Entscheidungskriterien beim Entwurf von Architekturen langlebiger Softwaresysteme im Kontext von Vendor-Lock-in- und Netz-Effekten

Dr. Simon Giesecke, Falko Basner, Dr. Jörg Friebe
BTC Business Technology Consulting AG
Kurfürstendamm 33, 10719 Berlin
{simon.giesecke,falko.basner,joerg.friebe}@btc-ag.com

Zusammenfassung

Der Vendor-Lock-in-Effekt bezeichnet die Abhängigkeit einer Investition von einem bestimmten Anbieter. Dieser kommt umso stärker zur Geltung, je länger die Nutzungsdauer der Investition (also z.B. die voraussichtliche Lebensdauer eines Softwaresystems) ist. Aus dem Vendor-Lock-in entstehen Risiken. Netzeffekte hingegen bezeichnen den Nutzen, den ein Anwender einer Technologie aus der Tatsache ziehen kann, dass es neben ihm weitere Anwender dieser Technologie gibt. Eine vollständige Vermeidung des Vendor-Lock-ins ist sowohl wirtschaftlich unsinnig als auch praktisch unmöglich. Dem (negativen) Vendor-Lock-in-Effekt gegenüber stehen (positive) Netzeffekte bei der Nutzung verbreiteter Technologien.

Ökonomisch rationale Entwurfsentscheidungen beim Entwurf von Architekturen langlebiger Softwaresysteme können nur unter sorgfältiger Abwägung beider Arten von Effekten getroffen werden. Wir begründen in diesem Artikel, dass Maßnahmen zur Reduzierung des Vendor-Lock-ins getroffen werden sollten, die die Wirkungen des Netzeffekts nur minimal verringern.

1 Einleitung

Bei der Entwicklung komplexer Softwaresysteme werden immer auch Komponenten von Drittanbietern genutzt. Wir verwenden die Ausdrücke „Softwarekomponenten“ bzw. „Komponenten“ in einer sehr weit gefassten Bedeutung (z.B. im Vergleich mit komponentenorientierter Softwareentwicklung), die z.B. auch Bibliotheken, objektorientierte Frameworks und die Laufzeit- und Entwicklungs-Infrastruktur umfasst. Komponenten können unterschieden werden in individuell für das System entwickelte Komponenten und Standardkomponenten. Letztere werden typischerweise von Drittanbietern bezogen. Das „System“ in diesem Sinne besteht nur aus den individuell entwickelten Komponenten.

Die Nutzung von Standardkomponenten erzeugt immer Abhängigkeiten von dem Anbieter dieser Komponenten. Dies wird auch als Vendor-Lock-in-Effekt bezeichnet. Dieser wird üblicherweise je nach Art der Komponente mehr oder weniger bewusst in Kauf genommen oder zu minimieren versucht. Beispielsweise wird man selten eine eigene Programmiersprache definieren, sondern eine möglichst verbreitete Programmiersprache nutzen, die dem technischen Problem einigermaßen angemessen ist (nicht selten wird letzteres auch überhaupt nicht betrachtet). Die hierdurch erzeugte Abhängigkeit ist allerdings sehr stark, da im Falle eines Austauschs jede einzelne Codezeile ersetzt werden muss. Man kann sagen, die Komponente „Programmiersprache“ ist in hohem Maße invasiv. Die Vorteile ihrer Nutzung liegen jedoch auf der Hand: Netzeffekte rechtfertigen diese Entscheidung, betrachtet man allein die Verfügbarkeit von Entwicklern und Werkzeugen.

Mit diesen Abhängigkeiten korrespondieren Risiken für die Weiterentwicklung des Systems. Diese Risiken bestehen in einer Auseinanderentwicklung von Anforderungen des Systems und Eigenschaften der Standardkomponenten. Dies kann seine Ursache in einer Änderung der Anforderungen des Systems oder der Eigenschaften der Standardkomponente, in einer Diskontinuität der Entwicklung der Standardkomponente, schlimmstenfalls in der Einstellung ihrer Wartung. Die Wahrscheinlichkeit, dass ein solches Ereignis irgendwann eintritt, steigt natürlich mit der Zeit. Daher kommt diese Problematik bei langlebigen Softwaresystemen besonders zum Tragen, bzw. muss überhaupt erst explizit adressiert werden.

Ein weiterer Grund für die besondere Bedeutung für langlebige Softwaresysteme liegt darin, dass mit steigender Lebensdauer eines Systems der Gesamtentwicklungsaufwand sowie eine spätere Flexibilität einen immer höheren Stellenwert gegenüber der Zeitdauer bis zur initialen Verfügbarkeit des Systems gewinnt (natürlich nur in gewissen Grenzen), so dass bei einem über 30 Jahre genutzten Netzleitsystem Entwurfsentscheidungen sicher anders ausfallen werden als bei einer Webanwendung für die Entgegennahme von Anträgen auf „Abwrackprämie”, das nach wenigen Monaten wieder außer Betrieb genommen wird.

Die hier angestellten Überlegungen ergänzen in gewisser Weise ein systematisches Abhängigkeitsmanagement [3], das zunächst völlig unabhängig von der Frage der Nutzung von Individual-

im Vergleich zu Standardkomponenten ist.

Mangels gesicherter Erkenntnisse werden diese Entscheidungen allerdings oft ohne einen expliziten Entscheidungsprozess getroffen, im günstigsten Fall aufgrund früherer Erfahrungen, deren Übertragbarkeit auf die aktuelle Situation unterstellt wird. Ziel dieses Artikels ist es, die Voraussetzungen dafür zu schaffen, diese Art sehr folgenreicher Entwurfsentscheidungen auf eine fundiertere Grundlage stützen zu können.

Die Alternativen bei einer solchen Entwurfsentscheidung umfassen typischerweise folgende Kategorien von Möglichkeiten:

- Nutzung einer Standardkomponente ohne risikomindernde Maßnahmen
- Nutzung einer Standardkomponente mit risikominderndern Maßnahmen
- Vollständige Individualentwicklung

Im Folgenden werden wir Entscheidungskriterien vorstellen, die in diesem Zusammenhang eine Rolle spielen.

2 Entscheidungskriterien

Wir haben eine initiale Liste von Entscheidungskriterien identifiziert, die für den Vendor-Lock-in- und den Netz-Effekt eine Rolle spielen. Diese sind:

- Kosten für Austausch der Standardkomponente
  - Invasivität der Standardkomponente
- Kosten und Nutzen von Maßnahmen zur Reduzierung des Vendor-Lock-ins, soweit technisch machbar
  - Kosten für Entwurf, Implementierung, Wartung einer Abstraktionsschicht
  - Auswirkung auf Nutzen der Standardkomponente
  - Auswirkung auf die Invasivität
- Kosten für die Beendigung der Abhängigkeit von dem Anbieter
  - Interne Übernahme der Quellbasis der Standardkomponente
  - Vollständiger Austausch der Standardkomponente durch andere Standardkomponente
    - Verfügbarkeit von Alternativen
    - Konfliktierende Architekturstile

Eine zentrale Rolle bei der Entscheidung spielen nach unserer Einschätzung Invasivität und konfliktierene Architekturstile. Beide Faktoren spielen nicht ausschließlich bei der Nutzung Standardkomponenten eine Rolle. Sie treten aber in besonderem Maße bei von Drittanbietern bezogenen Komponenten in Erscheinung, da deren Evolution nicht koordiniert erfolgt.

Invasivität Ein Aspekt des Vendor-Lock-in ist die Invasivität einer Komponente im Hinblick auf den Code des entwickelten Systems. Eine einfache Metrik für die Invasivität ist der Anteil der Codezeilen, die bei einem Austausch der Standardkomponente potenziell geändert werden müssen. Dies exakt zu ermitteln, ist auch nicht einfach, aber eine grobe Abschätzung lässt sich in vielen Fällen leicht ermitteln. Dies ist jedoch nur ein Aspekt, da dies noch keine Aussage über den Aufwand macht, der zur Änderung dieser Codezeilen nötig ist. Die notwendigen Aktivitäten können von automatisierbaren Transformationen bis zu tiefgreifenden Änderungen in der Architektur des Systems reichen.

Konfliktierende Architekturstile Bei der Verwendung verschiedener Komponenten können konfliktierende Grundannahmen über den verwendeten Architekturstil Probleme bereiten, die wiederum Adaptionslösungen erfordern, die z.B. Wartbarkeits- oder Performanceprobleme mit sich bringen [2]. Zu solchen Konflikten gehören z.B. unterschiedliche Annahmen a) hinsichtlich der Synchronität der Kommunikation, b) bezüglich der Zentralität und Synchronisation der Datenhaltung, c) bezüglich der Datenhoheit, d) bezüglich der Ausführungsmodelle.

3 Ausblick

Die Überlegungen zu den Entscheidungskriterien müssen vertieft werden. Da die tatsächlichen Konsequenzen einer Entwurfsentscheidung bzgl. der Nutzung externer Komponenten nur langfristig sichtbar werden, ist es wichtig, ein geeignetes theoretisches Modell zu finden und dies mit (ggf. historischen) Daten zu validieren. Als Grundlage für die theoretische Modellierung erscheint die Realoptionen-Theorie geeignet, die bereits zur Modellierung verschiedener Arten von Entwurfsentscheidungen verwendet wurde [1, 4].

Aus den bisherigen Überlegungen wird jedoch bereits deutlich, dass die grundlegende Entscheidung zugunsten oder zulasten der Nutzung irgendeiner externen Komponente für ein bestimmtes Anwendungsgebiet nicht abstrakt getroffen werden kann, d.h. ohne die verfügbaren Komponenten und ihre Auswirkungen auf die Architektur zu bewerten. Konkret haben wir bereits mit vertiefte Überlegungen zur Auswirkung von Abstraktionsschichten zur Risikominimierung begonnen.

Literatur

[1] Bahsoon, R., W. Emmerich und J. Macke: *Using real options to select stable middleware-induced software architectures.* Software, IEE Proceedings, 152(4):167–186, 2005.

[2] Garlan, David, Robert Allen und John Ockerbloom: *Architectural Mismatch: Why Reuse Is So Hard.* IEEE Softw., 12(6):17–26, 1995.

[3] Martin, Robert C.: *Clean Code.* mitp, 2009.

[4] Sullivan, Kevin J., William G. Griswold, Yuanfang Cai und Ben Hallen: *The structure and value of modularity in software design.* In: *ESEC / SIGSOFT FSE*, Seiten 99–108, 2001.