HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins institute for networked solutions

IoT - Provisioning und Management

Bachelorarbeit FS 2017

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren: Andreas Stalder, David Meister
Betreuer: Prof. Beat Stettler, Urs Baumann
Projektpartner: INS Institute for Networked Solutions
Datum: 7. März 2017

Inhaltsverzeichnis

1 Projektplan

1.1 Projektübersicht

In diesem Projekt soll der Stand der Entwicklungen im Bereicht „Internet of Things“ aufgezeigt werden. Die verschiedenen Arten der eingesetzten Sensoren sollen ermittelt- und die Einsatzgebiete untersucht werden. Heutzutage werden in der Industrie bereits verschiedenartige Sensoren eingesetzt. Die Anzahl der eingesetzten Sensoren steigt drastisch. Schon bald stellt sich die Frage, wie man mit der steigenden Anzahl Sensoren deren Management realisieren soll.

1.1.1 Zweck und Ziel

Die Bachelorarbeit soll den Nachweis der Problemlösungsfähigkeit unter Anwendung ingenieurmässiger Methoden nachweisen. Entsprechend verfügt die Arbeit über einen konzeptionellen, theoretischen und einen praktischen Anteil.

1.1.2 Projektorganisation

Vorname	Name	E-Mail
Andreas	Stalder	astalder@hsr.ch
David	Meister	dmeister@hsr.ch

Tabelle 1.1: **Teammitglieder**

Das Projekt wird von Prof. Beat Stettler und Urs Baumann betreut und benotet. Experte und Gegenleser sind zur Zeit noch nicht bekannt.

1.2 Management Abläufe

1.2.1 Zeitbudget

Der Projektstart ist am Montag, dem 20. Februar 2017.
Die Projektdauer beträgt 17 Wochen, und das Projektende ist am Freitag, dem 16. Juni 2017.

Während diesen 17 Wochen sind 360 Arbeitsstuden pro Projektmitglied eingeplant. Das entspricht pro Mitglied eine Arbeitszeit von ca. 22 Stunden pro Woche. Dies ergibt einen totalen Aufwand von ca. 720 Stunden.

Die wöchentliche Arbeitszeit von 22 Stunden kann bei Verzug oder bei unerwarteten Problemen auf maximal 30 Stunden erhöht werden.

Es sind gegenwärtig keine Absenzen während dieser Zeit geplant.

1.2.2 Projektphasen

Das Projekt wird in fünf Phasen unterteilt: Initialisierung, Analyse, Design, Realisierung und Abschluss.

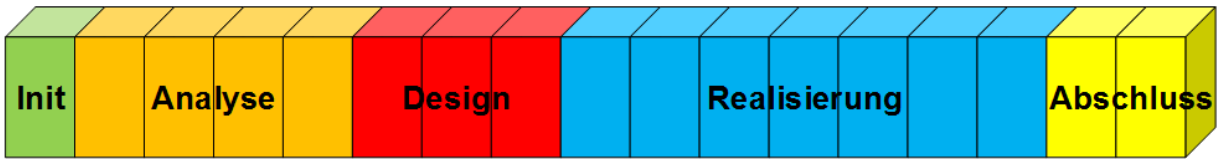

Abbildung 1.1: Projektphasen

1.2.3 Meilensteine

Das Projekt beinhaltet insgesamt vier Meilensteine.

Meilenstein	Beschreibung	Datum
MS1	Anforderungen und Scope definiert	26.03.2017
MS2	Architektur und Design beschrieben	16.04.2017
MS3	Software fertiggestellt, Codefreeze	04.06.2017
MS4	Arbeitsabgabe	16.06.2017

Tabelle 1.2: **Projekt Meilensteine**

1.2.4 Iterationen

Die Dauer eines Iterationszyklus beträgt jeweils eine Woche.

Iteration	Inhalt	Start	Ende
Initialisierung 1	Kickoff Meeting, Projektplanung, Infrastruktur	20.02.2017	26.02.2017
Analyse 1	IoT Analyse Allgemein	27.02.2017	05.03.2017
Analyse 2	IoT Analyse Allgemein	06.03.2017	12.03.2017
Analyse 3	Sensoren Analyse technisch & Evaluation	13.03.2017	19.03.2017
Analyse 4	Requirements definieren	20.03.2017	26.03.2017
Design 1	-	27.03.2017	02.04.2017
Design 2	-	03.04.2017	09.04.2017
Design 3	-	10.04.2017	16.04.2017
Realisierung 1	-	17.04.2017	23.04.2017
Realisierung 2	-	24.04.2017	30.04.2017
Realisierung 3	-	01.05.2017	07.05.2017
Realisierung 4	-	08.05.2017	14.05.2017
Realisierung 5	-	15.05.2017	21.05.2017
Realisierung 6	-	22.05.2017	28.05.2017
Realisierung 7	-	29.05.2017	04.06.2017
Abschluss 1	-	05.06.2017	11.06.2017
Abschluss 2	-	12.06.2017	16.06.2017

Tabelle 1.3: **Projekt Iterationen**

1.2.5 Arbeitspakete (Tickets)

Name	Inhalt	Iteration	Wer	Soll	Ist
Initialisierung					
Kickoff-Meeting	Allgemeine Besprechungen zum Projektstart	Initialisierung 1	Alle	1	1
Dokumenterstellung	Erstellung LaTeX Vorlagen	Initialisierung 1	Alle	4	5
Projektplan	Zeitplanung, Phasen, Meilensteine	Initialisierung 1	Alle	5	6
Einrichtung Projektmanagement Software	Installation und Einrichtung Jira	Initialisierung 1	Alle	3	-
Analyse					
-	-	-	-	-	-
Design					
-	-	-	-	-	-
Realisierung					
-	-	-	-	-	-
Abschluss					
-	-	-	-	-	-

Tabelle 1.4: **Arbeitspakete**

1.2.6 Teammeetings

Besprechungen finden drei mal wöchentlich jeweils an den vorgesehenen Arbeitstagen statt. Besprechungen dauern in der Regel 10-15 Minuten. Es wird das weitere Vorgehen, sowie durchgeführte Arbeiten, fällige Arbeiten und auftretende Probleme besprochen. Weiter werden Arbeitspakete verteilt, damit beide Projektmitglieder wissen was zu tun ist.

1.2.7 Meeting mit Betreuern

Die Meetings mit den Betreuern finden jeden Freitag um 14:00 Uhr statt. Die Meetings werden mit den Betreuern Prof. Beat Stettler und Urs Baumann in ihrem Büro durchgeführt. Die Meetings dauern normalerweise zwischen 30-60 Minuten.

1.3 Qualitätsmassnahmen

1.3.1 Versionierung

Wie die Dokumentation wird auch der Sourcecode mit git versioniert und auf GitHub abgelegt. Es wird darauf geachtet, möglichst häufig auf den Stamm zu commiten.

1.3.2 Reviews

Regelmässige Reviews sind in einem iterativen Vorgehen unerlässlich. Die getätigte Arbeit muss ständig abgeglichen und in Frage gestellt werden. Aus Kosten-Nutzen Sicht sind Reviews das effektivste Mittel um die geforderte Qualität zu erreichen.

In diesem Projekt werden drei verschiedene Arten von Reviews durchgeführt. Zum einen sind dies regelmässige Code Reviews, zum anderen sind dies Requirement- und Architekturreviews.

Bei den regelmässigen Code Reviews wird besonders auf die gewählten Namen (Packages, Klassen, Methoden, Variablen), die Verständlichkeit vom Code und Code Smells geachtet.

Bei den Requirement Reviews wird sichergestellt, dass man den Wuenschen des Auftraggebers entsprechend entwickelt. Die Frage nach dem „Was“ wird erneut gestellt und somit sichergestellt, dass man beim Projektende nicht ein qualitativ hochwertiges Produkt entwickelt hat, welches aber nicht die Wünsche des Auftraggebers abdeckt.

Beim Architektur Review wird besonders auf die nicht-funktionalen Anforderungen (NFA) geachtet. Diese wirken sich in den meisten Fällen auf die gewählte Architektur aus. Die gewählte Architektur muss mit den NFA's verträglich sein. Wenn zu spät im Projekt bemerkt wird, dass die Architektur geändert werden muss, kann dies sehr aufwändig sein.

1.3.3 Code Metriken

Code Metriken zeigen mögliche Fehler oder Schwachstellen im entwickelten Code auf. Es wird grundsätzlich zwischen statischen- und dynamischen Metrik Tools unterschieden. Bei den dynamischen Metrik Tools wird der Code ausgeführt. Beispiele wären Unit Tests und die dazugehörige Coverage. Bei den statischen Analysetools wird der Code nicht ausgeführt. Ein Beispiel wäre Checkstyle. Dieses Tool überprüft vor allem die Einhaltung von Style Richtlinien.

Abbildungsverzeichnis