```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDescription xmlns="http://example.com/schemas/ordermanagement"
                    xml:base="http://om.example.com/">
  <link rel="all"           href="/orders/" />
  <link rel="received"      href="/orders/?state=received/" />
  <link rel="accepted"      href="/orders/?state=accepted/" />
  <link rel="rejected"      href="/orders/?state=rejected/" />
  <link rel="cancelled"     href="/orders/?state=cancelled/" />
  <link rel="fulfilled"     href="/orders/?state=fulfilled/" />
  <link rel="cancellations" href="/cancellations/" />
  <link rel="reports"       href="/reports/" />
</serviceDescription>
```

Listing 15–1 *Servicedokument für Discovery*

Der Client kann auf Basis dieses Dokuments, das er als Resultat einer HTTP-GET-Anfrage erhält, einen logischen Namen (z. B. »cancelled«) in eine physische Adresse (Base-URI + Wert des href-Attributs, also http://om.example.com/orders/?state=cancelled/) umsetzen.

Auf ähnliche Art und Weise können dem Client für unterschiedliche Kundensegmente URIs unterschiedlicher Server übermittelt werden, vielleicht noch einmal unterteilt nach Anfangsbuchstaben des Firmennamens (A-E, F-K usw.). Vergleicht man dies mit den Mechanismen aus der WS-Architektur[4], stellt sich heraus, dass der Hypermedia-Mechanismus deutlich einfacher und trotzdem mächtiger ist.

Im Rahmen einer unternehmensweiten SOA auf REST-Basis kommt somit der Beschreibung der Ressourcen durch Hypermedia eine besonders herausgehobene Rolle zu: Darüber wird die Evolution der Service- bzw. Ressourcenlandschaft ohne eine grundsätzliche Notwendigkeit zur Modifikation der Clients ermöglicht.

15.4 Orchestrierung und Choreografie

Mit Orchestrierung bezeichnet man den koordinierten Aufruf mehrerer Serviceoperationen zur Realisierung einer höherwertigen Funktionalität, der durch eine geeignete Laufzeitkomponente (eine *Orchestration Engine*) unterstützt wird. Choreografie dagegen wird (leicht vereinfacht) positioniert als Koordination zwischen kooperierenden Partnern ohne ein zentrales steuerndes Element.[5]

Eine Orchestrierungs-Engine wäre nicht mehr als eine Laufzeitumgebung für Programme, vergleichbar einem Interpreter für eine Skriptsprache, wenn sie nicht noch ein besonderes Merkmal unterstützen würde: den (auch parallelen) Aufruf von Serviceoperationen, die ihr Resultat asynchron und möglicherweise erst nach längerer Zeit zurückmelden. Für die Beschreibung der Orchestrierungen, die von

4. Z.B. UDDI, einem nahezu absurd komplizierten Standard.
5. Die Analogie: Bei einem Orchester gibt es einen Dirigenten, der den Ton angibt; bei der Choreografie folgen die Tänzer ihren jeweils eigenen Vorgaben und berühren sich nur gelegentlich.

solchen Engines ausgeführt werden, gibt es verschiedene Standards. Der populärste ist WS-BPEL, die Business Process Execution Language [102], die auf die Koordination von Web Services ausgelegt ist, die mit WSDL beschrieben sind.

Aktuell gibt es weder für den Architekturstil REST auf konzeptioneller Ebene noch für RESTful HTTP auf technischer Ebene einen akzeptierten Standard oder eine Best Practice zu Orchestrierung oder Choreografie. Es gibt einige Forschungsansätze und recht neue Implementierungen in Open-Source-Produkten, nichts davon hat bislang jedoch eine nennenswerte Verbreitung erlangt.

Ich selbst bin der Ansicht, dass die Notwendigkeit für eine Orchestrierungsunterstützung stark überbewertet wird und auch in einer T-SOA auf WS-Architektur-Basis nur eine optionale Komponente darstellt. Man kann anderer Meinung sein: In diesem Fall kann REST bzw. RESTful HTTP keinen alternativen Ansatz bieten – außer natürlich der Möglichkeit zur Implementierung der Orchestrierungen mit »normalen« Programmiersprachen, ggf. in Kombination mit einer einbettbaren Prozess-Engine[6].

15.5 Enterprise Service Bus (ESB)

Anders sieht es beim Konzept eines Enterprise Service Bus (ESB) aus. Dieser wird selbst von T-SOA-Verfechtern unterschiedlich bewertet: Die einen sehen ihn als zentrales Element und fordern eine Entscheidung für ein spezifisches ESB-Produkt schon in einer sehr frühen Phase einer SOA-Initiative. Andere sprechen von einem virtuellen ESB als Summe der technischen Entscheidungen, die für die Schnittstellen der Services und der Kommunikation zwischen ihnen getroffen wurde. Wieder andere lehnen das ESB-Konzept grundsätzlich ab und bevorzugen eine »dumme« Infrastruktur mit intelligenten Services an den »Endpoints«.

In einer RESTful-HTTP-Architektur existieren unterschiedliche Alternativen zum ESB-Konzept: Zum einen können im Bereich der Ablaufumgebungen für HTTP-Anwendungen in aller Regel die gleichen Produkte eingesetzt werden wie für den Betrieb von Web Services. Zum anderen existiert gerade für HTTP eine Unzahl von Laufzeitkomponenten, wie Proxy-Server, Gateways, Firewalls oder Cache-Server, die konzeptionell den gleichen Zweck haben wie eine ESB-Laufzeitstruktur: die Anreicherung bzw. Interpretation von Nachrichten, die zwischen Consumer und Provider bzw. Client und Server ausgetauscht werden.

15.6 WSDL, SOAP & WS-*: WS-Architektur

Im Folgenden möchte ich zum Abschluss RESTful HTTP auf der einen und die Architektur SOAP/WSDL-basierter Web Services (»WS-Architektur« in der Terminologie dieses Kapitels) auf der anderen Seite gegenüberstellen.

6. Z.B. JBoss jBPM in der Java-Welt.