



Technische
Universität
Braunschweig

Institut für
Flugführung



Softwareprojektmanagement

Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc.,
16. April 2019

Agenda

- 09. April Einführung
- 16. April Softwareprojektmanagement**
- 23. April Entwicklungstools
- 30. April GitHub
- 07. Mai Dokumentation und Bug-Reporting
- 14. Mai Einführung Arduino
- 21. Mai Dateieingabe und -ausgabe
- 28. Mai Einführung von Qt
- 4. & 11. Juni Tag der Lehre und Exkursionswoche
- 18. Juni GUI-Erstellung mit Qt
- 25. Juni Serielle Kommunikation
- 02. Juli API-Anleitung und Projektarbeit
- 09. Juli Vorbereitung der Abgabe
- 16. Juli Fragen
- 12. August 10:00 Abgabe



Lehrziele

Softwareprojektmanagement

Als Teilnehmer soll ich am Ende dieser Übung...

- Methoden kennen, die Softwareentwicklung planbar machen
- klassische & agile Softwareentwicklung abgrenzen können
- mit der Projektmappe beginnen können
- Sprints planen und durchführen können
- das API-Vorgehensmodell kennen



Einführung Projektplanung



Realität?

Problematik



Wie Softwareprojekte bearbeitet werden...



[www.projectcartoon.com](http://projectcartoon.com)

[<http://projectcartoon.com>]

Wie der Kunde es erklärt hat



Technische
Universität
Braunschweig

16. April 2019 | Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc. | Seite 5
Softwareprojektmanagement

Institut für
Flugföhrung



Wie Softwareprojekte bearbeitet werden...



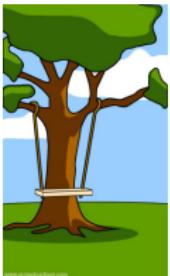
[<http://projectcartoon.com>]



Wie der Projektleiter es verstanden hat



Wie Softwareprojekte bearbeitet werden...



[<http://projectcartoon.com>]



Wie der Analyst es auffast



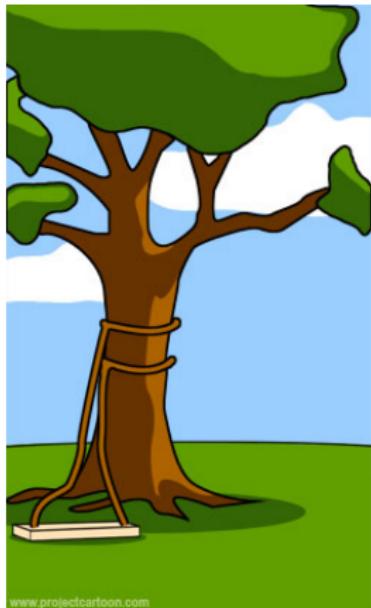
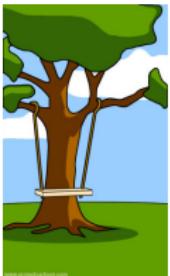
Technische
Universität
Braunschweig

16. April 2019 | Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc. | Seite 5
Softwareprojektmanagement

Institut für
Flugföhrung



Wie Softwareprojekte bearbeitet werden...

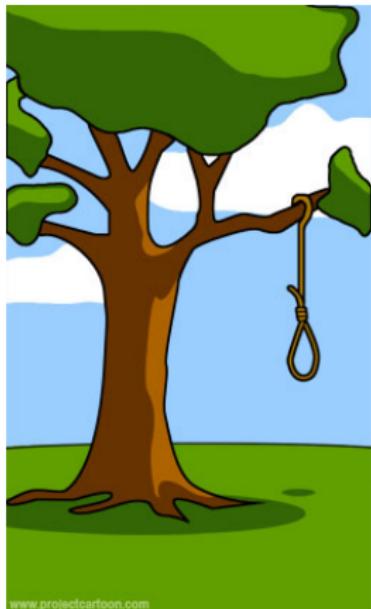
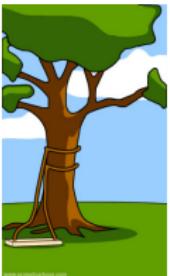
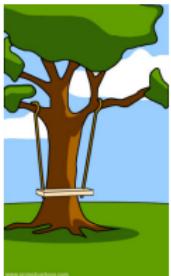


[<http://projectcartoon.com>]

Wie der Programmierer es geschrieben hat



Wie Softwareprojekte bearbeitet werden...

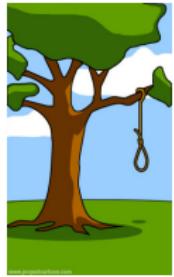
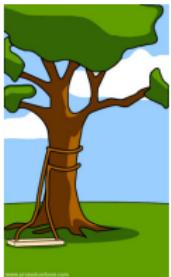
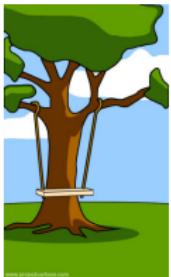


[<http://projectcartoon.com>]

Was die Beta-Tester erhalten



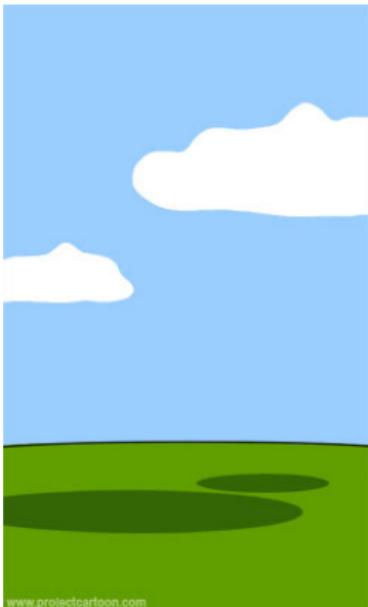
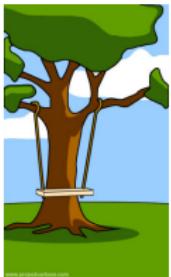
Wie Softwareprojekte bearbeitet werden...



[<http://projectcartoon.com>]

Wie der Wirtschaftsberater es verkauft

Wie Softwareprojekte bearbeitet werden...

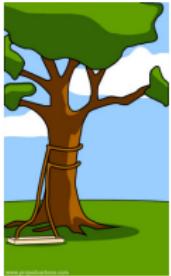
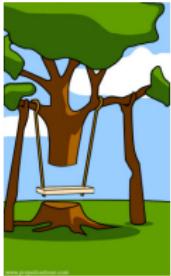


[<http://projectcartoon.com>]

Wie das Projekt dokumentiert wurde



Wie Softwareprojekte bearbeitet werden...



[<http://projectcartoon.com>]

Was der Kunde wirklich gebraucht hätte



Allgemeine Problematik

- Anforderungen der Kunden werden nicht erfüllt
- Kosten für Software sind zu hoch
- Methoden der Softwareentwicklung ändern sich schnell
- Spezialisierung von Entwicklern kurzlebig (Innovationsdruck)
- Weiterverwendung alter Technologien
- Fokus auf Technologien zur Realisierung, nicht bei der eigentlichen Anwendung



Probleme der Anforderungsdefinition [1]

- Kundenspezifische Problematik

- Es ist nicht genau bekannt, was entwickelt wird
 - Anforderungen werden ungenau/falsch formuliert
 - Fachsprache kann für Verständnisprobleme sorgen
 - Unterschiedliche Anforderungen der Projektbeteiligten (Stakeholder)

- Randbedingungen

- Unternehmenspolitische Einschränkungen
 - Änderungen der Anforderungen möglich

Notwendigkeit, Anforderungen klar und möglichst einfach auszuformulieren



Spezifische Probleme der Softwareentwicklung

- Software ist schwer messbar/beurteilbar
 - (Zwischen-) Ergebnisse sind für IT-Laien oft schwer beurteilbar
 - Es besteht ein hoher Verifizierungsaufwand
- Zusammenhang zwischen Anforderung und Kosten ist nicht intuitiv
 - Kleine Anforderungsänderungen können große Auswirkungen haben
 - Oftmals viele Änderungen der Anforderungen während des Projektverlaufs
- Es besteht starke Personalabhängigkeit
 - Programmierer sind nicht einfach austauschbar
 - Es bestehen erhebliche Produktivitätsunterschiede zwischen Programmierern



Zusammenhang zwischen Anforderungen und Kosten



Abbildung 1: Zusammenhang zwischen Anforderungen und Kosten [<https://imgs.xkcd.com/comics/tasks.png>]

Vorgehensmodelle



Realität

Wasserfallmodell

Spiralmodell



Einfach machen...

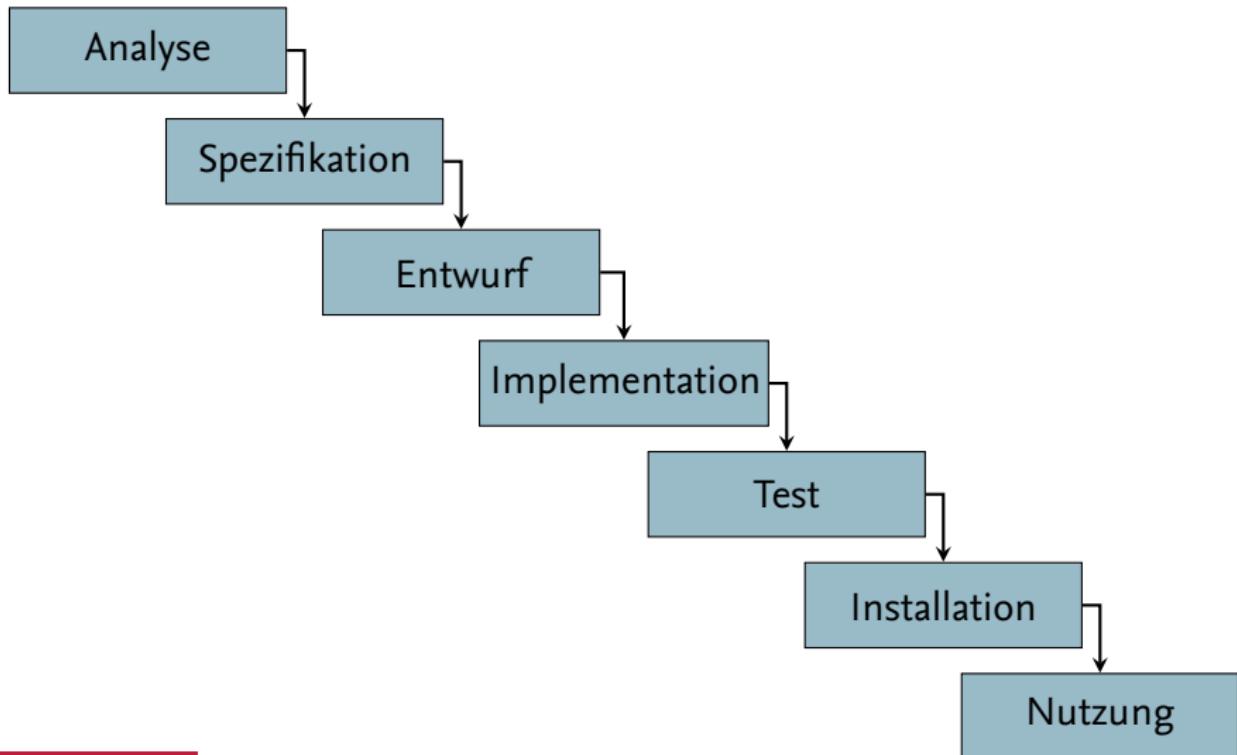
1. Quellcode schreiben
2. Fehler beheben

Nachteile:

- Fehlerbehebungen führen meistens zu Umstrukturierungen
 ⇒ Weitere Behebungen werden aufwendiger
- Wenig Akzeptanz des Produkts beim Endnutzer
- Fehleridentifikation ist sehr schwierig, weil Tests nur unzureichend vorbereitet wurden



Das Wasserfallmodell [2]



Wasserfallmodell

Vor- und Nachteile [3]

Vorteile:

- Einfach
- Geringer Managementaufwand

Nachteile:

- Nicht immer sinnvoll, Phasen komplett abzuschließen
- Nicht immer sinnvoll, alle Phasen sequentiell abzuarbeiten
- Dokumente haben zum Teil eine höhere Priorität als das Produkt
- Zu spät identifizierte Risiken können unter Umständen nicht abgefangen werden



Spiralmodell nach Boehm



[Conny, Wikimedia (CC-BY-SA-3.0)]



Spiralmodell

Vor- und Nachteile [3]

Vorteile:

- Periodische Überprüfung kann ein Abdriften von Zielen und Risiken verhindern
- Je nach Zyklus kann ein geeignetes Prozessmodell ausgewählt werden
- Flexibel
- Erfahrungen können im nächsten Zyklus eingebracht werden

Nachteile:

- Hoher Managementaufwand für kleine und mittlere Projekte
- Risiken müssen identifizierbar sein



Gibt es Fragen oder Anmerkungen zu dem Thema
Vorgehensmodelle?



Abgehakt

Softwareprojektmanagement

Als Teilnehmer soll ich am Ende dieser Übung...

- Methoden kennen, die Softwareentwicklung planbar machen
- klassische & agile Softwareentwicklung abgrenzen können
- mit der Projektmappe beginnen können
- Sprints planen und durchführen können
- das API-Vorgehensmodell kennen



Agile Software-entwicklung

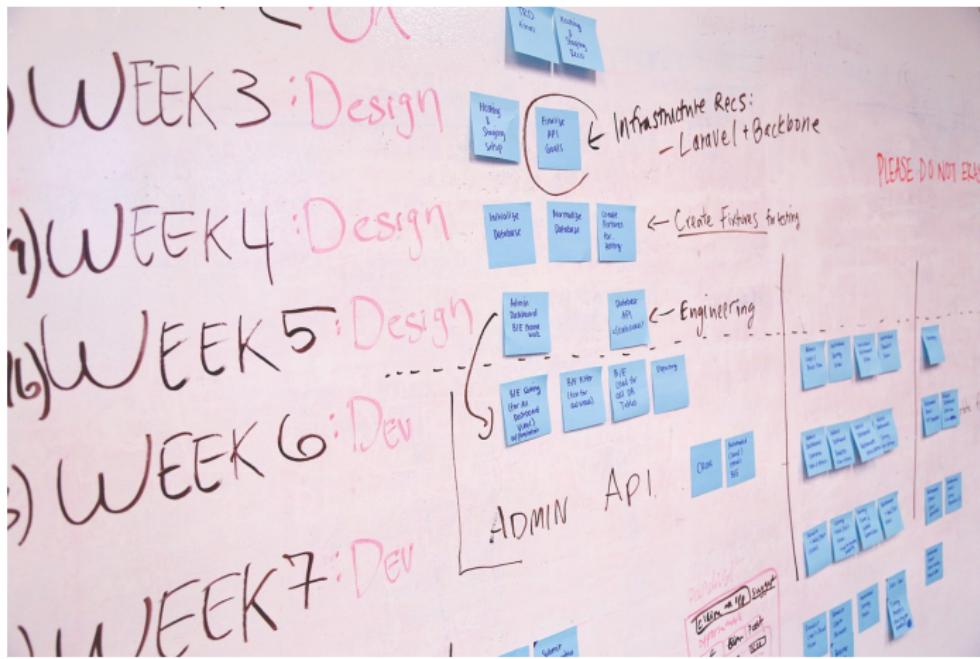


Definition

Scrum



Vorstellung von agiler Entwicklung



Historische Entwicklung [4]

- ab den 90ern Diskussion der ersten agilen Managementansätze
Welt zu schnelllebig für Wasserfallmodell
- 1999 Veröffentlichung von „Extrem Programming“
Lösen von Programmieraufgaben rückt in den Vordergrund
- 2001 Agile Manifest
Etablierung des Begriffs der agilen Softwareentwicklung



Manifest für agile Softwareentwicklung

In der agilen Softwareentwicklung werden

**„Individuen und Interaktionen mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen eines Plans“**

geschätzt.

[<http://agilemanifesto.org>]



Was ist die agile Softwareentwicklung?

Bei der agilen Softwareentwicklung wird ein besonderes Augenmerk auf die Kommunikation und Transparenz im Entwicklungsprozess gelegt.

Die folgenden Punkte sind charakteristisch für die agile Entwicklung:

- Das Team organisiert sich selbst
- Der Entwicklungsprozess wird stets hinterfragt und verbessert
- Anforderungen können im Entwicklungsprozess verändert werden
- Funktionierende Software ist das wichtigste Fortschrittsmaß
- Es herrscht ein reger Informationsaustausch unter allen Beteiligten

Das Commitment aller Teammitglieder ist erforderlich!



Agile Softwareentwicklung

Vor- und Nachteile

Vorteile:

- Softwareinkremente sorgen für stetiges Feedback
- Auf Anforderungsänderungen kann schnell reagiert werden
- Geringer bürokratischer Aufwand
- Entwicklungsprozess kann auf das jeweilige Projekt abgestimmt werden

Nachteile:

- Entwicklungsprozess hängt sehr stark von den planenden Rollen ab
- Schwierige Vertragsdefinition



Gibt es Fragen oder Anmerkungen zu dem Thema
Agile Softwareentwicklung?



Scrum



Was ist Scrum? [5]

Scrum ist ein offenes Framework für das Management agiler Softwareprojekte.

Diese Scrum-Regeln sollen helfen, das Produkt nah an den Kundenanforderungen zu entwickeln:

- Entwicklung ist in Sprints unterteilt
- Rolleneinteilung
 - Entwicklungsteam
 - Scrum Master
 - Product Owner



Anforderungen als User-Stories

User-Stories formulieren Kundenanforderungen aus.

User-Stories sollen...

- I Independent ...unabhängig voneinander sein.
- N Negotiable ...verhandelbar sein.
- V Valuable ...einen Wert für den Kunden haben.
- E Estimatable ...schätzbar sein.
- S Small ...klein sein.
- T Testable ...testbar sein.



User-Story Aufbau [5]

Eine User-Story hat 3 Bestandteile:

1. Benutzerrolle
2. das Ziel
3. Grund für das Ziel

Als *<Rolle>* möchte ich eine *<Aktion>* ausführen, weil sie diesen *<Nutzen>* bringt.



Product Backlog [5]

Das Product Backlog umfasst die Anforderungen (z.B. als User-Stories) der zu entwickelnden Software und stellt die Produktvision dar.

- Anforderungen werden nach ihrer Wichtigkeit priorisiert
→ Bestimmt deren Entwicklungsreihenfolge
- Die Umsetzungsdauer ist geschätzt
- Enthaltene Anforderungen können modifiziert werden
- Verantwortlich für die Anforderungen ist der Product Owner



Scrum

Akteure: Entwicklungsteam [6]

Entwicklungsteam (3-9 Personen):

- Selbst organisierend
- Interdisziplinär
- Liefert Produkt
- Arbeitet eigenständig an den User-Stories eines Sprints
- Implementiert und testet Anforderungen

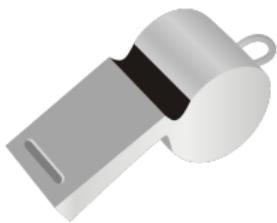


Scrum

Akteure: Scrum Master [6]

Scrum Master:

- Treibt den Prozess
- Wahrung und Vermittlung der Scrum-Werte und -Regeln
- Schützt das Team vor Störungen
- Löst Blockaden
- Vermittelt zwischen Team und Product Owner

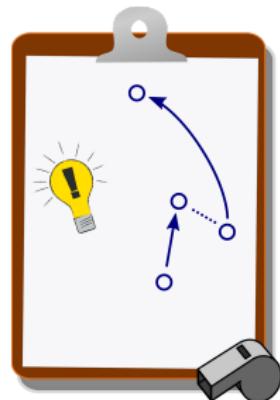


Scrum

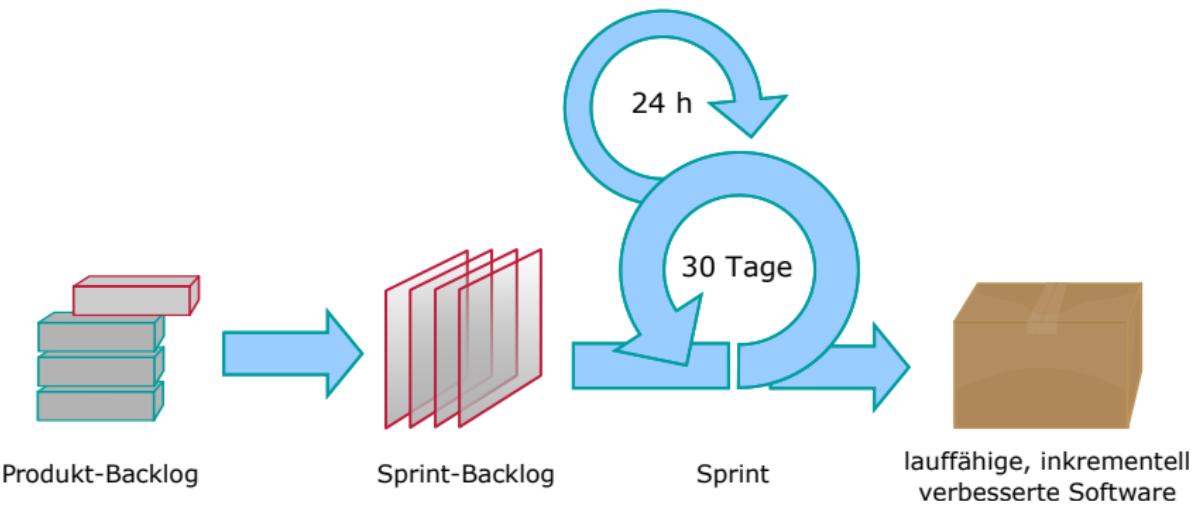
Akteure: Product Owner [6]

Product Owner:

- Verkörpert die Projektidee
- Maximiert den Wert des Produktes
- Vermittelt die Vision an das Team
- Stellt das Team zusammen
- Erstellt ein Product Backlog
- Priorisiert das Product Backlog



Scrum-Prozessverlauf

Abbildung 2: Scrum-Prozessverlauf¹

¹Von Scrum_process.svg: Lakeworksderivative work: Sebastian Wallroth (talk) - Scrum_process.svg, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=10772971>

Phasen eines Sprints

Der Sprint ist ein endlicher Zeitraum, in dem das Entwicklungsteam geschützt an ausgewählten ausgewählten Product Backlog Items arbeitet.

Ein Sprint beinhaltet

- die Sprintplanung
- die Sprintdurchführung
- die Retrospektive

Das Ergebnis eines Sprints soll ein Softwareinkrement sein, dass von den Kunden eingesetzt werden kann und Feedback liefert.



Sprintplanung

In der Sprintplanung werden die Ziele für den jeweiligen Sprint festgelegt. Die Ziele leiten sich aus den Product Backlog Items ab und sollen innerhalb eines Sprintzeitraums fertig gestellt werden können.

- Auswahl von User-Stories bzw. einzelner Zielkriterien
- Zerlegung in zu bearbeitende Aufgaben
- Abhängigkeiten unter den Aufgaben identifizieren
- Zuteilung der Aufgaben

Es entsteht ein Sprint-Backlog, den es im Sprint abzuarbeiten gilt.



Kanban Board

Mithilfe eines Kanban Boards können die Aufgaben eines Sprints und deren Bearbeitungsstand visualisiert werden.



Abbildung 3: Kanban Board Tutorial [Jeff.lasovski, <https://commons.wikimedia.org/wiki/File:Simple-kanban-board-.jpg> (CC BY-SA 3.0)]

Sprintdurchführung

Während des Sprints arbeitet das Team eigenständig an den definierten Aufgaben.

Daily-Scrum-Treffen sollen helfen, alle Teammitglieder zu synchronisieren und Probleme offen zu legen.

Das Treffen ist verbindlich für das gesamte Entwicklungsteam und darf nicht länger als 15 Minuten dauern

Jedes Teammitglied steuert die folgenden Beiträge bei:

- Was wurde seit dem letzten Meeting bearbeitet?
- Die Zwischenstände der Aufgaben und ob es Schwierigkeiten gab
- Was wird bis zum nächsten Meeting bearbeitet?



Sprintretrospektive

In der Sprintretrospektive soll der abgeschlossene Sprint hinsichtlich der Ergebnisse und der Produktivität bewertet werden. Die Bewertung des Sprints soll helfen, Probleme zu identifizieren und diese im folgenden Sprint zu beheben.

- Wurden die gesteckten Ziele erreicht?
- Gab es Schwierigkeiten bei der Bearbeitung?
- Wie kann der Entwicklungsprozess verbessert werden?



Gibt es Fragen oder Anmerkungen zu dem Thema
Agile Softwareentwicklung?



Abgehakt

Softwareprojektmanagement

Als Teilnehmer soll ich am Ende dieser Übung...

- Methoden kennen, die Softwareentwicklung planbar machen
- klassische & agile Softwareentwicklung abgrenzen können
- mit der Projektmappe beginnen können
- Sprints planen und durchführen können
- das API-Vorgehensmodell kennen



API-Vorgehensmodell

38

[API]

2019

Einführung

User-Stories

Projektschema

Sprint



Motivation

Kein bestehendes Vorgehensmodell ist perfekt für API.

- API soll moderne agile Softwareentwicklung demonstrieren
→ Wasserfall- und Spiralmodell ungeeignet
 - Alle Studierenden sollen gleich bewertet werden können
→ Verschiedene Scrum-Akteure sind schlecht abbildbar
- ⇒ Gesondertes API-Vorgehensmodell



API-Vorgehensmodell

Vorgehensmodell inspiriert von Scrum

- Nur ein Akteur: Studierende
 - Verkörpern gemeinsam *Product Owner* und *Entwicklungsteam*
 - Erstellen User-Stories und verwalten Backlog
- Anzahl und Dauer der Sprints wird angepasst
 - Drei Sprints in ca. 12 Wochen

Zusätzlich:

- Erstellung eines Projektschemas als Planungshilfe



User-Stories

Jedes Gruppenmitglied erstellt eine User-Story.

Die einzelnen User-Stories der Gruppenmitglieder sollen voneinander abgrenzbar sein, d. h. verschiedene Funktionalitäten und Nutzungszenarien abbilden.

Für die Umsetzung der User-Stories ist das gesamte Team gemeinsam verantwortlich.



Zielkriterien der User-Stories

Zu jeder User-Story werden vier Zielkriterien erstellt.

Zielkriterien sollen:

- Die Funktionalitäten der User-Story exakt definieren
- Es ermöglichen, den Fortschritt bei der Implementierung der User-Story zu quantifizieren

Dazu müssen die Zielkriterien

1. zueinander abgrenzbar und
2. einzeln messbar sein, vorzugsweise als Ja/Nein-Bedingung.



ÜBUNG

Aufgabe 1

Dauer: 5 Minuten

Jedes Gruppenmitglied überlege sich eine User-Story für das gemeinsame Projekt.

Beispielformulierung:

Als *<Rolle>* möchte ich eine *<Aktion>* ausführen, weil sie diesen *<Nutzen>* bringt.



Beispielprojekt

Rahmenbedingungen

IFF: 55+ Mitarbeiter

250+ Kaffebezüge pro Woche

Zwei Preise: Kaffee, Kaffeespezialität

Abrechnung erfolgt per Strichliste

Grobe Idee

Elektronisches Abrechnungssystem als
Ersatz für die Strichliste.



ÜBUNG

Aufgabe 2

Dauer: 8 Minuten

Jedes Gruppenmitglied überlege sich zu seiner User-Story vier Zielkriterien.

Beispiel:

Das persönliche Kaffeeguthaben wird nach Eingabe der Benutzer-ID auf dem Bildschirm angezeigt.



Gibt es Fragen oder Anmerkungen zu dem Unterthema
User-Stories?



Projektschema



Projektschema

Das Projektschema soll als Hilfe bei der Planung der Projektarchitektur und der Sprints dienen.

- ⇒ Erstellung des ersten Entwurfes vor Beginn des ersten Sprints.
Aktualisierung bei Voranschreiten des Projektes.

Problemstellungen, bei denen das Projektschema helfen soll:

- In welche Komponenten lassen sich die User-Stories aufteilen?
→ Benutzeroberfläche, Datenauswertung, welche Sensoren? ...
- Bestehen Überschneidungen zwischen den User-Stories?
→ Können Sensoren gemeinsam verwendet werden? ...
- Bestehen Abhängigkeiten zwischen den Komponenten?



Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen

<Systemgrenze>
z.B.: Rechner, User-Story, Zielkriterium,...

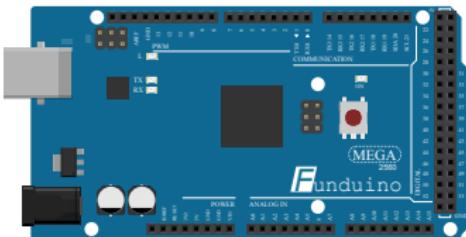


Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards

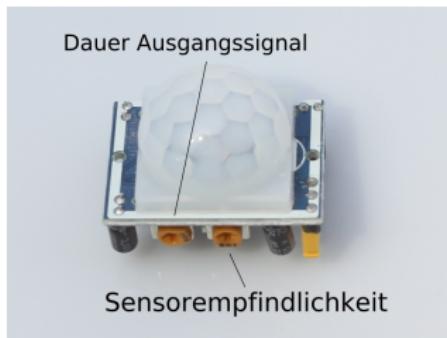


Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards
 - Sensoren



Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards
 - Sensoren
 - Bauteile

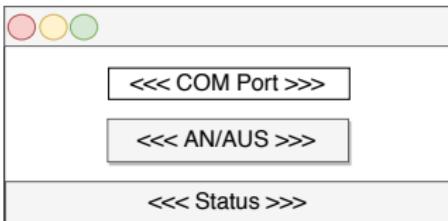


Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards
 - Sensoren
 - Bauteile
- Software
 - Graphische Benutzeroberfläche

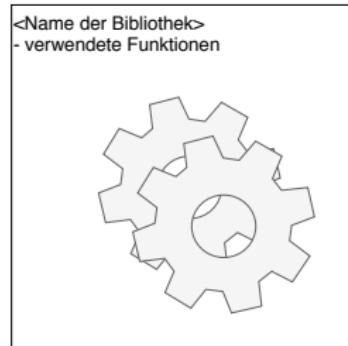


Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards
 - Sensoren
 - Bauteile
- Software
 - Graphische Benutzeroberfläche
 - Bibliotheken

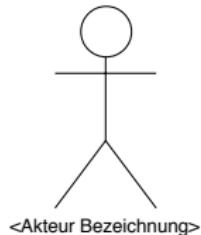


Projektschema

Komponenten

Nachfolgend sind mögliche Komponenten eines Projektschemas aufgelistet:

- Systemgrenzen
- Hardware
 - Microcontroller-Boards
 - Sensoren
 - Bauteile
- Software
 - Graphische Benutzeroberfläche
 - Bibliotheken
- Akteure

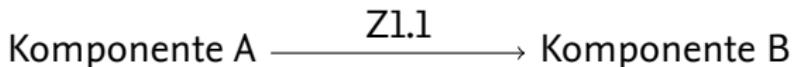


Projektschema

Interaktionen

Pfeilrichtung

- Komponente A steuert Komponente B oder
- Komponente A ist die Quelle der Information



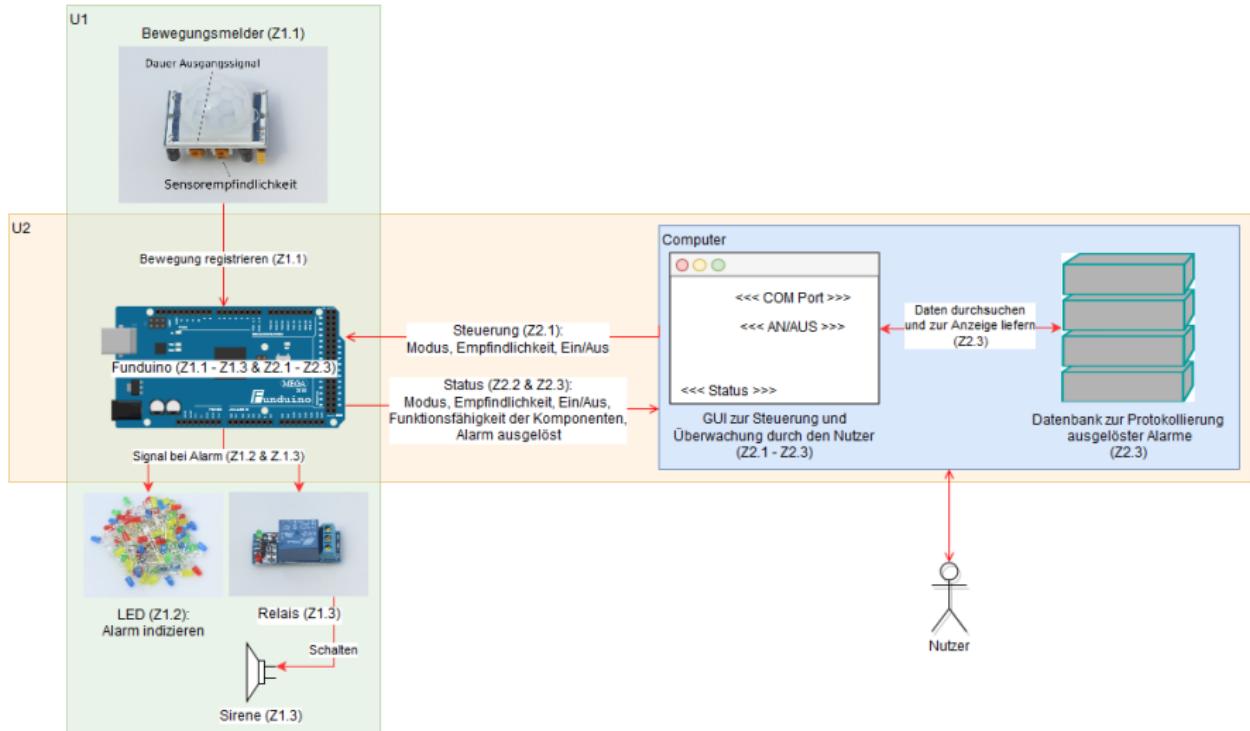
Pfeilbeschriftung

- Protokollbezeichnung
Welche Daten werden übertragen?
- Zielkriterien
Welche Zielkriterien adressiert diese Interaktion?



Projektschema

Beispiel



Projektschema

Anforderungen

Vollständigkeit des Projektschemas zur Abgabe:

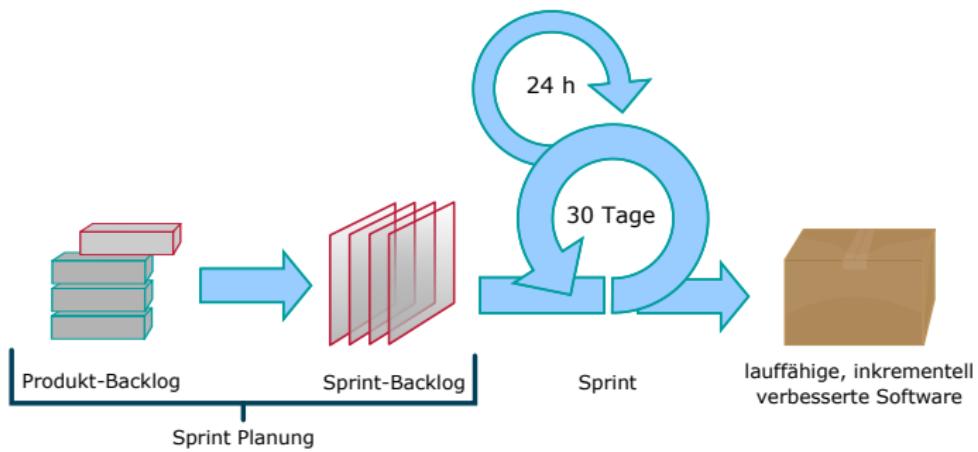
- Alle Komponenten zur Umsetzung der User-Stories und ihrer Funktionalitäten sind im Schema dargestellt und zugeordnet
- Alle Komponenten mit Einfluss auf ein Zielkriterium sind entsprechend markiert
- Jede Komponente ist mit einer Bezeichnung und Aufgabe beschriftet
- Verknüpfungen und Schnittstellen zwischen den Komponenten sind vollständig dokumentiert



Gibt es Fragen oder Anmerkungen zu dem Unterthema
Projektschema?



Sprint



Sprint

Planung

Vorgehen

1. Eine oder mehrere User-Stories auswählen
2. Aus den Zielkriterien Aufgaben generieren
3. Abhängigkeiten unter den Aufgaben identifizieren und dokumentieren
4. Aufgaben einem oder mehreren Mitgliedern zuweisen

Wichtig

- Jede Aufgabe soll eine eindeutige Nummer erhalten
→z.B. A1.2.1 (1. Userstory, 2. Zielkriterium, 1. Aufgabe),
- Formulierung interdisziplinärer Aufgaben möglich (Design, Lötarbeiten, Einkauf ...)



Dokumentation der Sprintplanung

Dokumentation

- Schriftliches Verfassen der Aufgaben in ganzen Sätzen
- Verknüpfung mit Zielkriterium herstellen
- Kontext bereitstellen (Zugehörigkeit zu User-Story / Zielkriterien)
- Darstellung der Abhängigkeiten (in Textform oder als Schema)

Beispielaufgabe

Lfd. Nr.: A1.1.2

Bearbeiter: Paul Frost

Aufgabe: Beispieltext auf Display anzeigen

Abhängigkeiten: A1.1.1



Sprint

Durchführung

Während des Sprints können die Gruppenmitglieder parallel an ihren Aufgaben arbeiten.

In 10 wöchentlichen Sprintmeetings soll für jedes Teammitglied folgendes protokolliert werden:

- Was wurde seit dem letzten Meeting bearbeitet?
- Was wird als nächstes bearbeitet?
- Optional: *Gab es Probleme?*

Wichtig

Ein Sprintmeeting sollte nicht länger als 15 Minuten dauern.



Dokumentation eines Sprint-Meetings

Beispielmeeting

Meeting: 16.04.2019

| | |
|-------------------------|---|
| Teilnehmer: | Paul Frost |
| Bearbeitet: | A1.1.1, A1.1.2 |
| Geplant: | A1.1.2 und A1.1.3 |
| Schwierigkeiten: | Display war defekt und musste getauscht werden. |
| Teilnehmer: | Andreas Dekiert |

...



Sprint

Retrospektive

Vorgehen

1. Aktuellen Projektstatus dokumentieren

- Zwischenstand der User-Stories, Zielkriterien und Aufgaben
Einteilung in: nicht begonnen, in Bearbeitung, abgeschlossen
- Bei den erfüllten Zielkriterien soll die Revisionsnummer angegeben werden

siehe GitHub-Übung

Abweichungen zwischen Zuweisung und Bearbeitung angeben

2. Reflexion der Sprintplanung und des Sprints

- Waren die Aufgaben zu umfangreich/zu klein?
- Sollte die Methodik geändert werden?
- Konnte die Arbeitslast auf alle Mitglieder gleichmäßig verteilt werden?
- ...



Dokumentation des Sprint-Abschlusses

Beispiel

Sprint 1 28.05.2018

| | |
|-----------------------|---|
| Projektstatus: | US1 begonnen - Z1.1 abgeschlossen (5115d7abb7c6... US2 begonnen - Z2.1 begonnen; Z2.2 begonnen; ... Änderungen am Backlog sind nicht erforderlich |
| Aufgaben: | A1.1.1 abgeschlossen A2.1.2 in Bearbeitung ... |
| Bewertung: | Die generierten Aufgaben waren nicht kleinteilig genug und müssten genauer definiert werden. Das Team konnte so nicht parallel an den Aufgaben arbeiten... |



Vorschlag Sprinteinteilung

Sprint 1 23. April

Sprint 2 14. Mai

Sprint 3 18. Juni

Abgabe 12. August 10 Uhr

Wichtig

Sollten mehr als drei Sprints durchgeführt werden. Werden ausschließlich die ersten drei Sprints bewertet.



Gibt es Fragen oder Anmerkungen zu dem Thema
API-Vorgehensmodell?



Abgehakt

Softwareprojektmanagement

Als Teilnehmer soll ich am Ende dieser Übung...

- Methoden kennen, die Softwareentwicklung planbar machen
- klassische & agile Softwareentwicklung abgrenzen können
- mit der Projektmappe beginnen können
- Sprints planen und durchführen können
- das API-Vorgehensmodell kennen



Aufgaben zur Nachbereitung

Aufgabe 1

Erstellte User-Stories im Team besprechen und sicherstellen, dass sie individuelle (abgrenzbare) Funktionalitäten beschreiben.

Aufgabe 2

Zielkriterien im Team besprechen, bewerten und ggf. optimieren.



Aufgaben zur Nachbereitung

Aufgabe 3

1. User-Stories und Zielkriterien hinsichtlich ihrer Umsetzung analysieren. Wie können die User-Stories mit ihren Zielkriterien umgesetzt werden?
2. Erforderliche Projektkomponenten (Softwareteile, Sensoren etc.) bestimmen und Überschneidungen zwischen den User-Stories suchen.
3. Erste Version des Projektschemas zeichnen.

Hinweis

Tipp: MS Visio oder <https://www.draw.io/> können für das Projektschema verwendet werden



Literatur

- [1] T. Thüm, "Vorlesung Software Engineering, Anforderungsanalyse",
- [2] B. B.W., *Software Engineering Economics*. 1981.
- [3] H. Balzert, *Lehrbuch der Software-Technik*. 1998.
- [4] N. Urbach. (), Agiles IT-Projektmanagement, Adresse:
<http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Software-Projektmanagement/agiles-it-projektmanagement>.
- [5] J. Mainusch, *Scrum mit User Stories*. München: Hanser, 2017, Source:
<https://katalog.ub.tu-braunschweig.de/vufind/Record/877054657>.
- [6] B. Gloger, *Scrum Produkte zuverlässig und schnell entwickeln*. München: Hanser, 2016.



Ende

Vielen Dank für eure Aufmerksamkeit!

