SOA@T-Mobile (3)

Nachdem in der vorigen Ausgabe der ESB der T-Mobile im Detail dargestellt wurde, geht es nun an die vollautomatische Serviceprovisionierung. Gezeigt wird die konzeptionelle und die technische Umsetzung des Systems sowie Erfahrungen im internationalen Einsatz.

von Carsten Sensler und Andre Karalus

Die Serviceprovisionierung, die Rekonfiguration der Runtime mit neuen Routing-Informationen, ist ein wesentlicher Bestandteil der technischen Umsetzung der SOA Backplane. Durch die Ausführung einer Serviceprovisionierung werden Kommunikationsbeziehungen in der SOA Backplane Runtime beeinflusst (freigeschaltet, verändert oder gelöscht). Es wurde sehr stark darauf geachtet, dass der Automatisierungsgrad sehr weit fortgeschritten ist, um schnell, einfach und kostengünstig eine Rekonfiguration durchführen zu können. Bei der Konzeption und Umsetzung dieses Aspekts der SOA Backplane galt es auf alle Fälle zu vermeiden, dass der Betreiber bei jeder Änderung der Konfiguration der Laufzeitumgebung, aufgrund von Änderungen der Kommunikationsbeziehungen, eingebunden werden muss. Wäre dieses Ziel nicht erreicht worden, so hätte es bedeutet, dass Kosten bei Änderungen der Kommunikationsbeziehungen entstünden und man nicht so flexibel gewesen wäre, wie wir es heute sind.

Zur Configuration-Time wird bei der Serviceprovisionierung festgelegt, welche bereits an eine Infrastruktur gebundenen und in der Architektur festgelegten Kommunikationsbeziehungen (Routings) in den CALs und EMS-Servern konfiguriert werden sollen. Dabei sind verschiedene Granularitäten (Model-Targets) der Serviceprovisionierung möglich (nach absteigender Granularität):

- *Per Umgebung*: Im Binding spezifiziert man die Umgebung, an die die Applikation mit seinen Ports gebunden und dementsprechend auch ausgeliefert werden soll. Die Umgebung meint alle gebundenen Komponenten und ist faktisch eine komplette Konfiguration (wird praktisch nur für kleine Testumgebungen genutzt, da für die richtige Umgebung der Umfang extrem groß wäre) – es werden alle EAR-Files der Providing Ports und alle WAR-Files der Using Ports aller Applikationen, die an die Umgebunden gebunden sind, in die CAL-Instanzen eingespielt.
- *Per Binding-Set*: In einem BindingSet sind alle Ports (Using und Providing) einer Applikation aus der Architektur an eine Umgebung (an einen konkreten CAL aus der Infrastruktur) gebunden – alle entsprechenden WAR-Files und EAR-Files werden in die CAL-Instanzen eingespielt.
- *Per Providing oder Using Port Binding*: In einem Providing Port Binding ist nur ein Providing Port einer Applikation aus der Architektur an eine Umgebung (an einen konkreten CAL aus der Infrastruktur) gebunden. Beim Using Port Binding ist entsprechend nur ein Using Port gebunden – es wird ein EAR-File in die CAL-Instanzen (Providing Port Binding) und nur ein WAR-File in die CAL-Instanzen eingespielt (Using Port Binding).

Eine beliebige Kombination dieser Punkte ist möglich, z.B.: Provisioniere bitte die Applikation XYZ und weitere drei Using und fünf Providing Ports – wichtig ist dabei nur, dass sich die verschiedenen Model-Targets auf dieselbe Umgebung beziehen. Eine Serviceprovisionierung wird durch die Ausführung eines Jobs gegen CEISeR ausgelöst. Es sind entsprechende Tasks vorhanden, um einen Service-Provisioning-Job mit dem Xplor zu erstellen.

Die technische Serviceprovisionierung

Der technische Service Provisioning Task besteht aus drei Schritten:

1. *Generierung aller Artefaktdateien aus dem Service Repository:* (a) Ermittlung aller gebundenen Ports für die Infrastruktur, die der Service Provisioning Task spezifiziert (Targets), (b) Konsistenzprüfung über diesen Teil des Repositories, ob alle Informationen vollständig sind (fehlt z.B. ein technischer Endpoint eines Service Providers oder ist die WSDL, auf die der Service verweist, bereits ins Repository importiert etc.), (c) Generierung der *adapter.config.xml*, *web.xml*, *ant.build.xml* (2. Schritt), XSLTs etc. mittels oAW.
2. *Assemblierung der JEE-Artefakte*: Hierbei werden generierte Ant-Skripte ausgeführt, die alle benötigten Artefakte in WAR-Files und EAR-Files packen – gegebenenfalls wird dabei Java-Code gebaut, falls es sich um spezielle Plug-ins handelt. Anschließend werden noch alle JEE-Artefakte pro CAL in eine Datei gezippt.

3. *Generieren eines so genannten Configuration-Deskriptors*: Dies ist eine SOAP-Nachricht, die neben dem (base64 encoded) Artefakt-ZIP auch die Informationen zu den EMS-Servern enthält, beispielsweise die zu erzeugenden Home und Remote Queues.

Grundsätzlich erhält der Service Provisioning Task folgende Parameter:

- Die Umgebung (Infrastruktur), für die Services auf den CAL-Instanzen provisioniert werden sollen. Sie ist implizit, es kann nicht „gemischt" werden – eine Serviceprovisionierung bezieht sich auf genau eine Umgebung.
- Bindings: Alle gebundenen Ports, die neue oder geänderte Konfiguration erhalten sollen bzw. deren Konfiguration entfernt werden soll (Dekommissionierung von nicht mehr benötigten Routings).
- Branch bzw. Label (der Default ist natürlich main latest) bezüglich des Repositories, auf den sich die Serviceprovisionierung beziehen soll. Wichtig ist dies, wenn eine Kommunikationsbeziehung auf dem letzten Stand des Repositories bereits geändert wurde, andere QoS-Parameter gelten oder das verbundene Plug-in geändert worden ist. So kann man z.B. auf einem Branch im Repository einen Hotfix an Routings für die Produktion ausüben und dies auch provisionieren, während der letzte Stand des Repositories bereits nicht getestete Änderungen an den betroffenen Elementen enthält.

Der Configuration-Deskriptor (Abb. 1) wird mittels Web Service zum GSPA geschickt und vom ConfigStore ebenfalls auf Konsistenz geprüft und dann in seine „landesspezifischen" Bestandteile aufgelöst. Diese werden über den ESB auf dedizierten Queues verteilt (asynchrones MEP). In jeder Landesgesellschaft ist ein LSPA installiert, der diese Nachricht entgegennimmt und interpretiert. Dabei werden von ihm über eine Administrationsschnittstelle des EMS-Servers die Queues erzeugt und über das Remote-Interface des verwendeten JBoss-Applikationsservers die JEE-Artefakte deployt. Das Remote-Interface des JBoss

?-? xml	version="1.0" encoding="utf-8"
▼ e dpl:soa-bp-dpl	
a dpl:created-by	sensler
a dpl:created-on	2008-08-11T06:54:57.002
a dpl:environment	tmo.ei.test.common.TestInfrastructure1
a dpl:global-destination-name	TST-TMO-EMS
a dpl:global-destination-url	tcp://10.100.xxx.xx:2xxxx,tcp://10.100.xxx.xx:2xxxx
a dpl:version	2.0
a xmlns:dpl	http://sbp.ei.tmobile.net/dpl/datatypes
a xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
a xsi:schemaLocation	http://sbp.ei.tmobile.net/dpl/datatypes soa-bp-dpl2.xsd
▼ e dpl:configSpec	
--	Branch: main
e dpl:config-item	* 207
▼ e dpl:service-assembly	
e dpl:servcoDomain	TMDE
e dpl:auto-deploy	false
▶ e dpl:identification	
▼ e dpl:service-unit-container	
e dpl:name	tmo.ei.test.common.physical.TestInfrastructure1-PE:WSG_DE1-AS
e dpl:service-unit-container-type	WSG
e dpl:management-url	http://10.100.xxx.xx:2xx80/jmx-console
e dpl:deployment-url	jnp://10.100.xxx.xx:2xx99
e dpl:destination-url	tcp://10.100.xxx.xx:2xxxx,tcp://10.100.xxx.xx:2xxxx
e dpl:library-name	Gateway-WSG_DE-LBC-deploy.zip
▼ e dpl:service-unit	
▼ e dpl:identification	
e dpl:name	tmd.tmobile.demo.jm.JavaMagazin.Demo.testPing
e dpl:description	Using port for synchronous porttype testPing
▶ e dpl:connections	

Abb. 1: Configuration-Deskriptor

erlaubt es über JNDI, auf die Dienste der JMX-Beans zuzugreifen und damit eine JMX-Bean anzusprechen, die das Hot Deployment über JSR88 unterstützt.

Über die (erfolgreiche) Ausführung dieses Vorgangs wird eine Protokollnachricht erzeugt und über den ESB asynchron an den ConfigStore gemeldet, der dies in

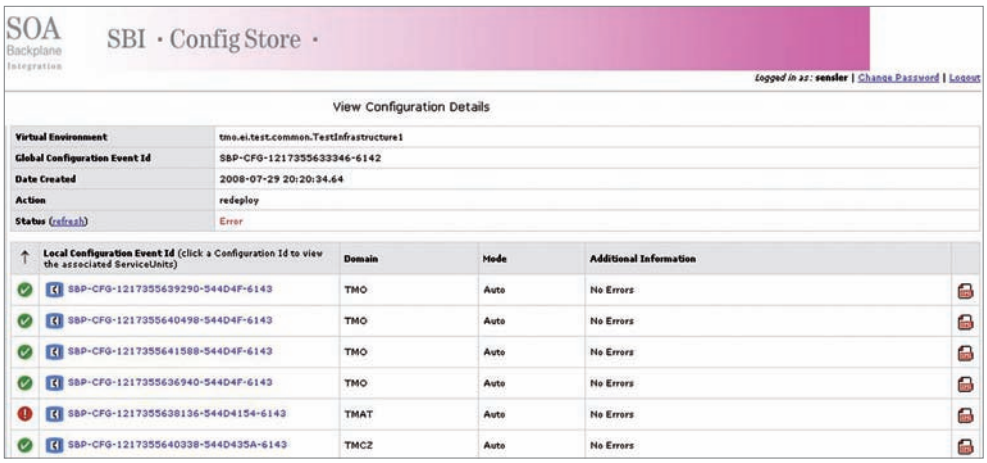

Abb. 2: Schematische Darstellung der Serviceprovisionierung

seiner Datenbank vermerkt und mittels einer Webanwendung zur Abfrage für eine Verifikation zur Verfügung stellt.

Abbildung 2 zeigt den schematischen Verlauf einer Serviceprovisionierung, ausgehend nach der Ausführung eines entsprechenden Jobs gegen CEISeR bis hin zu den CAL-Instanzen bzw. EMS-Servern.

Die Kommunikation des ConfigStores mit den LSPA wird dabei genauso im LMS geloggt, wie der Nachrichtenaustausch der Runtime zwischen den CAL-Instanzen. Damit wird nicht nur der ESB sowohl für die Runtime als auch für die Configuration-Time benutzt, auch der Logstore-/MessageStore dient als international verfügbares Medium, um den kompletten Nachrichtenaustausch verfolgen zu können.

Der ConfigStore

Der ConfigStore ist die zentrale Monitoring-Komponente der Infrastruktur innerhalb der SOA Backplane. Durch ihn erhält man einen Liveeinblick in die jeweilige Laufzeitumgebung. Es werden sämtliche, der Umgebung angeschlossenen EMS-Server sowie die CALs überwacht. Jede Serviceprovisionierung wird protokolliert und reportet. Dabei wird genau festgehalten, welche Serviceartefakte in welche Landesgesellschaft geliefert wurden und welche Queues auf welchen EMS-Servern angelegt worden sind. Wenn das Service Provisioning Event (SOAP Request mit dem Configuration-Deskriptor) den ConfigStore erreicht, wird eine eineindeutige ID vergeben, mit der sich das Service Provisioning Event im ConfigStore jederzeit wiederfinden lässt. Der ConfigStore bietet die Möglichkeit zu analysieren, welche Konfiguration in welche Landesgesellschaft eingespielt wurde. Abbildung 3 zeigt einen Screenshot der Webanwendung des ConfigStores. Auf dem Screenshot ist zu erkennen, dass es sich um eine internationale Serviceprovisionierung handelte und dass ein Problem in T-Mobile Austria (TMAT) aufgetreten ist – diese Serviceprovisionierung war also nicht in jeder Landesgesellschaft erfolgreich. Durch die gute Logging- und Monitoring-Fähigkeiten des ConfigStores sind derartige Fehler leicht festzustellen. Folgt man dem Link hinter der lokalen Service-Configuration-ID, würde man auf eine detaillierte Ansicht des Configuration-Events weitergeleitet werden. In diesem Fall konnte der LSPA eine CAL-Instanz in TMAT nicht erreichen (bedingt durch Wartungsarbeiten in TMAT) und daher konnten dort keine Serviceartefakte provisioniert werden.

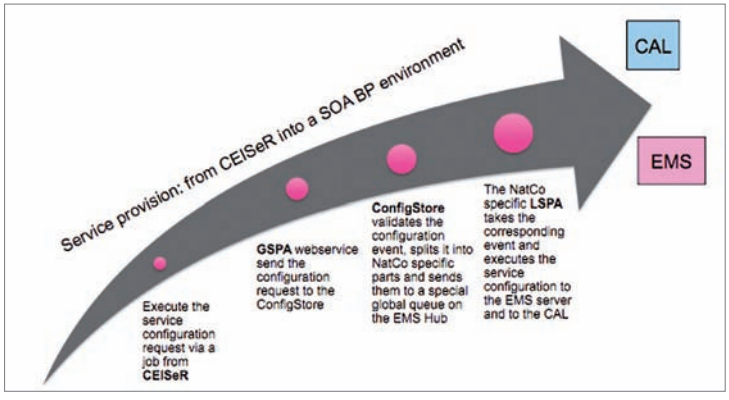

Abb. 3: Screenshot: ConfigStore

Grad der Automatisierung

Der Grad der Automatisierung der Serviceprovisionierung ist auf einer Skala von 1 bis 10 bei ca. 7 anzusiedeln. Aus prozesstechnischen Gründen und auch aus Sicherheitsgründen ist es zurzeit nicht möglich, eine hundertprozentige Automatisierung zu erreichen. Rein technisch könnten wir die Serviceprovisionierung vollständig automatisiert ausführen lassen, aber die fachliche Verifikation ist nur schwer automatisierbar. Technisch verifiziert CEISeR die generierten Artefakte bereits und der ConfigStore nimmt auch eine Überprüfung des Configuration-Deskriptors vor. Aber eine fachliche Verifikation, ob die erwarteten Services auch tatsächlich in der Runtime konfiguriert worden sind, ist im Augenblick nicht automatisiert. Auch die Spezifikation der Targets der Serviceprovisionierung (welche Services sollen in welcher Umgebung provisioniert werden) ist auch nur durch manuelle Eingriffe möglich. Vielmehr arbeiten wir zurzeit an einer Workflow-Unterstützung für die Provisionierung, da viele Artefakte von unterschiedlichen Rollen und Teams zum CEISeR beigesteuert werden und mit einem Workflow-System diese verteilte Arbeit sehr gut unterstützt werden könnte.

Das Service-Provisioning-Team

Es ist ein internationales Team mit Mitgliedern aus jeder Landesgesellschaft aufgebaut worden, wobei die Verantwortlichkeiten klar zwischen lokalen und internationalen Aktivitäten getrennt sind (Abb. 4).

Jede Landesgesellschaft ist für Serviceprovisionierungen, die nur ihre Laufzeitkomponeten der SOA Backplane konfigurieren, selbst verantwortlich – also für

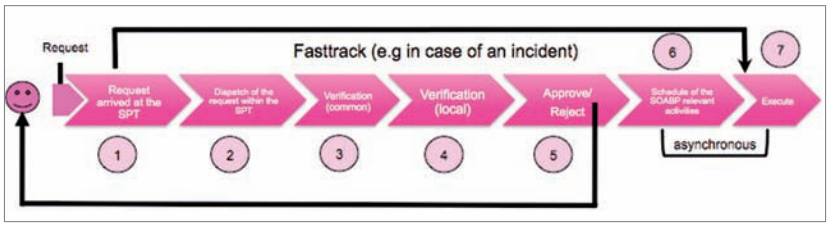

Abb. 4: Internationales Service-Provisionierungsteam – Aufbau und Verantwortlichkeiten

lokale Services. Sobald eine Serviceprovisionierung mehr als eine Landesgesellschaft betrifft, handelt es sich jedoch um eine internationale Serviceprovisionierung und die Verantwortung geht im Service-Provisioning-Team eine Instanz höher, auf die internationale Ebene des Teams. Bei einer internationalen Serviceprovisionierung sind deutlich mehr Abhängigkeiten zwischen den beteiligten Systemen aufzulösen und zu koordinieren. Es gilt dabei, unterschiedliche Change-Prozesse anzustoßen und Freigaben von unterschiedlichen verantwortlichen Personen (diese kann je Landesgesellschaft sehr stark variieren) einzuholen. Diese Koordination wird landesgesellschaftsübergreifend vorgenommen.

Der Service-Provisioning-Prozess

Der Prozess der Serviceprovisionierung gliedert sich in sieben Prozessschritte, wobei vereinzelte Prozessschritte je nach Landesgesellschaft eine spezialisierte Ausprägung aufweisen können. Abbildung 5 verdeutlicht den Service-Provisioning-Prozess.

Im Folgenden werden die einzelnen Prozessschritte erläutert:

1. Ein Request bezüglich der Ausführung einer Serviceprovisionierung erreicht das SPT. In dem Request müssen verschiedene Informationen angegeben werden, z.B.: gewünschter Ausführungszeitpunkt, Requestor, Umgebung, Targets (Was soll provisioniert werden?), beteiligte Landesgesellschaften etc.
2. Innerhalb des SPTs wird der Request je nach Verantwortlichkeit verteilt (Stichwort: lokale oder internationale Serviceprovisionierung).
3. Es erfolgt mithilfe des Constraint-Frameworks von CEISeR und der Ausführung eines entsprechenden Provisionierungsjobs eine Validierung (Dry-Run).
4. Es werden unter Umständen weitere landesgesellschaftsspezifische Validierungen durchgeführt.
5. Wenn die Schritte 3 und 4 erfolgreich durchlaufen sind, erfolgt eine Genehmigung. Andernfalls wird der Request abgewiesen und der Requestor kontaktiert.
6. Der Request wird terminiert und es werden die entsprechenden Kollegen aus dem SPT eingeteilt
7. Zum geplanten Zeitpunkt wird die Serviceprovisionierung ausgeführt und dem Requestor nach der Ausführung eine Bestätigung zugestellt.

Hat sich der Aufwand gelohnt

Ja! Eine Betrachtung des Businessnutzens der automatischen Serviceprovisionierung darf nicht fehlen. Dabei werden die Vorteile aus wirtschaftlichen Gesichtspunkten dargestellt und Bezug auf Reaktionszeiten vs. Imageverlust und Geschäftsausfall im Incident-Fall genommen – auch der Aspekt Time-to-Market wird diskutiert. Es wurden insgesamt 220 Serviceprovisionierungen über alle Landesgesellschaften hinweg in den Jahren 2007 und 2008 durchgeführt. Üblicherweise wäre bei jeder Serviceprovisionierung das Eingreifen des IT-Operator nötig gewesen, das entsprechende Kosten verursacht hätte. Aufgrund unseres Grades der Automatisierung entfallen diese Kosten.

Es ist davon auszugehen, dass durch die stetig wachsende Nutzung der SOA Backplane durch weitere Kommunikationspartner die Kostenersparnis durch die Automatisierung deutlich ansteigen wird.

Beispiel eines Incidents: Ausfall eines Netzabschnitts

Durch die Tatsache, dass alle Kommunikationsbeziehungen in dem zentralen

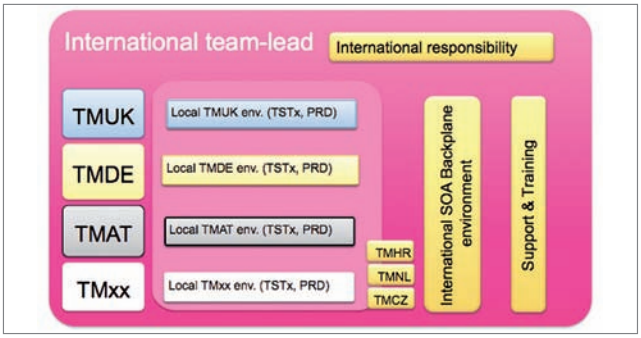

Abb. 5: Service-Provisioning-Prozess

Service Repository modelliert und gespeichert sind, ist es ein Leichtes für den Operational Support im Falle eines Netzabschnittsausfalls den Impact zu bewerten. Der Operational Support kann den Informationen des Service Repositories direkt entnehmen, welche Komponenten von der Netzstörung betroffen sind und entsprechende Gegenmaßnahmen einleiten. Eine mögliche Maßnahme könnte sein, dass die betroffenen Anwendungen in einem anderen Netzsegment gestartet werden. Der IT-Operator muss sich dabei nicht um die Umkonfiguration der Services auf der SOA Backplane kümmern, sondern er teilt die neuen Serviceendpunkte dem Service-Provisioning-Team mit und es wird direkt eine entsprechende Serviceprovisionierung durchgeführt. Es vergehen nur wenige Minuten von Erhalt der neuen Endpunkte bis zu Provisionierung auf dem ESB – und dies nicht nur bei lokalen Services, sondern auch europaweit bei internationalen Services.

Abkürzungen

CAL – Common Access Layer
CEISeR – Central Enterprise Intergation Service Repository
CR – Change Request
EMS – Enhanced Messaging Service
ESB – Enterprise Service Bus
GSPA – Global Service Provisioing Agent
LSPA – Local Service Provisioning Agent
MEP – Message Exchange Pattern
oAW – openArchitectureWare
Payload – fachliche Nutzdaten einer Nachricht
SOA BP – SOA Backplane
SP – Service Provider: serviceanbietende Komponente
SC – Service Consumer: servicenutzende Komponente
SPT – Service-Provisioning-Team

Time-to-Market

Die SOA Backplane ist innerhalb der T-Mobile mit ihren Landesgesellschaften eine Infrastukturkomponente, die dem Bestreben nach möglichst kurzer Time-to-Market unterstützt. Nach erfolgtem Import aller benötigten Artefakte (WSDL, Architektur, Binding) steht einer Serviceprovisionierung nichts mehr im Weg. Man kann die SOA Backplane auch als Enabler bezeichnen. Sie enabled die Kommunikation, national oder international, von Serviceanbietern und Nutzern, ohne dass erneute Koordinationsaufwände investiert werden müssen.

Internationaler Rollout der SOA Backplane

Es war eine große Herausforderung, sämtliche Runtime-Komponenten in fünf Landesgesellschaften auszurollen. Doch nicht zuletzt durch den herausragenden Managementsupport und den guten Verbindungen in die anderen Landesgesellschaften, haben wir diese Herausforderung innerhalb eines Jahres umsetzen können. Ein sehr gutes Beispiel für die gute internationale Zusammenarbeit ist MyFaves. MyFaves wurde innerhalb kürzester Zeit mithilfe der SOA Backplane umgesetzt und in drei Landesgesellschaften nahezu zeitgleich livegeschaltet. Kurze Zeit später folgten die zwei weiteren Landesgesellschaften. Mittlerweile geht MyFaves in die nächste Phase und ist in allen fünf Landesgesellschaften verfügbar.

Fazit und Ausblick

Wir hoffen, dass wir in den drei Teilen dieser Artikelserie einen Einblick in die Konzeption sowie Funktionsweise der SOA Backplane vermitteln konnten und dass wir erfolgreich aufzeigen konnten, dass eine statisch geroutete SOA mit ein wenig Geschick keine Dynamik bezüglich der Kommunikationsbeziehungen einbüßen muss und man umgekehrt viele Vorteile der Statik ausnutzen kann (Wer spricht mit wem?). Die nächste große Herausforderung wartet bereits: Mit unserem strategischen Partner arbeiten wir zurzeit an einer Evolution von CEISeR, deren Ziel ist es, dass unsere Anforderungen von einem Produkt unterstützt und so die Entwicklungs- und Wartungskosten für Eigenentwicklung gesenkt werden und dass wir weitere Funktionen des Produkts nutzen können. ■

Dipl.-Ing. Carsten Sensler ist Angestellter der T-Mobile Deutschland GmbH und füllt dort die Rolle des Teamleiters eines internationalen Teams zur Serviceprovisionierung aus. Zudem arbeitet er an den Konzepten der SOA Backplane mit und gibt interne, internationale Trainings bezüglich SOA Backplane. Kontakt: http://www.sensler.de

Dipl.-Inform. Andre Karalus ist freiberuflicher Softwarearchitekt mit den Schwerpunkten Architektur, modellgetriebene Entwicklung und Integrations-Frameworks. Er arbeitet zurzeit für die T-Mobile Deutschland GmbH; dort gestaltet er am Design der SOA Backplane mit und unterstützt in der Umsetzung.

Links & Literatur

[1] Carsten Sensler, Andre Karalus: Subconf 2007, Integration eines Service Repositories mit Subversion zur Anbindung an den ESB: 2007.subconf.de/fileadmin/PDF_Dateien/SubConf_2007/Vortraege/Integration_eines_SOA_Repositories_Sensler_Karalus.pdf

[2] Carsten Sensler, Andre Karalus: SOA@T-Mobile – Vollautomatische Serviceprovisionierung auf dem ESB – Teil 1, in: Java Magazin 10.2008.

[3] Carsten Sensler, Andre Karalus: SOA@T-Mobile – Vollautomatische Serviceprovisionierung auf dem ESB – Teil 2, in: Java Magazin 11.2008.