

2014

# Live Service Composition

AUSARBEITUNG  
SOFTWAREARCHITEKTUR  
PHILIPP EWEN  
JAN ZIPFLER

## Inhalt

Abstract .....	2
Grundlagen .....	3
Serviceorientierte Architektur .....	3
Registry .....	3
Service .....	3
Service Composition .....	3
Choreografie .....	3
Orchestrierung .....	4
Execution Engine .....	4
Implementierung .....	5
Programmiersprache .....	5
Registry .....	5
Random Service .....	6
IsPrime Service .....	7
Execution Engine/Concatenate Service .....	7
Menü .....	8
Programmablauf mit Bildschirmfotos .....	9

## 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 zu anzulegen. Um mehrere Services miteinander zu verknüpfen setzen wir eine sogenannte Execution Engine ein, die wir beispielhaft für die Verknüpfungsart „verketteten“ implementieren. Bei dieser Verknüpfungsart handelt es sich um ein Verketteten von zwei Funktionen im mathematischen Sinne, wie zum Beispiel:  $f(g(x))$ . 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 einer 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)<sup>1</sup>

### 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 Services erfragen.

### Service

Bei einem Service handelt es sich um eine autarke Einheit, die eine bestimmte Funktionalität zur Verfügung stellt. Diese Funktionalität wird dann im 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.“<sup>2</sup>

Er 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.

---

<sup>1</sup> [https://de.wikipedia.org/wiki/Serviceorientierte\\_Architektur](https://de.wikipedia.org/wiki/Serviceorientierte_Architektur) (16. März 2014)

<sup>2</sup> Ingo Melzer – Service-orientierte Architekturen mit Web Services

## Orchestrierung

„Eine Orchestrierung beschreibt die ausführbaren Aspekte eines Geschäftsprozesses aus der Sicht des Prozesses.“<sup>3</sup>

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

## Execution Engine

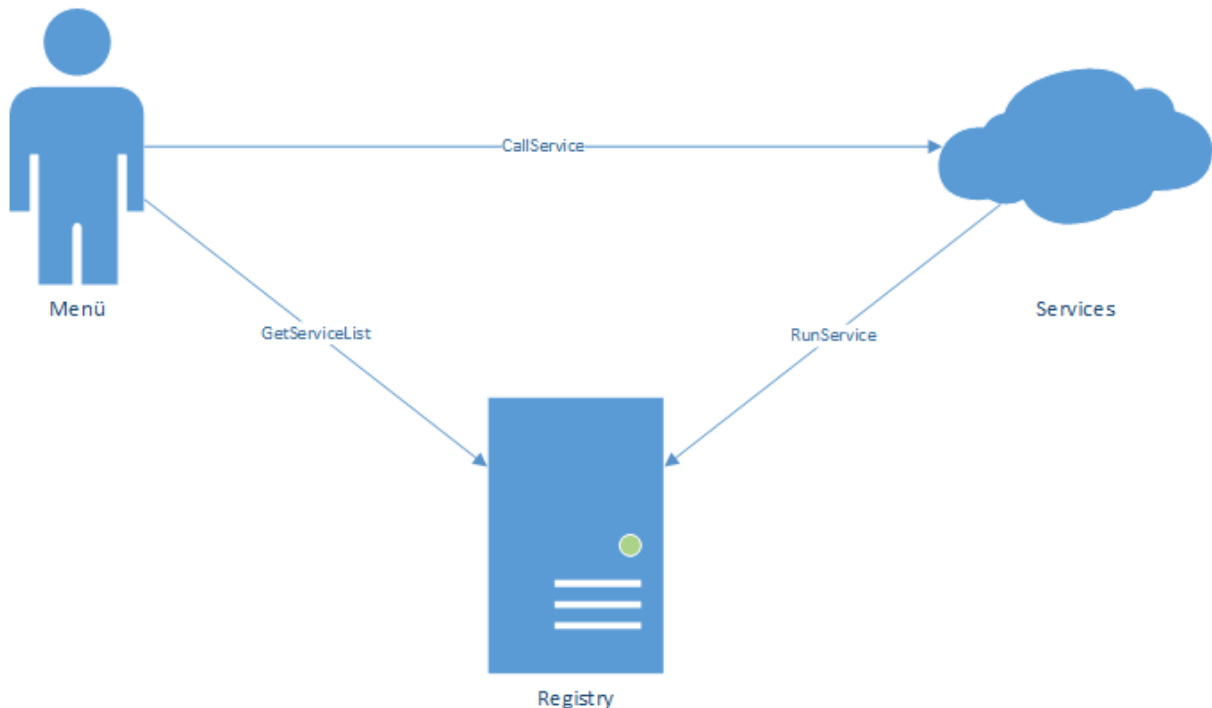
Die Execution Engine dient dem Ausführen eines zusammengesetzten Services (Service Composition). Dazu benötigt die Execution Engine eine Definition der Service Composition. Hierfür verwenden wir in dieser Arbeit die Namen der Services, welche in der Form  $f(g(x))$  verkettet werden sollen.

---

<sup>3</sup> Ingo Melzer – Service-orientierte Architekturen mit Web Services

## 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 und der Registry. Die folgende Übersicht illustriert die Struktur unserer Anwendung und vermittelt einen Überblick:



## Programmiersprache

Als Programmiersprache haben wir Go verwendet. Go wurde von Google entwickelt und entstand 2007 im Rahmen eines 20% Projekts. Zu den Stärken 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 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.

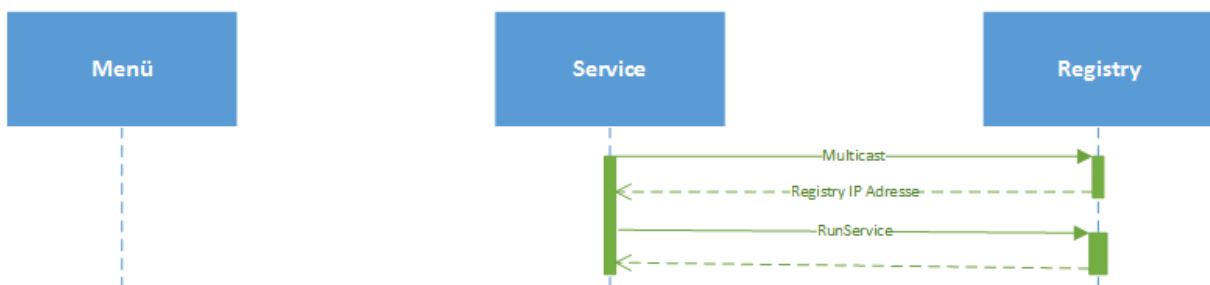
## Random Service

Dieser Service generiert eine Zufallszahl im Bereich von 0 bis  $2^{31} - 1$ .

Service Contract:

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

Sequenzdiagramm: Service registrieren und aufführen



Definition des Random Services:

```
1 // Service Contract definieren
2 serviceRandom := service.ServiceInfo{
3     "random",
4     "int",
5     "Generates a random integer.",
6     []service.ArgumentInfo{
7         {"void", "void", "no arguments"},
8     },
9 }
10
11 // ...
12
13 // Hauptfunktion des Services
14 func randomHandler(servicecall *service.ServiceCall) string {
15     number := rand.Int()
16
17     return strconv.Itoa(number)
18 }
19
20 // ...
21
22 // Service registrieren und ausführen
23 service.RunService(&serviceRandom, randomHandler)
```

## IsPrime Service

Dieser Service testet, ob die übergebene natürliche Zahl eine Primzahl ist. Hierzu wird ein 16-Facher Miller-Rabin-Test durchgeführt. Dabei handelt es sich hierbei um einen probabilistischen Test, d.h. das Ergebnis gilt mit einer bestimmten Wahrscheinlichkeit (in diesem Fall:  $1 - \left(\frac{1}{4}\right)^{16} = 99,999999977\%$ ).

Service Contract:

<i>Service Name</i>	isprime
<i>Beschreibung</i>	Test if x is prime. (Ausgabe als Text)
<i>Parameter</i>	x, int, number to test
<i>Returns</i>	string

## Execution Engine/Concatenate Service

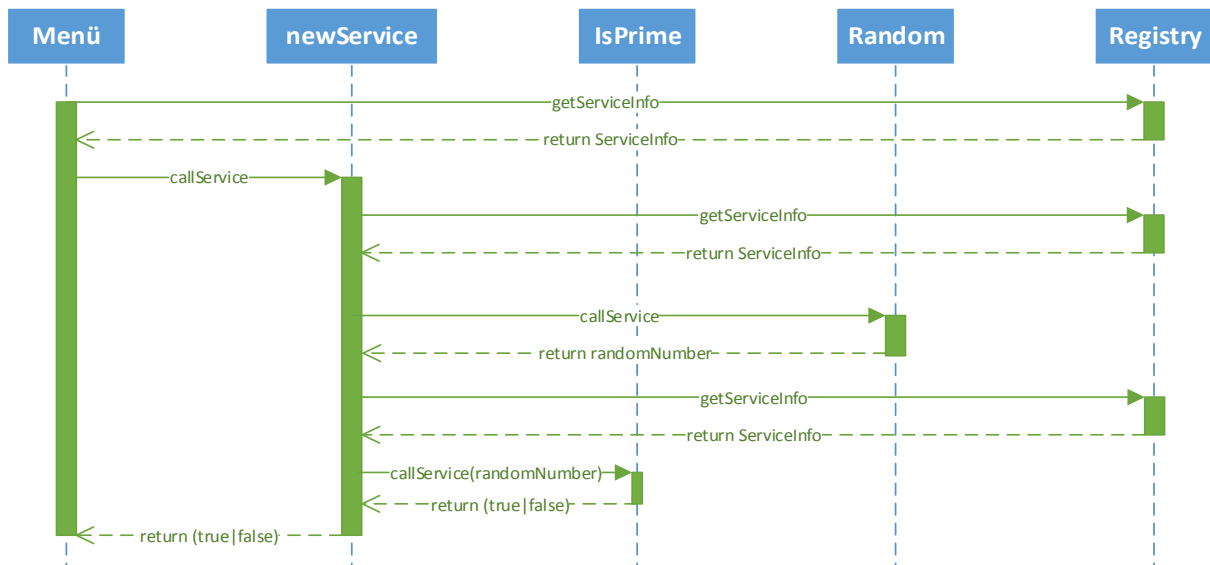
Die Execution Engine führt zwei Services hintereinander aus. Dieses Hintereinander Schalten entspricht der mathematischen Operation verketteten, und zwar im Sinne von  $newservice := service1(service2())$ . Dabei hat „service1“ aus Einfachheitsgründen keine Übergabeparameter, es wäre jedoch ohne weiteres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

Service Contract:

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

Sequenzdiagramm: Zusammengesetzten Service ausführen



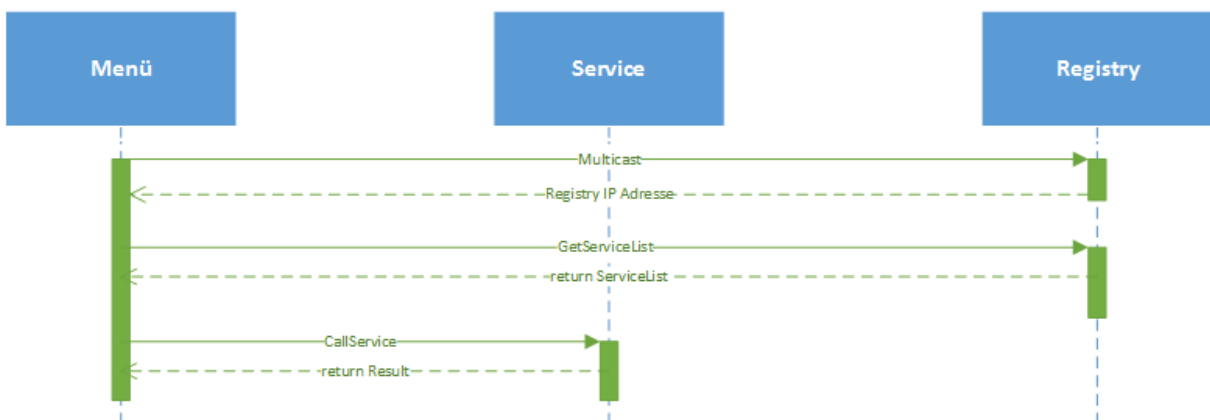


## Menü

Das Menü stellt die Schnittstelle zwischen Mensch und unserer 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 soll erläutert werden, wie die Implementierung der Service Komposition verwendet werden kann. Dazu werden einige Bildschirmfotos bereitgestellt, die die einzelnen Schritte beschreiben.

Um diese Implementierung verwenden zu können, werden mindestens 3 Programmteile benötigt:

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

Da es sich um eine Service Komposition handeln soll, werden die drei Services verwendet, die bereits im Abschnitt Implementierung erläutert wurden:

- Random Service
- Primzahltest Service
- Kompositionsservice

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

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

Abbildung 2 zeigt, den Aufruf des ConcatenateService, der die erwähnten 2 Services als Parameter benötigt. Hierbei wird zuerst die äußere (isPrime) und dann die innere Funktion (Random) angegeben. Der dritte Parameter (randomprimetest) ist der Name des neuen Services.

Die durchgeführte Aktion, kann ebenso auf dem Server, auf dem die Registry läuft, nachvollzogen werden. Ein Beispiel für eine solche Protokollierung ist in **Fehler! Verweisquelle konnte nicht gefunden werden.**3 zu sehen.

Die letzte Abbildung, zeigt das ausführen des neuen Services, der die Zufallszahl einem Primzahlentest unterzieht und das Ergebnis ausgibt.

```
C:\Users\Jan Zipfler\Documents\GitHub\HTW-SwArchitektur\bin\menu.exe
-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe: 1
-----Ausgabe-----
      Service
      concatenate
-----
SERVICE CONTRACT:
Adresse: 127.0.0.1:6701
Service-Name: concatenate
Service-Beschreibung: Concatenate two services service1(service2()) as service.
Rückgabewert: var
1. Argument:
      Bezeichnung:  service1
      Typ:          string
      Beschreibung: first service name
2. Argument:
      Bezeichnung:  service2
      Typ:          string
      Beschreibung: second service name
3. Argument:
      Bezeichnung:  service
      Typ:          string
      Beschreibung: new service name
-----
-----Ausgabe-----
      Service
      isprime
-----
SERVICE CONTRACT:
Adresse: 127.0.0.1:6697
Service-Name: isprime
Service-Beschreibung: Performs 16 Miller-Rabin tests to check whether x is prime
.
Rückgabewert: string
1. Argument:
      Bezeichnung:  x
      Typ:          int
      Beschreibung: number to test
-----
-----Ausgabe-----
      Service
      random
-----
SERVICE CONTRACT:
Adresse: 127.0.0.1:6693
Service-Name: random
Service-Beschreibung: Generates a random int
Rückgabewert: int
1. Argument:
      Bezeichnung:  void
      Typ:          void
      Beschreibung: no arguments
-----
-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe:
```

Abbildung 1 Auflistung aller Service-Contracts

```
C:\Users\Jan Zipfler\Documents\GitHub\HTW-SwArchitektur\bin\menu.exe

-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe: 3

Sie wollen einen Service ausführen.
Geben Sie dazu bitte den Namen des Services an.
Eingabe des Service-Namen: concatenate

Nachdem Sie den Service gewählt haben,
müssen Sie nun die erforderlichen Parameter
eingeben. Danach wird der Service ausgeführt.

Der 1 te Parameter ist vom Typ:      string
Dazu gehört folgende Beschreibung:  first service name
Parameter 1 eingeben: isprime

Der 2 te Parameter ist vom Typ:      string
Dazu gehört folgende Beschreibung:  second service name
Parameter 2 eingeben: random

Der 3 te Parameter ist vom Typ:      string
Dazu gehört folgende Beschreibung:  new service name
Parameter 3 eingeben: randomprimetest

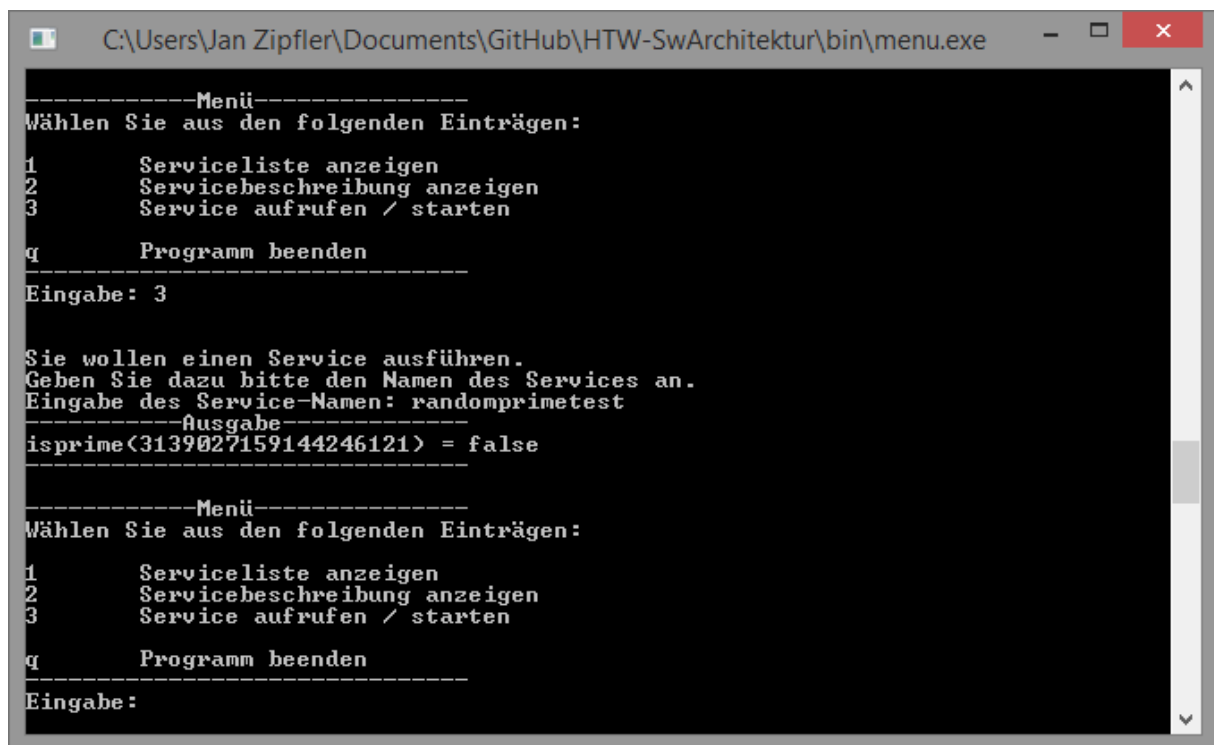
-----Ausgabe-----
1
-----
-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe:
```

Abbildung 2 Erstellung eines neuen Services

```
C:\Users\Jan Zipfler\Documents\GitHub\HTW-SwArchitektur\bin\registryserver....

running...
service registered: random
service registered: isprime
service registered: concatenate
service list
service info: randomservice
service info: isprime
service address: isprime
service info: concatenate
service address: concatenate
service registered: randomprimetest
```

Abbildung 3 Protokollierung, dass neuer Service angelegt wurde



```
-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe: 3

Sie wollen einen Service ausführen.
Geben Sie dazu bitte den Namen des Services an.
Eingabe des Service-Namen: randomprimetest
-----Ausgabe-----
isprime(3139027159144246121) = false
-----
-----Menü-----
Wählen Sie aus den folgenden Einträgen:
1      Serviceliste anzeigen
2      Servicebeschreibung anzeigen
3      Service aufrufen / starten
q      Programm beenden
-----
Eingabe:
```

Abbildung 4 Ausführung des zusammengesetzten Service