Feature-Entwicklung und interne Qualitätsarbeiten verzahnen

Bereich
Realisierung

Aktivität
Softwareinkrement realisieren

schnell durchführbar ★★☆
einfach durchführbar ★★☆
agil einsetzbar ★★☆

Ziele

- Mittel- und langfristige Bewahrung interner Produktqualität
- Vereinheitlichung von Konzepten
- Verbesserung bestehender Konzepte

Motivation/Problemstellung

Wird die Verteilung der Entwicklungsressourcen rein aus Feature-Sicht getrieben, so leidet häufig die interne Qualität der Software darunter. Die komplette Entwicklungskapazität wird dabei in die Erstellung von Features investiert, die direkt sichtbaren Kundennutzen generieren. Prinzipiell ist es wünschenswert den Fokus auf den beim Kunden wahrgenommenen Wert zu legen. Jedoch führt eine zu starke Konzentration auf die kurzfristige Generierung dieser Werte mittel- bis langfristig zu einer Herabsetzung der internen Qualitäten der Software. Dies schlägt sich vor allem in schlechter Wartbarkeit und damit in geringerer Flexibilität nieder. Sichtbar werden diese Effekte in einer geringeren Produktivität des Entwicklungsteams. Für die Realisierung neuer Features wird ein immer größerer Aufwand benötigt, da die Systemarchitektur mehr und mehr degeneriert. Es werden größere Refactorings in jeder Iteration notwendig um auf die neuen Anforderungen zu reagieren, so dass immer weniger Aufwand zur eigentlichen Realisierung der neuen Features zur Verfügung steht. Alleine durch Refactorings können die Probleme jedoch mittel- und langfristig nicht gelöst werden. Nur durch eine bessere Ausrichtung der Architektur auf die Anforderungen, auch auf zukünftige, kann hier Abhilfe schaffen.

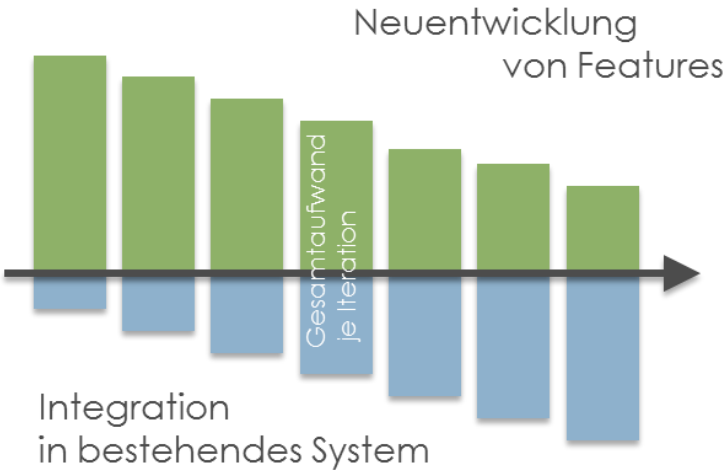

Abbildung 1: Reduktion der verfügbaren Entwicklungskapazität über die Zeit

Kurzbeschreibung

Die Kernidee dieser Best Practice ist es, nicht den kompletten Entwicklungs-Aufwand in die Realisierung von Features zu stecken, sondern einen gewissen Teil auch zur Bewahrung der internen Produktqualität zu investieren. Die genaue Verteilung muss das Team einmalig festlegen.

Während der Entwicklung sammelt das Team Indikatoren für Qualitäts-Defizite, die bei der Realisierung eines Features aufgefallen sind.

Bei der Planung der Aufgaben für die nächste Iteration (hier als Iteration n bezeichnet) werden die gefundenen Indikatoren für Qualitäts-Defizite im Team analysiert und der durch das Defizit potentiell entstehende Schaden bewertet.

Aus den hoch priorisierten Indikatoren für Qualitäts-Defizite werden Aufgaben abgeleitet, die dazu dienen, Konzepte zu erarbeiten, die diese Defizite beheben. Das können zum einen neue Konzepte sein, die ein Konzept ersetzen, das in einer bestehenden Realisierung verwendet wurde, aber dessen Qualität als nicht mehr ausreichend erachtet wird. Trotz sorgfältiger Arbeit gibt es immer wieder solche Stellen im Produkt, die entweder unter Zeitdruck entstehen mussten, oder die im Lichte neuer Erkenntnisse als nicht mehr geeignet erachtet werden. Weiterhin kann auch die Vereinheitlichung von mehreren ähnlichen Konzepten angestrebt werden.

In dieser Iteration n werden im Team diese geplanten Konzeptionierungsaufgaben erledigt. Das Ergebnis daraus ist eine Menge an Realisierungsaufgaben, die der Umsetzung von Lösungsansätzen der gesammelten Qualitäts-Defizite dient.

In der nächsten Iteration (hier als Iteration n+1 bezeichnet) liegen damit für die hoch priorisierten Indikatoren für Qualitäts-Defizite sowohl Lösungskonzepte als auch konkrete Aufgaben vor.

Während der Planung der nächsten Iteration n+1 wird nun das interne Qualitätsbudget zu einem großen Teil mit der Realisierung dieser Aufgaben belegt, im anderen Teil werden Lösungen für neue Refactoring-Wünsche konzipiert, die dann wieder in Iteration n+2 umgesetzt werden. Die genaue Verteilung muss das Team einmalig festlegen. Somit werden in jeder Iteration sowohl neue Konzepte erstellt als auch umgesetzt.

Insgesamt findet somit kontinuierlich eine Investition in die interne Qualität und die Infrastruktur innerhalb der Software statt.

Beispielsweise könnte ein Indikator für ein internes Qualitätsproblem sein, dass während der Umsetzung eines neuen Features zwar eine Ähnlichkeit zur Realisierung mehrerer anderer Features erkannt wird, die dabei verwendeten Architektur-Konzepte jedoch voneinander abweichen. Um die Wartbarkeit und Erweiterbarkeit zu erhöhen wird hier ein gemeinsames Architektur-Konzept erarbeitet und umgesetzt, dass alle Features konsistent umsetzt und dabei gemeinsame Bestandteile verwendet.

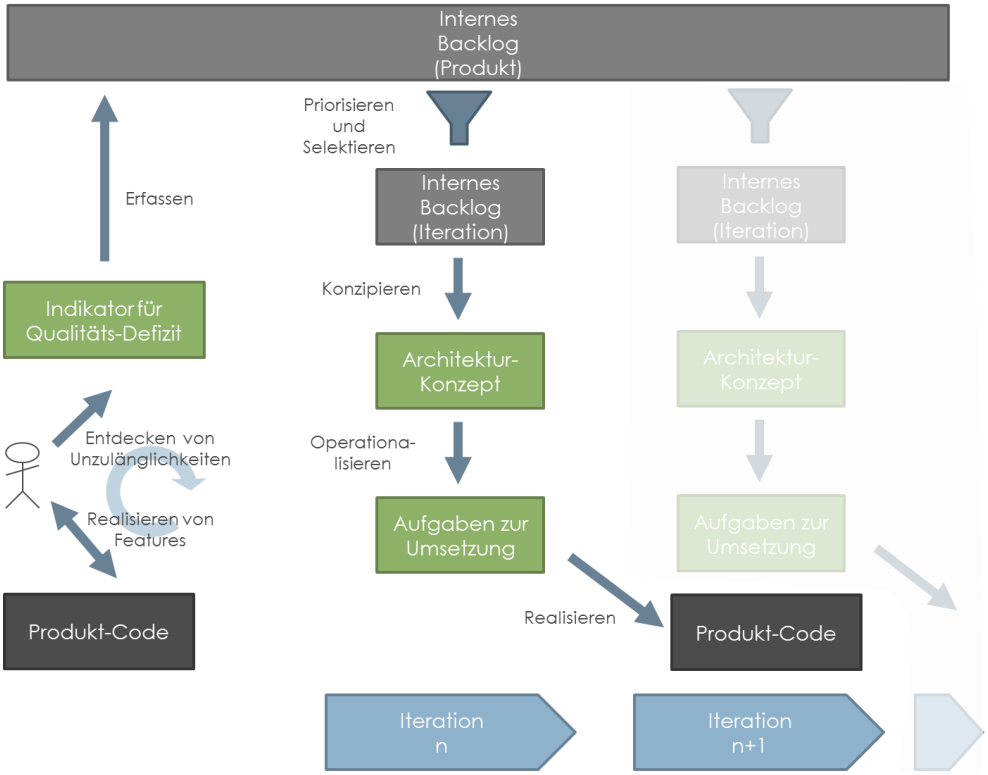

Abbildung 2: Benutzung des internen Backlogs

Input

- Sammlung von Indikatoren für Qualitäts-Defizite aus dem Team

Output

- Kontinuierliche Reduktion von offenen Refactoring-Bedarfen innerhalb der Software
- Erhöhung der internen Produktqualität

Rahmenbedingungen

Ausführender
Verantwortlicher für internes Qualitätsbudget

Werkzeuge, Hilfsmittel
Liste zum Erfassen der Refactoring-Wünsche des Entwicklungsteams

Vorkenntnisse/Erfahrungen
Erfahrung in der Beurteilung von Lösungskonzepten

Ort/Umgebung
Kein spezieller Ort

Weitere Teilnehmer
Entwicklungsteam

Voraussichtliche Dauer
Planung der Aufgaben, die innerhalb des Qualitäts-Budgets erledigt werden sollen: 20% der Dauer der Planung für das „Feature-Budget“

Vorgehensweise

Vorbereitung

- Kommunikation des Zieles der Best Practice an Budgetverantwortliche
- Etablierung eines dedizierten Budgets für interne Qualitätsarbeiten
  - Beispielsweise: 20% des Personalbudget für die Entwicklung innerhalb einer Iteration(also der verfügbaren Gesamtkapazität)
- Einrichtung eines internen Backlogs für Refactoring-Wünsche
- Entscheidung im Team erarbeiten, welche Personen die Rolle des Budget-Verantwortlichen übernehmen sollen

Durchführung

- Sammlung von Indikatoren für Qualitäts-Defizite im internen Backlog während der Realisierung eines Features durch jedes Team-Mitglied
- Priorisierung des internen Backlogs während der Planung der nächsten Iteration. Eine absolute Bewertung findet dabei nicht statt, die Priorisierung ist relativ zueinander. Beispiel: Bewertung durch Punktesystem
  - Vergabe gleicher Anzahl von Punkten an jedes Team-Mitglied
  - Verteilung von Punkten an interne Backlog-Einträge anhand des erwarteten Nutzens für die interne Qualität der Software
  - Ergebnis: Priorisierung des internen Backlogs
- Konzeptionierungs-Aufgabe für die geplante Iteration: Erarbeitung von Konzepten zur Lösung der hoch priorisierten internen Backlog-Items
- Gesamtes Team erledigt innerhalb des Budgets, das für interne Qualitätsarbeiten vorgesehen ist, diese Konzipierungs-Aufgaben. Ergebnisse:
  - Neue Konzepte, die ein Konzept ersetzen, das in einer bestehenden Realisierung verwendet wurde, aber als suboptimal erkannt wurde
  - Vereinheitlichung mehrerer ähnlicher Konzepte
- Ende der Iteration: Menge an definierten Realisierungs-Aufgaben zur Lösung der hoch priorisierten Refactoring-Wünsche im internen Backlog
- Bei Planung der folgenden Iterationen liegen für hoch priorisierten Refactoring-Wünsche sowohl Lösungskonzepte als auch konkrete Aufgaben vor
- Während der Planung der folgenden Iterationen wird nun das interne Qualitätsbudget zu 70% mit der Realisierung dieser Aufgaben belegt, zu 30% werden Lösungen für neue Refactoring-Wünsche konzipiert.
- Innerhalb der Iteration erledigt das Team die geplanten Konzeptionierungs- und Realisierungs-Aufgaben

Nachbereitung/Auswertung

- Sammlung von häufig vorkommenden Refactoring-Wünschen, Beachtung bei der Erarbeitung neuer Konzepte zur Vermeidung sich wiederholender Fehler

Gütekriterien/Empfehlungen

- Kommunikation mit Management über Nutzen der internen Qualitätsarbeiten, um die Akzeptanz für dieses Vorgehen zu erhöhen

Risiken

- Interne Qualität wird während der Realisierung von Features bewusst niedrig gehalten, um die Geschwindigkeit zu erhöhen. Man wartet auf das Aufräumen innerhalb des internen Qualitätsbudgets.
- Erledigung von Dingen, die nicht direkt notwendig wären
- Aufopfern des Budgets für Features

Einordnung in den agilen Referenzprozess

Mögliche Vorgänger

- Requirements Engineering\Systemkontext und -umfang festlegen
-

Mögliche Nachfolger

Mögliche Alternativen, verwandte Praktiken

- Architektur\Grob- und Detailplanung bei der Implementierung nutzen
- Architektur\Kontinuierliche Architekturbewertung durchführen
- Testen\Fehlermanagement einsetzen
- Testen\Teststrategie festlegen und Teststufen aufeinander abstimmen
- Unterstützung\Checklisten verwenden

Einordnung in das PQ4Agile-Qualitätsmodell

Alle internen Produktqualitäten werden beeinflusst, insbesondere Wartbarkeit.
Es gibt auch indirekte Auswirkungen auf die extern sichtbaren Qualitäten des Produktes.

Schlagworte

Qualitätsbudget, Internes Backlog, Refactoring, Vereinheitlichung der Architektur

Weiterführende Informationen

Das Projekt PQ4Agile wird vom Bundesministerium für Bildung und Forschung im Rahmen der Maßnahme KMU-innovativ: IKT (01|S13032) gefördert.