

Nearshoring

KM-Prozesse und Technologien als Erfolgsfaktor

Gunther Popp
Eric Wirth-Durst

Cirquent - Daten und Fakten

- Business- und IT-Consulting
- **Cirquent ist unter den Top 10 in Deutschland**
- Geschäftsentwicklung 2006: 10% Wachstum
300 Neueinstellungen
- 2007: 1800 Mitarbeiter & 284 Mill.€ Umsatz
- Branchenschwerpunkte
 - Fertigungs- und Automotive Industrie
 - Telekommunikation
 - Finanzdienstleister
 - Versicherungen
- SAP Service Partner

cirquent | softlab
credible consulting | group

Vereinigung aller
Beteiligungen als
Cirquent GmbH
2008

softlab
group

1700 Mitarbeiter
262 Mio. Euro
Umsatz
2006

softlab

Gründung in
München als
„Software Labor“
1971

100%ige
BMW Tochter
1992

nexolab

Gründung
Nexolab
2000

axentiv

Übernahme
Axentiv
2004

entory

Übernahme
Entory
2005

Cirquent Portfolioübersicht



Branchen

Finanzdienstleistungen, Versicherungen, Fertigungsindustrie, Telekommunikation, weitere Branchen

Unterstützende Prozesse

Application Management
Customer Management
Finance Transformation
IT-Management
SAP-Consulting

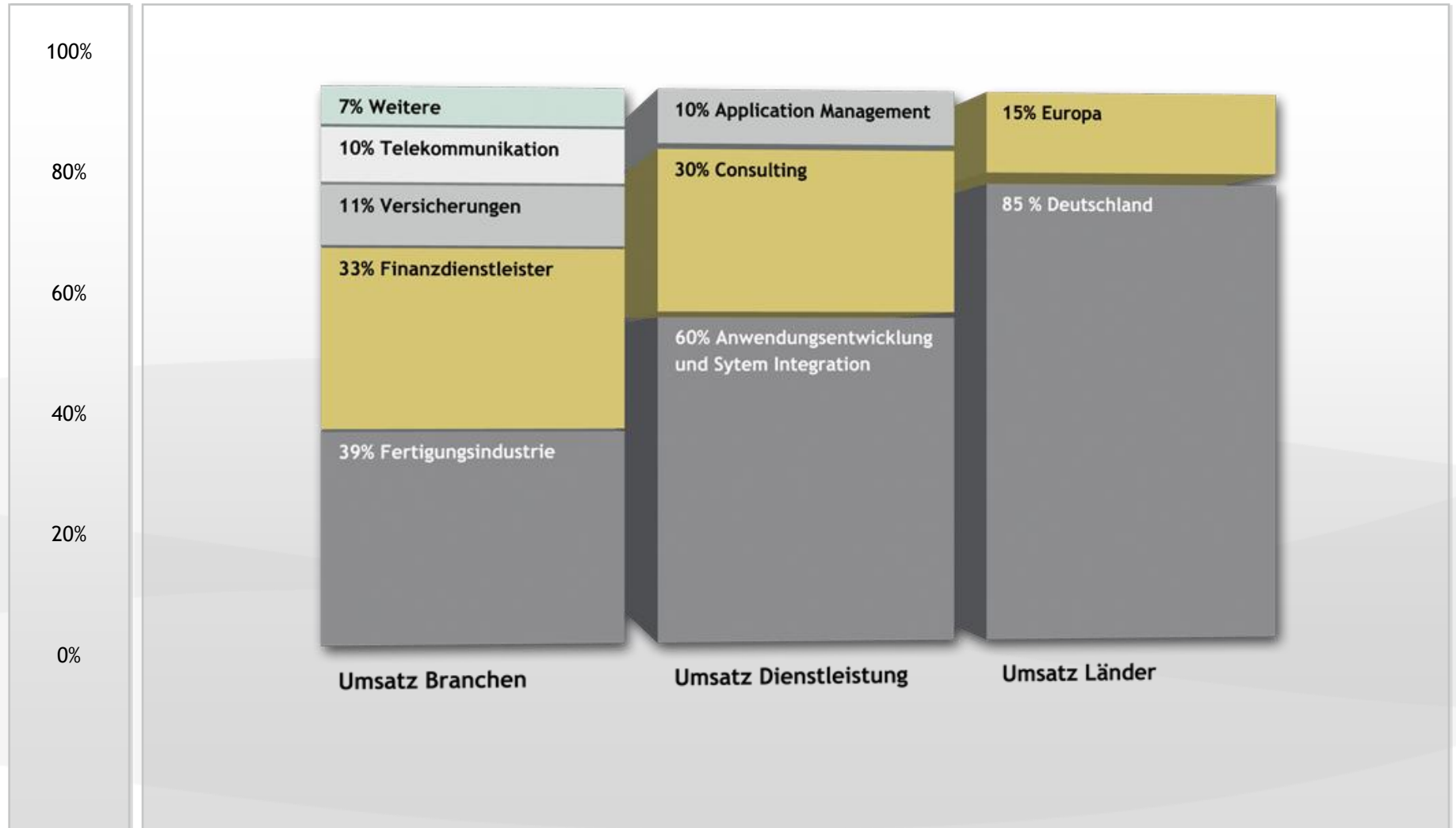
Methoden und Technologien

Business Consulting
Business Process Management
Business Intelligence
Portale
IT-Engineering
System Integration
Project Management

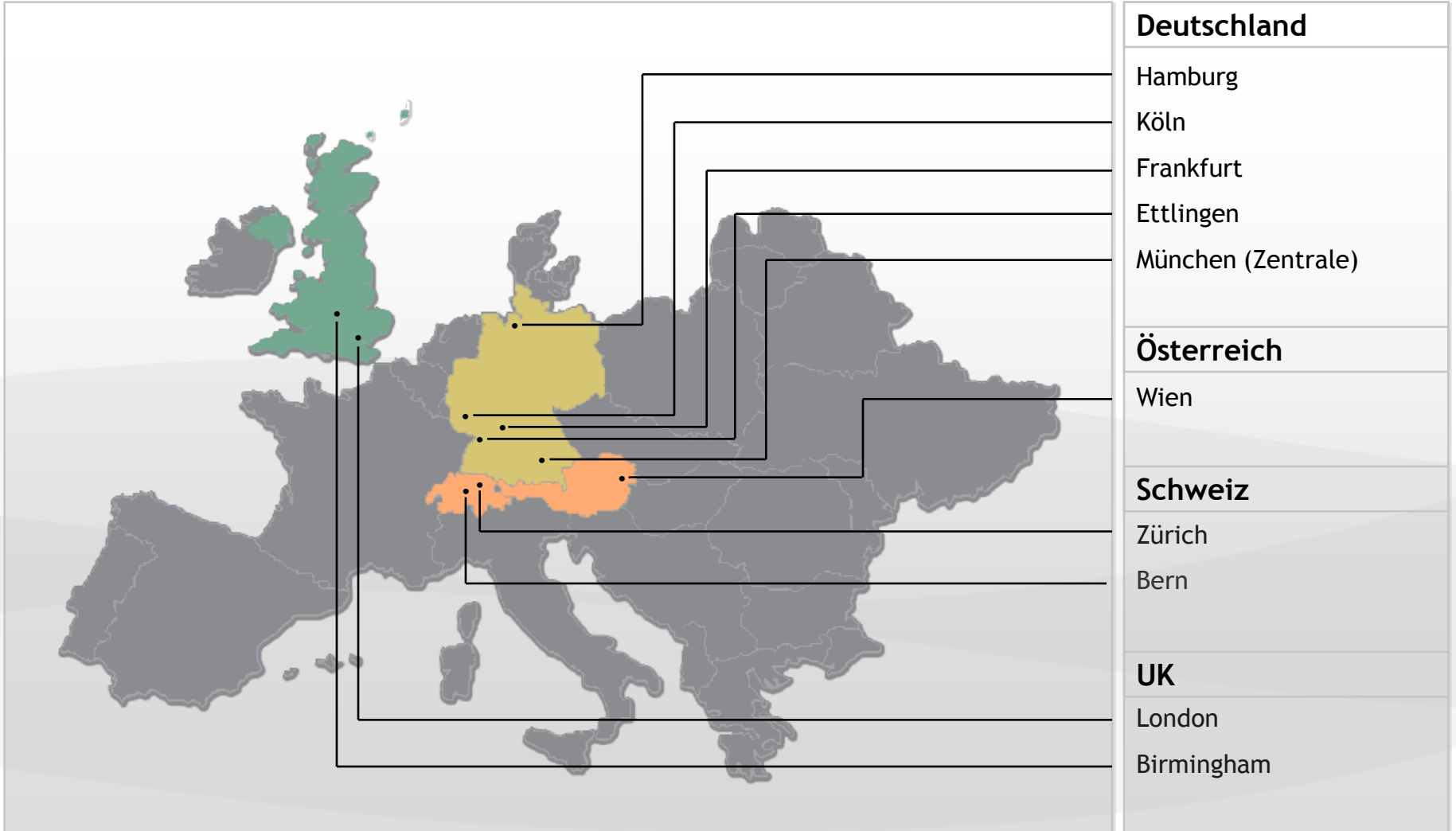
Partner

Avaya, BMC, Genesys, Oracle, SAP und weitere

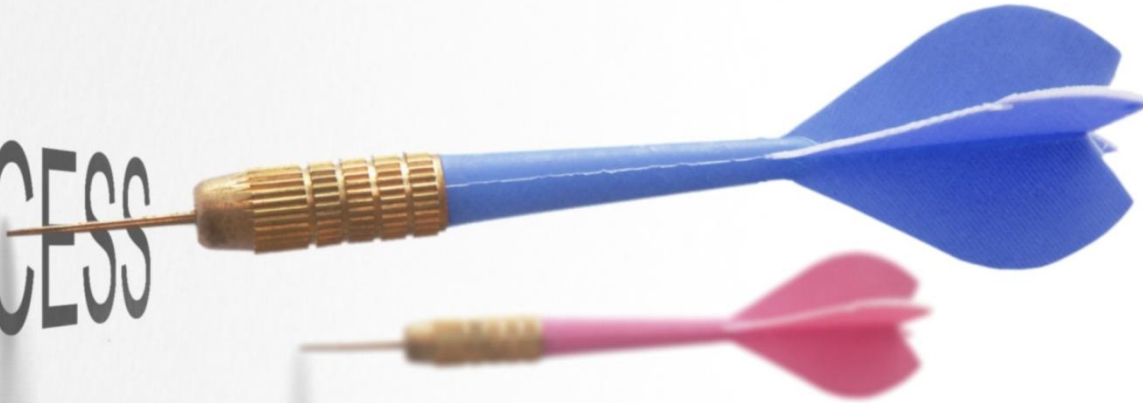
Erfolgreiche Ausrichtung (Stand 2006)



Cirquent Niederlassungen



SUCCESS



Was ist Nearshore / Offshore?

- Verlagerung von “teuren Arbeitsplätzen” in so genannte Niedriglohnländer nach Osteuropa (Nearshore) bzw. Indien (Offshore)
- Ersetzen von nicht vorhandenen Arbeitskräften

Nearshore vs. Offshore

- Kleiner Zeitunterschied (1-2 Std.) (statt 5 Std. Indien)
- Geringe bis keine kulturelle Unterschiede
- Sprache (oft wird Deutsch verstanden)

Warum Nearshore?

- Hoher Preisdruck am Markt und verlorene Chancen
- Spitzenlastausgleich über freie Mitarbeiter

Risiken des Nearshore

- Qualitätseinbussen
- Höherer Aufwand durch zusätzliche Kommunikation und Management
- Akzeptanzprobleme beim eigenen Entwicklerteam
- Datenschutz
- Know-How-Schutz
- Anlaufphase wird immer wieder unterschätzt



Chancen beim Nearshore

- Verbesserung der internen Entwicklungsprozesse
- Verbesserung der Infrastruktur für die SW-Entwicklung
- Kosteneinsparung
- Zusätzliche Möglichkeit zur Beherrschung von Auslastungsspitzen
- Ausbau des Technologie Know-How
- Partnerschaften



Lessons Learned

Kommunikation

- Kompetente und zeitnah verfügbare Ansprechpartner auf beiden Seiten.
- Gemeinsam nutzbare Plattformen (z.B. Skype, Sharepoint u.a.)

Qualität

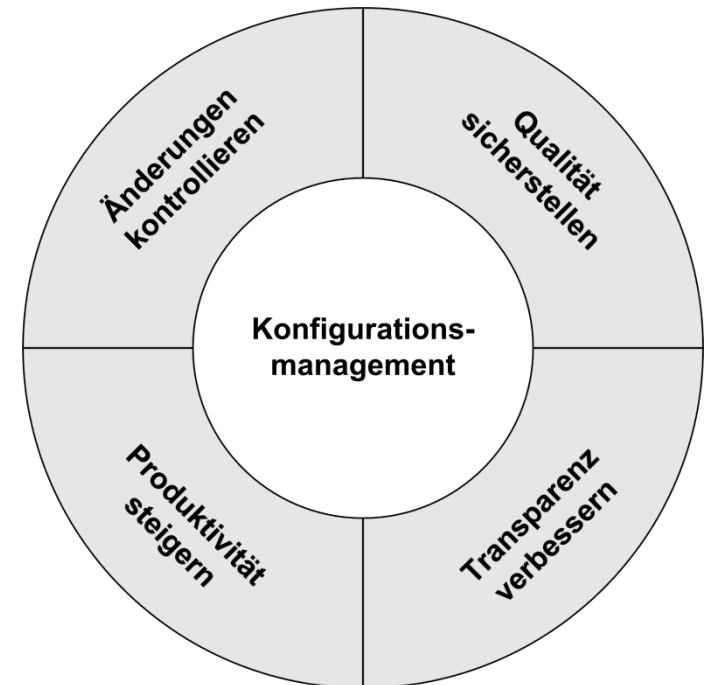
- Aufbau einer gemeinsam verwendbare Infrastruktur.
- Abstimmung der SW-Entwicklungsprozesse.
- Abstimmung der Styleguides und Regeln.
- Vollständige und gemeinsam verfügbare Dokumentation (Konzepte, Architektur, Testfälle, u.a.).



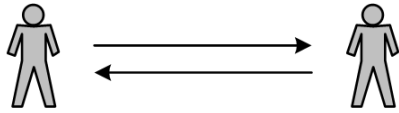
Konfigurationsmanagement als Erfolgsfaktor für Nearshoring

- Nearshoring - die typischen Probleme
 - **Kommunikation:** Missverständnisse, keine gemeinsame Linie
 - **Effizienz:** Langes „Einschwingen“, hohe Anlaufverluste
 - **Qualität:** Willkürliche Änderungen im System, keine stabilen Releases
 - **Transparenz:** Wer arbeitet an was? Wo stehen wir?

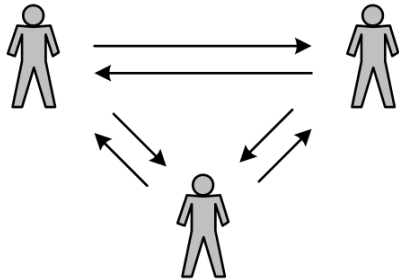
Ein KM-Prozess hat das Ziel, genau diese Probleme zu beheben



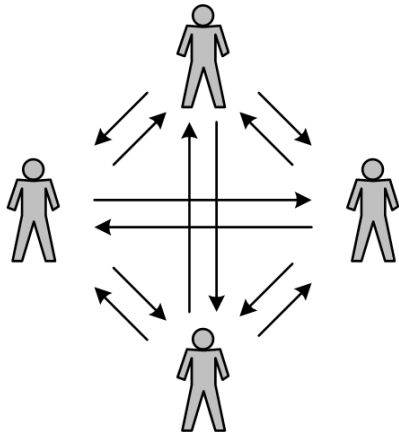
Lösung des Kommunikationsproblems



2 Personen im Team, 2 Kommunikationswege



3 Personen im Team, 6 Kommunikationswege



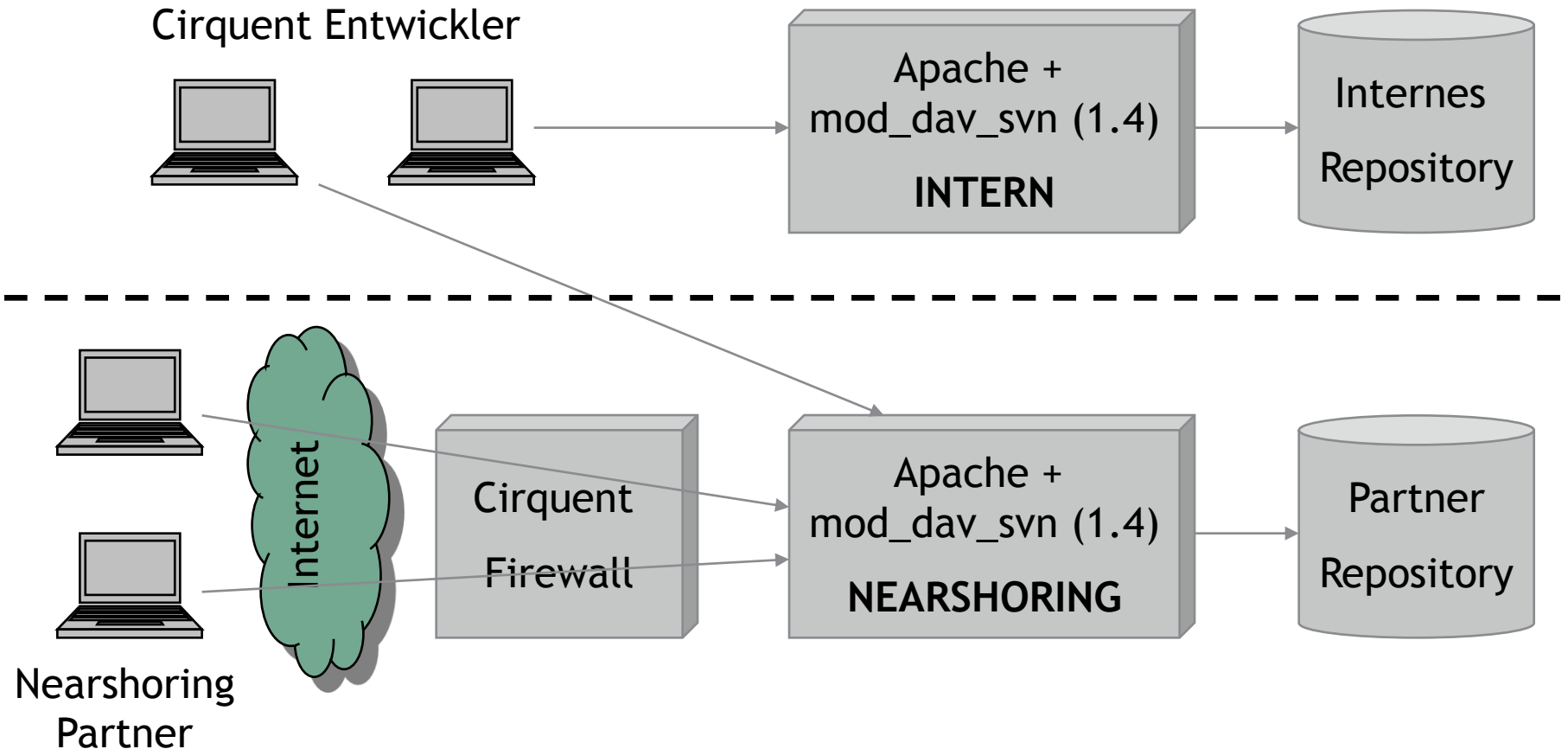
4 Personen im Team, 12 Kommunikationswege

- Der Abstimmungsaufwand steigt überproportional mit der Teamgröße
- Zusätzliche Herausforderung beim Nearshoring: International verteilte Teams mit Sprach- und Kulturunterschieden

Lösungsansatz des KM-Prozesses

- Zentrale Verwaltung der Konfigurationselemente (Source-Code, Test-Cases, etc.) in Repositories
→ Nachvollziehbarkeit von Änderungen
- Zusätzlich Einrichtung von Projekthomepages und Wikis in MS Sharepoint
→ Zentrale Kommunikationsplattform

Subversion-Infrastruktur bei Cirquent



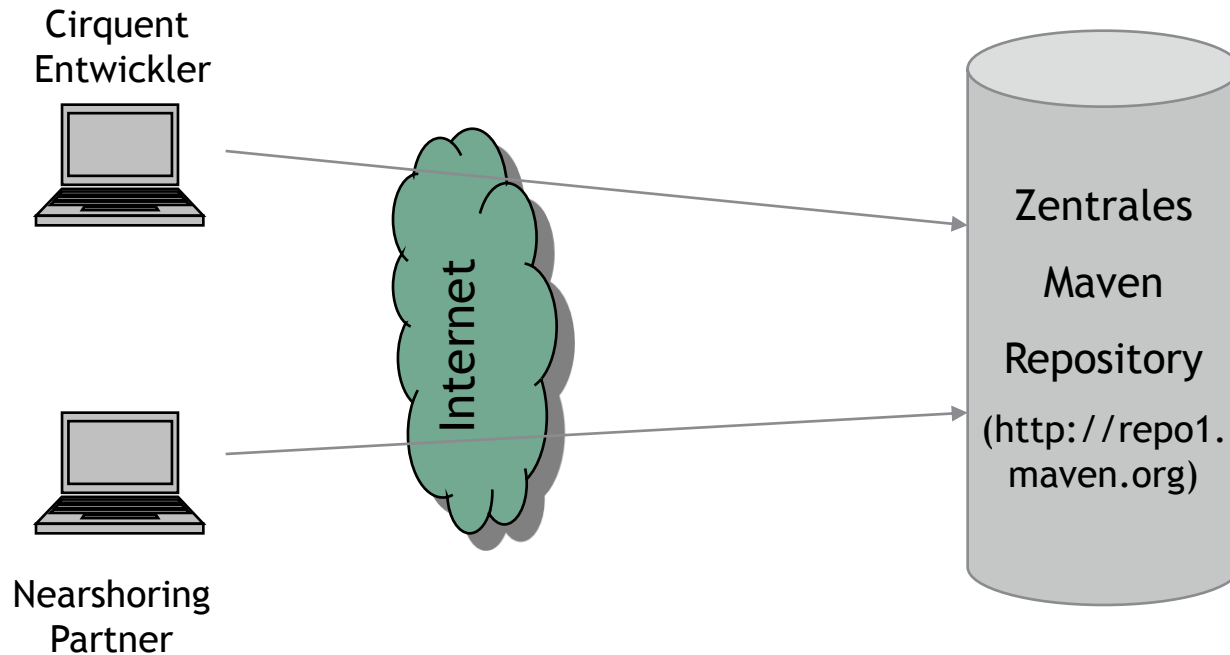
Lösung des Effizienzproblems

- Entwicklungsteams brauchen eine gewisse „Einschwingphase“
- In dieser Zeit wird z.B. die Entwickler-Infrastruktur eingerichtet (Eclipse, ...), die Projektstruktur angelegt und der Build-Prozess implementiert
- Sitzen alle Entwickler „in einem Raum“, können Probleme und Fragen sofort geklärt werden
- In gemischten Teams mit einem Nearshoring-Partner funktioniert das nicht
- Gefahr: Die „Einschwingphase“ zieht sich inakzeptabel in die Länge, das Team arbeitet nicht effizient

Lösungsansatz des KM-Prozesses

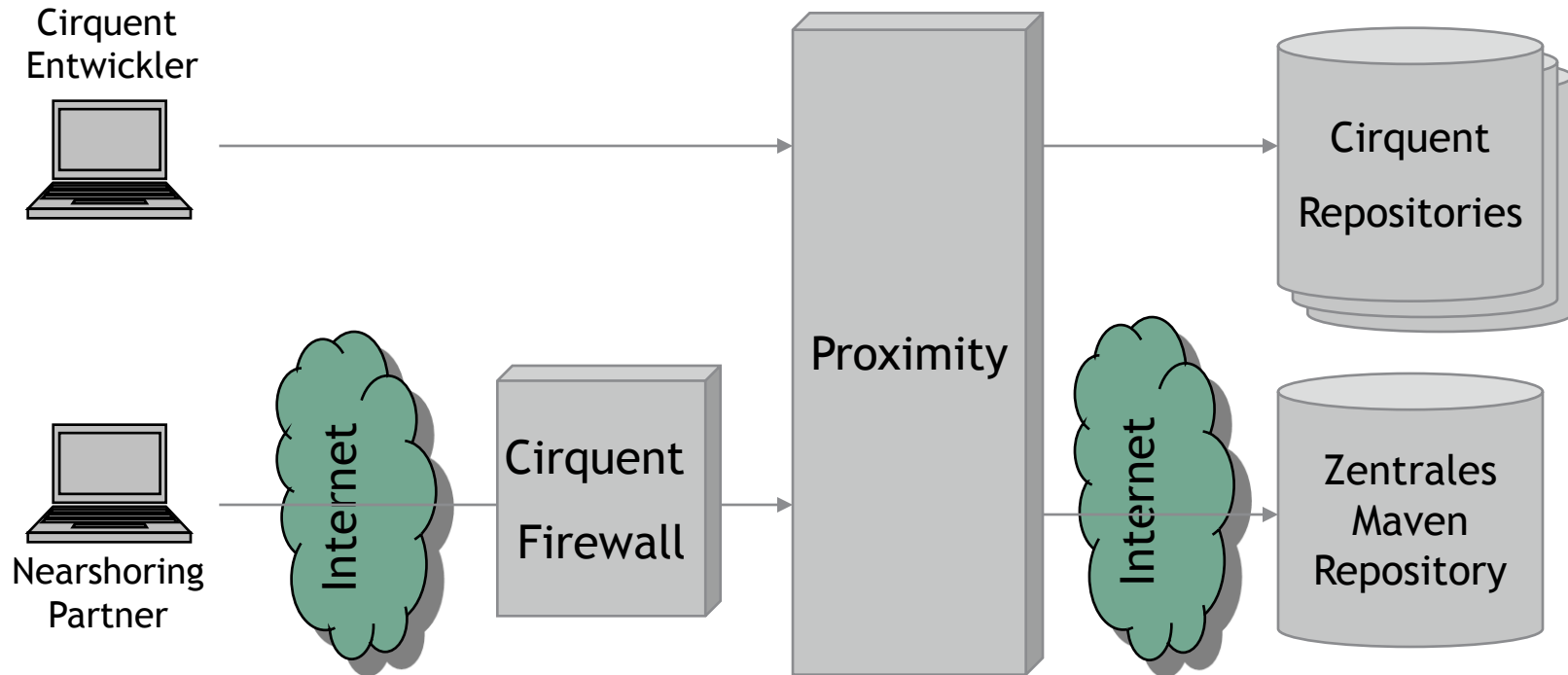
- Templates und Hilfestellungen für die Einschwingphase:
 - Muster für Projektstrukturen bis hin zu kompletten Musterprojekten
 - Einheitlicher Build-Prozess (Maven)
 - Templates für die Entwicklerdokumentation

Maven-Infrastruktur - Standardvariante



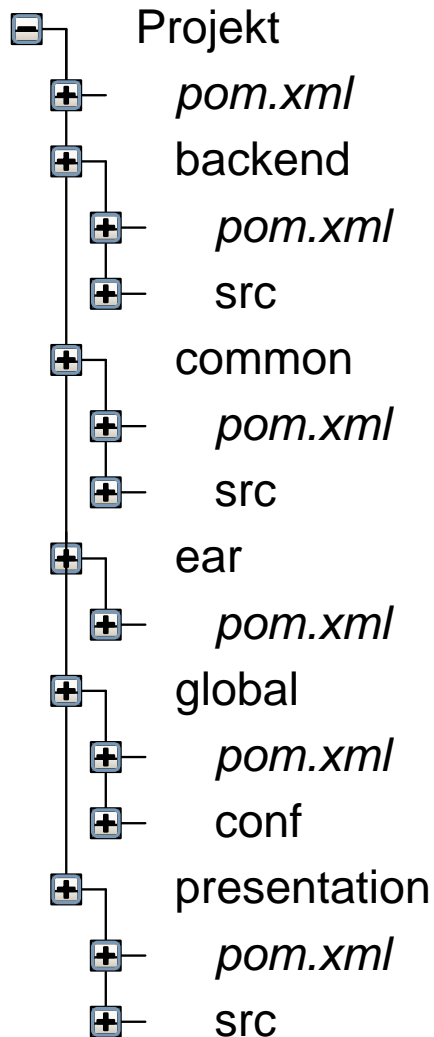
- Die von Maven standardmäßig verwendete Infrastruktur basiert auf zentralen Maven-Repositories im Internet
- Für Unternehmen ist diese Variante völlig ungeeignet

Maven-Infrastruktur bei Cirquent



- Cirquent verwendet „eigene“ Maven-Repositories im Intranet auf Basis von Proximity
- Neben einem internen Spiegel von <http://repo1.maven.org> existieren auch projektspezifische Repositories

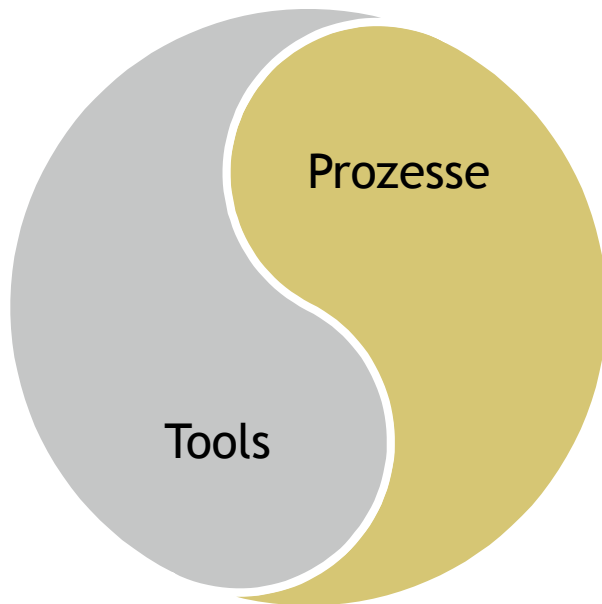
Bereitstellung von Maven-Archetypes



- Maven implementiert von Haus aus einen einheitlichen Build-Prozess
 - Vorteil: Nur einmalige Einarbeitung der Entwickler (auch beim Partner!)
 - Nachteil: Hoher Aufwand für die initiale Konfiguration eines Projektes
- Ein Maven-Archetype ist nichts anderes als ein Template, das die grundlegende Projektstruktur und die dafür passende Maven-Konfiguration enthält
- Features eines Cirquent-Archetypes:
 - Vorschlag für eine bewährte Projektstruktur
 - Lauffähige Testanwendung für einen bestimmten Applikationstyp
 - Konformität zu Architekturvorgaben
 - Basisfunktionalität für Exception-Handling, etc.
 - Vorkonfigurierte Plugins für QS, Unit-Tests, Reports (z.B. CheckStyle)
 - Zugriff auf die Cirquent-Infrastruktur (z.B. Proximity)

Lösung des Qualitätsproblems

Softwarequalität



- Hohe Softwarequalität ist nur durch ein abgestimmtes Paket von Prozessen und Tools zu erreichen - insbesondere im Nearshoring-Umfeld!
- Prozesse bei Cirquent:
 - Anforderungen und Softwarearchitektur entstehen immer bei Cirquent
 - Bildung abgeschlossener Pakete für Nearshoring Partner
 - Kurze Iterationen, regelmäßige Lieferungen
- Tools
 - Automatisierte Integrationsbuilds (CruiseControl)
 - Automatisierte technische QS (Metriken)
 - Automatisierte Modultests
 - Zentrale Task- und Defectmanagement (Jira)

Automatisierte technische QS bei Cirquent

- Cirquent setzt eine Suite von verschiedenen Tools ein:
 - Checkstyle: Konformität zu Coding-Standards, Metriken
 - FindBugs: Findet typische „Fehlermuster“
 - Dependometer: Prüft Konformität des Codes zur Architektur. Gibt Liste von ungültigen Beziehungen zwischen Modulen, Paketen, etc. aus (siehe unten)

package violations

```
[15] com.bmw.eric.backend.kernel.base.jms => javax.jms <external>
[4] com.bmw.eric.backend.kernel.adapter.dmsoutbound.commands.v101 => com.bmw.eric.ifc.kernel.base.customer
[4] com.bmw.eric.backend.kernel.adapter.dmsoutbound.vo => com.bmw.eric.ifc.kernel.base.customer
[4] com.bmw.eric.backend.kernel.service.e4npublishing.ba.inbound => javax.jms <external>
[4] com.bmw.eric.backend.kernel.service.e4npublishing.ba.inbound => org.apache.xmlbeans <external>
[3] com.bmw.eric.backend.kernel.service.e4npublishing.ba.outbound => org.apache.xmlbeans <external>
[2] com.bmw.eric.backend.kernel.adapter.dmsinbound.commands.v101 => com.bmw.eric.backend.kernel.service.resourceearch.ba
[2] com.bmw.eric.backend.kernel.adapter.dmsinbound.commands.v101 => com.bmw.eric.ifc.kernel.service.resourceearch
[2] com.bmw.eric.backend.kernel.adapter.dmsoutbound.vomapper => com.bmw.eric.ifc.kernel.base.customer
[2] com.bmw.eric.backend.kernel.adapter.migration.connection => oracle.jdbc.pool <external>
[2] com.bmw.eric.backend.kernel.adapter.migration.utils => com.bmw.eric.backend.kernel.base.exceptionhandler
[2] com.bmw.eric.backend.kernel.base.dom => com.bmw.eric.backend.kernel.logic.masterdata.vo
[2] com.bmw.eric.backend.kernel.base.util => com.bmw.eric.backend.kernel.logic.base.dom
[2] com.bmw.eric.backend.kernel.service.e4npublishing.ba.reply => org.apache.xmlbeans <external>
```

Lösung des Transparenzproblems

- Typische, banale Fragen im Projektalltag:
 - Wer arbeitet an was?
 - Wo stehen wir?
- Lösungsmöglichkeiten:
 - MS Project
 - Excel
 - Statusmeetings mit Kontrolle der zugewiesenen Tasks
 - ...
- In Nearshoring-Projekten braucht man eine professionellere Infrastruktur
- Cirquent setzt daher auf Jira als zentrales Instrument zur Taskverwaltung und Fortschrittskontrolle
 - Verwaltung von Entwickleraufgaben
 - Verwaltung von Defects
 - Verwaltung von Change Requests

Jira bei Cirquent

- Allgemeine Funktionalität
 - Umfangreiche Funktionalität für Task-Tracking Aufgaben
 - Zentrales Management von Projekten und Benutzern
 - Weitreichende Möglichkeiten zur Rechte und Rollenvergabe
 - Freie Gestaltbarkeit der Arbeitsabläufe und Masken sowie des Reportings
 - Sehr gute Eclipse Integration
 - Offenes API und Plugin Erweiterungen
- Einsatz bei Cirquent
 - Zugriff über das Internet per https möglich - Nearshoring-Partner und interne Mitarbeiter arbeiten mit demselben Tool
 - Vorkonfigurierte Arbeitsabläufe und Masken für unterschiedliche Projekttypen
 - Integration in die Entwicklerarbeitsplätze (Eclipse & Mylyn)

Fragen ?

Gunther Popp

Telefon: + 49 160 820 89 13

WWW: <http://www.guntherpopp.de>

E-Mail: gpopp@guntherpopp.de

Eric Wirth-Durst

Telefon: + 49 89 9936 - 1275

Fax: + 49 89 9936 - 1443

E-Mail: Eric.Wirth-Durst@cirquent.de



<http://www.km-buch.de>