Software-Wiederverwendung (Stichworte)

Udo Kelter

16.05.2018

Zusammenfassung dieses Lehrmoduls

Software-Wiederverwendung wird als ein entscheidendes Mittel angesehen, die Entwicklungszeit und -Kosten von Software zu senken und die Qualität zu erhöhen. Dieses Lehrmodul analysiert zunächst verschiedene Arten von Wiederverwendung (ungeplante, geplante), die möglichen Kosteneinsparungen und an der Wiederverwendung orientierte Vorgehensmodelle.

Vorausgesetzte Lehrmodule:

empfohlen: – Softwarearchitekturstile

Stoffumfang in Vorlesungsdoppelstunden: 0.5

Inhaltsverzeichnis

1 Wiederverwendung

Grundidee: Produkte möglichst weitgehend aus vorhandenen / vorgefertigten Standard-Komponenten konstruieren.

Vorteile:

- Kostensenkung, weniger Neuentwicklung
- Erhöhung der Qualität

eigentlich keine neue Idee, passiert tausendfach:

- Benutzung von Bibliotheken
- Muster, ...

1.1 "Wiederverwendung" von Erfahrung

Wiederverwendbare Artefakte nach Größe sortiert:

1. Programmstrukturen, Konzepte der Programmiersprache
2. Entwurfsmuster, Analysemuster
3. Architekturmuster
4. Standardarchitekturen

sind nur Gestaltungsregeln, die erst "instantiiert" werden müssen, keine Komponenten, die man i.w. unverändert einbaut

überlappt thematisch mit Wiederverwendung

1.2 Wiederverwendung vs. technologische Arbeitsteilung

Fremdherstellung von Komponenten ist in klassischen Industrien üblich:

- ggf. eigenes, umfangreiches Know-How (Domänenwissen) erforderlich, das nicht zu den Kernkompetenzen des Anbieters gehört
- Ausnutzung von Skaleneffekten bei kleinen Produktionsmengen

allgemeines Prinzip der industriellen Arbeitsteilung:

- spezialisierte Marktteilnehmer,
- geringe Fertigungstiefe,
- Konzentration auf Kernkompetenzen

wichtig: nicht zu viele Modelle, Normen / offene Spezifikationen ("DIN-Schrauben")

Besonderheiten in der Informatik:

- Größe der Produktionsmengen irrelevant für SW-Produkte
- "versteckte" technologische Arbeitsteilung durch Benutzung von Netzwerkprotokollen, Graphikpaketen, DBMS, mathematische Bibliotheken, Transaktionsmonitore usw.

  wenig sichtbar, weil keine expliziten Kopien bzw. Lizenzgebühren

→ technologische Arbeitsteilung auch in der Informatik selbstverständlich

ist auch ohne Wiederverwendung sinnvoll (zugekaufte Komponenten werden nur 1* verwendet)

1.3 Wiederverwendung im engeren Sinne

Fälle, wo *mehrere einander ähnliche Systeme* in der gleichen Technologie entwickelt werden

- ggf. in verschiedenen Zeiträumen
- insb. bei Systemen, die nicht Varianten voneinander sind, sondern unabhängig voneinander entwickelt werden
- sofern gleichzeitig: Entwicklung einer **Systemfamilie**

  Besonderheit: gemeinsame, wiederverwendbare Komponenten können *gezielt* bestimmt werden

1.4 Typen wiederverwendbarer Dokumente

im Prinzip beliebige Typen, in beliebigen Entwicklungsstufen:

- Anforderungen, Testfälle

- Quelltexte von Programmen
- Modul-/API-Spezifikationen
- Architekturen bzw. Architekturfragmente
- Datenbankschemata
- Gestaltungselemente von GUIs
- Dokumentation: Bedien-, Installationshandbücher, Glossare

ggf. zusammenhängende Gruppen, z.B. GUI-Komponente bestehend aus Code, Gestaltung, Hilfesystem, Manual

Schwerpunkt in der Praxis: Quelltexte, Architekturen, API-Spezifikationen, Datenbankschemata

1.5 Geplante vs. ungeplante Wiederverwendung

ungeplante Wiederverwendung:

- Komponente wird nach ihrer Entwicklung für Wiederverwendung entdeckt
- muß i.d.R. abgeändert / verbessert / nachdokumentiert werden

geplante Wiederverwendung / Wiederverwendbarkeit:

- Komponente von vornherein zwecks Wiederverwendung entwickelt und gestaltet
- (a) Bibliotheksfunktionen
- (b) Teile eines konkreten Systems, die vermutlich später noch einmal verwendet werden können

2 Grundformen wiederverwendungsorientierter Entwicklungsprozesse

- primär in den mittleren bis späten Phasen
- ohne genaue Kenntnis der Wiederverwendungen

2.1 Grundform: Architektur- und Komponentenwiederverwendung

1. Bilden der (Grob-) Architektur des Systems
2. Suche nach geeigneten Komponenten in einem Vorrat wiederverwendbarer Komponenten;
   ggf. anpassen der Architektur (d.h. zurück zu Schritt 1)
3. gefundene Komponenten in die Entwicklungsversion übertragen
4. Anpassen (abändern) bzw. ggf. Konfigurieren der übernommenen Komponenten

Anmerkungen, insb. zu Schritt 1 + 3:

- i.d.R. bestimmte architektonische Strukturen und Konventionen bei Komponenten vorausgesetzt, z.B. Fehlerbehandlung
  Problem: versteckte Abhängigkeiten
- Probleme umso größer, je heterogener die Entwicklungssprachen / -Technologien
- bei Frameworks ist das Hauptprogramm komplett vorgegeben
  → individuelle Teile des Systems anpassen

Anmerkung zu Schritt 4:

- **black-box-Wiederverwendung**: Komponente wird völlig unverändert wiederverwendet
- **white-box-Wiederverwendung**: Komponente wird verändert

2.2 Kostenanalyse

Vergleich mit klassischer Neuentwicklung:

Schritt 1: wahrscheinlich kein großer Unterschied

Schritt 2: Suche nach Komponenten ist ohne besondere Vorbereitungen problematisch!

Hindernisse:

- finden der entsprechenden Dateien (ggf. auf Archivierungsmedien)
- schlechte Trefferqualität bei Stichwortsuche
- hoher Aufwand zur Analyse der einzelnen Fundstellen, Einschätzung des erforderlichen Anpassungsaufwands unsicher

Schritt 3: Übernahme der Komponente in die Entwicklungsversion arbeitsaufwendig, falls es sich z.B. um verstreute Codefragmente handelt

Schritt 4: ggf. umfangreiche Anpassungsarbeiten;

anschließend kompletter Test der Komponente erforderlich (Aufwand!)

→ reduziert Kosteneinsparung durch die Wiederverwendung

Rechenbeispiele für eine Komponente, deren Neuentwicklung 15 Stunden dauert:

- black-box-Wiederverwendung 0.5 Stunden, kein Suchaufwand
  → Aufwand um den Faktor 30 reduziert!
- langwierige Suche: 2 Stunden
  Änderungen an der Komponente: 6 Stunden incl. Test
  im Durchschnitt 2 vergebliche Suchvorgänge auf einen erfolgreichen
  → Aufwand nur um 20 % reduziert,
  ungünstige Risikostruktur: möglicher Gewinn 7 Stunden,
  möglicher Verlust 2 Stunden

Schlußfolgerungen:

- sehr kleine Komponenten (Aufwand ca. eine Stunde): Wiederverwendung lohnt i.a. nicht

- umfangreiche Komponenten: lohnt im Prinzip, aber reduzierte Wahrscheinlichkeit, daß die Komponente unverändert zum aktuellen Bedarf paßt
  → erhöhte Wahrscheinlichkeit von Änderungen

2.3 Geplantes Wiederverwenden

Voraussetzungen / positive Einflußfaktoren für Wiederverwendung:

- "Firmenkultur", Anreizsystem
- gute Suchfunktionen
- leichte Beurteilung gefundener Komponenten
- möglichst unveränderte Übernahme der Komponenten

... können gezielt verbessert werden → geplante Wiederverwendung

Denkbare vorbereitende Maßnahmen

- Sammlung aller potentiell wiederverwendbaren Komponenten in einem (logisch) zentralen Repository
- zusätzliche Beschreibung der Komponenten anhand eines Klassifikationsschemas und durch Schlagworte
  Möglichkeit, Klassifikation / Schlagworte bei der Suche auszunutzen
- erhöhte Qualität der technischen Dokumentation der Komponenten, der Strukturierung und Lesbarkeit des Programmcodes usw.
- andere (allgemeinere) Gestaltung der Funktionalität und/oder Schnittstellen der Komponenten
- besonders sorgfältiger Test der Komponenten (Vertrauen der Entwickler und Akzeptanz erhöhen)

Mehraufwand im Bereich von 30 - 60 % des normalen Entwicklungsaufwands

Faustregel: Mehraufwand amortisiert sich erst nach nach 3 Wiederverwendungen