Advicio Research Note
Datum: 2. September 2005
Autor: Tonio Grawe
tonio.grawe@advicio.com
Datei: ARN20050905

Advicio®
The Service Architects

IT-Service-Katalog: Was folgt der Bestandsaufnahme?

IT-Dienstleister sind heute dabei, ihre Services zu erfassen, um einen IT-Service-Katalog zu erstellen. Meist wird dabei eine Dekomposition komplexer Services durchgeführt, um überschaubare Teilleistungen zu erhalten. So entsteht eine Sammlung von einigen hundert Service-Komponenten, und man stellt möglicherweise fest:

Eine Sammlung von Service-Komponenten ist noch keine Service-Architektur

Das erhoffte Ziel, die identifizierten Service-Komponenten als Bausteine bei der Entwicklung neuer Services wiederzuverwenden, wird nicht erreicht, weil die Komponenten nicht kompatibel zueinander sind. An dieser Stelle muss ein Entwicklungsprozess einsetzen, durch den die Komponenten aneinander angepasst werden. Nur so können die Komponenten in einem Netzwerk gegenseitiger Abhängigkeiten zusammenarbeiten. Die beiden häufigsten Ursachen von Inkompatibilitäten sind:

1. Eine unterschiedliche Terminologie. Beispielsweise wenn für eine Komponente „QoS“ definiert ist, obwohl Verfügbarkeit gemeint ist. Und wie ist eigentlich „Verfügbarkeit“ definiert? Eine gemeinsame Sprache muss durch ein Glossar geregelt sein.
2. Grundsätzliche Design-Aspekte. Dies betrifft einheitliche Regelungen für Verfügbarkeiten, Betriebszeiten et cetera. Sind solche Parameter für jede Service-Komponente individuell festgelegt, so ist es mutig bis riskant, für einen daraus aggregierten Service eine Qualitätsgarantie abzugeben.

Die Anzahl der Service-Komponenten auf das Mindestmaß reduzieren durch Konsolidierung und Varianten

Typischerweise sind es zu viele Service-Komponenten. Mehrere hundert sind üblich, in Einzelfällen wurden auch über tausend Service-Komponenten identifiziert. Eine reiche Auswahl ist in einem gut sortierten Service-Baukasten erwünscht. Allerdings wirken sich mehrfach erfasste, identische Service-Komponenten kontraproduktiv aus. Erfahrungsgemäß kann die Anzahl durch Konsolidierung um 15% reduziert werden. Je nach Historie des IT-Dienstleisters – insbesondere Fusionen wirken sich aus – kann die Konsolidierung auch weit größere Effekte haben.

Desweiteren hilft die Einführung von Varianten, die Anzahl zu minimieren. Allerdings ist die Entscheidung, ob zwei ähnliche Service-Komponenten Varianten voneinander sind, oder ob es sich doch um verschiedene Service-Komponenten handelt, nicht trivial. Advicio empfiehlt diesbezüglich folgende Regelung: Varianten haben eine identische Funktionalität und unterscheiden sich lediglich in der Qualität (Verfügbarkeit, Bereitstellungsdauer, Betriebszeit, etc.), in der Kapazität und im Preis bzw. in den Kosten.

Übersicht gewinnen durch Strukturierung

Im nächsten Schritt soll die Übersichtlichkeit erhöht werden, indem die Service-Komponenten in sinnvolle Gruppen eingeteilt werden. Doch nach welchem Kriterium soll gruppiert werden? Drei Ansätze sind augenscheinlich möglich:

1. Nach der Organsiationsstruktur, also welche Einheit die Leistung erbringt.
2. Nach der verwendeten Technologie.
3. Nach der durch den Service bereitgestellten Funktionalität.

Eine Reihe von Gründen spricht für die Funktionalität als Kriterium für die Gruppenbildung. So entsteht aus den Teilleistungen „Betrieb Oracle“, „Betrieb DB2“ und „Betrieb MySQL“ die Gruppe „Betrieb RDBMS“.

Die Gruppe ist eine Oberklasse. Wir bezeichnen diese auch als abstrakte Service-Komponente. Abstrakte Service-Komponenten können nicht instantiiert werden, hingegen können konkrete Service-Komponenten wie „Betrieb MySQL“ tatsächlich umgesetzt werden. Die abstrakten Oberklassen dienen lediglich der Strukturierung des Komponentenkatalogs.

Entsprechend kann nun wiederum für mehrere Gruppen eine gemeinsame Oberklasse gefunden werden, so dass letztlich alle Service-Komponenten in einer übersichtlichen Baumstruktur einsortiert sind. Eine solche Struktur nennen wir eine Service-Taxonomie.

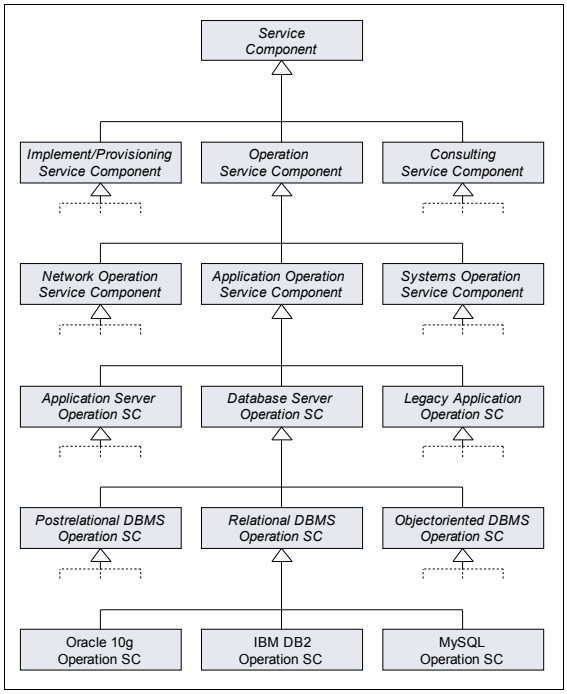

Abb. 1: Service-Taxonomie.

Eine Service-Architektur entsteht durch die semantischen Beziehungen der Service-Komponenten zueinander

Wie in Abbildung 1 zu sehen ist, entstehen jede Menge dieser abstrakten Oberklassen, deren Namen gemäß UML-Notation kursiv geschrieben werden. Was ist der praktische Nutzen von diesem Overhead?

Wie eingangs beschrieben unterscheidet sich eine Service-Architektur von einer Ansammlung unterschiedlicher Service-Komponenten durch die Beziehungen der Service-Komponenten zueinander. Durch die Service-Taxonomie werden die semantischen Beziehungen der Service-Komponenten hergestellt. Dadurch wird semantisches Wissen erfasst. Wenn man Beispielsweise nicht wüsste, was *MySQL Operation SC* sein könnte, so verrät die Taxonomie, dass es sich um den Betrieb einer relationalen Datenbank handelt.

Die semantischen Beziehungen helfen auch im IT-Service-Engineering bei der Spezifikation von Service-Komponenten. Soll etwa definiert werden, dass ein Anwendungsservice die Betriebsdienstleistung einer relationalen Datenbank erfordert, ohne festlegen zu wollen, welche dies ist, so wird auch auf eine abstrakte Oberklasse verwiesen. Man erhält dadurch einen Freiheitsgrad, den man ohne Taxonomie nur durch wartungsintensive Konfigurationsregeln erreichen könnte.

Vererbungsmechanismen aus der Objektorientierung nutzen, um Standards und Konsistenz zu wahren

Aus der objektorientierten Software-Entwicklung sind die Vererbungsmechanismen bekannt, die sich nun auch beim IT-Service-Engineering bewähren. Damit können Eigenschaften bei den abstrakten Service-Komponenten definiert werden und gelten dann durch die Vererbung für alle davon abgeleiteten Service-Komponenten. Zwei Beispiele dazu:

- Wir definieren ganz oben in der Taxonomie, dass es zwei Qualitätsvarianten geben soll: Standard und Premium. Durch Vererbung gilt dies für alle Service-Komponenten.
- Alle Betriebsdienstleistungen in der Standard-Variante haben Betriebszeiten zu Bürozeiten, die Premium-Varianten 24x7. Dies wird in der *Operation Service Component* definiert und von dort vererbt.

Diese Beispiele verdeutlichen, dass die Vererbung dazu führt, Service-Komponenten gewissen Standards und Vorgaben folgend zu spezifizieren. Wie eingangs festgestellt, ist das die Voraussetzung dafür, aufeinender abgestimmte, integrierbare Service-Komponenten – eben eine Service-Architektur – zu erhalten.

Ein weiterer Nutzen ist, dass die Entwicklung neuer Service-Komponenten wesentlich schneller geht, weil sie auf bereits bestehenden aufsetzen und lediglich in Details angepasst werden müssen.