

XION IT SYSTEMS

AKTIENGESELLSCHAFT

Dresdnerstraße 81-85/8.Stock
A-1200 Wien

Tel: 0664-8242-600

E-mail: office@xion.at

Web: xion.at

Festnetz: +43/1/333 91 99-0

Fax: +43/1/333 91 99-199

x i o n . i t s y s t e m s . a g



Software Wartung und Evolution

Dipl.-Ing. Dr. techn. Johannes Weidl-Rektenwald
Xion IT Systems AG

XION IT SYSTEMS

AKTIENGESELLSCHAFT

Dresdnerstraße 81-85/8.Stock
A-1200 Wien

Tel: 0664-8242-600

E-mail: office@xion.at

Web: xion.at

Festnetz: +43/1/333 91 99-0

Fax: +43/1/333 91 99-199

x i o n . i t s y s t e m s . a g



Lecture 7

Lecture 7

- Inhalt
 - Spezielle Kapitel der Software Wartung
 - Program Comprehension
 - Change Impact Analysis
 - Change Propagation
 - Wartungsdokumentation
 - Maintainability (Wartbarkeit)
 - Definition
 - Ensuring Maintainability
 - Design for Change: Ausgewählte Kapitel
 - » Cohesion, Coupling, Model Driven Architecture (MDA)
 - Software Wartung im unternehmerischen Kontext

© J. Weidl-Rektenwald 02-07

210

Spezielle Kapitel der Software Wartung

Aktivitäten im Wartungsfall

- Fehlermeldung bzw. Änderungsantrag
 - Dokumentieren/Einpfelegen
 - Life Cycle Management
 - Evaluierung/Reporting
- **Analyse bzw. Planung der Änderung**
 - Program Comprehension
 - Change Impact Analysis
- **Implementierung der Änderung**
 - Restructuring
 - Change Propagation
 - Verwalten der Artefakte
- **Verifikation und Validierung**
- **Re-Dokumentation**
- Produktivstellung der Änderung

Spezielle Kapitel der Software Wartung

- Program Comprehension
- Change Impact Analysis
- Change Propagation
- Wartung und Test
- Wartungsdokumentation

Program Comprehension

Program Comprehension

- Definition
 - **Program Comprehension**
 - Research into how software engineers understand existing systems
 - by Malcolm Munro

Program Comprehension

- [R. Brooks 1983]
 - Programmer creates assumptions or hypotheses based on both acquired or existing knowledge to arrive at an understanding
 - Hypotheses are checked against the source code to prove their validity
 - *Beacons* are places in the source code that prove or falsify a hypotheses

Program Comprehension - Approaches

- Top-down approach
 - Taken by the original developer
 - The software engineer takes a top-down approach starting with the most abstract concepts, refines them, and transforms them into a computer program
- Bottom-up approach
 - Taken by the maintenance engineer
 - Programmers learn by focusing on small pieces of code and later combine this information together

Span of Understanding

- Span of Understanding
 - Nennt man die Zeitspanne, die der Programmierer zum Verstehen eines definierten Programmstückes benötigt
- Ziel der Forschung: Wie kann man den *Span of Understanding* verkürzen bzw. minimieren

Change Impact Analysis

Change Impact Analysis

- Die Change Impact Analyse versucht, den so genannten *Ripple-Effekt* erschöpfend zu beschreiben
 - Ripple Effekt einer Änderung
 - Effekt der sequentiellen Programm-Inkonsistenzen aufgrund einer initialen Änderung
- Der sogenannte „Change Impact“ kann dann
 - Abgeschätzt bzw. quantifiziert werden
- Damit werden
 - Änderungen planbar und alternative Vorgehensweisen abwägbar
 - Änderungen in der Software konsistent und vollständig durchführbar

© J. Weidl-Rektenwald 02-07

220

Change Impact Analysis

- Definition
 - **Impact Analysis**
 - Research into the development of a method for predicting the effect of a change to an existing system as early as possible in the change cycle. The research is investigating how changes can be represented and methods for the automatic propagation of changes.
 - by Malcolm Munro

© J. Weidl-Rektenwald 02-07

221

Change Impact Analysis

- Mehrere Ansätze, z.B.
 - analytisch
 - graphbasiert (Vaclav Rajlich)
- Input
 - Meist Abstract Syntax Tree (AST)
- Tools
 - Es gibt heute sehr leistungsfähige high-level Cross-Referencer, die detaillierte Change Impact Analysen erlauben

© J. Weidl-Rektenwald 02-07

222

Change Propagation

Change propagation

- Definition
 - Change propagation ist der Prozess der sequentiellen Behebung von Programm-Inkonsistenzen aufgrund einer initialen Änderung
- Input
 - Korrektes Programm vor der Änderung
 - Change Impact Analysis
- Output
 - Korrektes Programm nach Durchführung der Änderung

© J. Weidl-Rektenwald 02-07

224

Change Propagation spreads to Documentation

- Technical documentation has to be updated whenever a change to dataflow, design, architecture, module procedure, or any other related artefact is made
- Inaccurate documentation can be worse than no documentation
- Entire documentation should be previewed prior to re-release of the software

© J. Weidl-Rektenwald 02-07

225

Wartungsdokumentation

Wartungshandbuch

- Beschreibt für die Wartung relevante Inhalte
 - Einführung in Architektur, Design, Systemumgebung, ... für den Wartungsingenieur
 - Wie ist im Wartungsfall vorzugehen?
 - Was sind die relevanten Dokumente für die verschiedenen Wartungsfälle?
 - Welche Einstiegspunkte in den Code gibt es?
 - Wie ist die Produktivstellung zu planen, was ist dabei zu berücksichtigen?
 - Wie wird die Wartung dokumentiert?

Projekttagbuch Wartung

- Der Wartungsmanager führt ein Projekttagbuch, das die einzelnen Wartungsfälle dokumentiert und zueinander in Beziehung stellt
- Management Summaries für das Top-Management können daraus periodisch produziert werden
- Das Projekttagbuch liefert Daten für die Prozessoptimierung der Wartung und die langfristige Nachvollziehbarkeit

© J. Weidl-Rektenwald 02-07

228

Maintainability (Wartbarkeit)

Definition: Maintainability

- Maintainability is the ease of maintenance
- Can be decomposed as
 - Repairability
 - Ability to correct defects in reasonable time
 - Evolvability
 - Ability to adapt software to environment changes and to improve it in reasonable time

Maintainability - Design for Change

Design for Change

- Problem
 - Wie entwirft und implementiert man Software, sodass zukünftige absehbare bzw. nicht absehbare Änderungen möglichst leicht einzuarbeiten sind?
- Lösung
 - Es gibt mannigfaltige Ansätze in Forschung und Industrie
 - Aber keine „Silver Bullet“ Lösung

© J. Weidl-Rektenwald 02-07

232

Design for Change: Ausgewählte Kapitel

- Design Metriken
 - Cohesion und Coupling
- Model-driven Architecture

© J. Weidl-Rektenwald 02-07

233

Cohesion

- Beschreibt den Grad der logischen Abhängigkeiten innerhalb eines Software Moduls
- Je größer die Cohesion desto besser das Software Design
- Hinter einem Interface sollte die Cohesion maximal sein
- Große Cohesion erlaubt die einfache Wiederverwendung von Software Modulen

© J. Weidl-Rektenwald 02-07

234

Coupling

- Coupling beschreibt den Grad der logischen Abhängigkeiten zwischen verschiedenen Software Modulen
- Je größer das Coupling desto schlechter das Software Design
- Das Coupling über Interfaces hinweg sollte minimal sein
- Großes Coupling verhindert die einfache Wiederverwendung von Softwaremodulen

© J. Weidl-Rektenwald 02-07

235

Model Driven Architecture (MDA) und Software Wartung / Evolution

MDA - Motivation

- Technologie entwickelt sich in kurzen Zyklen weiter
 - Betriebssysteme, Datenbanken, Middleware, Komponentenmodelle, (GUI) Bibliotheken, Programmiersprachen
- Es gibt zu jedem Zeitpunkt mehr als eine adäquate Technologie, um ein System umzusetzen
 - z.B. CORBA, J2EE, .NET
- Die Geschäfts- oder Fachlogik ist beständiger als Technologien
 - Trotzdem wird die Fachlogik bei einem Technologiewechsel in der Regel neu implementiert

Was ist die MDA?

- Vorgehensmodell
 - Das *Platform Independent Model* (PIM) modelliert die Fachdomäne
 - Das *Platform Specific Model* (PSM) beschreibt die Anwendung in der konkreten Implementierungstechnologie
 - Transformation PIM -> PSM -> Code erfolgt im Idealfall voll automatisch
- MDA basiert auf Ideen der modellgetriebenen und generativen Softwareentwicklung

© J. Weidl-Rektenwald 02-07

238

Technologien der MDA

- Unified Modeling Language (UML)
 - OCL für Pre-/Postconditions
 - Action Language für Semantik
- UML Profiles
 - Tailor the language to particular areas of computing (such as EDOC) or particular platforms (such as EJB or CORBA)
- Meta Object Facility (MOF)
 - Defines a standard repository for Meta-Models
 - Used to define information models for particular domains
- XML Metadata Interchange (XMI)
 - The XML-UML standard
- Common Warehouse Metamodel (CWM)
 - Forms the MDA mapping to database schemas

© J. Weidl-Rektenwald 02-07

239

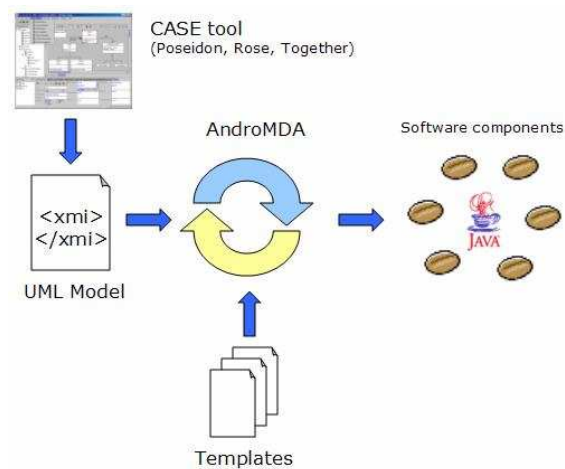
AndroMDA: „MDA light“

- Open Source Tool (www.andromda.org)
- PIM wird in UML modelliert
- Auf ein eigenständiges PSM wird verzichtet
- Template basierter Codegenerator
 - IN: UML Modell (fachlich, technisch attribuiert)
 - OUT: Softwarekomponenten (J2EE, etc.)

© J. Weidl-Rektenwald 02-07

240

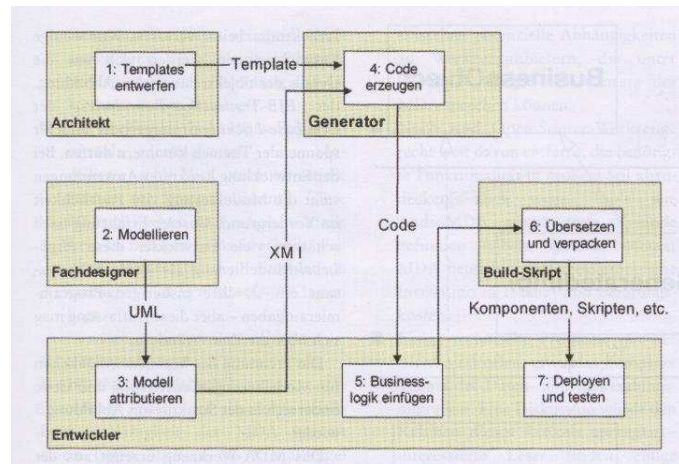
AndroMDA: Konzept



© J. Weidl-Rektenwald 02-07

241

AndroMDA: Vorgehensmodell



© J. Weidl-Rektenwald 02-07

242

AndroMDA: Was wird generiert?

- Entity Beans (inkl. CMP, allen Interfaces, Utility Klassen, Implementation Stubs)
- Session Beans (alle Interfaces, Implementation Stubs)
- Struts Komponenten (JSP Page Stubs, alle Forms, Action Stubs, Config File!)
- Value Objects
- Deployment Deskriptoren (viele!)
- .ear File (Deployment Package)

© J. Weidl-Rektenwald 02-07

243

AndroMDA: Womit wird generiert?

- Apache Komponenten
 - Ant (XML Build Engine)
 - Jakarta
 - Commons (allgemeine Utility Bibliotheken)
 - Taglibs (Tag Libraries)
 - Struts (MVC Web Application Framework)
 - Velocity (Template Sprache)
 - XML (xerces)
- Xdoclet (Code generation engine)
 - Reads meta data from javadoc tags

© J. Weidl-Rektenwald 02-07

244

MDA: Bottom-line Benefits

- The benefits of MDA are significant to business leaders and developers alike
 - Reduced cost throughout the application life-cycle
 - Reduced development time for new applications
 - Improved application quality
 - Increased return on technology investments
 - Rapid inclusion of emerging technology benefits into their existing systems

[<http://www.omg.org/mda/>]

© J. Weidl-Rektenwald 02-07

245

Probleme der MDA

- Stabilität des Datenmodells
 - Änderungen im Modell bedingen Datenmigration
- Entfremdung des Entwicklers vom Code
 - Generierung von zig Implementierungsklassen
- Unzulänglichkeiten in den Tools
 - z.B. Ändern eines UML Klassennamens
- Inkrementelle Generierung
- Debugging auf unteren Schichten / Error Reporting

© J. Weidl-Rektenwald 02-07

246

Was erspart mir die MDA nicht?

- Software Artifact Management
 - Auf Modellebene!
- Verifikation / Test
 - Nur auf Modellebene?
- Optimieren
 - Nur auf Modellebene?
- Software Wartung und Evolution(splanung)

© J. Weidl-Rektenwald 02-07

247

MDA und Wartung

- Generierter Code muss nicht gewartet werden
 - Fall-Beispiel „Xion xbib“ (AndroMDA)
 - J2EE Bibliotheksverwaltung im Applikationsserver Cluster (Failover, Load Balancing, Skalierbarkeit, CMP O/R Mapping)
 - Gesamt: 4308 SLOC
 - Manuell: 1815 SLOC (42,13%)
 - Generiert: 2493 SLOC (57,86%), 60% der Klassen, 80,79% der Methoden
- Ohne Code keine quick-fix Wartung möglich (?)
- Wartung quasi nur auf Modellebene möglich
 - Hier sind qualitativ hochwertige Informationen vorhanden
 - Anforderungen, Design, Design Decisions, Dokumentation, ...
 - Damit „automatisch“ Iterative Enhancement Wartung

© J. Weidl-Rektenwald 02-07

248

MDA und Evolution

- Three problems of evolution
 - How to express the change?
 - How to propagate the change?
 - How to manage/analyse the change?
- For software evolution to be automated, design and dependency information must be preserved.
- The MDA codifies design information in high-level instances, and dependency information in transformations.
- Different propagation techniques can be applied to the MDA to achieve automated software evolution.

© J. Weidl-Rektenwald 02-07

[David Hearnden]

249

Software Wartung im unternehmerischen Kontext

Software Wartung im Unternehmen

- Stakeholder („beteiligte Parteien“)
 - Hardware/Systeme
 - Softwareentwicklung
 - Testabteilung
 - Betriebsführung
 - Rechenzentrum
 - Call Center
 - Benutzer
 - Kunden

Wartungsphilosophien

- „Throw it over the wall“ – someone else is responsible for maintenance
 - Investment in knowledge and experience is lost
 - Maintenance becomes a reverse engineering challenge
- „Mission orientation“ – development team makes a long term commitment to maintaining the software

© J. Weidl-Rektenwald 02-07

252

Software Wartung im Unternehmen

- Linienorganisation
 - CTO
 - Bereichsleiter
 - Abteilungsleiter
 - Teamleiter
 - Ingenieur/Techniker
- Rollen
 - Planung/Management/Projektleitung
 - [Wartungsmanager](#)
 - Operative Durchführung
 - [Wartungsingenieur](#)

© J. Weidl-Rektenwald 02-07

253

Wartungsmanager

- Aufgaben
 - Ganzheitliche Planung des Wartungsprozesses
 - Evaluierung, Machbarkeit, Kostenschätzung von Change Requests
 - Beauftragung und Coaching der Wartungsingenieure
 - Controlling und Qualitätssicherung der Durchführung
 - Planung von Wartungsfenster und Produktivstellung
 - Planung von Schulung
 - Dokumentation

Wartungsingenieur

- Aufgaben
 - Umsetzung der Wartungsaufträge (vgl. „Aktivitäten im Wartungsfall“)
 - Analyse
 - Design
 - Implementierung
 - Dokumentation
 - Test
 - Durchführung der Systemmodifikation im Wartungsfenster
 - Pflege des Wartungshandbuchs

Software Wartungs-Verträge

Gewährleistung

- Modalitäten der Gewährleistung (GWL)
 - Dauer der GWL
 - Standardsoftware (zeitlich unbegrenzte Überlassung)
 - GWL-Bestimmungen des Kaufrechtes
 - Individualsoftware
 - GWL-Bestimmungen des Werkvertrages (wenn nicht anders vereinbart: 6 Monate)
 - Überlassung auf bestimmte bzw. unbestimmte Zeit
 - GWL-Bestimmungen des Mietrechts
 - Wann beginnt die GWL-Frist
 - Mit dem Datum des Abnahmeprotokolls, wenn nicht anders vereinbart
 - Wo sind die GWL Bedingungen geregelt
 - Oft in den Allgemeinen Geschäftsbedingungen, ansonsten (bzw. Ausnahmen und Erweiterungen) im Vertrag

Gewährleistung

- Modalitäten der Gewährleistung (GWL)
 - Definition der Modalitäten der Behebung von Mängeln
 - Vor allem Reaktionszeit, Zeitdauer bis zur Behebung
 - Abgrenzung der Mängel, die unter GWL fallen
 - Keine GWL z.B. für Mängel, die aufgrund der Hardware-Konfiguration bzw. besonderer Beschaffenheit von Fremdsoftware auftreten
 - Klassifikation von Mängeln (z.B. betriebsverhindernd, betriebsbehindernd, nicht betriebsbehindernd)
- Die Rechtsfolgen der GWL treten nicht schon mit dem Vorhandensein der fehlerhaften Beschaffenheit ein, sondern müssen vom Erwerber **vor Gericht** durch Klage geltend gemacht werden

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

258

Software Wartungs-Verträge

- Obwohl Software ohne sachgemäße Pflege schnell unbrauchbar werden kann, wird diese vom Lieferanten nicht automatisch mit der Überlassung geschuldet
 - Kauf: Mängelbeseitigung während der GWL-Frist
 - Miete: Mängelbehebung während der Vertragsdauer
- Die erbrachten Leistungen der Software Wartung (korrektiv nach Ablauf der GWL, adaptiv, perfektionierend, präventiv) können
 - Teil der Leistung gemäß des Überlassungsvertrages sein (meist bei wiederkehrenden Zahlungen)
 - gemäß eines Programmwartungs-Vertrages gegen gesondertes Entgelt erbracht werden

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

259

Inhalt von Software Wartungs-Verträgen

- **Herkömmlich**
 - Korrektive, adaptive Wartung, perfektionierende Wartung, präventive Wartung
- **Zusätzlich**
 - Überlassung von neuen Programmversionen
 - Einweisung von Personal in neue Programmversionen
 - Technische Hilfe (z.B. telefonisch, über ein Portal)
 - Beratung beim Einsatz der Software
 - Aufklärung von Bedienungsfehlern
 - Beseitigung der Auswirkungen von Bedienungsfehlern
 - Nachrecherchieren von ungerechtfertigten Mängelrügen

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

260

Arten von Software Wartungs-Verträgen

- Gekoppelte Software Wartung als Nebenleistung im Rahmen eines Hardware-Miet oder –Wartungsvertrags
 - z.B. für Betriebssystem
- Nebenleistung im Rahmen eines Softwarelizenzvertrages
 - Mit periodischen Lizenzgebühren einschließlich Wartung
- Softwarewartung während der GWL-Frist eines Softwareentwicklungsvertrags bzw. eines Softwarelizenzvertrages
 - Mit pauschaler Gebühr, wobei das Entgelt Teil der Kosten der Entwicklung bzw. der Lizenz ist und sich die Leistungen auf die korrektive Wartung beschränken
- Selbständige Leistung im Rahmen eines Softwarewartungsvertrages
- Softwareunterstützung nach Aufwand
 - Wenn der Anwender aus irgendeinem Grund auf die dauernde Wartung nicht angewiesen ist

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

261

Mitwirkung des Anwenders

- Ist der Anwender verpflichtet, die letztgültige Programmversion einzusetzen?
 - Was passiert, wenn er dies nicht tut?
- Ist der Anwender zur Mitteilung der Änderung der Einsatz- und Betriebsbedingungen verpflichtet?
- Der Anwender soll die gemeinsam festgelegten Richtlinien für Fehlermeldungen einhalten
- Wie weit muss der Anwender bei der Analyse, Test, und Rekonstruktion der Fehlerbedingung mitwirken, welche Ressourcen muss er zur Verfügung stellen?

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

262

Software Wartungs-Verträge – Weitere Vertragsbestandteile

- Abgrenzung von Wartung und GWL
- Upgrade Policy bei neuen Versionen
- Übertragung der Software Wartung (z.B. bei Veräußerung des Systems)
- Infrastruktur für die Wartung (Lieferant und Kunde)
- Zeitpunkt der Ausführung von Fehlerkorrekturen, Produktivstellung
- Reaktionszeiten und -modalitäten und Eskalationsverfahren bei Nicht-Reagieren
- Sorgfaltspflicht in Bezug auf Wahrung von Datensicherheit und Datenschutz bei der Ausführung von Wartungsarbeiten
- Entgelt und Zahlungsbedingungen
- Vertragsdauer

© J. Weidl-Rektenwald 02-07

nach [Wolff01]

263

Software Wartungs-Verträge

- Weiterführende Literatur
 - „Das Vertragsrecht der Computer Software“
 - Skriptum zur Vorlesung „EDV Vertragsrecht“ an der Abteilung für Verteilte Systeme
 - Dr. Arthur Wolff, 2001
 - Jaburek, Handbuch der EDV-Verträge

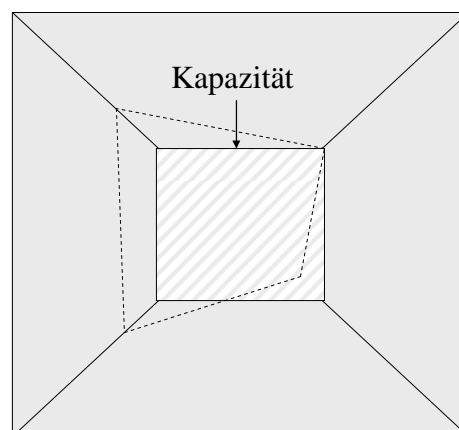
© J. Weidl-Rektenwald 02-07

264

Teufelsquadrat

Qualität

Ressourcen



Projekt-dauer

Wirtschaft-lichkeit²⁶⁵

© J. Weidl-Rektenwald 02-07

Ende