Universität Zürich[UZH]

Institut für Informatik

Martin Glinz Thomas Fritz

Software Engineering

Kapitel 18

Produktivitätsfaktoren

18.1 Überblick

18.2 Mehrfachverwendung

Was beeinflusst die Software-Produktivität?

- Menschen
  - Ein dominierender Produktivitätsfaktor
    - Ausbildung
    - Motivation
    - Förderung

    → Folienkapitel 19
- Technische Hilfsmittel: Werkzeuge (→ Folienkapitel 22)
- Projektmanagement (→ Folienkapitel 14 und 19)
  - Gut geplanter Personaleinsatz
  - Vermeidung von Kontextwechsel-Verlusten
  - Vermeidung von Koordinations- und Kommunikationsverlusten
- Mehrfachverwendung

18.1 Überblick

18.2 Mehrfachverwendung

Motivation

Die Masse macht es.

Ziel: Kosten senken für Software

- Entwicklungsproduktivität steigern – Möglichkeiten begrenzt
  - Stückzahlen erhöhen
  - Das Geheimnis der billigen Hardware
  - Bei Software durch Mehrfachverwendung

Verschiedene Formen von Mehrfachverwendung möglich

Formen der Mehrfachverwendung

- Große Produktserien mit identischer Software
- Familien ähnlicher Produkte mit gemeinsamem Kern (software product lines)
- Wiederverwendung selbst entwickelter Software (typisch Komponenten oder Frameworks) in mehreren Produkten
- Verwendung von Bibliotheken und Frameworks Dritter
  - durch Beschaffung
  - durch Nutzung von quelloffener (open source) Software
- Beschaffung oder Lizenznahme (Miete) von Software-Systemen oder -Komponenten
- Nutzung von Software-Diensten (services)

Wiederverwendung selbst entwickelter Software

- Wiederverwendbare Software muss erst einmal geschaffen werden
  - ➪ Komponenten/Dienste bilden, entwickeln und dokumentieren
- Entwicklung wiederverwendbare Software ist teurer als die von Einzweck-Software
  - ➪ Explizite Anreize für Entwicklung wiederverwendbarer Software
- Wiederverwendbares wird nicht gefunden
  - ➪ Kataloge
- Wer pflegt wiederverwendete Software
  - ➪ Pflegeverträge
- Förderung der Wiederverwendung
  - Bereitstellung / Nutzung von Komponenten und Diensten
  - interner Handel mit Komponenten und Diensten
  - Informationsstelle für Wiederverwendung

Verwendung von Bibliotheken und Frameworks

- Das Passende suchen / finden
- Lizenzproblematik, v.a. bei Nutzung quelloffener Software
- Anpassungsproblematik (“white-box re-use”)
- Architekturprobleme bei Verwendung mehrerer Frameworks (“architectural mismatch”)

Beschaffung / Lizenznahme / Nutzung

- Beschaffung
  - Erwerb der Rechte am Objektcode durch Kauf
  - Rechte am Quellcode oft nicht inbegriffen (vertraglich zu regeln)
  - Pflege und Weiterentwicklung zu regeln
  - Als Bestandteil eines eigenen Produkts vertreibbar
- Lizenznahme
  - Erwerb der Nutzungsrechte auf Zeit
  - Keine Rechte am Quellcode
  - Pflege und Weiterentwicklung in der Regel inbegriffen
  - Betrieb der Software auf Rechnern des Lizenznehmers
- Nutzung
  - Bezahlung nach Nutzungsmenge oder –intensität (pay per use)
  - Betrieb auf Rechnern des Lizenzgebers

Verwendung von Komponenten und Diensten

- Finden geeigneter Komponenten bzw. Dienste (services)
  - mit der gewünschten Funktionalität
  - mit den notwendigen Leistungs- und Qualitätsmerkmalen
- Einbettung in eigene Software
  - Semantik der Schnittstellen?
  - Erprobung/Test erforderlich
- Bei Diensten: Verfügbarkeit sichern
  - Dienstverfügbarkeit
  - Netzverfügbarkeit
- Kosten: Abwägen der Kosten für:
  Nutzung – Lizenzierung – Beschaffung – Eigenentwicklung
- Verwendungsrechte: Verträge erforderlich

Verwendung quelloffener Software

- In der Regel kostenlos (!)
- Aber
  - Anpassungs- und Parametrierungskosten berücksichtigen
  - Vorab zu klären
    - Qualität
    - Wer übernimmt die Pflege
    - Lizenzfragen
  - Wirtschaftlichkeit im Einzelfall prüfen

Mehrfachverwendung im Entwicklungsprojekt

- Entscheidungen müssen früh getroffen werden
  - Teilweise schon vor Projektbeginn
  - Spätestens im Architekturentwurf
- Spezifikation, Test und Installation auch bei Verwendung von Software Dritter notwendig
- Die Beteiligten müssen mitmachen:
  - Nicht alles selbst machen wollen
  - Nicht das Perfekte (und Teure), sondern das Gebrauchstaugliche schaffen
  - Mitarbeiterinnen und Mitarbeiter nicht nach der produzierten Software-Menge beurteilen
  - Suche nach verwendbarer Software von Dritten zum selbstverständlichen Bestandteil jedes Entwicklungsprojekts machen

Literatur

B.H. Barnes, T.B. Bollinger (1991). Making Reuse Cost-Effective. *IEEE Software* **8**(1):13–24.

P. Clements, L. Northrop (2002). *Software Product Lines: Practices and Patterns*. Addison-Wesley.

S. Dustdar, H. Gall, M. Hauswirth (2003). *Software-Architekturen für Verteilte Systeme: Prinzipien, Bausteine und Standardarchitekturen für moderne Software*. Berlin: Springer.

W.B. Frakes, K. Kang, (2005). Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering* **31**(7):529–536.

M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann (2007). Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer* **40**(11):38–45.

C. Szyperski, D. Gruntz, S. Murer (2002). *Component Software: Beyond Object-Oriented Programming*, 2nd edition. Harlow: Addison-Wesley.