Live Service Composition

AUSARBEITUNG SoftwareArchitektur

Philipp Ewen

Jan Zipfler

2014

Inhalt

[Abstract 2](#_Toc382772760)

[Grundlagen 3](#_Toc382772761)

[Serviceorientierte Architektur 3](#_Toc382772762)

[Registry 3](#_Toc382772763)

[Service 3](#_Toc382772764)

[Service Composition 3](#_Toc382772765)

[Choreografie 3](#_Toc382772766)

[Orchestrierung 4](#_Toc382772767)

[Execution Engine 4](#_Toc382772768)

[Implementierung 5](#_Toc382772769)

[Programmiersprache 5](#_Toc382772770)

[Registry 5](#_Toc382772771)

[Random Service 6](#_Toc382772772)

[IsPrime Service 7](#_Toc382772773)

[Execution Engine/Concatenate Service 7](#_Toc382772774)

[Menü 8](#_Toc382772775)

[Programmablauf mit Bildschirmfotos 9](#_Toc382772776)

# Abstract

Service Composition beschreibt ein Verfahren in serviceorientierten Architekturen (SOA), bei dem neue Services aus bereits bestehenden Services erzeugt werden. Dies kann zum einen über Choreografie und zum anderen über Orchestrierung erfolgen. In dieser Arbeit werden wir anhand einer Beispielimplementierung zeigen, wie es mittels Orchestrierung möglich ist aus vorhandenen Services einen neuen Service zu erstellen. Dieser Erzeugungsprozess findet zur Laufzeit statt, das heißt es ist möglich, während das System läuft einen neuen, zusammengesetzten, Service anzulegen. Um mehrere Services miteinander zu verknüpfen setzen wir eine sogenannte Execution Engine ein, die wir beispielhaft für die Verknüpfungsart „verketten“ implementieren. Bei dieser Verknüpfungsart handelt es sich um ein Verketten von zwei Funktionen im mathematischen Sinne, wie zum Beispiel: . Zur Implementierung benutzen wir die Programmiersprache Go, welche sich besonders gut zur Entwicklung von nebenläufigen und verteilten Anwendungen eignet.

# Grundlagen

Dieser Abschnitt behandelt alle, für das Verständnis, notwendigen Grundlagen und Begriffserklärungen. Dabei werden wir insbesondere die Begriffe Service und Service Composition genau definieren. Außerdem wollen wir hier Techniken erläutern, die später für das Implementieren der Software notwendig sind. Dies sind im Wesentlichen Registry und Execution Engine.

## Serviceorientierte Architektur

Bei eine serviceorientierten Architektur (SOA) handelt es sich laut Wikipedia um folgendes: „SOA ist ein Paradigma für die Strukturierung und Nutzung verteilter Funktionalität, die von unterschiedlichen Besitzern verantwortet wird.“ (Wikipedia, 2014)[[1]](#footnote-1)

## Registry

In einer serviceorientierten Architektur wird eine Registry zum lokalisieren von Services verwendet. Services können sich entweder bei der Registry anmelden, um ihren Standort (z.B.: IP) mitzuteilen oder den Standort eines anderen Services erfragen.

## Service

Bei einem Service handelt es sich um eine autarke Einheit, die eine bestimmte Funktionalität zur Verfügung stellt. Diese Funktionalität wird dann in einem Netzwerk bereitgestellt. Damit ein Service benutzt werden kann, stellt dieser eine Schnittstelle nach außen bereit, welche im Service Contract definiert ist. Dieser Contract umfasst insbesondere: Name und Beschreibung des Services, Rückgabewert und Parameter mit Typ und Beschreibung.

## Service Composition

Service Composition beschreibt das Erzeugen eines neuen Services aus bereits im Netzwerk vorhandenen Services. Um dies zu bewerkstelligen, gibt es zum einen den Ansatz der Choreografie und zum anderen den Ansatz der Orchestrierung. Der Vollständigkeit halber werden im Folgenden beide Varianten definiert. Die später beschriebene Implementierung, wird sich jedoch nur mit dem Thema Orchestrierung beschäftigen.

### Choreografie

„Eine Choreographie beschreibt die Aufgaben und das Zusammenspiel mehrerer Prozesse unter dem Aspekt der Zusammenarbeit.“ [[2]](#footnote-2)

Ingo Melzer beschreibt weiterhin, dass Prozesse miteinander kooperieren und diese Kooperation in dazugehörigen Choreographie-Dokumenten beschrieben wird. Dazu wird kein neuer Service erzeugt, sondern die erwähnten Regeln für die Zusammenarbeit angewandt. Jeder Prozess trägt somit einen bestimmten Teil zum gesamten Prozess bei. Außerdem kann jeder Service nur seinen eigenen Kontext sehen und nicht den gesamten Prozess. Ein Beispiel hierzu bilden nachrichtenbasierte Systeme.

### Orchestrierung

„Eine Orchestrierung beschreibt die ausführbaren Aspekte eines Geschäftsprozesses aus der Sicht des Prozesses.“ [[3]](#footnote-3)

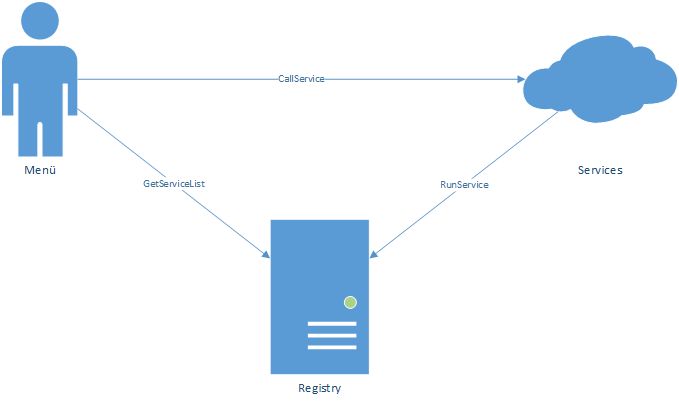
Ingo Melzer beschreibt weiterhin, dass Orchestrierung die klassische Sicht beschreibt, bei der eine dedizierte Instanz (Prozess), alle Aktivitäten steuert.

## Execution Engine

Die Execution Engine dient dem Ausführen eines zusammengesetzten Services (Service Composition). Dazu benötigt die Execution Engine eine Definition dieser Service Composition. Hierfür verwenden wir in dieser Arbeit die Namen der Services, welche in der Form verkettet werden sollen.

# Implementierung

In diesem Kapitel werden wir erläutern, welche Methoden und Werkzeuge wir zum Implementieren unserer Live Service Composition verwendet haben. Dazu zählen vor allem detaillierte Informationen über die Kommunikation zwischen den einzelnen Services, der Registry und dem Menü. Die folgende Grafik illustriert die Struktur unserer Anwendung und vermittelt einen Überblick darüber, wie die einzelnen Komponenten miteinander kommunizieren:



## Programmiersprache

Als Programmiersprache haben wir Go verwendet. Go wurde von Google entwickelt und entstand 2007 im Rahmen eines 20% Projekts. Zu den Stärken von Go zählen unter anderem Netzwerk- und nebenläufige Programmierung. Außerdem ist Go Plattformübergreifend nutzbar, d.h. die Programme können u.a. unter Windows und Linux kompiliert werden.

## Registry

Zu den Aufgaben der Registry in unserem Projekt gehören:

* Das zuordnen vom Namen eines Services zu dessen IP-Adresse und dem dazugehörigem Service Contract.
* Verwalten einer Liste mit allen vorhandenen Services.
* Eigene IP-Adresse über Multicast bekannt geben, damit die Registry ohne Konfiguration von anderen Netzwerkteilnehmern gefunden werden kann (Multicast Discovery).

Aus der Verwendung von Multicast folgt insbesondere, dass für die Multicast Discovery das UDP Protokoll zum Einsatz kommt. Da mit UDP keine gesicherte Verbindung möglich ist kommt sonst immer TCP zum Einsatz. Außerdem verwenden wir für das Übertragen von Daten in allen Fällen JSON (JavaScript Object Notation), welches von Go (ohne zusätzliche Bibliotheken) unterstützt wird.

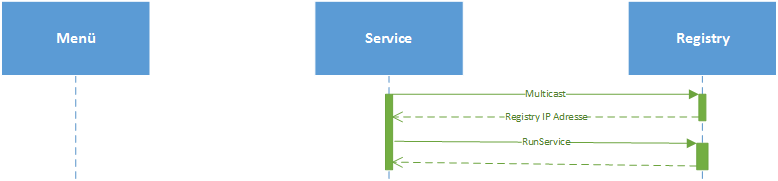
## Random Service

Dieser Service generiert eine Zufallszahl im Bereich von 0 bis .

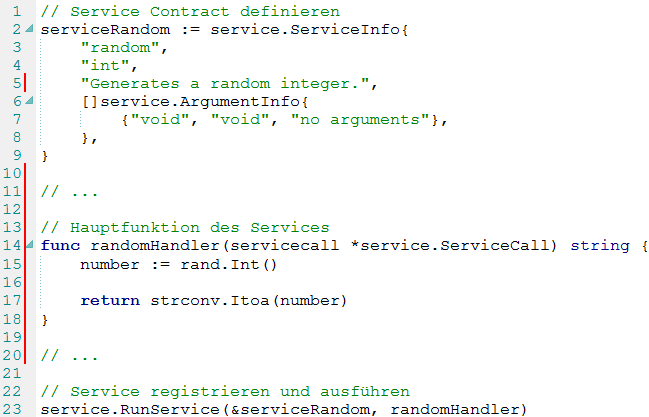
Service Contract:

|  |  |
| --- | --- |
| Service Name | random |
| Beschreibung | Generates a random integer. |
| Parameter | Keine |
| Returns | int |

Sequenzdiagramm: Service registrieren und aufführen



Definition des Random Services:



## IsPrime Service

Dieser Service testet, ob die übergebene natürliche Zahl eine Primzahl ist. Hierzu wird ein 16-Facher Miller-Rabin-Test durchgerührt. Dabei handelt es ich um einen probabilistischen Test, d.h. das Ergebnis gilt mit einer bestimmten Wahrscheinlichkeit (in diesem Fall: ).

Service Contract:

|  |  |
| --- | --- |
| Service Name | isprime |
| Beschreibung | Test if x is prime. (Ausgabe als Text) |
| Parameter | x, int, number to test |
| Returns | string |

## Execution Engine/Concatenate Service

Die Execution Engine führt zwei Services hintereinander aus. Dieses hintereinanderschalten entspricht der mathematischen Operation verketten, und zwar im Sinne von *newservice := service1(service2()).* Dabei hat „service1“ aus Gründen der Einfachheit keine Übergabeparameter, es wäre jedoch ohne weiteres möglich „service2“ mit Parametern aufzurufen.

Service Contract:

|  |  |
| --- | --- |
| Service Name | concatenate |
| Beschreibung | Passes output of service2 to service1 (service1(service2()) and creates a new service. |
| Parameter 1 | service1, string, first service name |
| Parameter 2 | service2, string, second service name |
| Parameter 3 | newservice, string, new service name |
| Returns | string |

Sequenzdiagramm: Zusammengesetzten Service ausführen

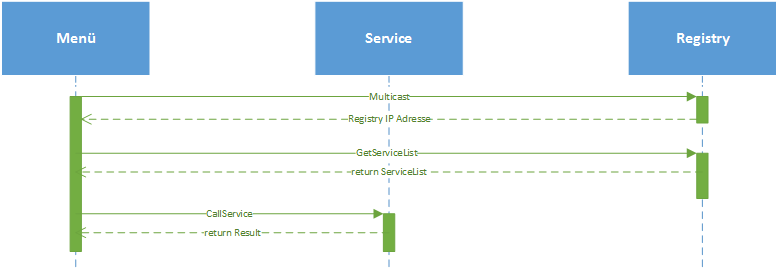


## Menü

Das Menü stellt die Schnittstelle zwischen Mensch und unsrer Implementierung einer serviceorientierten Architektur dar. Dies haben wir durch eine einfache Kommandozeilenschnittstelle realisiert. Die folgenden Operationen stehen dem Benutzer zur Auswahl.

|  |  |
| --- | --- |
| Liste anzeigen | Zeigt eine Liste aller Services an (jeweils mit IP und Service Contract). |
| Service Info anzeigen | Zeigt Informationen für den angegebenen Service an (IP und Service Contract) |
| Service aufrufen | Führt den angegebenen Service aus. Die Parameter, welche vom Benutzer eingegeben werden müssen, werden dynamisch aus dem Service Contract generiert. |

Sequenzdiagramm: Service lokalisieren und aufrufen



# Programmablauf mit Bildschirmfotos

In diesem Abschnitt werden wir erläutern, wie unsere Implementierung der Service Composition verwendet wird. Dazu stellen wir einige Bildschirmfotos bereit, welche alle Schritte zum Erzeugen eines zusammengesetzten Services beschreiben.

Um unsere Implementierung verwenden zu können, werden mindestens die drei nachfolgend aufgelisteten Programmteile benötigt:

1. Registry Server
2. Service(s)
3. Menü / Benutzeroberfläche

Da wir eine Live Service Composition demonstrieren möchten, werden wir alle Services verwenden, die im Abschnitt Implementierung erläutert wurden:

* Random Service
* Primzahltest Service
* Concatenate Service /Execution Engine

Nachdem die Registry und alle Services gestartet wurden, kann das Menü gestartet werden und alle, im Abschnitt Implementierung erwähnten, Menüpunkte aufgerufen werden.

In Abbildung 1 ist die Ausgabe der Serviceliste zu sehen, die detaillierte Informationen zu jedem Service anzeigt.

Abbildung 2 zeigt, den Aufruf des Concatenate Services, der die im Kapitel Implementierung erwähnten Services als Parameter benötigt. Hierbei wird zuerst der äußere (isprime) und dann der innere Service (random) angegeben. Der dritte Parameter (randomprimetest) ist der Name des neuen Services, der zur Laufzeit neu erstellt wird.

Die durchgeführten Aktionen, können außerdem auf dem Server, auf dem die Registry läuft, nachvollzogen werden. Ein Beispiel für eine solche Protokollierung ist in Abbildung 3 zu sehen.

Die letzte Abbildung, zeigt das Ausführen des neuen Services (randomprimetest), welcher eine Zufallszahl einem Primzahlentest unterzieht und das Ergebnis ausgibt.



Abbildung 1 Auflistung aller Service Contracts

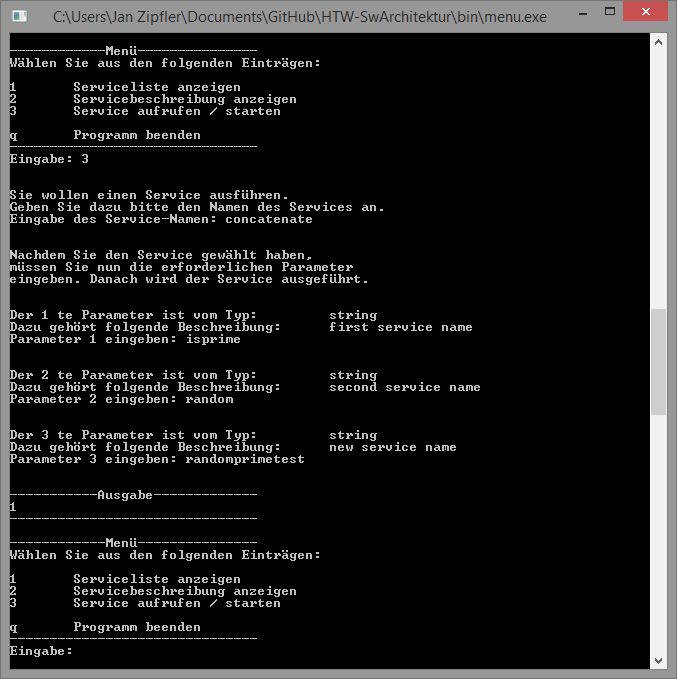


Abbildung 2 Erstellen eines neuen Services

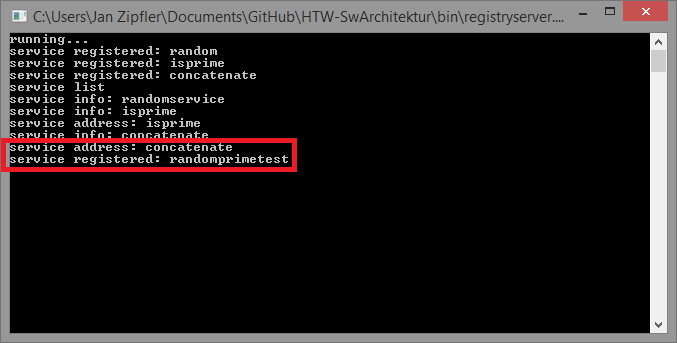


Abbildung 3 Protokollierung: Neuer Service angelegt wurde

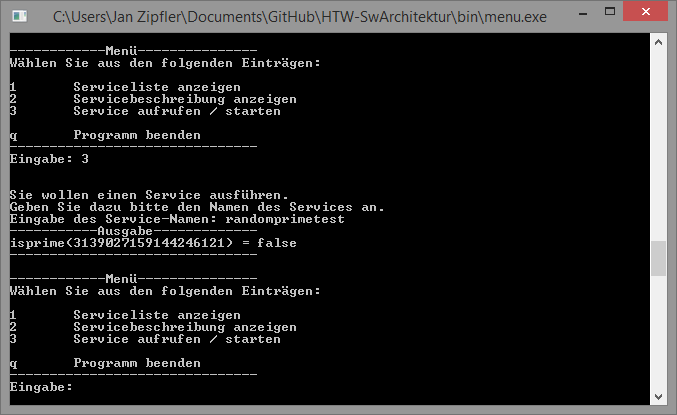


Abbildung 4 Ausführung des zusammengesetzten Services

1. https://de.wikipedia.org/wiki/Serviceorientierte\_Architektur (16. März 2014) [↑](#footnote-ref-1)
2. Ingo Melzer – Service-orientierte Architekturen mit Web Services [↑](#footnote-ref-2)
3. Ingo Melzer – Service-orientierte Architekturen mit Web Services [↑](#footnote-ref-3)