Fachhochschule Nordwestschweiz
Hochschule für Architektur, Bau und Geomatik

Agiles Projektmanagement

Einführung

Was bedeutet “agile”?

- Iterativer Ansatz
- Unterstützt das Projektmanagement eines Teams
- Verstärkt die Reaktionsfähigkeit über den Verlauf eines Projekts
- Erlaubt Kurskorrekturen um am Ende den Bedürfnissen des Kunden oder Anwenders gerecht zu werden

Charakteristiken von agilem Projektmanagment

- Schwerpunkte
  - Adaptive Planung
  - Evolutionsbasierte Softwareentwicklung (Iterationen)
  - Early delivery
  - Fortlaufende Verbesserungen
  - Reaktionsfähigkeit auf Unvorhergesehenes

Das agile Manifest (2001)

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

agilemanifesto.org

Das agile Manifest (2001)

Kernaussagen

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
- Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
- Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
- Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist das wichtigste Fortschrittsmaß.
- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
- In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

Agile Softwareentwicklung

- Iterativer Ansatz zur Softwareentwicklung der konsistent ist mit den agilen Prinzipien
- Fokus auf Flexibilität, Interaktion und Transparenz innerhalb des Teams
- Selbstorganisation
- Interdisziplinarität

Primärziel

Build what **is** needed, not what **was** planned

Agiles Projetmanagement in der Organisation

Conways Law

“If you ask an organisation with four separate teams to build a compiler…

….you will to get a 4-pass compiler!”

Melvin Conway 1968

Grundidee der iterativen Planung

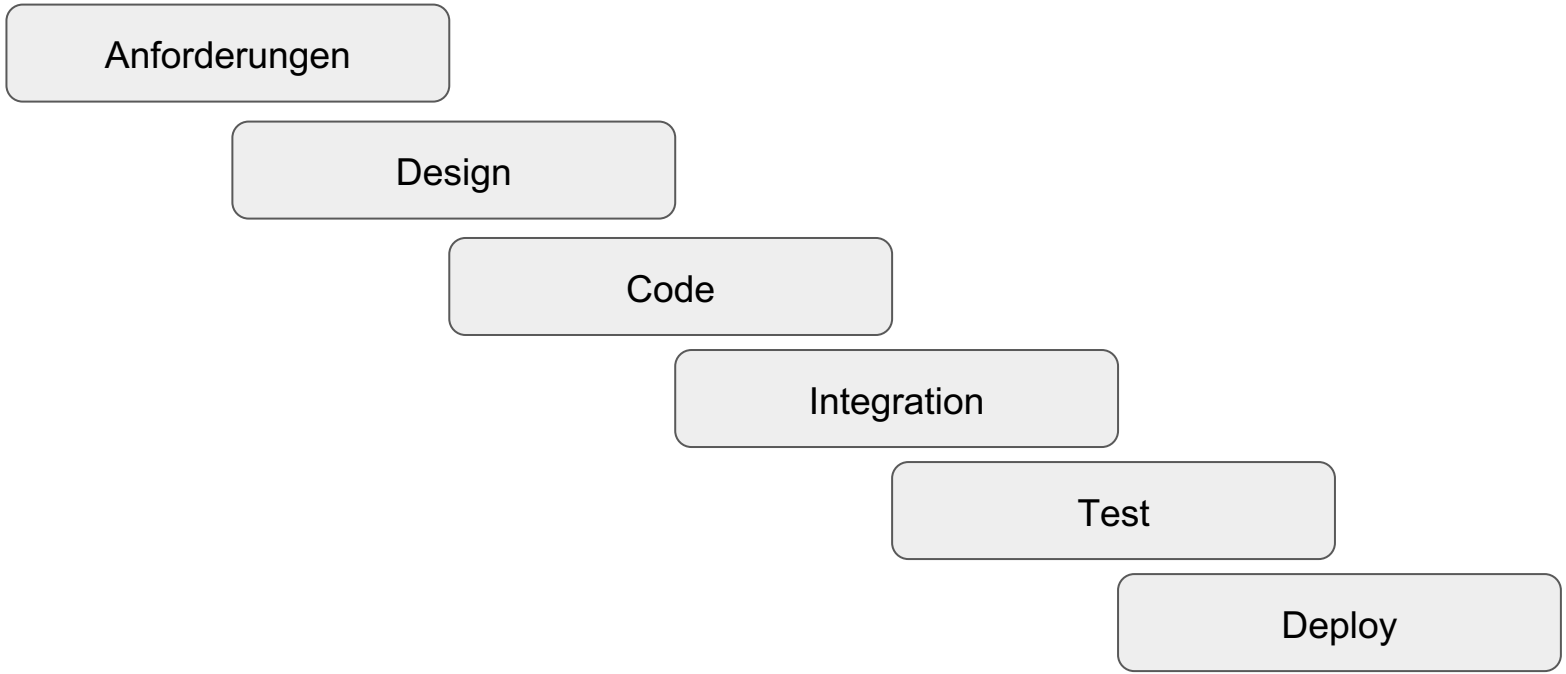
- Man entscheidet nicht Dinge zu einem Zeitpunkt wo man noch gar nicht das wesentliche Wissen über alle Details hat z.B. zu Projektbeginn
- Es wird das geplant worüber man gerade am meisten weiss um verlässliche Schätzungen zu machen
- Anpassung der Planung sobald neue Erkenntnisse vorliegen
- Zeiträume für die Planung müssen dementsprechend überschaubar gewählt werden

Gruppenübung

Spaghetti - Challenge

Methodologien

Methodologien - Wasserfall

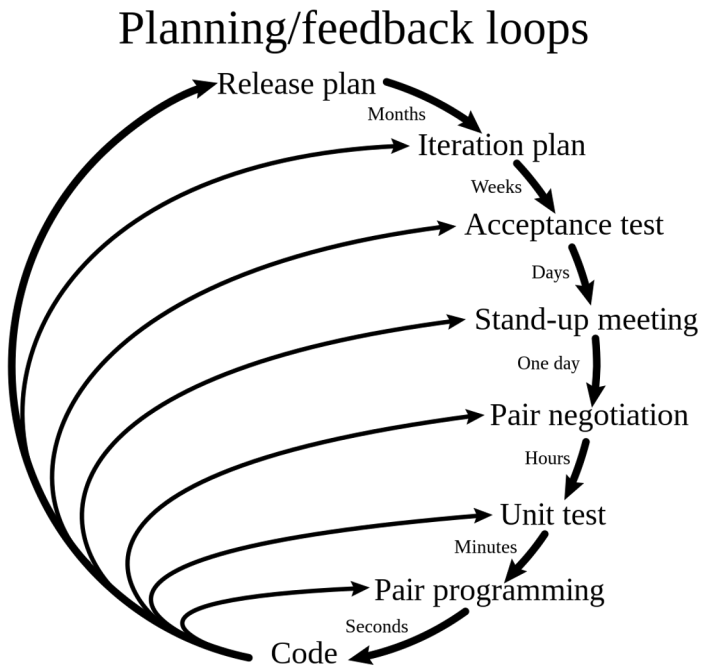


Methodologien - Wasserfall

- Kein Handhabe für nachträgliche Anpassungen von Anforderungen
- Keine Wahrnehmung über den momentanen Stand und die Qualität der Umsetzung
- Jeder Schritt endet wenn der nächste beginnt
- Fehler die spät im Projekt entdeckt werden sind teuer
- Lange Zeiträume bis ein Projekt in Test oder Produktion gehen kann. Keine Zwischen Reviews
- Verschiedene Teams arbeiten isoliert voneinander (Design, Code oder Test)

Methodologien - Extreme Programming (XP)

- Iterativer Ansatz
- Kleine Inkremente
- Prinzipien
  - Einfachheit
  - Kommunikation
  - Feedback
  - Respekt
- Ursprung der agilen Methode
- Ursprung: 1996 Kent Beck



Methodologien - Kanban

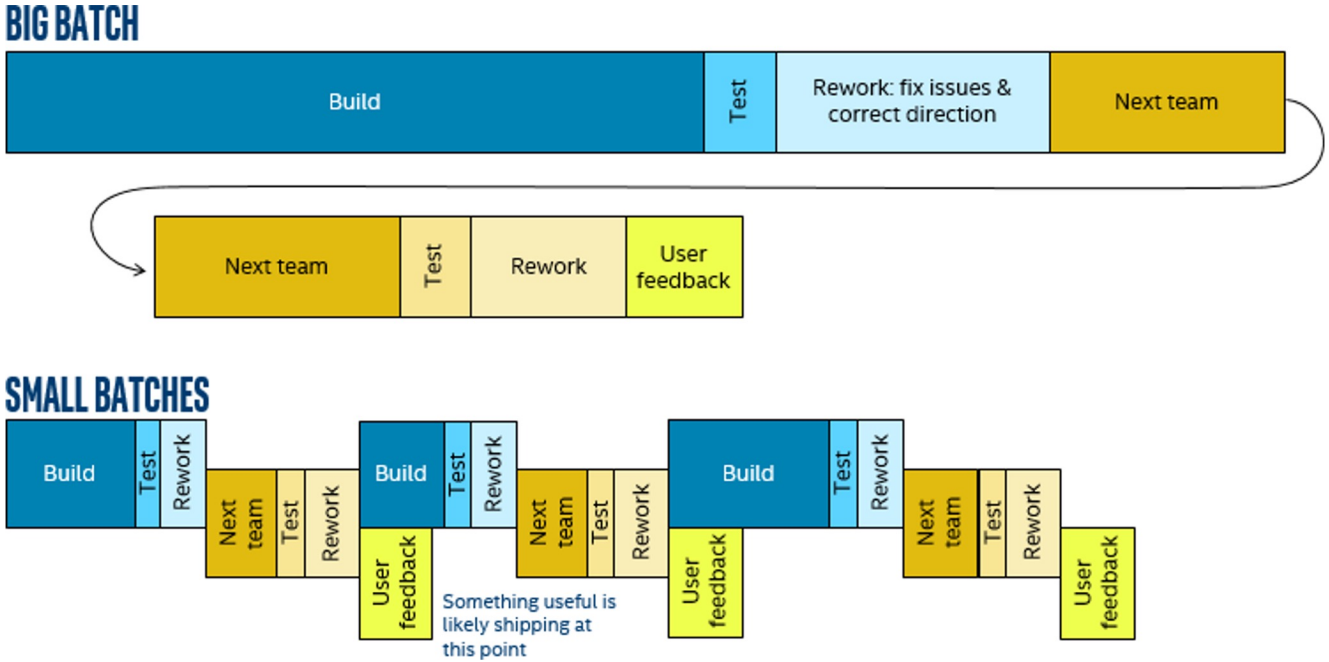
- 1970 aus Japan (aus der Fertigungsindustrie)
- Ursprünglich wurde es verwendet um Materialnachschub in den Produktionsketten zu verwalten mittels Anweisungskarten
- Prinzipien
  - Visualisierung des Workflows
  - Einschränken der momentan bearbeiteten Arbeitspaketen (Work in progress)
  - Kontrollieren und verbessern der Ablaufprozesse
  - Prozessrichtlinien mit expliziten Zieldefinitionen (Definition of done)
  - Fortlaufende Verbesserung (Continuous improvement)
  - Fortlaufende Lieferung (Continuous delivery)

Was ist agiles Arbeiten?

Agiles Arbeiten

- Arbeit in kleinteiligen Paketen (Small Batches)
- Minimum Viable Product (MVP)
- Verwerfen oder Behalten (Pivot or Persevere)
- Behaviour Driven Development (BDD)
- Test Driven Development (TDD)
- Pair Programming (PP)

Small Batches



intel.com

Small Batches

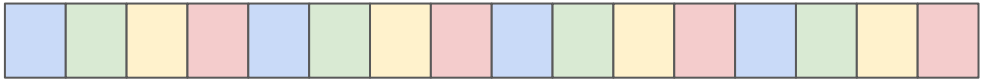
- Fallbeispiel: Versand von Broschüren oder Einladungsdossiers
- Die Arbeit gliedert sich in 4 Arbeitsschritte:
  - Falten, einfüllen, verschliessen, adressieren
- Wenn dies nun ca 50 mal gemacht werden muss und wir pro Arbeitsschritt 6 Sekunden pro Exemplar

Falten	Einfüllen	Verschliessen	Adressieren

- Wie lange dauert es, bis ich das erste Exemplar vollendet habe?
- Welche Probleme könnten hier auftauchen?

Small Batches

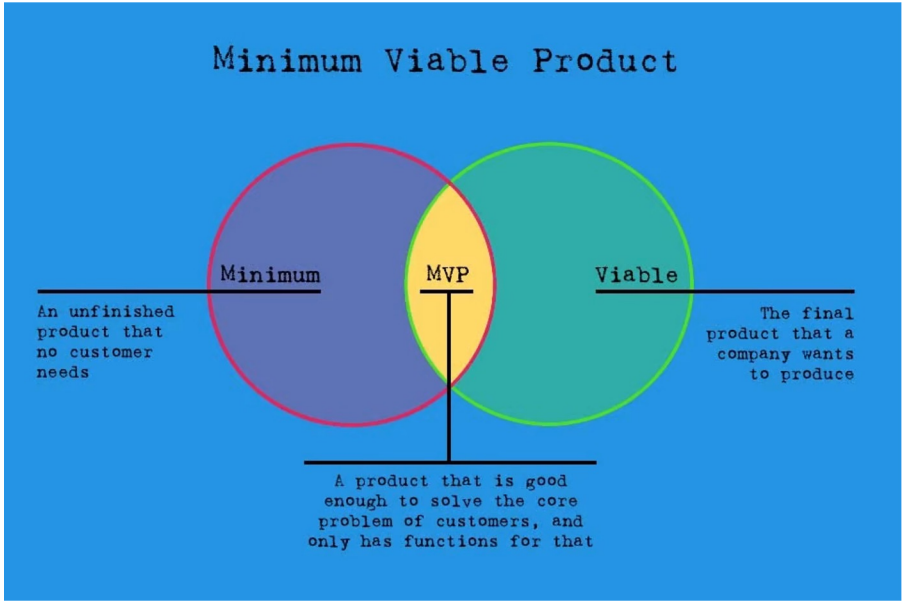
- Lösungsansatz: Verwendung des Single Piece Flow (SPF)



- Wie lange dauert es, bis ich das erste Exemplar vollendet habe?

Minimum Viable Product

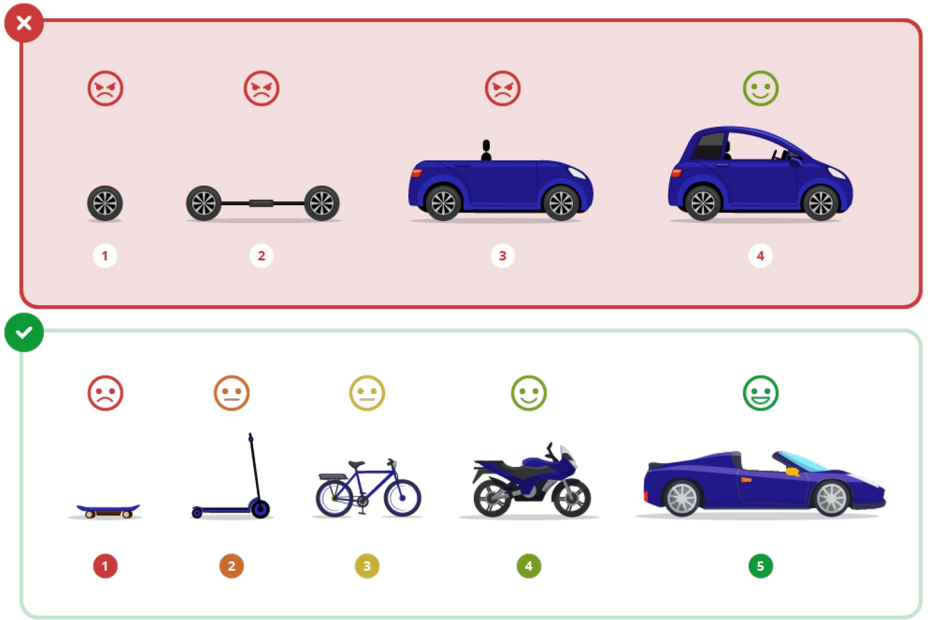
- Erste minimal funktionsfähige Iteration eines Produkts
- Testen einer Markthypothese
- Ausloten des brauchbaren Nutzens der Lösung



Minimum Viable Product

“The minimum viable product is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.”

Eric Ries



https://mlsdev.com/

Behaviour Driven Development

- Grundlage bietet das Verhalten eines Zielsystems von **aussen nach innen betrachtet**
- Definiert einen Test
- Wird meistens gegen ein Benutzerinterface durchgeführt um überprüft werden zu können
- Besteht aus einer simplen aber flexiblen Syntax

Behaviour Driven Development

Formulierung in sogenannten Stories (Geschichten)

- Rolle (Role)
- Funktionalität (Functionality)
- Geschäftsergebnis (Business Value)

Syntax: Wenn eine Menge von Vorbedingungen existiert und ein bestimmtes Ereignis eintritt, dann soll folgende Beobachtung gemacht werden können.

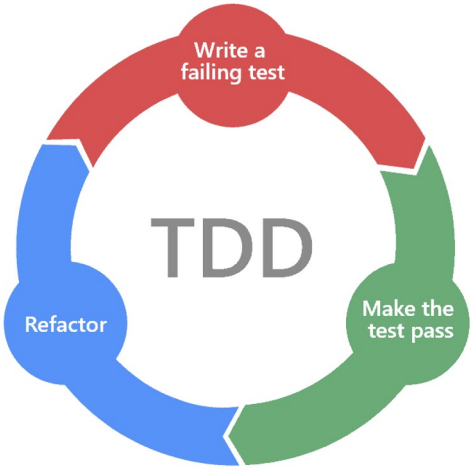
Beispiel: Als Benutzer <role> möchte ich eine Login-Eingabe <functionality> präsentiert bekommen, um mich im System anmelden <business value> zu können.

Test Driven Development (TDD)

- Testet Funktionen einen Systems von **innen nach aussen**
- Es wird zuerst ein Test formuliert und danach die entsprechende Funktionalität definiert
- Man schreibt den Testcode für Programmcode, den man danach bekommen möchte

Test Driven Development (TDD)

- Workflow für TDD
  - Schreiben des Tests und sehen wie er fehlschlägt (Write a failing test)
    - Beispiel: (testMyHelloWorldFunction() => Expected Result (Text: “Hello World” appears on Screen)
  - Schreiben von genug Programmcode um dafür zu sorgen, dass ein Test erfolgreich zurückkommt (Make the test pass)
  - Verbessern des Programmcodes falls ein Test fehlschlägt (Refactor)



Der grosse Vorteil dieses Ansatzes ist es, dass er automatisiert werden kann.

Pair Programming (PP)

- Ablauf
  - Zwei Entwickler arbeiten zusammen an einer Arbeitsstation
  - Ein Entwickler schreibt den Code
  - Der andere Entwickler überprüft und gibt Feedback
  - Regelmässiger Wechsel der Rollen (z.B. alle 20 min)
- Nutzen: Steigerung der Code Qualität
- Kritik des Chefs: “Warum soll ich zwei Leute für die Arbeit einer Person bezahlen?”

Writing code is the easy part <> Debugging and maintaining is the hard part

Paper Plane Challenge