Ausarbeitung- 30.01.2008

Wie unterscheiden sich Top-Down und Bottom-Up Integration? Was kommt häufiger vor, und welche Vorraussetzungen müssen gegeben sein?

Bottom-Up Integration
Hier arbeitet man sich ausgehend von den hardwarenahen Schichten nach oben vor. Vorteil ist die frühe Integration der risikobehafteteren Teile (wodurch man später auf ein stabiles Framework aufbauen kann), Nachteil ist dass man Driver (Softwaremodule, oben liegende Schichten simulieren, z.B. GUI-Interaktionen durch ein Script simulieren) schreiben muss. Der Kunde sieht lange keinen Fortschritt, weil die sichtbaren Teile erst spät integriert werden. Es müssen daher zusätzlich Prototypen erstellt werden.

Top-Down Integration
Die Komponenten werden von höher liegenden Schichten der Software-Architektur ausgehend nach unten integriert.
Beispielsweise wird zuerst das GUI mit der Business Logic zusammengef¨ugt, sp¨ater kommt unten noch der Data-Access-Layer hinzu.
Vorteil ist, dass die höher liegenden Schichten - das sind die für den Benutzer sichtbareren - früh verfügbar sind, was sich für Demo-Zwecke gut eignet. Allerdings sind Stubs zu schreiben (das sind Softwaremodule, die die noch nicht integrierten unteren Schichten simulieren). Außerdem werden hardwarenahe Teile (das sind üblicherweise die fehleranfälligeren) erst spät integriert werden.

Welche Arten des Tailoring im V-Model XT gibt es? Beschreiben Sie diese.

Statisches Tailoring:
- ▪Zu Projektbeginn
- ▪Charakterisierung eines Projekts anhand vorgeg. Projektmerkmale (Profil)
- ▪Profil legt verpflichtend bzw. optional zu verwendenden Vorgehensbausteine sowie die möglichen Durchführungsstrategien fest
- ▪Durch Hinzunahme; nicht durch Streichung (nur relevante VB/DS wichtig)

Dynamisches Tailoring
- ▪Während des Projekts
- ▪Durch Hinzunahme/Entfernung von VB (Ausnahmen, Regeln)

Detaillierte Anpassung erst bei Projektplanung
- ▪Produkte (Abhängigkeiten), Aktivitäten

Erweiterungen/Anpassungen (Tool: V-Modell XT-Editor)
- ▪z.B. von Vorgehensbausteinen, Entscheidungspunkten, …

Wie unterscheidet sich die Anforderungsanalyse im Unified Process (RUP) und im XP?

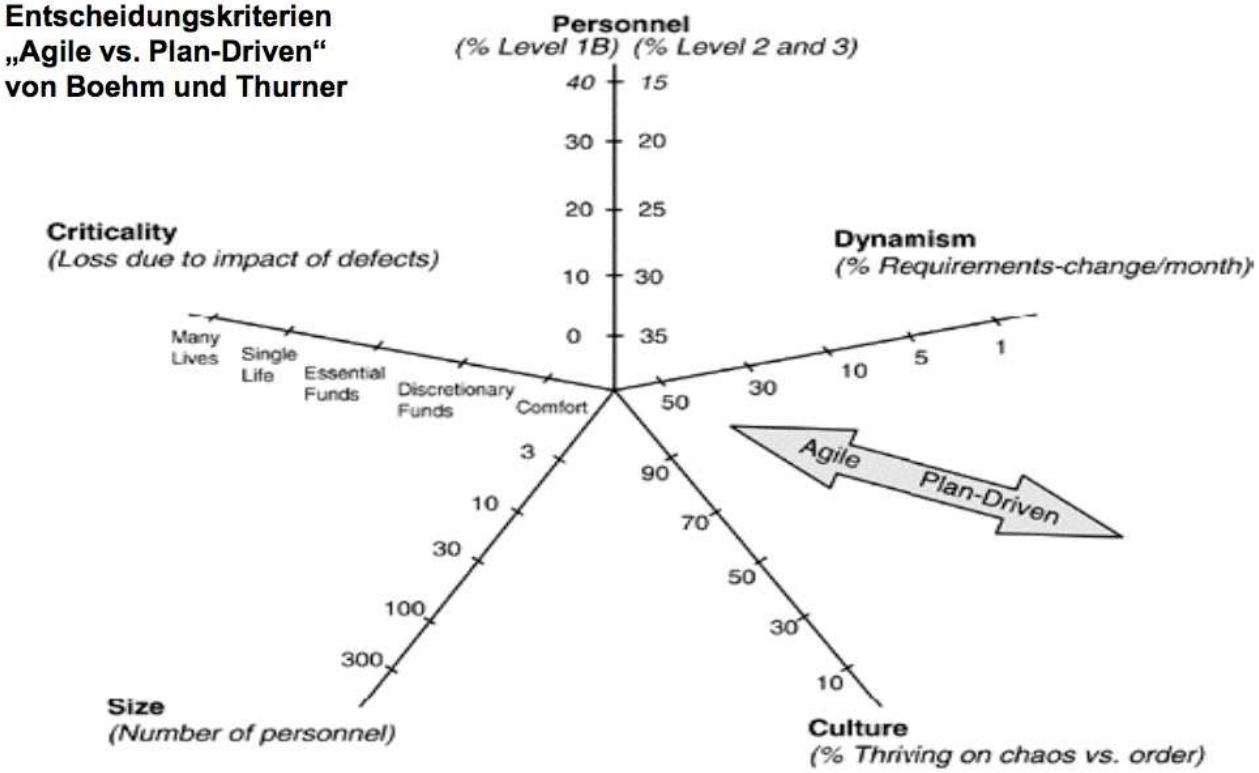
RUP:
-) Das Analysemodell soll die Anforderungsanalyse auf Vollständigkeit, Konsistenz und Machbarkeit überprüfen.
-) Das System wird nach objektorientierten Gesichtspunkten von innen beschrieben.
-) Das System wird in die Klassentypen Schnittstelle, Kontroller und Entitäten unterteilt. Durch die Zusammenarbeit dieser Klassen sollen alle Systemfunktionen für die Benutzer verfügbar gemacht werden.

-)Das Analysemodell entspricht einer Aufteilung des Systems nach dem MVC-Paradigma.
-)Dynamische Diagramme dienen zur Darstellung von komplexen Abläufen in oder zwischen einzelnen Anforderungen oder innerhalb des Analysemodells.
-)Dynamische Diagramme werden nur für besondere Fälle eingesetzt, in denen das Verständnis der dynamischen Funktionsweise des Systems nicht trivial erkennbar ist.
-)Die Analyse stellt die Vorbereitung für den Entwurf dar. Während im Entwurf die technische Umsetzung des Systems geplant wird und daher alle technischen Faktoren wie Zielumgebung oder Systemarchitektur eine große Rolle spielen, werden diese in der Analyse außer Acht gelassen.

XP:
Anforderungen werden mit User-Stories beschrieben. Diese werden vom Kunden verfasst und dienen auch als Grundlage für die Aufwandsabschätzung.
Der Kunde muss das ganze Projekt über vor Ort sein.
Die Metapher ist, dass der Kunde und das Entwicklerteam
-) eine gemeinsame Sprache im Bezug auf das System sprechen
-) eine gemeinsame Vision/Sicht auf das Problem haben
-) Generativität: Projektmitarbeiter und Kunde müssen sich gegenseitig umeinander "kümmern".
-) Architektur ?

Softwarewartung im Unified Process. Welche Rollen gibt es, welche Produkte? Wieso wird zwischen Fehlerbehebung und Erweiterung entschieden?
Rollen: Architekt, Dokumentierer, Wartungsteam
Produkte: Auslieferungsplan, Abnahmeprotokoll, Integrations- Migrationsplan, Anwenderdokumentation, Fehlerbericht, Änderungswunsch
Fehlerbehebung ist wichtiger als Erweiterung, da ein funktionierendes Programm wichtiger ist als eines das viele Funktionen hat die nicht gut funktionieren.

Anwendungskriterien von Agiler und Plan-Driven Entwicklung nach X und Y erläutern.



Agil (engl. agile)
▪Ergebnisse früh & danach laufend sichtbar (kurze Iterationszeiten)
▪Vage Anforderungen und Änderungen sind kein Problem
▪Produkt, Mensch und Kommunikation stehen im Mittelpunkt
▪Miteinbeziehung des Kunden wird angestrebt
▪Traditionell (engl. oft plan-driven)
▪Späte Ergebnisse
▪Exaktes Verständnis der Anforderungen und somit stabiles Ziel
▪Prozess, Werkzeuge und das Verfolgen eines Plans stehen im Mittelpunkt
▪Direktes Mitwirken des Kunden eher unwahrscheinlich

Rational Unified Process (RUP)
Erläutern Sie die grundlegenden Konzepte des Rational Unified Process (RUP). Aus welchen Phasen besteht der RUP? Welches sind die wesentlichen Vorteile von RUP? Wann wird es typischerweise eingesetzt?
Phasen:
Inception phase
Hier werden eine Projektbeschreibung, die Erfolgsfaktoren des Projektes sowie eine Kosten- und Risikoabschätzung erstellt. Weiters wird ein grundsätzliches Use Case-Modell und ein Basis-Projektplan festgelegt. Auch Qualitätsstandards für dieses Projekt werden hier definiert.
Grob zusammengefasst: hier geht es um die Ziele des Projektes.
Elaboration phase

Diese Phase entspricht in etwa der klassischen Design-Phase: Hier wird die grundsätzliche
Software-Architektur festgelegt und die Problem-Domain-Analyse durchgef¨uhrt (was soll konkret umgesetzt werden? was existiert bereits?).
Grob zusammengefasst: hier geht es um die Architektur.
Construction phase
Dies ist die eigentliche Entwicklung und entspricht der klassischen Implementierungs-Phase. Es entsteht eine erste Version der Software, die auch nach außen released werden kann.
Transition phase
Hier findet die Einschulung der End-User, das Beta-Testing und die abschließende Qualitätskontrolle statt.

Vorteile:
Es gibt Tool-Unterstützung
Nachteile:
RUP ist komplizierter als viele andere Modelle,
Dokumentationslastig
Proprietär

Einsatzgebiet:
Wegen dem relativ komplexen Aufbau wird RUP vor allem bei großen Projekten eingesetzt.

Software Implementierung und Integration
Was versteht man unter der Integrationsphase im Softwareentwicklungsprozess? Welche Integrationsstrategien kennen Sie? Wie unterscheiden sie sich? Welche Vorteile / Nachteile haben diese unterschiedlichen Strategien?
Integration meint das Zusammenfügen einzeln entwickelter Komponenten zu einem Teil oder Gesamtsystem. Es gibt dafür verschiedene Strategien mit verschiedenen Vor- und Nachteilen.
Big-Bang-Integration

> Damit ist die gleichzeitige Zusammenfügung aller Komponenten gemeint. Sie lässt sich nur bei sehr kleinen Projekten anwenden. Vorteil ist, dass keine Stubs oder Driver (siehe unten) geschrieben werden müssen und dass keine Regressionstests notwendig sind. Nachteil ist aber, dass auftretende Fehler kaum lokalisierbar sind.

Top-Down-Integration

> Die Komponenten werden von höher liegenden Schichten der Software-Architektur ausgehend nach unten integriert.
> Beispielsweise wird zuerst das GUI mit der Business Logic zusammengefügt, später kommt unten noch der Data-Access-Layer hinzu. Vorteil ist, dass die höher liegenden Schichten - das sind die f¨ur den Benutzer sichtbareren - früh verfügbar sind, was sich für Demo-Zwecke gut eignet. Allerdings sind Stubs zu schreiben (das sind Softwaremodule, die die noch nicht integrierten unteren Schichten simulieren). Außerdem werden hardwarenahe Teile (das sind üblicherweise die fehleranfälligeren) erst spät integriert werden.

Bottom-Up-Integration

> Hier arbeitet man sich ausgehend von den hardwarenahen Schichten nach oben vor. Vorteil ist die frühe Integration der risikobehafteteren Teile (wodurch man später

auf ein stabiles Framework aufbauen kann), Nachteil ist dass man Driver (Softwaremodule, oben liegende Schichten simulieren, z.B. GUI-Interaktionen durch ein Script simulieren) schreiben muss. Der Kunde sieht lange keinen Fortschritt, weil die sichtbaren Teile erst spät integriert werden. Es müssen daher zusätzlich Prototypen erstellt werden.

Build-Integration

Darunter versteht man die gleichzeitige Integration mehrere Komponenten auf verschiedenen Ebene der Architektur, die gemeinsam einen Use Case umsetzen. Beispielsweise könnte GUI, Business Logic und Data-Access-Layer, die für den Abruf von Kontoauszügen benötigt werden, gemeinsam integriert werden. Andere Geschäftsfälle sind dann aber noch nicht abgedeckt. Vorteil ist, dass schnell ein Use Case vollständig durchgeführt werden kann (gut für Präsentationen). Allerdings bauen oft verschiedene Use Cases auf gleichen Modulen auf, wodurch eventuell später ein Fehler in einem Modul erkannt wird, das vorher in Verbindung mit einem anderen Use Case verwendet wurde und dort funktioniert hat. Es sind in diesem Fall Änderungen und Regressionstests notwendig.

Software Wartung

Welche Arten der Software Wartung kennen Sie? Wie unterscheiden sie sich? Wie kann die Software Wartung im Rational Unified Prozess umgesetzt werden (geben Sie 2 konkrete Prozessausprägungen an)?

Corrective: Bugs ausbessern (nur ca. 20% der gesamten Wartung)
Adaptive: Anpassung an geänderte Bedingungen in der Umgebung (z.B. Portierung auf ein neues Betriebssystem)
Perfective: Verbesserungen für den Benutzer (z.B. neue Features einbauen)
Preventive: Zukünftige Wartungsarbeiten erleichtern (z.B. durch Verbesserung der Dokumentation)

Die Ausarbeitungen von dodlhuat und max_rayman
Quelle: http://www.informatik-forum.at/showthread.php?t=61876