handy tool where one speaks out loud what one sees, asks a question or makes a remark. The recipient can react to this by, for example, saying that it is a conscious choice or can adjust his actions when an error is indeed in place.

“Fore?? Where??” a small golf ball will cause a massive wound on the head and leave one disorientated. Lesson ‘first aid at sport injuries’

During one of my first drills a gynecologist wanted to administer a medicine where he erred in dosage. In the scenario I played that I became worse instead of better, only because of quick responses and giving me the antidote I was saved. In the evaluation a student who participated in the drill noted that she had seen the error made but was afraid to say something to the highest ranking doctor in the room. The gynecologist reacted that he had made the error because he had had a really bad morning that day and that his thoughts weren’t at the drill a full hundred percent. He would have really appreciated that the student had said he made an error before giving the medication. A few weeks later a similar scenario was executed. There was a different doctor, but the student was there too. This time a medication protocol was used that wasn’t applicable to the trauma played. The student used ‘SpeakUp!’, spoke out about the thing she saw and asked the question about the protocol. The doctor thanked the student and

adjusted his action. He had reacted on instinct, using a method he always used, but the newest insights and developments weren't customary for him yet. The SpeakUp! method had helped that a better, and for the patient less invasive medicine was used.

SpeakUp! is also very applicable in any team in software development. I would like to add a bit more in this case by making this principle mandatory or giving a bonus when using the SpeakUp! technique. Simply stating "You can say anything in this team when something is up" won't work. The more present a hierarchy in an organization, the less people will use 'SpeakUp!'. By making it an obligation and even adding a reward it takes on a whole other character and one will see it as a "must use". As long as the 'SpeakUp' technique isn't common practice, this is the way to stimulate it. Also in evaluations the SpeakUp! usage should be actively discussed and should be a reoccurring item on the agenda.

4. For closure

Inspiration and passion from a hobby can give a great contribution in your work, but also the other way around. I experienced that having passion for a hobby radiates to your work and vice versa. And by applying the lessons learned from the one, the other has benefitted from that. Whether I use a software testing process in casualty simulation or using the evaluation lessons from casualty simulation in software testing.

In this paper I have shown you that a familiar testing process can be applied in an unfamiliar testing environment, like in casualty simulation. In doing so, I got to extract lessons learned from each phase in this process that I could in turn use in my work as software testing.

A quote from the FloodEx drill stated: "We learn much more from the few things that go wrong than from the many that pass without a glitch".

I feel that as a software tester it is important not to only look at these many things that go wrong and learn from them, but also look at the positive things we can offer an organization. By applying (one of) the four gems like 'SpeakUp!', 'added value', 'jargon' and 'checklisting' we can offer a positive contribution to our work and the organization we work for without too much investment. In my paper I have described both the casualty simulation situations as the experiences with the gems in my work as software tester, so that you have an idea how to apply the gems in your own working place.

And the most important thing is that you have fun in whatever you do!

References

Braxton, E. (1942) Casualties Union, http://www.casualtiesunion.org.uk/

Heres, M & Vermeulen H. (2010); Simulatie Teamtraining Acute Gezondheidszorg en Verloskunde, leer- en werkboek, Garant-Uitgevers, ISBN 978-90-441-2527-6 [Dutch]
(translation:) Heres, M & Vermeulen H. (2010); simulation teamtraining, acute healthcare and obstetrics, learn- and workbook…

Het oranje kruis, LOTUS leerboek, 2005, Drukkerij Groen [Dutch]
(translation:) The orange cross, LOTUS learning book, 2005, Publisher Groen.

More information

Dutch LOTUS organization: : www.organisatielotus.nl (Dutch)

UK Casualties Union : www.casualtiesunion.org.uk (English)

German casualties organization : http://rud-team.de/ (German)

Belgium casualties organization : http://www.hvk.be/venus_westvlaanderen/ (Dutch)

Training US (combat simulation) : http://www.militarymoulage.com/ (English)

Photographs

Jorrit Schlenter, photo of Livex 2009 drill
Joost van den Broek, Volkskrant, 2010, photo of hospital drill, pre-eclamptia
Erik Jonker, 2009, Photo of LOTUS lesson ‘model photo’ (wound on arm)
Other photographs: personal archive of Nathalie Rooseboom de Vries van Delft