Evolution wiederverwendbarer Schnittstellen in der Produktentwicklung

Simon Giesecke, Niels Streekmann

BTC Business Technology Consulting AG, Oldenburg / Berlin
Geschäftseinheit Energie Produkte
simon.giesecke@btc-ag.com
niels.streekmann@btc-ag.com

Abstract: Im Rahmen der Entwicklung individualisierbarer Standardprodukte bei der BTC AG spielt die Wiederverwendung von Standardschnittstellen eine besondere Rolle. Besonders relevant ist dabei die Betrachtung von API- und ABI-Kompatibilität. Dieser Beitrag beleuchtet die dabei zu betrachteten Dimensionen sowie Strategien zum Umgang mit inkompatiblen Änderungen.

In der Geschäftseinheit Energie Produkte entwickelt die BTC AG mehrere Softwareprodukte für die Energiewirtschaft, bei denen es sich vorwiegend um individualisierbare Standardprodukte handelt. Bei der Entwicklung stützt sie sich auf die Wiederverwendung, wobei ein Schwerpunkt auf einer Reihe von Standardschnittstellen liegt [Sie04], über die flexible Optionen hinsichtlich der Ausprägung von Qualitätseigenschaften und technischen Rahmenbedingungen geschaffen werden können [FFGL11]. Dabei bedeutet Wiederverwendung hier insbesondere, dass die Einsatzkontexte von entwickelten Schnittstellen, Komponenten und Frameworks nicht vorab abschließend bekannt sind und sich diese auch in getrennten Quelltextrepositories befinden. Hierbei stellen sich für die Evolution besondere Herausforderungen:

- Berücksichtigung der Abwärtskompatibilität beim Entwurf
- Unterstützung der Ersetzung von Schnittstellen und Komponenten durch kompatible Versionen durch Provisionierungswerkzeuge

Wir unterscheiden Schnittstellen und Komponenten im komponenten-orientierten Sinne, bei denen diese getrennte Artefakte mit separatem Lebenszyklus darstellen, und Frameworks, die zwar eine Schnittstelle exportieren, welche aber an eine einzige Implementierung gebunden ist. Für Schnittstellen besonders relevant sind dabei API- und ABI-Kompatibilität (Application Programming/Binary Interface).

Für Komponenten ist die API-Kompatibilität bei korrekter Anwendung des Liskov'schen Substitutionsprinzips nicht von Bedeutung. Die ABI-Kompatibilität ist nur von sehr untergeordneter Bedeutung, da hier lediglich eine sehr schmale Schnittstelle von außen genutzt wird (Factories u.ä.). Wesentlicher sind hier Kompatibilität der Konfiguration und ggf. von persistenten Daten (Datenbankschemata, Dateiformate), Übertragungsprotokollen und ähnlichem. Für Frameworks sind sowohl API- als auch ABI-Kompatibilität relevant, und zudem von besonderer Komplexität, z.B. wegen der Verwendung offener Templates (in C++).

Weitere Kompatibilitätsaspekte (z.B. konzeptionelle Kompatibilität) werden von uns nicht näher betrachtet. Von Kompatibilität zu unterscheiden sind zudem Portabilität zwischen verschiedenen Plattformen oder die Interoperabilität zwischen verschiedenen Compilern auf derselben Plattform.

API-/ABI-Kompatibilität hat verschiedene Dimensionen:

1. Syntaktische vs. semantische Kompatibilität
2. Quelltext- vs. Binärkompatibilität
3. Strenge vs. lockere Kompatibilität
4. Client- vs. Provider-Kompatibilität

Im Zusammenhang mit der Komponenten-orientierten Entwicklung hat die letzte Dimension besondere Bedeutung, wobei Standardschnittstellen besonders kritisch sind. Clients einer Schnittstelle A sind hierbei Module, die die Schnittstelle A nutzen, Provider sind Module, die die Schnittstelle A implementieren. Bei vielen Änderungen ist es so, dass sie entweder für Clients (z.B. Entfernen einer Methode) oder für Provider (z.B. Hinzufügen einer Methode) inkompatibel sind. Aus diesem Grund erlaubt z.B. auch COM keinerlei Änderungen vorhandener Schnittstellen. Eine Standardschnittstelle, wie sie hier verstanden wird, ist aber auf einer höheren Granularitätsebene angesiedelt und entspräche einer Sammlung von COM-Schnittstellen. Hierin liegt auch der Lösungsansatz: Ein Hinzufügen einer optionalen Schnittstelle zu einer Sammlung feingranularer Schnittstellen ist sowohl für Clients als auch Provider kompatibel. Damit diese sinnvoll genutzt werden kann, muss dies jedoch bereits vor der ersten Erweiterung bedacht und geeignete Maßnahmen müssen ergriffen werden.

Der Vortrag beleuchtet die genannten Dimensionen anhand von konkreten Anwendungsszenarien und erläutert ihre Bedeutung für die Produktentwicklung auf der Basis von Standardschnittstellen:

- Szenario 1: Identifikation eines Bugs in einer Komponente oder einem Framework nach Auslieferung an einen Kunden
- Szenario 2: Anforderungen aus der Produktentwicklung, die nur durch Erweiterung einer Standardschnittstelle umgesetzt werden können

Des Weiteren werden Strategien vorgestellt, um inkompatible Änderungen mittels eines geeigneten Prozesses zu steuern, mit Unterstützung von Werkzeugen zu erkennen und sie bekannt zu machen.

Literatur

[FFGL11] Frenzel, M.; Friebe, J.; Giesecke, S.; Luhmann, T.: Standardschnittstellen als nichtfunktionale Variationspunkte: Erfahrungen aus der EPM-Produktlinie. In Reussner, R.; Pretschner, A.; Jähnichen, S. (Hrsg.): Software Engineering 2011 Workshopband, Workshop Produktlinien im Kontext: Technologie, Prozesse, Business und Organisation (PIK2011), Lecture Note in Informatics, Volume P-184, S. 253–264, 2011.

[Sie04] Siedersleben, J.: Moderne Softwarearchitektur, dpunkt Verlag, 2004.