Developer Network

QA Milestones since the Foundation of the eHealth Framework

Markus Jenderek

Imagine that you are responsible for the quality of the underlying technical platform in the scope of an electronic health record that is being developed from scratch. The platform is the eHealth Framework (eHF). In addition, this solution is part of an integrated health suite with a software development kit (SDK) and various professional services products.

On the 02 December 2009 the tenth version of the eHealth Framework with the version number 2.9 was released. Therefore it is a good occasion to take a look back and remind ourselves of the challenges we faced so far in QA, how we solved them and how we generated value.

As the saying goes: On time, on scope, on budget! The role that quality plays in this mix can't be underestimated and it's the subject of this article.

QA from Scratch to First Release:

When developing software from scratch you first need to lay the foundations so that when it is time to roll out the software all the features and usage scenarios can be tested. Sounds easy but it isn't!

Here is a list of the highlights:

- Product Management produced the Software Requirement Specification (SRS) with an overview of the priorities for developing the basis of the new LifeSensor, a personal health record.
- Based on the Software Requirement Specification, QA extracted all the relevant test cases, these are then reviewed by Product Management.
- The Tooling to carry out the necessary tests was built. To ensure the web services testing could be carried out we built a Java-based web services test client that catered for maximum flexibility und automation. The test data was stored in an MS Excel sheet.
- The development cycle was such that every Thursday there was a new version to be tested. Based on our testing we created new bugs as necessary.
- The respective Product Manager created acceptance tests for the newly developed features that had to be completed for the current week. In this way everyone was involved in an operational feature.
- In such situation be careful with statements like "….I'm 90% finished…" when you are already in the 3rd day of finalizing the requested missing 10%. I don't want to criticize anyone because requirements can and do change and that causes effects from the DB straight to GUI and its usability.
- To steer the project there was a clear focus on pragmatic and sensible reporting: tested, not-tested and accepted Bugs including Top-Ten Risks.

Lessons learned:

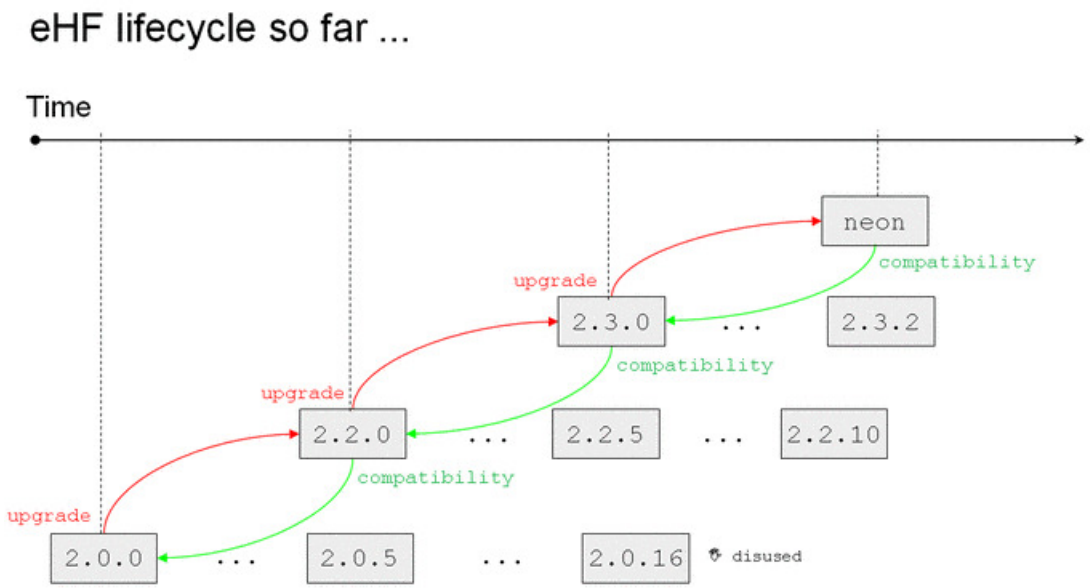
- It was quickly proved in practice that integrative tests between WS-to-GUI and GUI-to-WS were valuable.
- Another important point to share is: Feature or QA-ready is NOT Production ready. So involve administrators to test the software in operation as soon as possible.

QA after Initial Roll-out

Once the software has gone into production each new feature must fit into the existing application both conceptually and functionally without introducing any unwanted side effects. In addition we were confronted with the mandatory requirement that each new version of a framework with existing clients must face. The keyword here is: backwards compatibility. Here it is also important to mention that LifeSensor development was then split between developing its own specific components and its basis components, the latter becoming the backbone of ICW's health care platform: This was the birth of the eHealth-Framework with its own development team.

Here are some of the highlights of this new team:

- We changed our software development process to agile inside a V-Model and introduced Scrum. This means that we QAs worked in parallel with the development team and product mangers. We were no longer on the receiving end of getting a piece of software thrown over the fence and having to test it somewhere in isolation. The whole team was responsible for delivering the features. The whole team was responsible for quality. Not only QA was responsible for quality any more. This is the modern QA approach like in modern soccer. If the other team has the ball the forwards become the first line of defense.
- If you go deeper with this thought our approach and reason for existing is that QA has to deliver added value and support to development and product management in order to be accepted day by day. For instance, this means no longer being able to hide behind theoretical metrics discussions.
- We refactored the architecture of our test client and separated common components like object creation from the test data in MS Excel and test result reporting. The added value here was that other teams could easily reuse our test data and checks when using the eHealth Framework or if they wanted to build their own test client.
- We started to build up a continuous integration environment based on CruiseControl where every test in our test client was executed during the night.
- We developed further features in our test client in order to minimize duplication of test data based on localization needs. (for instance, different test data was necessary for Austria, Switzerland....)
- Once again: we had already released a software version and would have to release further software versions: So please think about this attached diagram from the outstanding Software Lifecycle Management session from the ICW Developer Conference 2008 and what that means for ensuring backwards compatibility.



Lessons learned:

- Do not underestimate the Software Lifecycle. And it is good to have the checks in a continuous integration process.
- QA is not only testing. QA is not Build Management.
- Build Management is an independent IT topic with special in-depth know-how and provides a crucial service for the development and the QA team.
- With software in production be prepared that QA will get new roles as supporter and investigator of potential bugs.

QA Between Version 2.1 and 2.8

The first part dealt with with the historical roots of QA. We now make a big jump over the released versions. Meanwhile, the amount of test data multiplied. We also supported new platforms such as Glassfish. Now the existing tools had to be made manageable and usable with this amount of necessary test data. Meaning usability is not only a topic for developing a GUI. We carried out about 50,000 tests per night on the Web services level, per code line, and rising.

Here is a collection of highlights and ideas:

- The test data in the Excel sheets were clustered together as fragments where it made sense. For example, now it's no longer necessary to include the address in each Emergency Contact as test data; you can pick the one you want from a pool of addresses and reference this. This saves lots of maintenance work when the interface changes.
- With each negative test, we check the expected exception. However, exceptions and their messages are also subject to change. So what happens now to our backward compatibility tests? They will all fail. What have we done about this? We reused the localization feature that I talked about in "QA after initial roll out" and include there the newly expected exceptions for backwards compatibility testing. And how does the test now knows its context, is it a backward compatibility test or a normal test for its own initial version? Talk with your build management team, they will provide you a switch.

- The Continuous Integration (CI) environment itself was also continuously developed. To deal with the quantity and to scale better we moved to Hudson. Appropriate monitoring solutions were designed as well as a “merge assistant” was developed for more control over this process. Additionally, the tomcat server logs are now published by Hudson for every test that is run.
- In order to test Auditing and Encryption on the database level we developed direct access to the DB with reuse of the already written and automated test cases. Of course everything can be run on local machines as well as in the CI environment.
- We made a test tool evaluation but at that time it did not look so good for support in the agile Java domain and an artifact based approach.

Lessons learned:

- Automation is king. Period! But be careful because automation can make you stupid. Systems and approaches are changing and therefore the test cases have to change respectively. Just configuring the test system and executing one BIG button is not QA but a hit-and-run approach - even when the test results are all “executed ok” in the log files. So we have to ask ourselves every time - Are we still testing the right things?
- Greatness must be worked hard for every day and is not a default value.
- Now take a deep breath and think about what is the process worth in time problems? Everybody should think about it at least once again after a release is delivered.
- Establish a weekly Bug smashing session. Here it’s important that not only the QA-Team works to resolve the bug but that decision makers from PM and Dev are also represented. The whole project will take a great benefit from that.

QA for the Latest Release - 2.9

It will not surprise anyone if I say the amount of work is no less. What was not mentioned so far is that we also have an outstanding development team in Bulgaria. That is, the communications must be guided by this. But there are enough tools to cater for this situation they only need to be properly implemented. Imagine that the whole team is located on the same floor, but everyone stays at their desk. Fortunately Scrum addresses this issue. It is not complicated.

So we're into the final stretch. Here are the main highlights and some associated ideas:

- We achieved the goal of including the whole system documentation (based on DITA) in our CI environment (for example, part of this documentation set is included on ICW’s partner DVDs).
- Not only bugs, but all story planning as well version tracking is completely mapped in Jira.
- This means we had absolutely everything that we need to create a release in our CI environment.
- Some bugs need to be fixed not only for future versions but also in older versions or code lines. How can I ensure that everything is remembered? During bug smashing if we determine that we need to provide a fix for an older version, we provide this information as an additional subtask in the Jira entry. Everything is now transparent and if there are side effects everyone can see if a fix was checked in and when it was checked in and QA can start with retesting.
- As far as our CI environment went we had to solve the problem that in the meantime we carry out so many tests that they were not finished in the morning but rather at midday. In this regard our build management with development and QA worked together to improve the feedback time by optimizing the build time. Some intermediate goals to achieve the required solution included: different load balancing and tuning of integration tests.

How to improve further

Have you ever experienced the situation where somebody calls you up and nervously asks "...QA! Do you have a test which checks the... (fill in some scenario you hope is covered) " – always if the customer claims to have found a bug , suddenly in that second for every stakeholder this is such a clear and logical never to be forgotten test from that moment on. So. Hmmmmm! Why not identify these "obvious" ones a little bit sooner? How? We have product managers, developers and QAs and everybody brings their own perspective to every single feature. We can use these perspectives and you will be surprised how your test cases increase. Every test case you identify in such sessions is an additional security net and a good day for the quality of your product. Donate this to your system as much as possible :-) It is of great benefit.

Some of next challenges will be adding micro-benchmarking tests in our test scenarios. This will never replace a mandatory performance test from this specially skilled people in our performance lab for every release but gives you soon information about possible side effects when your system is growing up.

I hope you liked this short overview of our work and that you can find something here that will help you in your own daily work. Feedback is very much appreciated. Thanks to my colleagues in technical writing and build management for their input and help with this post.

InterComponentWare AG Industriestraße 41 69190 Walldorf, Deutschland Tel.: +49 (0) 6227 385 0 Fax: +49 (0) 6227 385 199 www.icw-global.com info@icw-global.com