Software-Wiederverwendung

Julian Fietkau

Universität Hamburg

5. Dezember 2011

Übersicht

Was bedeutet **Software-Wiederverwendung?**

- Software zu erstellen ist teuer.
- Idee: Warum nicht Software mehrfach verwenden?
- Granularität: von einzelnen Objekten bis hin zu kompletten Systemen.
- Vorteil ist Kostenersparnis, aber viele Fallstricke (je nach Ansatz)

Arten von Wiederverwendung

1. Wiederverwendung von **Anwendungssystemen**
2. Wiederverwendung von **Komponenten**
3. Wiederverwendung von **Objekten und Funktionen**

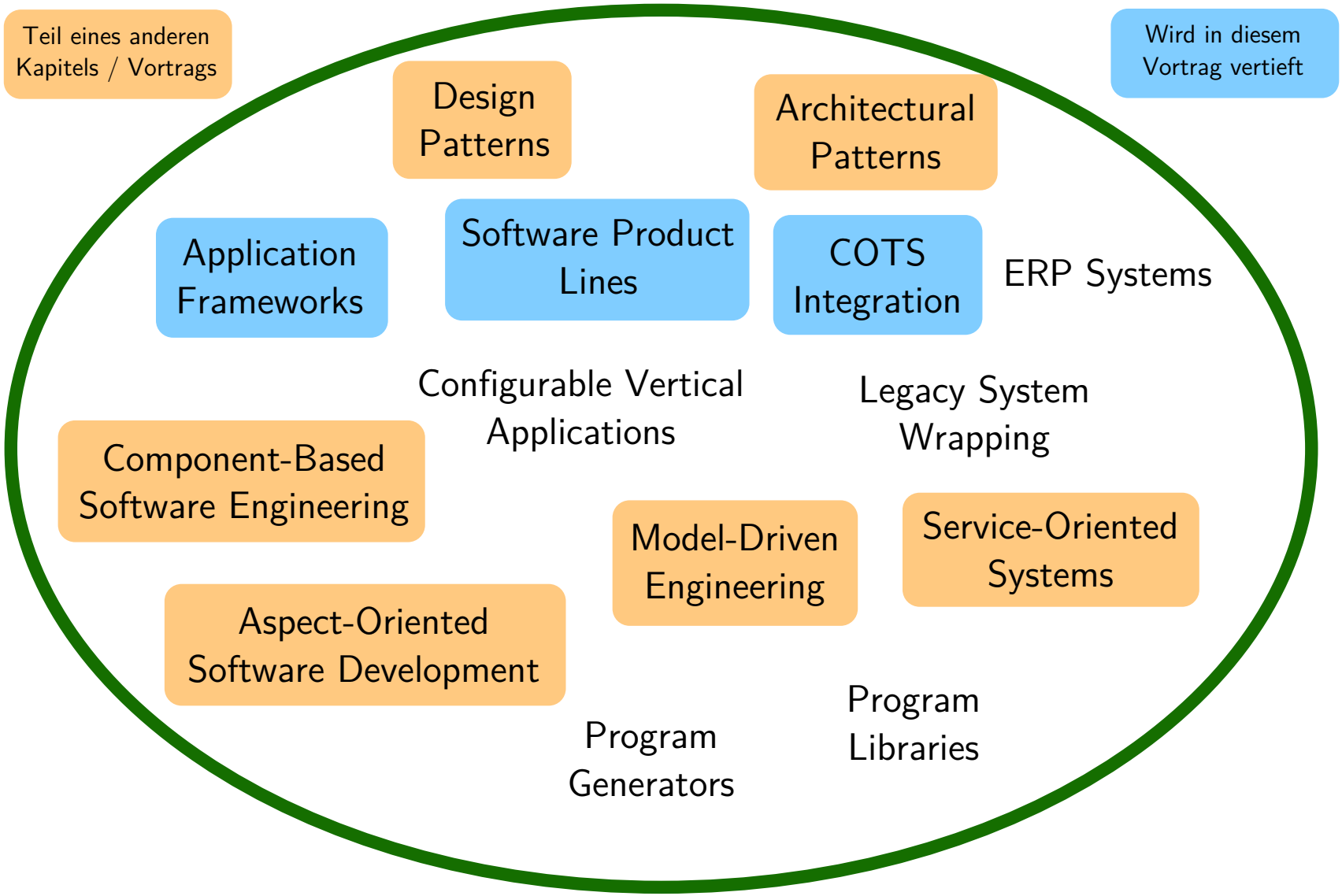
Vorteile und Probleme

Vorteile:

- Erhöhte Verlässlichkeit
- Geringeres Prozessrisiko
- Effektive Nutzung von Spezialisten
- Standardkonformität
- Schnellere Entwicklung

Probleme:

- Erhöhte Wartungskosten
- Mangelnde Werkzeugunterstützung
- „Not-invented-here“-Syndrom
- Pflege einer Komponentenbibliothek
- Korrektes Sammeln und Verstehen von Komponenten



(vgl. [Sommerville 2011], Abb. 16.3 auf S. 429)

Schlüsselfaktoren bei der Wahl des Ansatzes

- Zeitplanung
- Geplante Lebenszeit der Software
- Fähigkeiten der Entwickler
- Nicht-funktionale Anforderungen
- Anwendungsfeld
- Software-Plattform

Frameworks sind...

- ... Sammlungen von Software-Komponenten
- ... allein nicht sinnvoll ausführbar
- ... Schablonen und Hilfsmittel, die für Entwickler leicht nutzbar sind (oder sein sollen)

Arten von Frameworks

- System-Infrastruktur-Frameworks
- Middleware-Integrations-Frameworks
- Enterprise-Anwendungs-Frameworks

Eigenschaften und Auswirkungen

- Implementationszeit wird gespart, in der sonst das Rad neu erfunden worden wäre.
- Nicht-fachlicher Code ist besser gekapselt.
- Die Anwendungen sind an die Plattformen gebunden, die das Framework unterstützt.
- Die Einarbeitungszeit kann ggf. lang sein.

Produktfamilien sind...

- ... mehrere verwandte Anwendungen mit einer gemeinsamen technischen Grundlage / gemeinsamen Komponenten.
- ... beinhalten generische Funktionalität, die an verschiedene Kontexte angepasst werden kann.

Produkt-Instanz-Entwicklung

1. Anforderungsermittlung
2. Passendstes existierendes Produkt auswählen
3. Anforderungen erneut evaluieren
4. Bestehendes Produkt adaptieren
5. Neues Produktfamilien-Mitglied ausliefern

Eigenschaften und Auswirkungen

- Konfiguration hat einen hohen Stellenwert.
- Im Gegensatz zu Frameworks: Fachliche Konsolidierung, nicht ausschließlich technische.
- Eng verwandte Architektur der Anwendungen gewährleistet geringe Einarbeitungszeit für Entwickler beim Wechsel, sowie Wiederverwendung von (z.B. Test-)Infrastruktur.
- Kompromisse zwischen detaillierten Anforderungen und Verwendung der generischen Basis müssen gefunden werden.

COTS-Systeme sind...

- ... „commercial off-the-shelf“.
- ... für verschiedene Kunden anpassbar ohne Veränderungen am Quelltext.
- ... funktional oft sehr umfangreich.

COTS-Lösungs-Systeme...

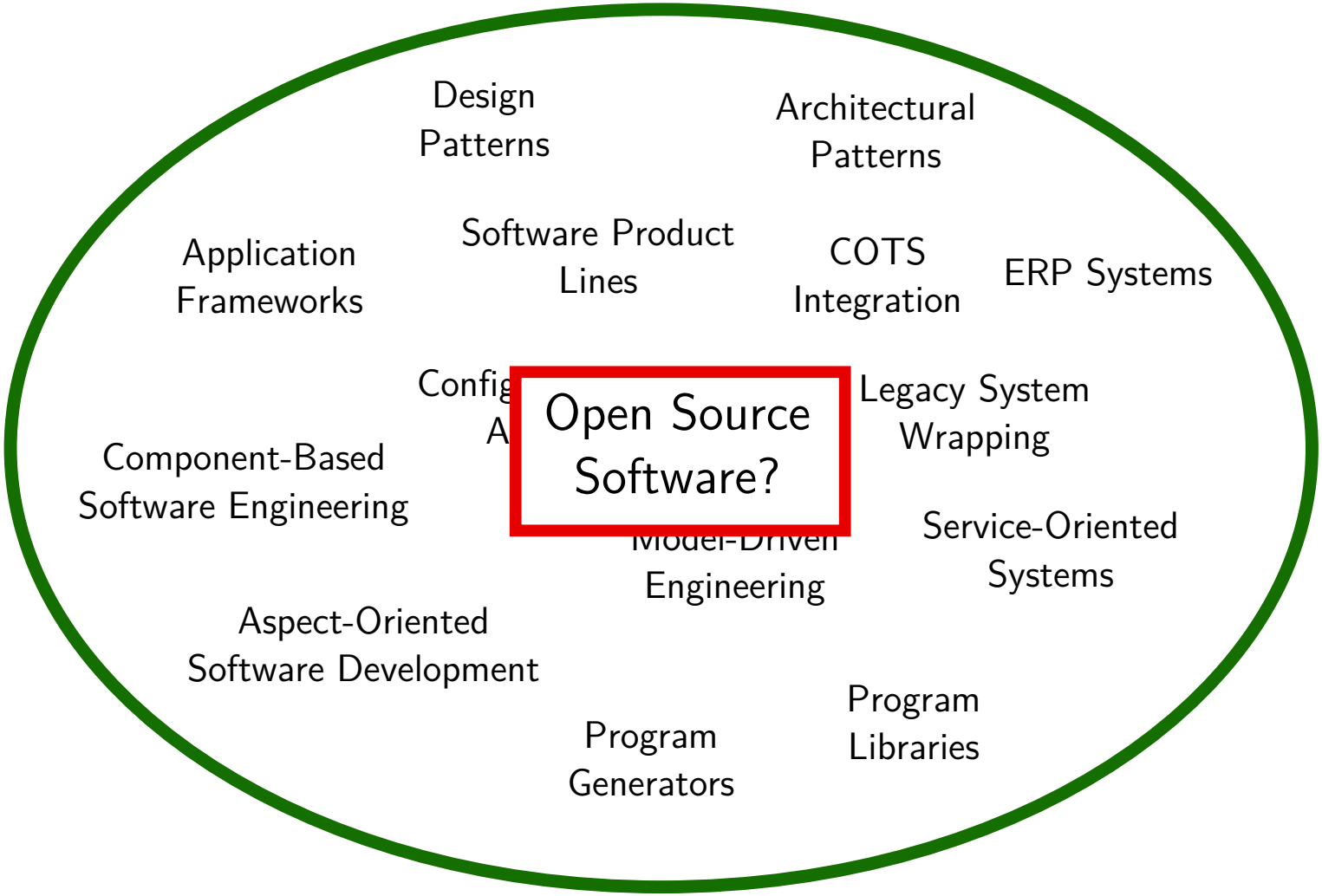
- ... sind generische Anwendungssysteme für einen bestimmten Einsatzzweck.
  - z.B. Universitäts-Verwaltungs-Systeme, Software für Arztpraxen, ERP-Systeme
- ... enthalten verschiedene Module, übergreifende Regeln und eine gemeinsame Datenbank.
- ... erfordern ein hohes Maß an Konfiguration vor der Einsatztauglichkeit.

Integrierte COTS-Systeme...

- ... bestehen aus zwei oder mehr COTS-Produkten.
- ... erfordern Lösungen zum Datenaustausch.
- ... können doppelte Funktionalitäten und größere Fehlerwahrscheinlichkeiten mit sich bringen.

Eigenschaften und Auswirkungen

- Kosten für COTS-Systeme stehen im Vorhinein sehr genau fest. (Aber: Vorsicht vor Betriebs- und Wartungskosten!)
- Kunden haben wenig Kontrolle über die angebotene Funktionalität.
- COTS-Systeme haben eine stetig ungewisse Zukunft.



(vgl. [Sommerville 2011], Abb. 16.3 auf S. 429)

Idee und Praxis

- Code ist nicht zwangsweise ein Geschäftsgeheimnis.
- **Open Source**: Der Code ist von jedermann einsehbar.
- Verschiedene Beschränkungen und Freiheiten zur Nutzung sind möglich (vgl. **Freie Software**).
- Open Source ist heute eigentlich überall irgendwie (vgl. Apache, Firefox, Linux, BSD, Android, SSH...).

OSS-Geschäftsmodelle

Im Unterschied zum Verkauf von Nutzungsrechten für Software:

- Verkauf von Services und Support rund um die Software
- „Commoditize complements“ – OSS-Produkte verleiten Nutzer dazu, verwandte Produkte zu kaufen.
- Kostenvermeidung / Kostenteilung
- Spenden, Crowd Funding

Geschäftsvorteile durch OSS-Engagement

Bei eigenen Produkten:

- Geringere Hürden bei der Markterschließung
- Nutzung von externer Kompetenz für Entwicklung und Qualitätssicherung
- Steigerung der Attraktivität für externe Entwickler
- „Versteinerung“ der Entwicklerbasis wird entgegengewirkt

Bei Beteiligung an bestehenden OSS-Produkten:

- Zugang zur vollständigen Code-Basis, ohne Beschränkungen
- „Meritocracy“-Prinzip: Vermeidung zukünftiger Kompatibilitätsprobleme und Steuerung des Projekts
- Geringerer Infrastruktur-Aufwand durch Vermeidung von „Forks“

„Google versteht Plattformen nicht“

Sagt wer? Und warum?

Steve Yegge und sein großes Versehen

- Steve Yegge: Softwareentwickler und Tech-Blogger, vormals bei Amazon, inzwischen bei Google
- Oktober 2011: Yegge schreibt ein langes Memo gedacht für seine Google-Kollegen. Er postet es versehentlich weltöffentlich.
- Themen seines Textes: Firmenpolitik, Management, aber auch **Plattformen und Software-Wiederverwendung**

Ausgewählte Zitate

Jeff Bezos' „SOA-Mandat“ (Amazon, ca. 2002)

> *„1) All teams will henceforth expose their data and functionality through service interfaces.*
> *2) Teams must communicate with each other through these interfaces.*
> *3) (. . .) The only communication allowed is via service interface calls over the network.*
> *4) It doesn't matter what technology they use. (. . .)*
> *5) (. . .) [T]he team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.*
> *6) Anyone who doesn't do this will be fired.“*

Ausgewählte Zitate

Über Google und Plattformen:

> *„That one last thing that Google doesn't do well is Platforms. We don't understand platforms. We don't "get" platforms. (. . .) It's a big stretch even to get most teams to offer a stubby service to get programmatic access to their data and computations. Most of them think they're building products.“*

> *„But when we take the stance that we know how to design the perfect product for everyone, and believe you me, I hear that a lot, then we're being fools. You can attribute it to arrogance, or naivete, or whatever – it doesn't matter in the end, because it's foolishness. There IS no perfect product for everyone.“*

Ausgewählte Zitate

Über Google+:

> *„Google+ is a prime example of our complete failure to understand platforms (. . .) The Google+ platform is a pathetic afterthought. We had no API at all at launch, and last I checked, we had one measly API call.“*

Über Facebook:

> *„Google+ is a knee-jerk reaction, a study in short-term thinking, predicated on the incorrect notion that Facebook is successful because they built a great product. But that's not why they are successful. Facebook is successful because they built an entire constellation of products by allowing other people to do the work. So Facebook is different for everyone.“*

Wichtigste Punkte

- Es gibt viele Wege zur Software-Wiederverwendung, vgl. Frameworks vs. Produktfamilien vs. COTS vs. Open Source.
- Geschickte Wiederverwendung bietet Möglichkeiten zur Kostensenkung, Beschleunigung der Entwicklung und Steigerung der Verlässlichkeit.
  - Fallstricke gibt es aber auch, man denke an Konfigurationsschwierigkeiten, Inkompatibilitäten etc.

Literatur

Ian Sommerville: *Software Engineering*

9th edition. **Pearson**, 2011

Steve Yegge auf Google+ über Google und Plattformen

Öffentlich durch Genehmigung des Autors, 12. Oktober 2011.

https://plus.google.com/112678702228711889851/posts/eVeouesvaVX

Weblinks

Ian Sommerville: Software Engineering 9

http://www.cs.st-andrews.ac.uk/~ifs/Books/SE9/

Wikipedia: Open source

http://en.wikipedia.org/wiki/Open_source

Wikipedia: Open-source software

http://en.wikipedia.org/wiki/Open-source_software

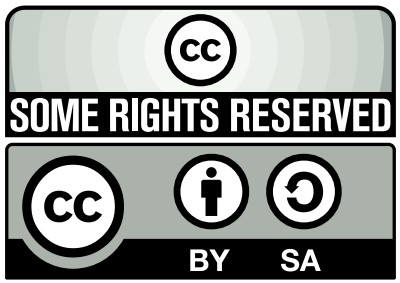
Wikipedia: Comparison of open source and closed source

http://en.wikipedia.org/wiki/Open_source_vs._closed_source

Reddit: Why should my team contribute to open source?

http://www.reddit.com/r/opensource/..._to_open_source/

Freigabe und Download



Folien-Download und Feedback-Möglichkeit:

http://www.julian-fietkau.de/software_wiederverwendung