SConf: Ein flexibles Architekturmodel für das Konfigurationsmanagement

Ralf Döring
Technische Universität Ilmenau
ralf.doering@tu-ilmenau.de

Abstract. Im folgenden Beitrag werden ausgehend von der Definition der Begriffe Dienst, Dienstmanagement und Dienstkonfiguration Anforderungen an ein experimentelles Architekturmodell für die Untersuchung unterschiedlicher Realisierungsaspekte eines Konfigurationsmodells für Dienste aufgestellt. Anhand dieser Anforderungen wird eine konkrete Modellarchitektur vorgestellt, die eine flexible Betrachtung unterschiedlicher Modellbestandteile sowie deren Implementierung ermöglicht.

1 Einführung

Aufbauend auf den mittels des Verbindungsnetzwerkes geschaffenen Kommunikationsgrundlagen bilden *Dienste* die zweite und aus Sicht des Anwenders wohl wichtigere Abstraktionsebene eines verteilten Systems. Traditionell nimmt jedoch die Verwaltung der Kommunikationsressourcen einen ungleich höheren Stellenwert ein. Die dazu notwendigen Technologien und Vorgehensweisen werden unter dem Begriff des *Netzwerkmanagementes* zusammengefasst. Das Dienstemanagementes nimmt demgegenüber einen untergeordneten Stellenwert ein, was sich sehr einfach an Anzahl und Entwicklungsgeschichte der verwendeten Managementsysteme veranschaulichen lässt. Im Kontext dieses Beitrages soll ein Dienst folgender einfacher Definition genügen:

> *Dienste sind* grobgranulare *Einheiten, die einem* Dienstnutzer *eine* Dienstleistung *zur Verfügung stellen. Im Gegensatz zum etablierten Dienstverständnis in verteilten Systemen fallen auch Anwendungsprogramme für Endnutzer in diese Kategorie.*

Ausgehend von dieser einfachen Definition fällt es leicht, eine Adhoc-Definition des Begriffes *Dienstemanagement* zu finden:

> *Dienstmanagement kann als Summe verschiedener Teilaufgaben beschrieben werden:*

	Dienstlokalisation
+	*Dienstkonfiguration*
+	*Dienstnutzung*
=	*Dienstmanagement*

Im weiteren Kontext dieses Beitrages soll der Begriff der *Dienstkonfiguration* im Vordergrund stehen. Analog zur Definition des Dienstmanagements lässt sich hier folgende Beziehung aufstellen:

	Initialkonfiguration des Dienstes
+	Dienstintrospektion
+	Parameteränderung im laufenden Betrieb
+	Zustandsbestimmung beim Shutdown
=	Dienstkonfiguration

Ausgehend von diesem Zusammenhang soll der weitere Fokus auf dem Aspekt der Dienstkonfiguration liegen.

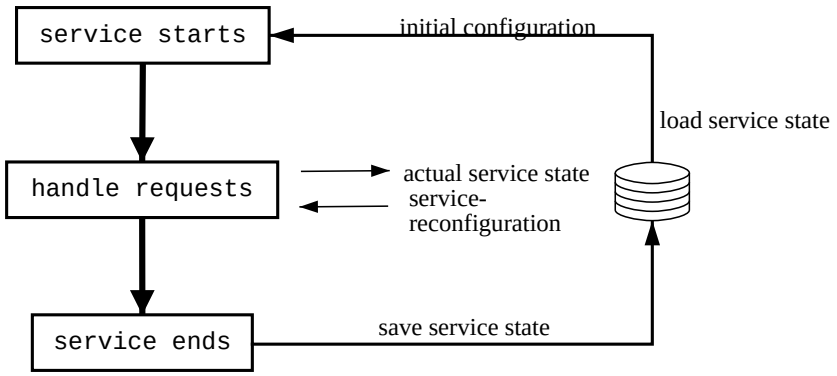


Abbildung 1. Lebenszyklus eines Dienstes

Im folgenden wird ein experimentelles Architekturmodell vorgestellt, welches für die Untersuchung von Aspekten der Dienstkonfiguration herangezogen werden soll. Dessen Aufgaben lassen sich am besten anhand der Beschreibung des Lebenszykluses eines Dienstes in Abbildung 1 erläutern.

Ein System zur Konfiguration von Diensten soll somit in der Lage sein

1. einen Dienst mit seiner Initialkonfiguration zu versorgen
2. im laufenden Betrieb den aktuellen Dienststatus zu bestimmen
3. die Konfiguration eines Dienstes im laufenden Betrieb zu ändern
4. den kompletten Status eines Dienstes beim Dienstende zu sichern und für die nächste Initialkonfiguration zu verwenden (wenn dies sinnvoll und machbar ist)

2 Anforderungen

Zur Untersuchung unterschiedlicher Ansätze und Realisierungen für die Speicherung, Verarbeitung und Abfrage von Parametern beliebiger Dienstinstanzen wurde ein flexibles Modell gesucht, welches als Testumgebung für unterschiedliche Modell- und

Implementierungsansätze dienen kann. An diese Testumgebung wurden folgende Anforderungen gestellt:

1. Die Testumgebung soll unabhängig von der Wahl des zugrundeliegenden Entwicklungsmodells für verteilte Anwendungssysteme sein. Eine parallele Verwendung verschiedener Zugriffsmechanismen soll nicht von vornherein ausgeschlossen werden. Als Beispiel soll der Zugriff sowohl mittels SOAP als auch mittels CORBA realisiert werden.

2. Unabhängigkeit der internen Realisierung des Konfigurationsmodell von den Zugriffsmechanismen des verwendeten verteilten Systems. Es muss somit eine Abstraktion stattfinden, mittels der aus Sicht des Modellkerns Unterschiede bezüglich des verwendeten Kommunikationsmodells ausgeglichen werden.

3. Das interne Informationsmodell zur Repräsentation von Konfigurationsparametern ist so einfach wie möglich zu halten, um eine weitestgehende Flexibilität zu erreichen.

4. Eine Einschränkung des Zugriffs auf Informationen soll möglich sein, wobei eine Klassifikation des zugreifenden Subjektes vorgenommen werden muss.

5. Das Testsystem muss die persistente Speicherung der enthaltenen Informationen unterstützen.

6. Das Informationsmodell ist unabhängig von der Art der Informationsspeicherung. Die gleichzeitige Nutzung verschiedener Speichermechanismen soll nicht von vornherein ausgeschlossen werden.

Bei genauer Betrachtung der obigen Anforderungen scheint eine Unterteilung in eine geschichtete Architektur sinnvoll. Als Hauptbestandteil kommt eine grobe Unterteilung in

- Distributed System Access Layer
- Information Layer
- Storage Layer

in Betracht. Die einzelnen Teile sind gegebenenfalls weiter zu strukturieren.

3 Das Architekturmodell

In diesem Abschnitt soll ein Architekturmodell vorgestellt werden, welches die aufgestellten Anforderungen erfüllen kann. Das Hauptaugenmerk wurde dabei auf einen möglichst flexiblen Aufbau des Gesamtsystems geachtet, der eine weitgehende Austauschbarkeit der Komponenten gewährleisten soll.

Abbildung 2 zeigt einen Vorschlag für eine Realisierung der aufgestellten Anforderungen mittels mehrerer Komponenten, auf die im folgenden genauer eingegangen werden soll. Dieses Modell sowie die daraus resultierende Implementierung erhalten den Namen *SConf* (*S*ervice *Conf*iguration).

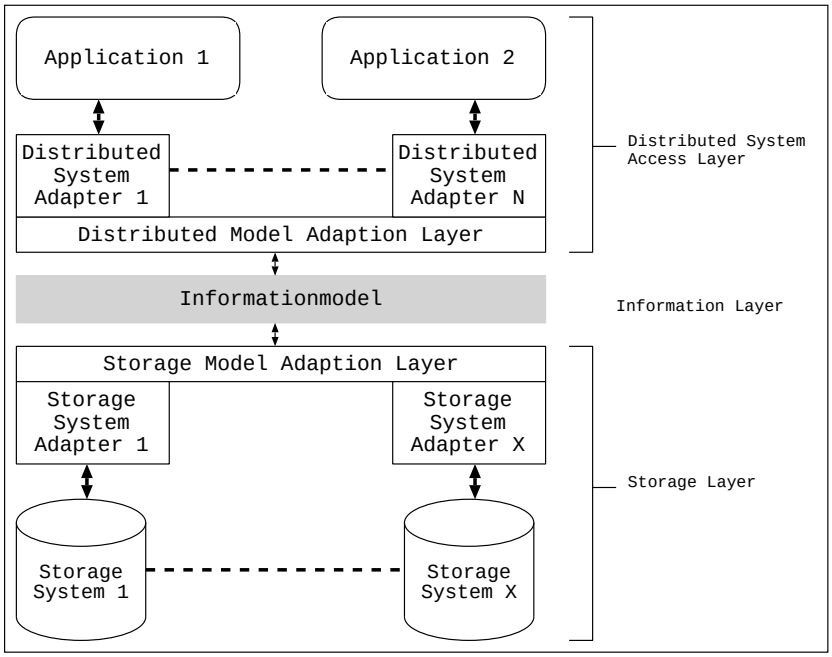


Abbildung 2. Architektur von SConf

3.1 Information Layer

Kernbestandteil der *Information Layer* und des kompletten Modells ist das *Informationmodel*. Hier werden Festlegungen darüber getroffen, welche Informationen das System bereithält und wie diese Informationen strukturiert sind. Zur Realisierung der Speicherung von Konfigurationsparametern für Dienste wird hier der bereits in [4] vorgeschlagene Ansatz verfolgt. Diese Schicht verwaltet somit die eigentlichen Parameter zur Konfiguration von Diensten. Wesentliche Teilaufgaben umfassen dabei:

- Schaffung eines Namensraumes für Dienste und deren Konfigurationsparameter.
- Eventbasierte Verteilung von Parameteränderungen an registrierte Observer.
- Definition von Parametern und Attributtypen sowie der dazu notwendigen Beschreibungsmittel.
- Verwaltung von Berechtigungen für den Parameterzugriff.
- Speicherung von Versions- und History-Informationen.

Als Beschreibungsmittel für die Struktur der verwalteten Information mittels XML steht eine DTD zur Verfügung, die u.a. *Dienstgruppen*, *Dienste*, *Dienstinstanzen* und *Parameter* mit ein oder mehreren *Attributen* enthält. Die eigentliche Speicherung von Werten für eine Dienstkonfiguration ist mittels integraler Datentypen (String, Integer,

Float, Boolean) sowie deren Zusammenfassung zu beliebig strukturierten Listen und Dictionaries möglich.

3.2 Distributed System Access Layer

Die *Distributed System Access Layer* umfasst alle Bestandteile, die für den Zugriff auf das System mittels unterschiedlicher verteilter Systemmodelle zur Verfügung stehen. Grundlegende Bestandteile sind die *Distributed System Adapter* sowie die *Distributed Model Adaption Layer*.

Aufgabe des *Distributed System Adapter* ist die Anbindung an die unterschiedlichen Modelle für verteilte Systeme sowie die Abbildung von Konzepten des Informationsmodells (Parameternamen und Attributtypen) auf die spezifischen Erfordernisse des jeweiligen Systems. Es findet somit eine Umsetzung von konkreten Konzepten unterschiedlicher verteilter Systeme hin zu einer einheitlichen Anbindung an das Informationsmodell. Derzeit existieren *Distributed System Adapter* für CORBA und (ansatzweise) für SOAP.

3.2.1 Distributed Model Adaption Layer

Die verschiedenen *Distributed System Adapter* kommunizieren nicht direkt mit dem zugrundeliegende Informationsmodell, sondern werden nochmals durch die *Distributed Model Adaption Layer* von diesem getrennt. Diese weitere Abstraktion erscheint sinnvoll, um bestimmte Basisfunktionalitäten zu separieren, die von den einzelnen *Distributed System Adaptern* genutzt werden können bzw. die deren zur Verfügung gestellte Funktionalität erweitern sollen.

Die bisher für diese Schicht spezifizierten Dienste sind

1. Nutzeridentifikation und Authentifizierung.
2. Eindeutige Requestidentifizierung und Zuordnung.
3. Schaffung eines einheitlichen Modells für die Benachrichtigung über Parameteränderungen.

3.3 Storage Layer

Die *Storage Layer* dient der persistenten Speicherung von Informationen über den Lebenszyklus der Implementierung hinaus. Auch hier sind, analog zu den *Distributed System Adaptern*, verschiedene *Storage System Adapter* vorgesehen, um eine Unabhängigkeit vom verwendeten Speichersystem zu erreichen. Derzeit existieren Adapter für ein sehr simples Speicherverfahren mittels XML, bei dem der gesamte Inhalt des Informationsmodells beim Shutdown des System in einer XML-Datei gespeichert wird. Als zweite Realisierung existiert eine Abbildung auf Berkeley `dbm` Datenbanken. An einer Anbindung an LDAP sowie Postgres wird derzeit gearbeitet. Die *Storage Model Adaption Layer* wiederum abstrahiert von den Eigenschaften der jeweils verwendeten *Storage System Adapter* und stellt eine einheitliche Schnittstelle zur persistenten Speicherung der Daten zur Verfügung.

4 Zusammenfassung und Ausblick

Das vorliegende Architekturmodell soll der Untersuchung unterschiedlicher Realisierungsmöglichkeiten und Implementierungsansätze für ein allgemeines Konfigurationsmodell für Dienste dienen. Es sollen unter anderem folgende Aspekte untersucht werden:

- Verteilungsaspekte der Bestandteile des Informationsmodells
- Realisierungsmöglichkeiten der Introspektion von Diensten im laufenden Betrieb mittels Push- oder Pullmodell.
- Welche Speicherkonzepte bieten sich an?
- Inwieweit sind Vorgaben für Parameter sinnvoll, die für mehrere Dienste oder Dienstegruppen gelten?
- Welche Verfahren sind anwendbar, wenn sich Parameter eines solchen gemeinsamen Parametersets ändern?

Im Rahmen der Architektur werden in einem nächsten Schritt die Verteilungsaspekte des Modells auf unterschiedliche Netzknoten untersucht.

Literatur

[1] : *Directory Enabled Networks Standards*. – URL `http://www.dmtf.org/spec/denh.html`

[2] ATTIYA, Hagit ; WELCH, Jennifer: *Distributed Computing: Fundamentals, Simulations amd Advanced Topics*. McGraw-Hill Publishing Company, 1998

[3] DEMMLER, Rudi: *Untersuchung von Konzepten zur automatisierten Verwaltung von Diensten und Ressourcen im Netz*, TU Ilmenau, Diplomarbeit, 2000

[4] DÖRING, Ralf: Towards a Service Configuration Model TU Ilmenau (Veranst.), 2002. – 8th Netties Conference, unpublished

[5] HEGERING, Heinz-Gerd ; ABECK, Sebastian ; NEUMAIR, Bernhard: *Integriertes Management vernetzter Systeme*. dpunkt-Verlag, 1999

[6] KAMPMAN, Kevin ; KAMPMAN, Christina: *All about Network Directories: Understanding Directory Services and Business Applications*. Wiley Computer Publishing, 2000

[7] STRASSNER, John: *Directory enabled networks*. Macmillan Technical Publ., Indianapolis, Ind., 1999. – ISBN:1-578-70140-6