Grob- und Detailplanung bei der Implementierung nutzen

Bereich
Realisierung

Aktivität
Softwareinkrement realisieren

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

Ziele

- Vermitteln einer Orientierungshilfe für alle Entwickler
- Etablierung einer gemeinsamen Sprache
- Entscheidungsgrundlage für schnelle Problemlösungen

Motivation/Problemstellung

Die Beschreibung von Aufgaben für die aktuelle Iteration ist häufig von unterschiedlicher Detailtiefe. Gerade in angespannten Situationen fällt sie besonders kurz aus und es ist mehr Interpretationsspielraum bei der Umsetzung gegeben. Jedoch auch in einer ausführlichen Aufgabenbeschreibung ist nicht alles ist dokumentiert, was zur Umsetzung an Wissen notwendig ist. Alle benötigten Informationen darin zur Verfügung zu stellen ist nicht praktikabel. Trotzdem ist es notwendig Entscheidungen während der Realisierung zu treffen, für die eigentlich mehr Informationen notwendig sind. Diesen Design-Entscheidungen Leitlinien zur Verfügung zu stellen ist eine der Hauptaufgaben von Software-Architektur. Dies erfolgt zum einen durch eine eher langfristige, größere Zusammenhänge darstellende Grob-Architektur. Zum anderen werden Fragestellungen, die aktuelle Entwicklungsdetails betreffen, eher durch ein Software-Design für die aktuelle Iteration beantwortet.
Umfassende Architekturdokumente, auch in einer digitalen Form, ermöglichen zwar einen umfassenden, aber nicht immer einen einfachen Zugang zu Architektur. Viele der Fragen, die während der Entwicklung auftreten, müssen in kürzester Zeit beantwortet werden, so dass ein Nachschlagen im Architekturdokument als zu langsam erachtet wird. Dagegen ist eine analoge, kondensierte Version der Architekturdokumentation direkt zugänglich, im Idealfall liegt diese im Blickfeld des Entwicklungsteams.
Ein positiver Effekt, der durch den „Zwang zur Konsolidierung auf wenige Diagramme“ zu beobachten ist, ist die Schärfung der Kernaussage der Architekturdokumentation. Da auch in großen Diagrammen der Platz begrenzt ist, muss eine Auswahl von wichtigen Konzepten stattfinden. Dies trägt zur Verständlichkeit bei, da mehr Übersichtlichkeit erreicht wird. Dabei fallen teilweise Informationen weg, die damit anderweitig dokumentiert werden müssen, entweder in den dazugehörigen Backlog Items, zusätzlichen Diagrammen oder auch in den Köpfen des Teams.

Kurzbeschreibung

Ziel der Praktik ist es die geplante Architektur so bald als möglich sichtbar machen. Dabei wird eine gemeinsame Sprache verwendet, die im gesamten Team die gleiche Semantik hat, um Missverständnisse zu vermeiden. Generell werden zwei Arten von Architekturdokumentationen benutzt: Eine abstraktere, die langfristige Architekturentscheidungen zueinander in Bezug setzt und zeigt, wie das Produkt die nächsten Blöcke an Anforderungen umsetzen soll. Weiterhin wird ein detailreicherer

Ausschnitt dargestellt, der die genauere Ziel-Architektur und damit das Design der aktuellen Iteration dokumentiert. Bei beiden wird zu mehr Details und anderen Artefakten verwiesen. Dies sind beispielsweise Anforderungen, die durch ein Architektur-Konzept umgesetzt werden sollen, oder auch konkrete Entwicklungstasks, die ein Ausschnitt der Architektur realisieren.

Hierbei wird von folgender gängiger Praxis ausgegangen: Zu den Anforderungen des Produktes werden durch das Team Entwicklungstasks definiert, die diese Anforderungen umsetzen. Diese Entwicklungstasks werden in Iterationen durch das Team umgesetzt, wobei anhand dieser Aufgaben eine Koordination der Entwicklung stattfindet. Dazu sind Tasks eindeutig bezeichnet und werden während ihrer Bearbeitung eindeutig Personen zugeordnet.

Sowohl die Grob-Architektur als auch das Design sollen verständlich für alle Team-Mitglieder dokumentiert werden. Hierbei ist eine immer präsente Dokumentationsform vorzuziehen. Diese kann direkt im Blickfeld des Teams untergebracht werden. Dies kann beispielsweise ein großes Diagramm am Whiteboard oder Flipchart sein, oder ein großformatiger Ausdruck. Dabei sollen diese Diagramme immer auch vom Schreibtisch aus lesbar sein, die Größe soll nicht dazu dienen sehr viele Details unterzubringen.

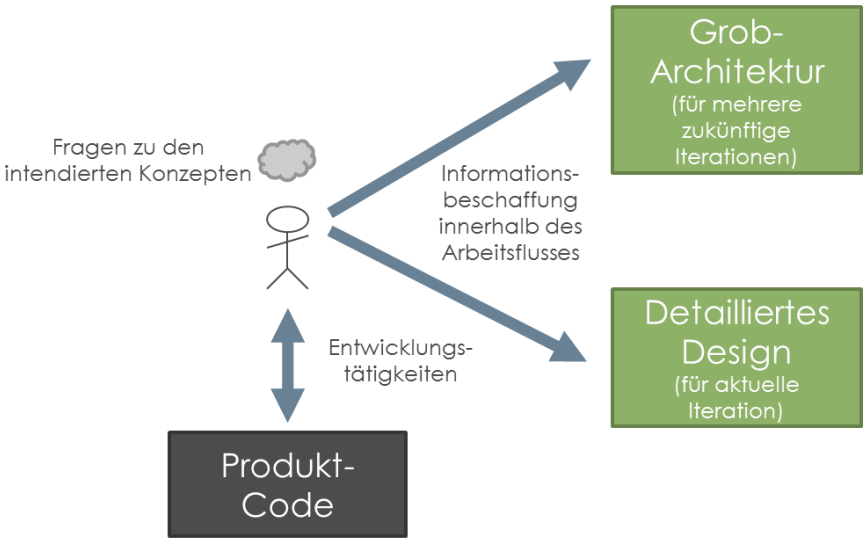

Abbildung 1:Benutzung der Grob- und Detailarchitektur während der Entwicklung

Input

- Grob-Architektur für nächste Anforderungen
- Design für aktuelle Iteration

Output

- Architekturentscheidungen nähere Zukunft und aktuelle Iteration bei allen Team-Mitgliedern verstanden und etabliert

Rahmenbedingungen

Ausführender
Architektur-Verantwortliche im Team

Werkzeuge, Hilfsmittel
Flipchart, Whiteboard

Vorkenntnisse/Erfahrungen
Richtlinien zur Dokumentation von Software-Architektur

Ort/Umgebung
Kein spezieller Ort

Weitere Teilnehmer
Alle entwickelnden Team-Mitglieder (Benutzen der Architektur-Darstellung, Annotieren von zusätzlichen Informationen)

Voraussichtliche Dauer
2 Stunden je Iteration

Vorgehensweise

Vorbereitung
- Konsolidierung gemeinsame Sprache und Begrifflichkeiten
- Erarbeitung und Konsolidierung der längerfristigen Grob-Architektur
- Erarbeitung des detaillierten Designs für die aktuelle Iteration

Durchführung
- Zu Beginn eines neuen Blocks an Anforderungen:
  - Mit Hilfe des Teams (siehe „Architekturlösungen im Team entwickeln“) wurde eine für den neuen größeren Block an Anforderungen (Beispielsweise ein „Epic“) adäquate Architektur erarbeitet (siehe „Für Planungshorizont angemessene Architektur erarbeiten“)
  - Erarbeitete Grob-Architektur auf Konsistenz mit der gemeinsamen Sprache prüfen und ggf. entsprechend anpassen
  - Grundlegende Konzepte der Grob-Architektur auswählen, die für deren Verständnis essentiell sind, beispielsweise:
    - Kernkomponenten
    - Verwendete Technologien
    - Kollaboration der Kernkomponenten um die wichtigen Anwendungsfälle umzusetzen
  - Visuelle Repräsentation wählen, die die Grundkonzepte sichtbar macht, beispielsweise:
    - Komponenten-Konnektor-Diagramme
    - Sequenz-Diagramme
    - Klassendiagramme
  - Annotationen mit ergänzendem Text zu wichtigen Aspekten hinzufügen
    - Beispiel für UML: Notiz-Element als Klebenotiz
  - An Bestandteile der Grob-Architektur Annotationen mit Links zu Stories hinzufügen, die durch diesen Aspekt realisiert werden sollen
- Zu Beginn einer neuen Iteration:
  - Mit Hilfe des Teams (siehe „Architekturlösungen im Team entwickeln“) wurde eine für die geplanten Anforderungen der nächsten Iteration adäquates Design erarbeitet (siehe „Für Planungshorizont angemessene Architektur erarbeiten“)
  - Erarbeitetes Design der aktuellen Iteration auf Konsistenz mit Grob-Planung überprüfen
  - Visuelle Repräsentation wählen, die es ermöglicht, alle aktuellen Entwicklungsaufgaben darzustellen
  - Links zu Tasks der Iteration als Annotationen anfügen
- Während einer Iteration:
  - Grob-Architektur und detailliertes Design sichtbar und verständlich für alle Team-Mitglieder
  - Aktualisieren der Grob-Architektur und des Designs um aktuelle Entscheidungen zu reflektieren
  - Einfaches Orientieren während der Entwicklung
  - Treffen von Entscheidungen auf Basis der längerfristigen Grob-Architektur und

des aktuellen detaillierten Designs

Nachbereitung/Auswertung

- Konsolidieren der Ergänzungen und Änderungen, die während einer Iteration sowohl am Design als auch an der Grob-Architektur gemacht wurden
- Eventuell Anpassen der getroffenen Architekturentscheidungen

Gütekriterien/Empfehlungen

- Bei der Darstellung der Grob-Architektur sollten nicht zu viele Details verwendet werden, damit die Hauptaussage eindeutig klar wird, also das Zusammenspiel der verwendeten Architektur-Entscheidungen. Trotzdem sollten alle wichtigen Aspekte genannt werden.
- Die Darstellung des detaillierten Designs ist so zu wählen, dass es möglich ist für alle Entwicklungsaufgaben deren Abhängigkeiten zu visualisieren. Dies bedeutet insbesondere die Repräsentation aller Beziehungen zwischen Software-Bestandteilen, die in dieser Iteration angepasst oder erstellt werden.

Risiken

- Generell ist diese Praktik nur in Verbindung mit einer gewissen Planung auf Architektur-Ebene möglich. Wie diese Planung erstellt werden kann und zu welchem Detailgrad sie erfolgen sollte ist mit der Praktik „Für Planungshorizont angemessene Architektur erarbeiten“ beschrieben. Diese Praktik baut auf deren Ergebnisse auf.
- Vergessene wichtige Informationen in der Grob-Architektur oder im detaillierten Design sorgen häufig dafür, dass diese auch von den Entwicklern bei der Umsetzung nicht beachtet werden. Daher sollte auf die Qualität dieser beiden Darstellungen wert gelegt werden und eine Überprüfung durch alle Team-Mitglieder erfolgen.
- Wenn ein unterschiedlicher Wissensstand bezüglich der Darstellung von Software-Architektur bei den Mitgliedern des Entwicklungsteams herrscht, so kann es dabei zu Missverständnissen kommen. Hier ist es hilfreich, die Diagramme kurz dem Team vorzustellen und auf die Semantik der Notation einzugehen.

Einordnung in den agilen Referenzprozess

Mögliche Vorgänger

- Architektur\Architekturentscheidungen dokumentieren
- Architektur\Architekturrelevante Anforderungen dokumentieren
- Architektur\Architekturlösungen im Team entwickeln

Mögliche Nachfolger

- Testen\Reviews von Entwicklungsartefakten durchführen
- User Experience\Template-basierte UI-Konzeption

Mögliche Alternativen, verwandte Praktiken

- User Experience\Informationsarchitektur erstellen
- Architektur\Feature-Entwicklung und interne Qualitätsaufgaben verzahnen

Einordnung in das PQ4Agile-Qualitätsmodell

Wartbarkeit als ein Vertreter der Qualitätsattribute zur Entwicklungszeit steht im besonderen Fokus dieser Praktik. Generell werden damit aber alle internen

Produktqualitäten beeinflusst, es gibt auch indirekte Auswirkungen auf die extern sichtbaren Qualitäten des Produktes.

Schlagworte

Architekturvision, Detail-Architektur, Grob-Architektur, Entwicklungsteam

Weiterführende Informationen

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