Analyse 2

Assignment 1

Robert Kraaijeveld, Steven Schenk, Cees-Jan Nolen, Robin Bakker

INF2C

Inhoudsopgave

[A 3](#_Toc429561835)

[B 3](#_Toc429561836)

[C 3](#_Toc429561837)

[D 4](#_Toc429561838)

[E 5](#_Toc429561839)

[F 6](#_Toc429561840)

# A

A client-software type application.

# B

We could change our client-software application to a data-collection system.

Our original project was built as an Android-app that collected tourist attractions from a standard database+ user suggestions, and then displayed those tourist attractions on a map,

Along with weather data. This could be changed into a data-collection system if the app would instead only ask the users to add new tourist attractions and subsequently rate them (with a star system for example). In this manner, the app would be collecting data which could then be reported to the owners of the aforementioned tourist-attractions, making the app a data-collection system.

# C

The downloading of local copies of patients’ files from the secure database could lead to a potential security/privacy risk, because these locally copied files are not going to be encrypted like the central database entries. On the other side, having local copies available in case of a system-failure might potentially save patients’ lives, especially when considering borderline- or suicidal patients.

# D

If we had used a waterfall-project method for Project 4, we would have started with analyzing our customers wishes and carefully preparing the system-requirements. After these requirements would have been validated and approved by our Manager/Stakeholder/Etc, we would have started to build the software according with our pre-made building plan. After our building-time, we would have created a test plan and tested our software according to this plan. When all of our tests would have passed, we would have validated and released our software.

In an incremental Project method (which we actually used) we would have started with our Product Owner delivering us a list of requirements for the entire project, known as a Product Backlog. After that, we would have gotten the team together and discussed which Items of this Product Backlog we wanted to tackle the following Sprint (2/3 week long Iteration) and also given each item an estimated time-rating. Then we would have started building and testing our software (we used Scrum, not XP so thusly we didn’t write our tests first) and at the end of our sprint, we would have looked with our Product Owner at how well our implementation fits the customers’ wishes. This process would repeat itself until the Project would be deemed complete.

If we were to use a Re-use-oriented method, we would have looked at our applications requirements first. After we would have been clear on what those requirements where, we would have chosen a software system that had similar requirements to the requirements we wanted to achieve. Once we would have found such a software system, we would have analyzed all the components of it in and modified our requirements to accompany the components that we had found. Then we would have started to develop our system, integrating the components that we had analyzed before. When our developing-time was over, we would have validated and tested our entire system to make sure it and its’ borrowed components behaved as expected and fulfilled our system-requirements.

# E

In the waterfall-method, I would have relied on a form of Prototyping in order to cope with changing requirements in the software. Prototyping in the waterfall-method only really works in the development phase (I do not think that making multiple requirements-prototypes is a very efficient use of time and resources) so I would add regular ‘pauses’ in the development phase to allow the customer to look at the prototype of the system thus far, and decide whether it fulfills their requirements or not.

In an Incremental Project Method like Scrum or XP, change-management activities are a regular part of the Project method. In fact, the Agile Manifesto (which contains the underlying principles that Agile Methods like Scrum and XP use) tells us to “embrace change”! In an Incremental Project Method the increments/sprints usually have a fixed time period; at the end of each increment/sprint the development team meets with the customer representative (Known as Product Owner in Scrum) and they discuss how the current increment holds up towards the customers’ requirements. If the customer wants any already present feature changed, or other changes, these can be easily accommodated in the next increment.

In a Reuse-oriented project I would (heavily) rely on prototyping to cope with changing requirements. Since a lot of the time of a Reuse-oriented project is spent analyzing requirements and adding useful pre-made components to the project as time passes, it would be easy to use prototyping: Every extra component that gets added could have its’ own prototype, so the customer can easily find out whether the new component satisfies their (changing) requirements or not.

# F

The parts of RUP that would have been applicable to our Project 4 are:

-Inception: Inception within RUP basically means that before you actually start your project, you should look into the various people and other systems that will be using your software, a and what kind of contribution your software will deliver to their experience. In the case of our tourist-attractions app, knowing what kind of contribution our app was going to make to the user was very important: Several questions arose, such as: Will we be focusing on the Netherlands, or will we provide attraction-information for other countries too? Are we mainly going to spark the users’ interest in hidden gems, or are we just going to point them to attractions everyone knows already? In the end we ended up with an app in which many kind of attractions were present. If we would have used the Inception principle, we would have ended up with an app that would have a more consistent tone and thusly a bigger contribution to the users’ experience.

The parts of RUP that would **not** have been applicable to our Project 4 are:

- Elaboration: The elaboration-part is basically concerned with specifying detailed requirements and project plans and identifying possible risks to the project. This part of RUP would not have been very useful in our Project, because we used a Iteration-approach (Scrum, with some Extreme Programming parts in it) in which we had neither the time to come up with advanced requirements and project plans nor the ability to do so, since the requirements made by our Product Owner were subject to a lot of change.

- Transition: This part of RUP is concerned with moving the project from a development environment into a user environment and producing the necessary documentation to do so.

In the Scrum style of Project method (which we used) quickly delivering quality software is valued over the creation of extensive documentation (which will often not be updated anyway)

- Construction: In the Construction part of RUP the system is designed, programmed and tested. Several parts of the system may be developed parallel to others. This approach would not have been very useful in our project, because we used a Scrum-style project method, developing the system in a ‘static’ way (2 weeks design, 10 weeks coding, etc) would not have worked very well with Scrums’ iterative approach. Also we did not have multiple teams available in order to develop several parts of the system parallel to eachother.