Extreme Programming (XP)

WANN: bei der agilen Implementierung

WER: Manager, Entwickler und Kunde in einem Team mit weniger als zwölf Personen

DAUER: aufgabenabhängig

Bei XP handelt es sich um eine von Kent Beck entwickelte agile Methode der Softwareentwicklung, die in einem kleinen bis mittelgroßen Team mit bis zu zwölf Personen einsetzbar ist. Sie eignet sich in einem Kontext mit vagen und sich schnell verändernden Anforderungen besonders gut. Bei der Methode werden anerkannte Prinzipien des Programmierens in ihrer Extremform angewendet, beispielsweise indem Prozesse, wie das Testen von Software, nicht nur regelmäßig, sondern kontinuierlich durchgeführt werden. Das Rahmenwerk von XP besteht aus fünf Werten, vierzehn Prinzipien und dreizehn Praktiken. Die genaue Anzahl kann bei verschiedenen Quellen leicht variieren, da die Entwickler der Methode diese im Laufe der Zeit überarbeitet haben. Um hier einen Einblick und Einstieg in die Methode zu finden, werden lediglich die Werte und ausgewählte Praktiken vorgestellt. Details zu dem kompletten XP-Rahmenwerk können in der weiterführenden Literatur nachgelesen werden.

DREIZEHN PRIMÄRPRAKTIKEN VON XP

1. Das Team sitzt **räumlich zusammen**.
2. Das Team muss **alle Fähigkeiten** vereinen, die zur Erfüllung der gestellten Aufgabe notwendig sind.
3. Die **Arbeitsumgebung** soll den Bedürfnissen des Teams angepasst sein und den aktuellen Stand des Projekts widerspiegeln.
4. Das Team sollte zu „**Energized Work**“ befähigt werden, also zu einer engagierten Arbeit voller Energie.
5. Das Programmieren findet **in Paaren** statt.
6. Anforderungen werden in **User Stories** formuliert.
7. Es wird nahegelegt, im **wöchentlichen Zyklus** zu arbeiten und Freitag als das Ende eines Arbeitsabschnittes zu betrachten.
8. Es wird nahegelegt, im **vierteljährlichen Zyklus** Releases durchzuplanen und durchzuführen.
9. Teammitglieder sollten den **Freiraum** haben, eigene Ideen entwickeln zu können.
10. Das Erstellen eines integrierten Versionsstandes des Software-Systems („Build“) inklusive aller automatisierten Tests sollte nicht länger als zehn Minuten dauern („**Ten-Minute Build**“). Dies minimiert Kosten.
11. Neuer Code wird **kontinuierlich integriert**.
12. Die Testfälle werden entwickelt, bevor die Komponente implementiert wird („**Test First Entwicklung**“).
13. Eine Funktionalität soll nicht früher als nötig eingebaut werden („**Inkrementelles Design**“).

XP EIGNET SICH BESONDERS GUT, WENN…

- … sich die Spezifikationen schnell verändern ☑
- …Kund*innen keine endgültige Vorstellung vom Funktionsumfang haben ☑
- … Kund*innen bereit sind, sich in hohem Maße in das Projekt einzubringen ☑
- … die Projektgruppe inkl. Manager*innen, Programmierer*innen und Kund*innen eine Größe von weniger als zwölf Personen hat ☑
- …die zu erstellende Software automatisiertes Testen erlaubt ☑

VORTEILE VON XP

- Reduktion von Projektrisiken bezüglich der Einhaltung von Termin-, Kosten- und Qualitätszielen
- Bessere Reaktionsmöglichkeiten auf sich verändernde Anforderungen
- Erhöhung der Produktivität über die gesamte Projektdauer hinweg

FÜNF WERTE ALS BASIS VON XP

1. **Kommunikation:** Intensive Kommunikation teamintern sowie mit den Kund*innen
2. **Einfachheit:** Suche nach der einfachsten Lösung bezüglich Design einer Software („Simple Design“) und Gestaltung der Projektabläufe
3. **Feedback:** Feedback über das Programmieren in Paaren von Teammitgliedern, über das Entwickeln von Testfällen von dem System selbst sowie über eine nahe Kundenbeziehung von den Kund*innen
4. **Mut:** Bestehende Ergebnisse nochmals neu aufrollen
5. **Respekt:** Respekt innerhalb des Entwicklungsteams und in der Kundenbeziehung

Mittelstand-Digital

Gefördert durch:
Bundesministerium für Wirtschaft und Energie
aufgrund eines Beschlusses des Deutschen Bundestages

Mittelstand 4.0
Kompetenzzentrum
Usability

Probieren Sie Praktiken von XP selbst aus!

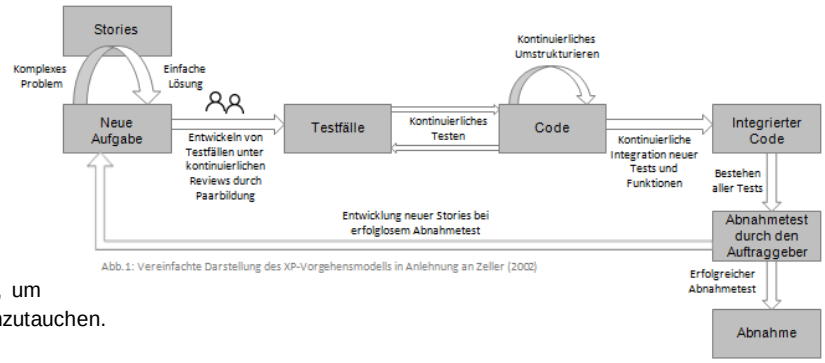

Abb.1: Vereinfachte Darstellung des XP-Vorgehensmodells in Anlehnung an Zeller (2002)

IHRE AUFGABE

Nutzen Sie die vorgestellten Praktiken als Inspiration, um tiefer in die agile Software-Entwicklung mittels XP einzutauchen.

PLANUNG

ANFORDERUNGEN IN USER STORIES

Getrieben wird die Entwicklung durch User Stories. In dieser Form werden die Anforderungen definiert. Jede Story beschreibt dabei einen konkreten Vorfall, der bei der Nutzung des Produktes auftritt. Als Regel gibt Kent Beck vor, dass User Stories so definiert werden sollten, dass ihre Bearbeitung etwa eine Woche dauert. Jede Story wird zur weiteren Bearbeitung direkt in einen Testfall umgewandelt.

RELEASE-PLANUNG

Die Release-Planung wird auch als Planungsspiel („Planning Game“) bezeichnet und basiert auf den erstellten User Stories. Ein Release-Plan sollte aus ungefähr 80 ± 20 User Stories bestehen. Die Planung besteht in der zeitlichen Anordnung dieser User Stories. Dabei werden Umfang, zur Verfügung stehenden Ressourcen, Marktanforderungen und Risiko in die Überlegungen einbezogen.

ITERATIONSPLANUNG

Eine Iteration dauert ungefähr eine Woche und wird in Form eines Iteration Planning Meetings gemeinsam mit den Kund*innen geplant. Hierbei werden User Stories priorisiert und in konkrete Aufgaben herunter gebrochen, die vom Entwicklungsteam bearbeitet werden sollen. Die Programmierer*innen machen für ihre Aufgabe bzw. ihre ausgewählte User Story eine Detailplanung, welche der Aufwandsschätzung und später der Messung des Projektfortschritts dienen kann.

DURCHFÜHRUNG

KONTINUIERLICHE REVIEWS

Das Programmieren findet in Paaren statt. Denn das stetige Gegenlesen steigert die Qualität und macht zusätzlich auch mehr Spaß. Dabei gucken zwei Programmierer*innen auf einen Monitor und es herrscht ein ständiger Dialog. Zudem kommt es zu keiner Rollenspezialisierung, sondern zu einem häufigen Tauschen der Aufgaben des Programmierens und Gegenlesens.

KONTINUIERLICHE TESTS

Die Testfälle werden entwickelt, bevor die Komponente implementiert wird. Ein Test beschreibt dabei die Funktion eines Moduls aus exemplarischer Sicht. Die Tests laufen hinterher automatisch ab, sodass selbst große Änderungen des Codes validiert werden. Außerdem ist die Funktionsfähigkeit des Systems so jederzeit sichergestellt.

KONTINUIERLICHE UMSTRUKTURIERUNG

Unter dem Refaktorisieren („Refactoring“) ist die Umstrukturierung des Designs oder Codes zu verstehen. Dies erfordert Mut, da bereits lauffähige Software umgeschrieben wird. Langfristig führt das klarere Design zu besseren Resultaten in den Software-Tests.

KONTINUIERLICHE INTEGRATION

Mehrmals täglich wird integriert und getestet. Dies soll bei XP durch die Programmier-Paare selbst erfolgen. Damit nur ein Paar zu einem Zeitpunkt integriert, erhält es einen Token. Bei diesem Token kann es sich beispielsweise um einen freien Integrationsrechner handeln.

KRITIK AN XP

- Die Beschreibung des Systems findet in Form von Stories statt. Dabei können in manchen Fällen Details unberücksichtigt bleiben.
- Das Vorgehensmodell trifft keine Aussagen bezüglich der Integration von Qualitätsstandards.
- Das Vorgehensmodell beinhaltet (abgesehen von der Anzahl abgeschlossener Stories) keine Metriken.

→ Eine Ergänzung des Konzeptes durch andere Vorgehensmodelle ist möglich.

WEITERFÜHRENDE INFOS

Beck K, Andres C (2004): *Extreme Programming Explained – Embrace Change*. 2nd Ed. Boston. MA: Addison-Wesley.

Hanser, Eckhart (2010): *Agile Prozesse: Von XP über Scrum bis MAP*. Heidelberg: Springer.

Zeller, Andreas (2002): *Extreme Programming. Perspektiven der Informatik*. Universität des Saarlandes. Saarbrücken, 28.10.2002. Online verfügbar unter https://www.st.cs.uni-saarland.de/edu/perspektiven/zeller.pdf, zuletzt geprüft am 26.03.2020.

Mittelstand-Digital

www.kompetenzzentrum-usability.digital

Gefördert durch:
Bundesministerium für Wirtschaft und Energie
aufgrund eines Beschlusses des Deutschen Bundestages