# Konfiguration dynamischer Webservices in adhoc Dienstenetzen

## - Eine Vorhabenbeschreibung -

Birgit Wendholt

HAW-Hamburg
Hochschule für angewandte Wissenschaften Hamburg
Berliner Tor 7
20099 Hamburg, Deutschland
wendholt@informatik.haw-hamburg.de

**Abstract.** Gegendstand der Untersuchung ist die Konfiguration von Webservices in adhoc Netzen. Adhoc Netze bilden sich zufällig aus Dienstanbietern (Providern) und Dienstnachfragern (Consumern). Typisch für diese Netze ist, dass Dienste zum Zeitpunkt (t) verfügbar, zum Zeitpunkt (t+x) nicht mehr auffindbar, und zum Zeitpunkt (t+x+k) erneut verfügbar sind. Damit ist eine neue Anwendungsklasse für Konfiguration und Rekonfiguration von Diensten identifiziert. Der Beitrag entwirft eine Peer-Architektur für Webservice Provider und Consumer von dynamisch rekonfigurierbaren Webservices und stellt eine Lösungsskizze für das Konfigurationsproblem vor. Abschließend werden vergleichbare Ansätze sowie Vorarbeiten besprochen.

## 1 Szenario und Problemstellung

Paul, wohnhaft in Hamburg, erfährt von einem Open Air Konzert in Berlin, das am folgenden Wochenende stattfindet. Er reserviert umgehend ein Ticket. Nun fehlt ihm nur noch die Mitfahrgelegenheit und das Begleitprogramm. Er beauftragt seinen digitalen Assistenten, die Teilnahme an dem Open Air in Berlin zu organisieren. Der digitale Assistent weiss, dass er dazu eine Mitfahrgelegenheit nach Berlin für den Freitag und die Rückfahrt für Samstag früh buchen muss. Als Zusatzbedingungen gelten: für die Wartezeiten vor dem Event soll ein Restaurant und für die nach dem Event soll eine Bar auszumachen werden. Der Besuch des Restaurants sollte möglichst mit Personen verabredet werden, die auch an dem Open Air teilnehmen. Die Wartezeiten sollten möglichst kurz, die Mitfahrgelegenheiten möglichst preiswert und die Personen möglichst freundlich sein. Kann der Transport nicht organisiert werden kann, dann soll die Reservierung für das Open Air Event rückgängig gemacht werden.

Als Randbedingungen gelten: Einzeldienste sollen solange reserviert werden, bis ein brauchbares Angebot konfiguriert werden konnte. Einzeldienste werden erst im zweiten Schritt verbindlich gebucht. Es sollte möglich sein, mehrere gleichwertige Dienste zu reservieren, um Alternativen konfigurieren zu können. Der

digitale Assistent soll auf allen Kanälen in stationären und insbesondere in adhoc Netzen um den Aktionsradius von Peter nach Dienstangeboten suchen.

Die Konfigurationsaufgabe besteht darin, für ein Vorhaben mehrere Einzeldienste (Webservices) für das Gesamtziel zu kombinieren. Die Webservices müssen vorgegebene Eigenschaften erfüllen, dürfen z.B. ein Budget an Zeit oder Geld nicht überschreiten, oder sollten gewisse Effekte haben, z.B. zur Zielerfüllung eines Nutzer (ereignisreicher Abend, informative Veranstaltung) beitragen. Die Eigenschaften der Einzeldienste sind die Constraints in der Optimierungsaufgabe für eine Konfiguration. Darüberhinaus muss sich die Konfiguration dynamisch an ändernde Situationen anpassen können, wenn Webservices adhoc hinzukommen oder sich abmelden. Die charakterischen dynamischen Komponenten für dieses Szenario sind:

- ein (oder mehrere) Anbieter, die zum Zeitpunkt der Buchung nicht mehr im adhoc Netz erreichbar sind, so dass Teildienste nicht mehr gebucht werden können.
- ein (oder mehrere) Anbieter, die während des Buchungsvorgangs das adhoc Netz verlassen.
- ungültige Dienste, wenn die Reservierungszeit für einen Dienst ausgelaufen ist
- Dienste, die erst während einer laufenden Konfiguration bekannt werden.

Konfiguration in adhoc Netzen ist daher charakterisiert durch ständige Korrektur und ist somit inhärent eine Rekonfigurationsaufgabe.

## 2 Architektur

Provider und Consumer sind Komponenten eines kollaborativen, dezentralen Peersystems, die autonom ohne Zentrale interagieren und adhoc untereinander ein Netzwerk aufbauen. Provider bieten Dienste an und liefern damit den Datenbasis für die Konfiguration, Consumer fragen Dienste nach und konfigurieren Aggregate, die die Randbedingungen erfüllen. Es können sowohl mehrere Provider als auch mehrere Consumer in einem adhoc Netzwerk miteinander kommunizieren. Um ein lose gekoppeltes, interoperables und standardisiertes Netzwerk zu etablieren, kommen die Webservice Standards SOAP und WSDL für die Verwendung von Diensten und der WS Dynamic Discovery Standard (vgl. [3]) für das Auffinden und Annoncieren von Diensten zum Einsatz. Die Rollen Provider und Consumer prägen sich in zwei spezifischen Architekturen aus.

Abb. 1 zeigt den typischen Aufbau einer Provider Architektur. Sie besteht aus einer Anwendungsschicht, die die typischen anwendungsbezogenen Dienste bereitstellt. Diese variieren je nach Szenario. Die Anwendungsschicht verwendet die Basisdienste einer adhoc Middleware, um u.a.

- Dienste im Netzwerk an- und abzumelden (WS Dynamic Discovery)
- Dienste auszuführen (Webservice Runtime)
- Reservierungen für Dienste mit Gültigkeitszeiträumen zu versehen (Leases).

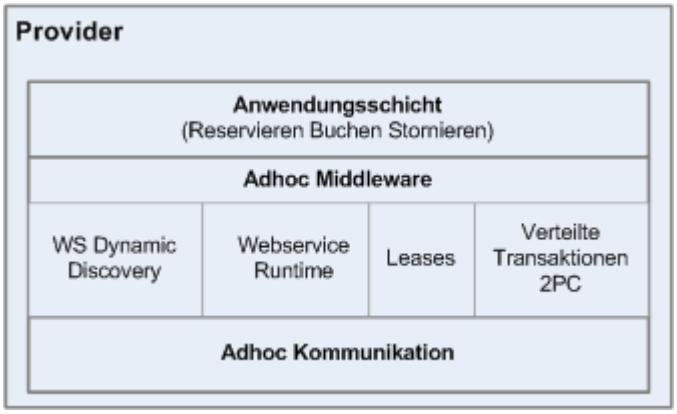


**Fig. 1.** Architekturkomponenten eines Providers

– an verteilten Buchungstransaktion teilnehmen zu können (2PC).

Eine vollständige Infrastruktur verfügt darüber hinaus über Benutzerverwaltung und Security Dienste. Die Protokolle und Basisdienste der Provider benötigen wenig Ressourcen, so dass als Laufzeitumgebung auch eingebettete Systeme (z.B. Mobiltelefone) möglich sind.

Den Verbindungsaufbau zu den Teilnehmern eines adhoc Netzes übernimmt die Schicht der Adhoc Kommunikation. Dazu gehören die Bekanntgabe und das Abmelden eines Teilnehmers (Provider / Consumer) im adhoc Netz sowie das Auffinden anderer Teilnehmer.

Consumer (siehe auch Abb. 2) sind die Komponenten, die komplexe Dienste aus Einzeldiensten der Provider konfigurieren. Consumer verfügen über Rechneressourcen, die es erlauben, einen komplexen Konfigurator zu betreiben. Hier kommen z.B. Subnotebooks als Laufzeitumgebungen in Frage. Die Anwendungsschicht des Consumers bildet die typische Funktionalität eines Szenarios ab. Die Anwendungsschicht verwendet die gleichen Basisdienste wie der Provider, um Dienste in einem Adhoc Netz auffinden, reservieren und buchen zu können.

Die Entscheidung darüber, welche Dienste in der Anwendungschicht verwendet werden, trifft der Konfigurator in einem definierten Protokoll zwischen den Einzelkomponenten:

Die Engine berechnet gültige Konfigurationen auf der Basis der verfügbaren Einzeldienste. Die Engine wird vom Konfigurator aktiviert, wenn neue Dienste entdeckt werden.

Der Tranformator übersetzt die Dienste in eine Engine konforme Darstellung und verwaltet die offenen und genutzen Dienste. Die Strategie für die Wahl der zu nutzenden Dienste, etwa dann, wenn mehrere Provider einen gleichwertigen Dienst anbieten, wird über Benutzerpräferenzen abgebildet.

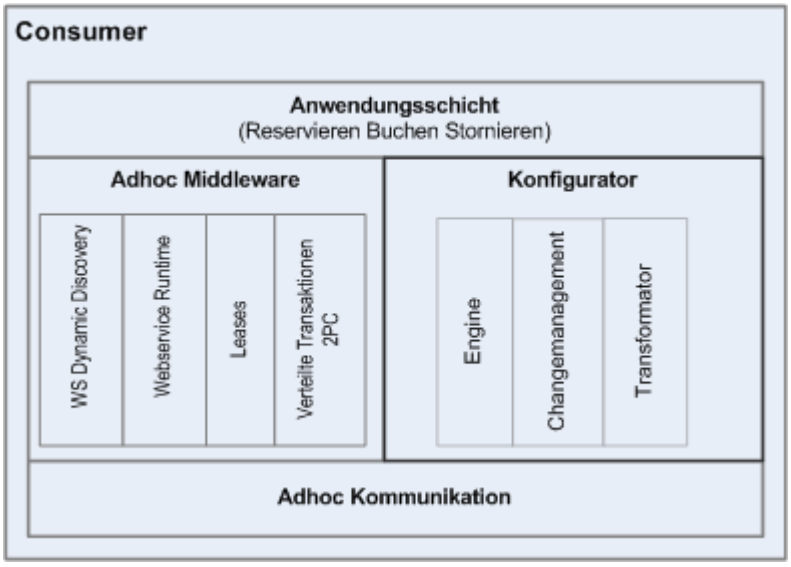


**Fig. 2.** Architekturkomponenten eines Consumers

Das Changemanagement

- verwaltet den Zustand gültiger Konfiguration
- registriert sich beim Konfigurator auf die Dienste, die in diesen Konfigurationen verwendet wurden
- informiert die Engine über ungültige Konfigurationen, wenn Vorbedingungen verletzt sind
- und iniitiert die Rekonfiguration durch die Engine

Wenn die Engine eine gültige Konfiguration an den Konfigurator meldet, dann initiiert dieser die Buchung der Einzeldienste. Sind während der Buchung Einzeldienste nicht verfügbar (nicht mehr erreichbar, bereits gebucht oder nicht mehr gültig, da die Reservierung ausgelaufen ist), dann informiert der Konfigurator das Changemanagement, die Konfiguration wird ungültig und die Rekonfiguration erneut angestossen.

## 3 Lösungsskizze für das (Re)Konfigurationsproblem

In erster Näherung soll das Konfigurationsproblem auf ein Constraint Solving Problem abgebildet werden. Dazu sollen die Dienste, die zur Erreichung einer komplexen Dienstleistung notwendig sind, in Form von Constraints dargestellt werden. Das Constraint Problem wird als Optimierung von QoS Eigenschaften

der Dienste, wie Preis, Dauer etc. formuliert. Die Belegungen der Constraintvariablen mit realen Diensten definieren die Lösung als Aggregat aus Einzeldiensten.

Die Changemanagement Komponente des Konfigurators verwaltet die Beziehung zwischen den Diensten und den Verarbeitungszuständen der Engine. Die Engine soll mittels eines existierenden Constraint Solvers realisiert werden. Aktuell werden die Engines ILOG Solver (siehe [4]) und der CLP Solver des Computational logic, Languages, Implementation, and Parallelism Laboratory der Unversität Madrid (siehe [5]) auf Eignung untersucht.

Die Engines sollten das Backtracking auf Konfigurationen und die Verarbeitung sich sukzessive aufbauender Datenmengen unterstützen sowie die Verarbeitung logischer Ausdrücke (Prädikate) in den Constraints zulassen.

## 4 Einordnung

Der hier vorgestellte Ansatz ist ähnlich zu dem Vorgehen in [2]. Legt vorliegende Arbeit den Schwerpunkt auf die Konfiguration von Webservices, ist dort der Fokus der Konfigurationsaufgabe die Allokierung von hardwarenahen Ressourcen für eine Anwendung.

CAWICOMS Fokus (vgl. [1]) liegt auf der semantischen Konfiguration von Webservices im B2B Umfeld. Der Ansatz berücksichtigt jedoch keine sich dynamisch verändernden Konfigurationsgrößen.

[6] gibt einen vollständigen Überblick über Konfigurations- und Rekonfigurationsprobleme in technischen und nicht technischen Domänen, diskutiert hingegen nicht die Domäne der dynamischen adhoc Systeme.

## 5 Vorarbeiten

Im Rahmen von Diplom und Bachelorarbeiten entstehen die Basisdienste für Provider und Consumer. Erste Ergebnisse sind die Implementierung eines WS-Dynamic Discovery für Webservices in adhoc Netzen sowie die Implementierung eines SOAP Server für Java ME.

## References

1. Felfering, A., Friedrich, G., Jannach D., Zanker M.: Semantic Configuration Web Services in the CAWICOMS Project, Proceedings of the First International Semantic Web Conference on the Semantic Web, p. 192-205, June 09-12, 2002
2. Poladian, V., Sousa P., Garlan, D., Shaw, M.: Dynamic Configuration of Resource-Aware Services, Proceedings of the 26th International Conference on Software Engineering (ICSE'04), 2004
3. Beatty J., et.al.: Web Services Dynamic Discovery (WS-Discovery), 2004, Microsoft
4. The ILOG Solver: http://www.ilog.com, last accessed: 24.06.2005
5. http://clip.dia.fi.upm.es
6. John, U.: Konfiguration und Rekonfiguration mittels Constraint-basierter Modellierung, Promotionsschrift, Feb., 2002, Akademische Verlagsgesellschaft