



Technische
Universität
Braunschweig

Institut für
Flugführung



Softwareprojektmanagement

Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc.,
10. April 2018

Agenda

03. April Einführung

10. April Softwareprojektmanagement

17. April Entwicklungstools

24. April GitHub

08. Mai Einführung Arduino/Funduino

15. Mai Dateieingabe und -ausgabe

22. Mai Exkursionswoche

29. Mai Dokumentation und Bug-Reporting

05. Juni Einführung von Qt

12. Juni GUI-Erstellung mit Qt

19. Juni Anleitung erstellen

26. Juni Projektarbeit

03. Juli Vorbereitung der Abgabe

10. Juli Abgabe



Lehrziele

Einführung API

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

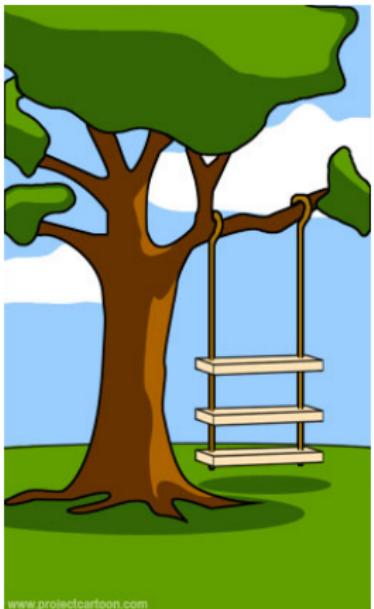


Realität?

Problematik



Wie Softwareprojekte bearbeitet werden...



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

[<http://projectcartoon.com>]

Wie der Kunde es erklärt hat



Technische
Universität
Braunschweig

10. April 2018 | Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc. | Seite 5
Softwareprojektmanagement

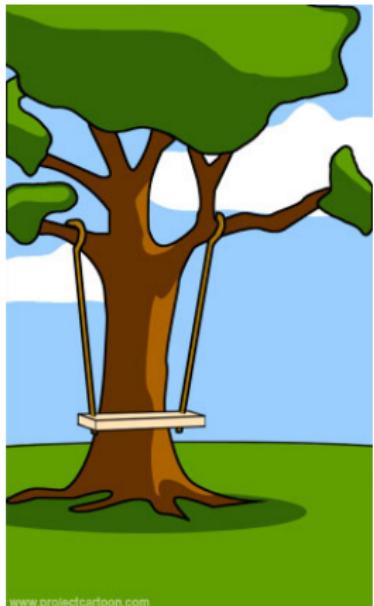
Institut für
Flugföhrung



Wie Softwareprojekte bearbeitet werden...



[<http://projectcartoon.com>]



Wie der Projektleiter es verstanden hat



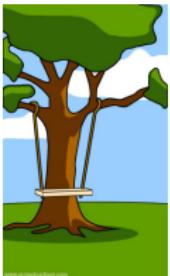
Technische
Universität
Braunschweig

10. April 2018 | 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 Analyst es auffast



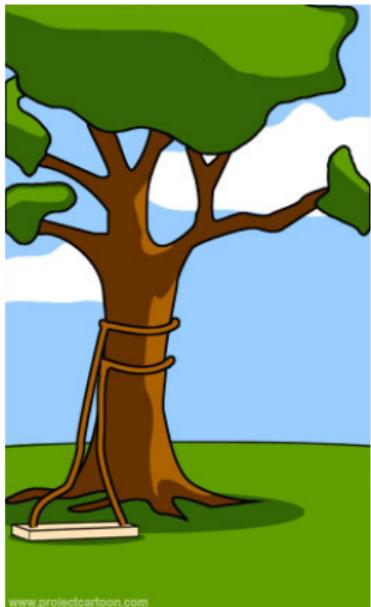
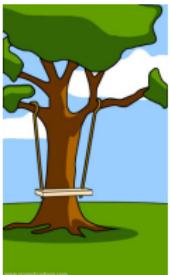
Technische
Universität
Braunschweig

10. April 2018 | Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc. | Seite 5
Softwareprojektmanagement

Institut für
Flugföhrung



Wie Softwareprojekte bearbeitet werden...

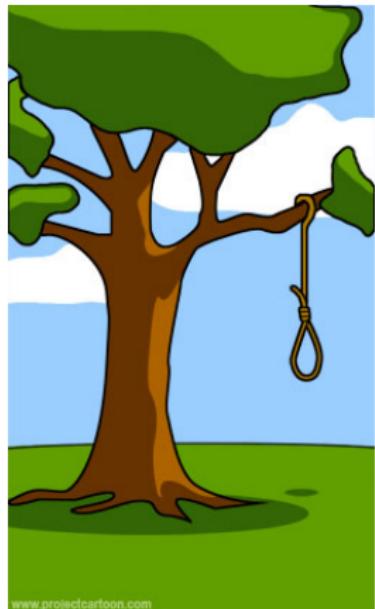
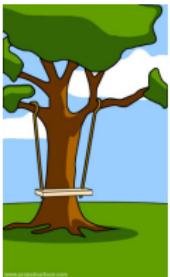


[<http://projectcartoon.com>]

Wie der Programmierer es geschrieben hat



Wie Softwareprojekte bearbeitet werden...

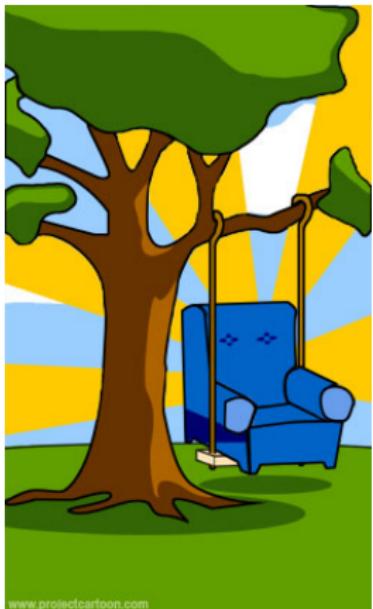


[<http://projectcartoon.com>]

Was die Beta-Tester erhalten



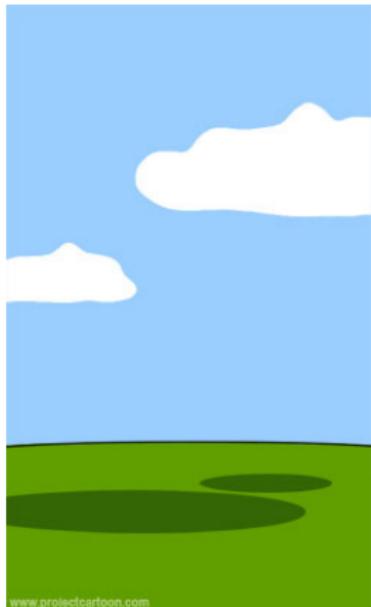
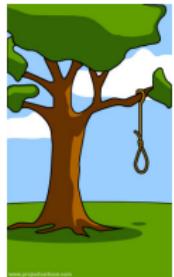
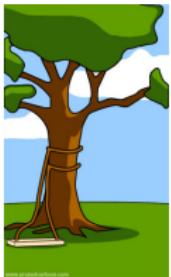
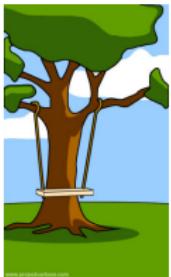
Wie Softwareprojekte bearbeitet werden...



[<http://projectcartoon.com>]

Wie der Wirtschaftsberater es verkauft

Wie Softwareprojekte bearbeitet werden...

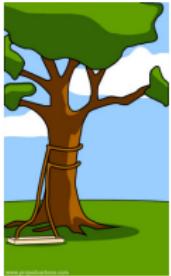
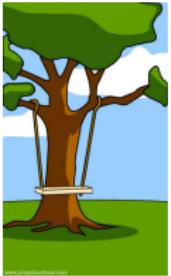


[<http://projectcartoon.com>]

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>]

Gibt es Fragen oder Anmerkungen zu dem Thema
Einführung Projektplanung?



Vorgehens- modelle



Simples
Modell

Wasserfall-
modell

Spiralmodell



Ein simples Modell

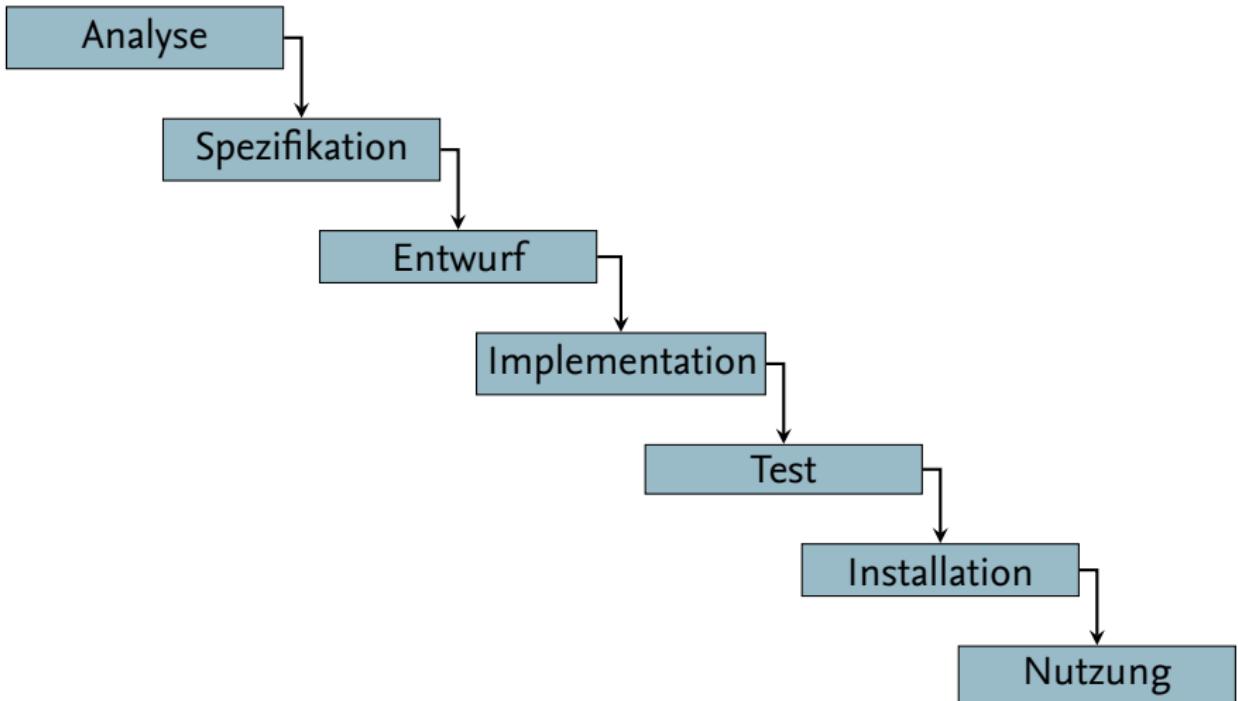
1. Quellcode schreiben
2. Fehler beheben

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

Einführung API

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



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
- Änderungen an den Anforderungen können leichter im Entwicklungsprozess berücksichtigt werden
- Entwicklungsphasen sollen einsatzfähige Softwareinkremente hervorbringen
- 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
- Nicht für alle sicherheitskritischen Applikationen geeignet



Was ist Scrum? [4]

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

Für API: Verwendung von User-Stories für die Definition von Kundenanforderungen



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 [4]

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 ^[4]

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 [5]

Entwicklungsteam (3-9 Personen):

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

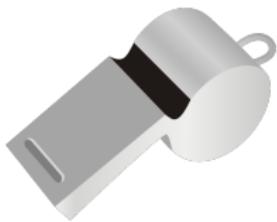


Scrum

Akteure: Scrum Master [5]

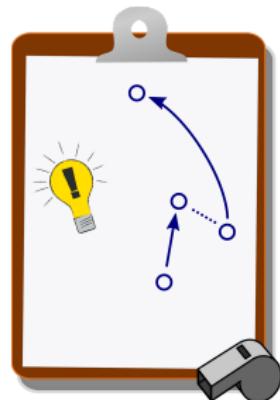
Scrum Master:

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

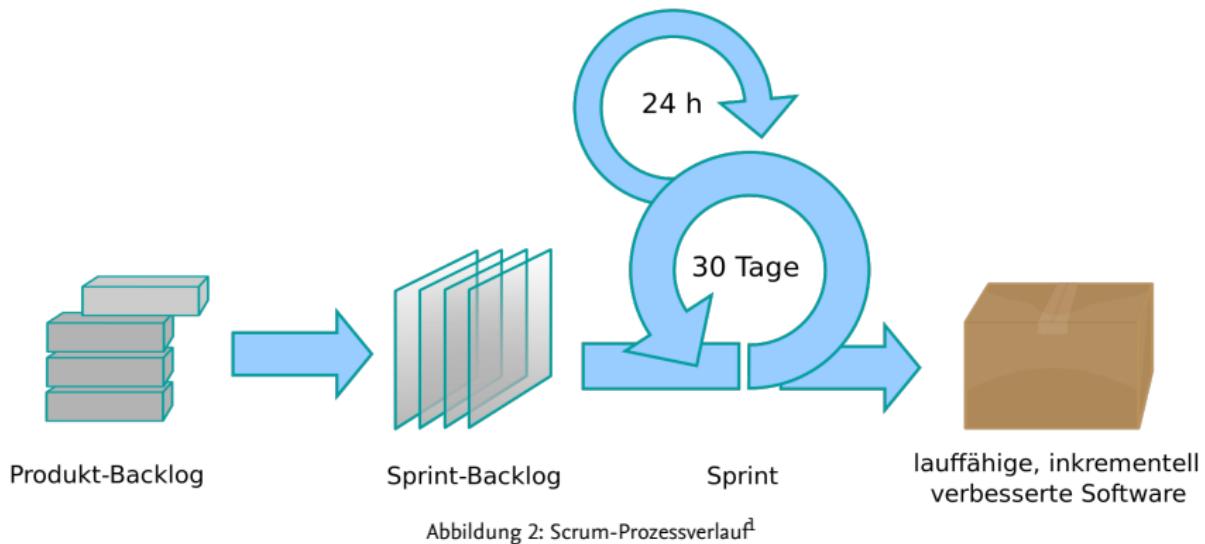


Product Owner:

- 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



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>

Sprint

Der Sprint ist ein endlicher Zeitraum, in dem das Entwicklungsteam geschützt an ausgewählten User-Stories arbeitet.

Ein Sprint beinhaltet

- die Sprintplanung
- die Sprintdurchführung
- die Retrospektive

Das Ergebnis eines Sprints soll ein Softwareinkrement sein, dass von dem 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 User-Stories 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?



Burndown Charts

Der Fortschritt eines Sprints kann nach jedem Daily-Scrum-Meeting in einem Burndown-Diagramm festgehalten werden.

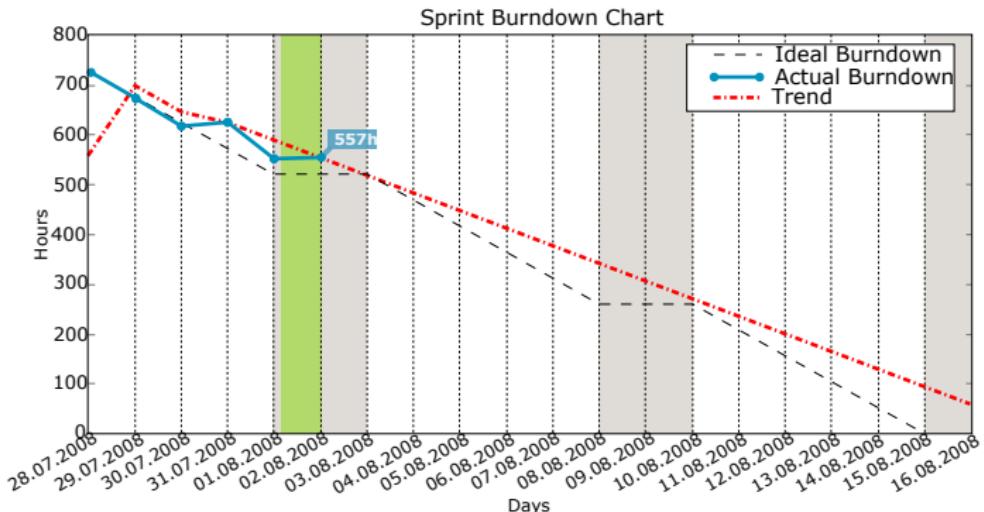


Abbildung 4: Burndown Chart

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

Einführung API

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

[API]
2018

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 herabgesetzt
 - Vier 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.



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.



Beispiel Projekt

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.



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 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

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

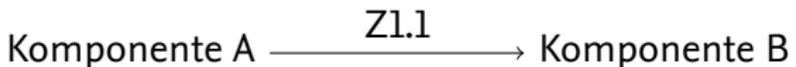


Projektschema

Interaktionen

Pfeilrichtung

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



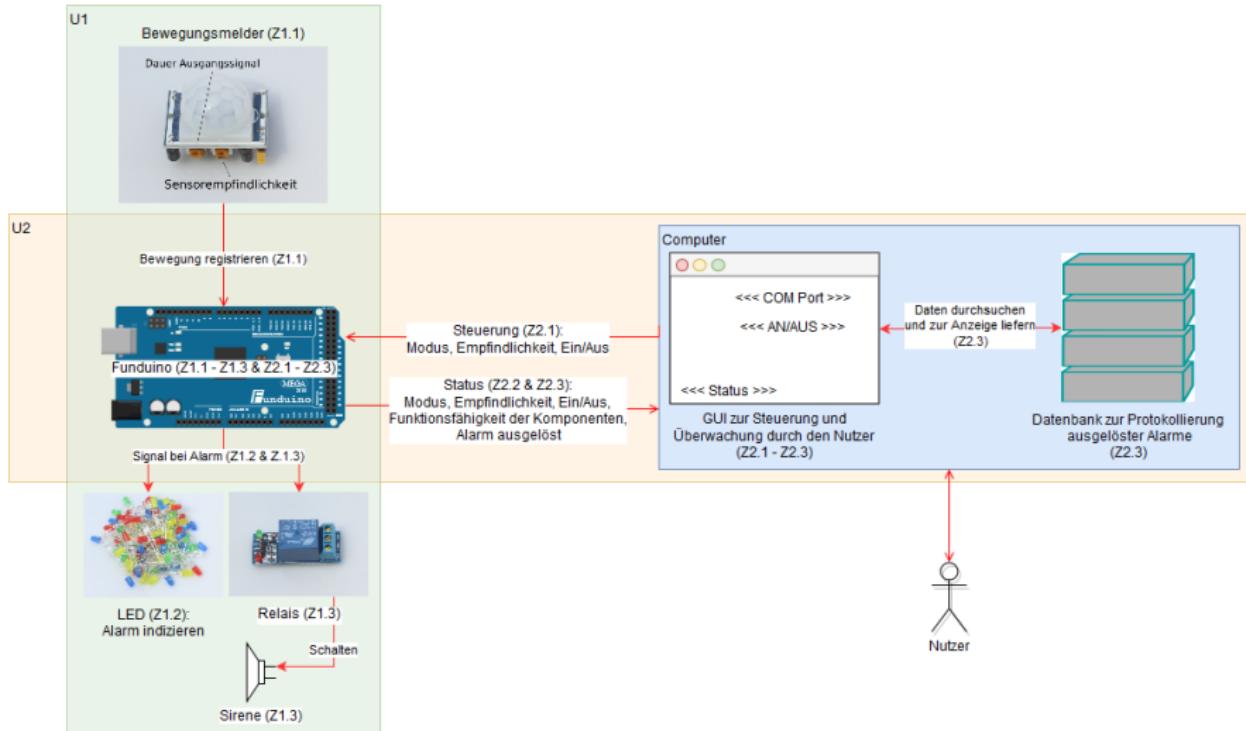
Pfeilbeschriftung

- 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

Planung

Es sollen vier Sprints von etwa drei Wochen Länge durchgeführt werden.

→ Der erste Sprint beginnt am 17. April

In der Sprintplanung zu Beginn jedes Sprints soll jedem Gruppenmitglied mindestens ein Zielkriterium als Aufgabe zur Umsetzung zugewiesen werden.

Zur Planung des Sprints ist ein gesondertes Treffen mit Anwesenheit aller Gruppenmitglieder vorteilhaft.



Sprint

Durchführung

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

In 10 wöchentlichen Sprintmeetings soll Folgendes protokolliert werden:

- Durch die einzelnen Teammitglieder bearbeitete Aufgaben und deren Zwischenstände
- Welche Aufgaben und Schritte sind für die Folgewoche geplant?

Für die Sprintmeetings werden während der API-Übungen wöchentlich 15 Minuten zur Verfügung stehen.



Sprint

Retrospektive

Jeder Sprint endet mit einem Abschlussmeeting. Dort wird protokolliert:

- Aktueller Projektstatus und dazugehörige Revisionsnummer
 - Welche Zielkriterien werden bereits erfüllt? ...
- Status der geplanten Aufgaben
 - Konnten die Aufgaben abgeschlossen werden? Gab es Probleme? ...
- Reflexion der Sprintplanung
 - Waren die geplanten Aufgaben im Sprintzeitraum zu schaffen? Kann die Planung verbessert werden? ...

Es ist vorteilhaft, das Abschlussmeeting und das Planungsmeeting des folgenden Sprints im Anschluss aneinander abzuhalten.



Vorschlag Sprinteinteilung

Sprint 1 17. April bis 7. Mai

Sprint 2 8. Mai bis 28. Mai

Sprint 3 29. Mai bis 18. Juni

Sprint 4 19. Juni bis 4. Juli

Abgabe 10. Juli

Zur Vorbereitung der Abgabe ist es sinnvoll, den letzten Sprint rechtzeitig zu beenden.



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



Abgehakt

Einführung API

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 bis nächste Woche

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 bis nächste Woche

Aufgabe 3

1. User-Stories und Zielkriterien hinsichtlich ihrer Umsetzung analysieren.
2. Erforderliche Projektkomponenten (Softwareteile, Sensoren etc.) bestimmen und Überschneidungen zwischen den User-Stories suchen.
3. Erste Version des Projektschemas zeichnen.

Tipp: MS Visio oder <https://www.draw.io/> sind gut geeignet.



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] J. Mainusch, *Scrum mit User Stories*. München: Hanser, 2017, Source: <https://katalog.ub.tu-braunschweig.de/vufind/Record/877054657>.
- [5] B. Gloger, *Scrum Produkte zuverlässig und schnell entwickeln*. München: Hanser, 2016.



Ende

Vielen Dank für eure Aufmerksamkeit!

