Continuous Integration im Umfeld der Oracle SOA Suite 11g

Markus Heinisch, Markus Zehnder
Trivadis GmbH
München / Bern

Schlüsselworte:

Continuous Integration, Oracle SOA Suite

Einleitung

Continuous Integration für eine Oracle SOA Suite 11g Umgebung enthält mehr als einen Build-Server wie z.B. Hudson. Für den erfolgreichen Einsatz von CI gehört neben entsprechend gelebten Prozessen auch der Einsatz einer Reihe von sich ergänzenden Tools. Der Vortrag erläutert im ersten Teil die Herausforderungen beim Aufbau einer Multi-Projekt CI Infrastruktur auf der Basis von Maven, Hudson, Nexus, Subversion, Sonar, MySQL und Jira. Der zweite Teil behandelt die spezifische CI-Realisierung für OSB / SOA Kundenprojekte. Anhand eines größeren Projektes wurde eine Referenzumgebung aufgebaut, welche nun als Master Image für neue Projekte zur Verfügung steht. Das Endergebnis ist eine voll automatisierte Infrastruktur, welche nach dem Einchecken in Subversion die OSB, SOA oder Java Artefakte neu erstellt, in Nexus ablegt, auf den entsprechenden Server installiert und die Integrationstests durchführt. Die Präsentation zeigt die Probleme und unsere Lösungen auf, welche für eine erfolgreiche Automatisierung umgesetzt werden mussten.

Continuous Integration Motivation und allg. Konzepte

Continuous Integration (CI) ist ein Prozess, der vom Projektteam und vom Auftraggeber gelebt wird. Dieser Prozess hat zum Ziel, dass die zu entwickelnde Software immer in einem benutzbaren Zustand zur Verfügung gestellt werden kann. Dies wird erreicht, in dem die Software bei jeder Änderung erneut gebaut, getestet, inspiziert und auf einem Zielsystem installiert wird. Eine Kernaufgabe zum Erreichen dieses Ziels ist es einen möglichst hohen Grad der Automatisierung der einzelnen Schritte zu erreichen; es sind 100% Automatisierung anzustreben. Somit kann die Software quasi auf Knopfdruck von jedem Teammitglied oder vom Auftraggeber neu erstellt und auf eine Zielplattform installiert werden, um beispielsweise Enduser-Tests durchzuführen oder eine kurzfristig anberaumte Demo des aktuellen Entwicklungsstands vorzuführen. Ein gelebter CI-Prozess kam mit ein paar grundlegenden Regeln beschrieben werden. Der Workflow eines Entwicklers besteht aus den folgenden Arbeitsschritten:

1. Aktuellen Stand der Software aus dem SCM Repository holen
2. Sourcecode weiterentwickeln
    a. Unittest entwickeln oder aktualisieren
3. Software auf dem eigenen Rechner bauen und testen
    a. Solange Bugs beheben, bis alles Tests durchlaufen
4. Änderungen in SCM Repository einspielen
5. Ergebnis des Build-Prozess auf dem Build-Server abwarten
6. Gegebenenfalls auftretende Bugs sofort beheben

Bei diesem Workflow ist es wichtig, dass möglichst viele Arbeitsschritte automatisiert sind, damit sie unabhängig von Person und Umgebung wiederholt werden können. Hier gilt das bekannt DRY

(„Don't repeat yourself") Prinzip: sobald eine Tätigkeit ein zweites Mal durchgeführt wird, sollte es durch ein Skript automatisiert werden. Ein wichtiges Augenmerk muss man auf Schritt 6 lenken. Sobald ein Build fehlgeschlagen ist gilt die höchste Aufmerksamkeit des Projektteams dem Beheben der Bugs des letzten Builds. Während dieser Zeit sollte auch kein neuer Build-Prozess gestartet werden. Damit ändert sich die Priorität im Team kurzfristig, damit mit voller Kraft an dem CI-Ziel „Software steht in einem benutzbaren Zustand zu Verfügung" gearbeitet werden kann.
Die Vorteile eines solchen CI-Prozesses liegen auf der Hand:

- Kontinuierliches, qualifiziertes und rasches Feedback über den Entwicklungsfortschritt
  - Frühestes mögliches Erkennen von Konflikten
  - Aussage zum Einhalten von Architektur- und Entwicklungsrichtlinien
  - Verlässliche Aussage zu Entwicklungsstand und Prognose des Restaufwands
  - Minimierung von Entwicklungsrisiken
- Erlaubt laufendes Testing durch Kunde ohne Mehraufwand seitens Entwicklung
- Schafft Voraussetzung für Continuous Delivery
- Eignet sich sowohl für agile wie auch konventionelle Vorgehensweisen
- Erlaubt kurze Entwicklungszyklen bei besserer SW Qualität

Continuous Integration for Java as a Cloud Service
Der initiale Aufwand bis eine solche CI-Umgebung nahezu reibungslos im Projekt seine Aufgabe erfüllt ist im Projektbudget einzuplanen und kann nicht nebenbei erstellt werden. Gerade bei kleineren Projekten könnten bei den Projektverantwortlichen trotz der Vorteile eines CI-Prozesses Zweifel an seiner Wirtschaftlichkeit aufkommen. Betrachtet man nicht nur das aktuelle Projekt, sondern auch alle anderen vergleichbaren Projekte im Unternehmen, dann wird schnell klar: Eine generische Lösung der CI-Prozesse muss für die entsprechenden Projekte vorab erstellt werden (Anwendung des DRY Prinzips).
Trivadis hat zu diesem Zweck in der eigenen Trusted Cloud einen „Continuous Integration for Java" Service implementiert. Die Trusted Cloud Umgebung wird von Kunden und Triviadis Mitarbeitern gemeinsam genutzt, um Softwareprojekte durchzuführen. Ein Projekt beantragt über ein Portal ein oder mehrere Virtuelle Server für z.B. Datenbank- und Applikationserver. Zusätzlich wird ein „Continuous Integration for Java" Server bestellt. Damit ergibt sich folgendes Bild:

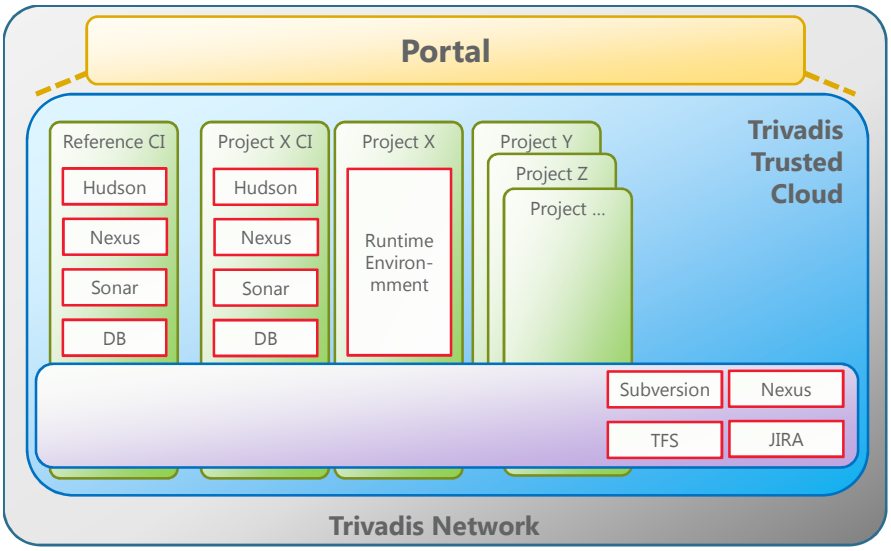

Abbildung 1 Trivadis Trusted Cloud

Die Komponenten der Trusted Cloud in zwei Gruppen aufgeteilt. Die vertikalen Komponenten bestehen aus der Referenz VM der CI for Java, den Projekt VM (DB-, App-Server, ...) und der Projekt CI VM, die eine projektspezifische Kopie der Referenz VM der CI for Java ist. Die horizontalen Komponenten bestehen aus den Shared Services, die von allen Projekten gemeinsam genutzt werden können. Dazu zählen Microsoft TFS, Subversion, Jira und zukünftig auch Nexus. Mit dieser Aufteilung der Komponenten auf verschiedene VMs wird die notwendige Flexibilität erreicht, wenn beispielsweise ein Kunde besonders hohe Sicherheitsanforderungen hat und sein Projekt gut isoliert von anderen Kundenprojekten halten will.
Der eigentliche „Continuous Integration for Java“ Service ist also eine projektspezifische Kopie der Referenz VM. Diese ReferenzVM hat folgenden Aufbau:

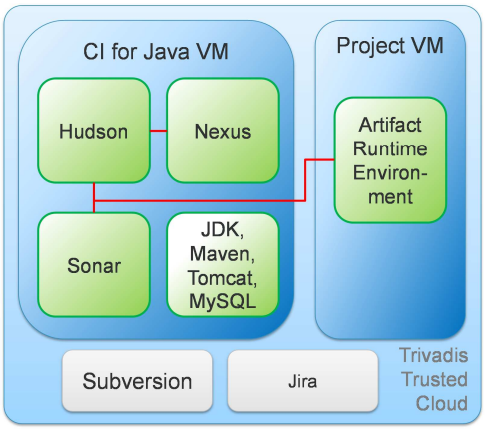

Abbildung 2 Referenz VM "Continuous Integration for Java"

Die Referenz VM wird durch den Bereitstellungsprozess zu Beginn des Projekts kopiert und dem Projekt zur Verfügung gestellt. Das Projekt bekommt vorinstallierte Komponenten, die nur noch individuell an die Bedürfnisse des Projekts angepasst werden müssen. Die wichtigste Konfiguration, die das Projekt vornehmen muss, ist die Integration des Build-Scripts in Hudson auf der Basis von Maven. Aktuell werden folgende Komponenten/Tools verwendet:

- Hudson 2.1.0
- Nexus 1.9
- Sonar 2.7 inkl. MySQL 5.5.9.
- Maven 2.2.1
- Sowie Tomcat 6 und JDK 1.6

Hat das Projekt die notwendigen Konfigurationen durchgeführt, dann wurden somit die wichtigsten CI-Prozessschritte automatisiert und standardisiert.
Selbstverständlich muss der „Continuous Integration for Java“ Service in der Trusted Cloud gepflegt werden. Ein Team von Trivadis Mitarbeitern ist mit dieser Aufgabe betraut und entwickelt die Referenz CI VM weiter. Dieses Team steht auch für Fragen und Support für die Projekt CI VM s zur Verfügung.

Mit Hilfe des „Continuous Integration for Java“ Service kann jedes Projekt schnell und unkompliziert den CI Prozess umsetzen und leben ohne zu Projektbeginn das Wer, Wie, Was der CI Thematik zu diskutieren. Don’t repeat yourself.

Continuous Integration im Oracle SOA Umfeld

Die Realisierung der Trivadis Java CI-Services war die Grundlage für die Erweiterung auf Oracle Fusion Middleware Projekte. Der anfängliche Fokus lag dabei auf OSB Projekte und SOA Composites. Ausschlaggebend war ein weiteres internes Projekt bei Trivadis, welches die SOA Suite inklusive OSB einsetzte. Vorgängige Projekte realisierten bereits gewisse Automatisierungen anhand projektbezogener Lösungen. Somit war eindeutig die Zeit gekommen eine einheitliche Lösung für alle Projekte zu realisieren (DRY Prinzip!). Die Lösung sollte sich am bereits vorhandenen „Continuous Integration for Java" Service orientieren und auf gleiche Art und Weise Services für OSB und SOA Projekte zur Verfügung stellen.

Herausforderungen

Für die Build und Deployment Automatisierung im Oracle SOA Umfeld mussten einige Hürden überwunden werden. Die Unterstützung seitens der eingesetzten Software Suiten ist zum Teil sehr dürftig, vor allem für den OSB. Die größte Herausforderung war das Maven basierte Bauen und Deployen von Artefakten. Anhand der relativ spärlichen Dokumentation blieb vielfach nichts anders übrig als Ausprobieren bis es geht oder auf Unterstützung aus dem Internet zu zählen.

Die Entwickler in den SOA Projekten hatten zudem oft Mühe im Umgang mit den WebLogic Servern und verbrachten viel Zeit für Anpassungen und Konfigurationsänderungen – ein weiterer Grund diese Problematik mit einem Set von Standardskripten für wiederkehrende Aufgaben zu lösen. Darunter fallen unter anderem die Konfiguration von Connection Pools und JMS Ressourcen, wie auch die Handhabung von den Managed-Servern.

Ziele

Im SOA Umfeld soll der Workflow eines Entwicklers genau gleich stattfinden wie es im vorgängigen Kapitel beschrieben ist. Schließlich handelt es sich abstrakt betrachtet um die gleichen Arbeiten: es bestehen Source Artefakte, welche gebaut oder zusammengestellt werden, und schlussendlich auf einem Server ausgeführt und getestet werden. Was gegenüber Java unterschiedlich ist, ist die Art des Bauen und Deployen, sowie die ganze Bereitstellung der Infrastruktur.

Der Infrastruktur kommt eine besondere Bedeutung zu. Sie ist das Fundamt eines Services in einer SOA. Ohne sie kann der Service weder ausgeführt, noch in den meisten Fällen getestet werden. Gleichzeitig ist sie von dynamischer Natur – es können langlaufende BPEL Prozesse oder Informationen in Queues vorhanden sein. Die wiederkehrenden Tests in einer CI Umgebung verlangen aber nach einer definierten Umgebung. Deshalb müssen Mechanismen vorhanden sein, um einen stabilen Ausgangszustand auf Knopfdruck zu erstellen. D.h. neben dem Bauen, Deployen und Testen von Software Artefakten eines Projektes ist ein weiteres Ziel das vollautomatische Erstellen der Infrastruktur. Dies umfasst alle benötigten Domänen, Admin- und Managed-Servern sowie den DB Schemas für die Metadaten.

Lösungsansätze

Das Anlegen der Metadaten Schemas wird mit dem Oracle Repository Creation Utility (RCU) bewerkstelligt. Mittels Ant Tasks wird es im Kommandozeilen-Modus mit den entsprechenden Parametern ausgeführt.

Das Erstellen der WebLogic Domäne(n) geschieht anhand Domänen Templates. Mit WebLogic Scripting Tools (WLST) wird danach eine neue Instanz angelegt, welche wiederum mit WLST gestartet, gestoppt und konfiguriert werden kann. Wiederkehrende Konfigurationen, wie JDBC Connection Pools und JMS Ressourcen, sind nicht im Template enthalten, sondern als parametrisierbare Skripte realisiert. Ein WLST Code Fragment zum Starten des AdminServer mit einer Reihe von ManagedServer ist in Abbildung 3 aufgeführt.

```
...
nmConnect(nm.user, nm.password, nmHost, nmPort, domainName, nmType=connectionType)
nmStart(adminServerName)
nmDisconnect()

connect(wlsAdminUser, wlsAdminPassword, wlsAdminHost);
nmEnroll(nmHome, nmHome=wlsDomainDir)

for mngtServerName in serverNameArray:
    start(mngtServerName, block='true')

disconnect()
```

Abbildung 3 WLST Beispiel: Admin & Managed Server starten

Die Möglichkeiten von WLST sind immens, alle Konfigurationseinstellung der WebLogic Konsole können skriptgesteuert durchgeführt werden.

Das Deployment von OSB Projekten basiert auf einem speziellen Configuration.jar. Diese Datei kann nur mit der OSB OEPE Eclipse Umgebung erstellt werden. Dank Kommandozeilen-Unterstützung kann die Datei dennoch skriptgesteuert erstellt werden. Eclipse wird zwar weiterhin verwendet, aber es ist zumindest keine graphische Oberfläche erforderlich. Der Import in den OSB ist mit WLST realisiert. Die einzelnen Schritte sind als Ant Tasks abgebildet und zusätzlich mit dem Maven Ant Task in den Maven Build Lifecycle (package, install, deploy) eingebunden. Der Build & Deployment Prozess ist in Abbildung 4 wiedergegeben.
Dieses Konzept ermöglicht die einfache Integration mit Nexus und separaten Hudson Jobs für das Bauen und Deployen von OSB Projekten. umgebungsspezifische Anpassungen wie Service Endpoints zum Testen sind mit sogenannten Customization XML Dateien möglich.

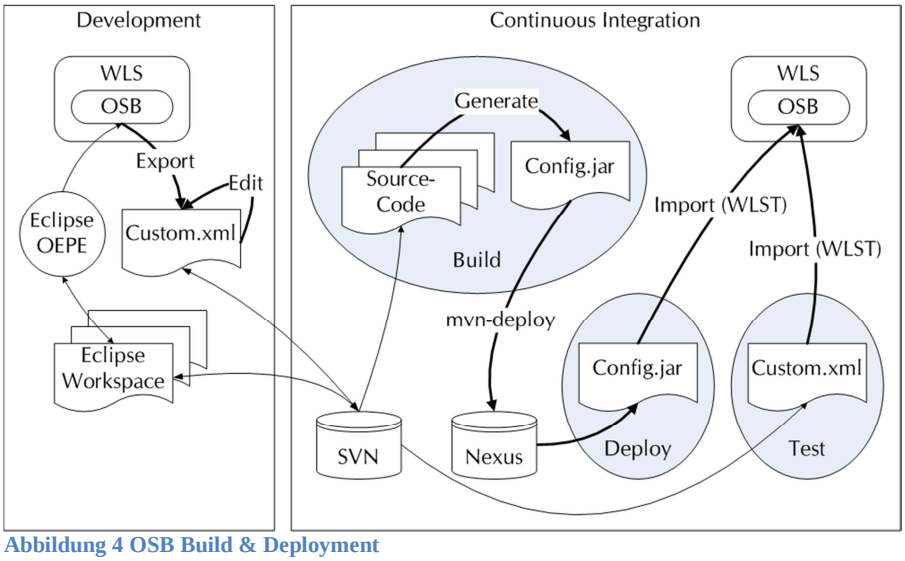

Abbildung 4 OSB Build & Deployment

In der SOA Suite Installationsumgebung existieren bereits vordefinierte Ant Skripte zum Bauen und Deployen von SOA Composites. Für eine konsistente Handhabung wurden die Ant Skripte wiederum in den Maven Build Lifecycle eingebunden. Die SCA Pakete werden mit einem Oracle spezifischen Namen erstellt der nicht Maven konform ist. Daher wird dieser vor dem Abspeichern in Nexus automatisch umbenannt und vor dem Deployment in die SOA Suite wiederum zurückgesetzt.
Der Build & Deployment Prozess ist in Abbildung 5 dargestellt.

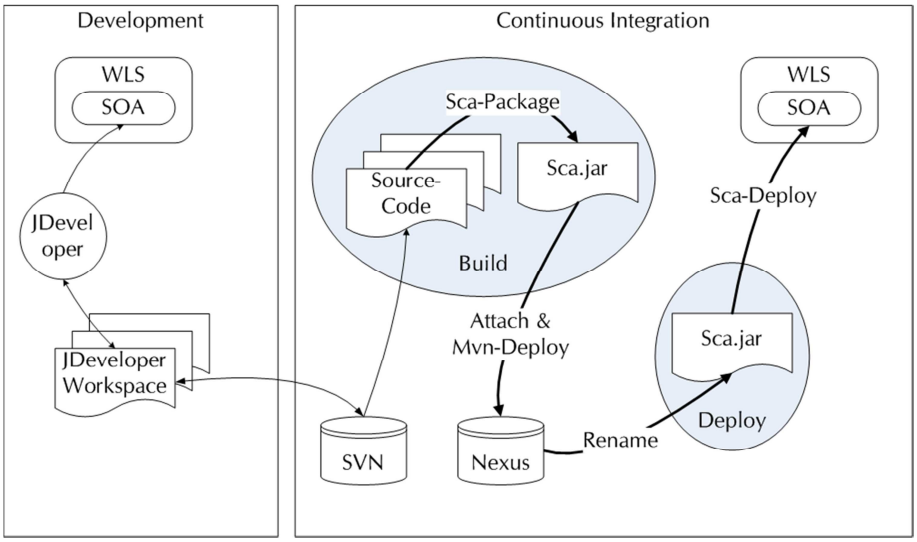

Abbildung 5 SOA Suite Build & Deployment

Die jeweilige Aufteilung von Bauen und Deployment erlaubt nicht nur die einfache Einbindung von einem Maven Repository (Nexus), sondern auch eine grössere Flexibilität beim Deployment. Die hinterlegten Artefakte im Maven Repository können mit dem Deployment Job auf verschiedene Umgebungen deployed werden. Warum sollte ein Deployment auf die Referenz- oder Abnahme-Umgebung von Hand geschehen, wenn dies mit einem Job-Parameter in Hudson elegant gelöst werden kann?

Umgesetzte Lösung: CI for SOA as a Cloud Service

Trivadis hat mit einem internen Experten Team aus den Fachbereichen Application Development und Infrastructure Management Services die SOA-CI Lösung umgesetzt. Als Ausgangslage wurde die „CI for Java“ Referenz VM verwendet und mit den benötigten Komponenten und Tools erweitert. Somit ergibt sich ein ähnlicher Aufbau wie für Java, was in Abbildung 6 ersichtlich ist. Gleich wie für den „CI for Java“ Service wird eine projektspezifische Kopie von der Referenz VM erstellt und dem Projekt zugewiesen.

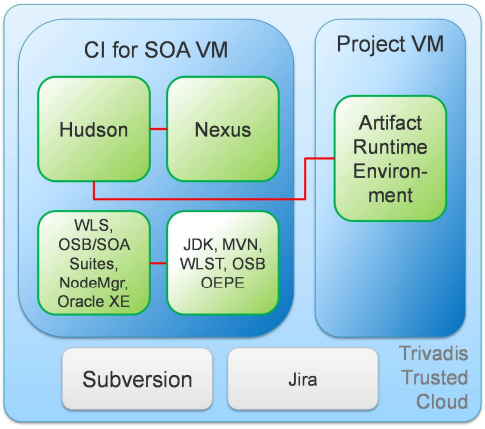

Abbildung 6 Referenz VM „Continuous Integration for SOA“

Aktuell werden folgende Komponenten/Tools verwendet:

- Hudson 2.1.0
- Nexus 1.9
- Ant 1.8.2 und Maven 2.2.1
- Oracle WebLogic Suite 11g und SOA Suite 11g
- Oracle XE Datenbank für die Metadaten der SOA Suite und ggF. auch für das Projekt

Die Vorinstallation umfasst eine startfähige WebLogic Domäne mit einem Admin-Server und zwei Managed-Server. Ein Server für den OSB und der zweite für die SOA Suite. Auf Hudson sind vordefinierte Jobs hinterlegt, welche die WebLogic Umgebung neu aufsetzt, die Server startet oder herunterfährt. Damit erhält ein Projekt eine einsatzfähige Umgebung, welche bereits vielen Anwendungsfälle abdeckt. Das Domänen Template und Anzahl Server kann vom Projekt angepasst werden. Die im vorgängigen Kapitel aufgeführten Skripte sind ebenfalls hinterlegt und müssen nur noch eingebunden werden.

Die Arbeiten sind damit nicht abgeschlossen. Analog dem Java-CI Service ist ein Team von Trivadis Mitarbeitern für den Support und Weiterentwicklung verantwortlich. Die SOA Suite ist umfangreich, und weitere Komponenten müssen für zukünftige Projekte miteinbezogen werden. Bereits geplante Arbeiten sind unter anderem die Erweiterung auf Hudson Build Slaves und die bessere Konfiguration von JCA Adaptern (momentan nur mittels Domänen Template konfigurierbar).

Fazit
Der initiale Aufwand um in einem Projekt kontinuierliche Integration zu betreiben ist mit den hier vorgestellten Services drastisch gesunken. Dies gilt insbesondere für SOA Projekte, wo Automatisierungen mehr Aufwand erfordern. Dies ist aber nur einer von vielen weiteren positiven Aspekten, welche anfangs in den CI Konzepten aufgeführt wurden. Man kann es nicht genug erwähnen: CI führt zu effizienterer Software Entwicklung und höherer Qualität! Egal mit welcher Sprache entwickelt wird.

Der „Continuous Integration for Java“ Service ist bei Trivadis bereits in mehreren Projekten erfolgreich im Einsatz. Die Implementierung der SOA-CI Umgebung kommt momentan in einem internen Referenz Projekt zum Einsatz. Der Release des darauf basierenden „Continuous Integration for SOA“ Services ist für Oktober 2011 geplant.

Zusätzlich entstanden aus den Arbeiten für die SOA-CI Referenz VM weitere Nebenerzeugnisse. Eines davon ist eine Referenz Entwicklungs-VM, welche den Aufwand für neue Projekt Mitglieder von ehemals gut einem Tag auf unter eine Stunde reduzierte. Zudem flossen WebLogic Erkenntnisse in die Trivadis Toolbox TVD-BasEnv™ ein, wovon wiederum ein größerer Personenkreis profitiert.

Kontaktadresse:

Markus Heinisch
Trivadis GmbH
Lehrer Wirth Str. 4
D-81829 München

Telefon:	+49 (0) 89 99275930
Fax:	+49 (0) 89 99275959
E-Mail	markus.heinisch@trivadis.com
Internet:	www.trivadis.com