2009

ZIELORIENTIERTE DATENMODELLIERUNG FÜR ITILBASIERTE INTER-ORGANISATIONALE CONFIGURATION MANAGEMENT DATABASES

Wolfgang Hommel
Leibniz-Rechenzentrum, Garching b.München

Silvia Knittl
Technische Universität München

ZIELORIENTIERTE DATENMODELLIERUNG FÜR ITIL-BASIERTE INTER-ORGANISATIONALE CONFIGURATION MANAGEMENT DATABASES

Wolfgang Hommel[1], Silvia Knittl[2]

Kurzfassung
Die essentiellste Aufgabe des IT-Service-Managements (ITSM) ist die Ausrichtung der IT an Kundenbedürfnissen. Zunehmende Dynamik, Infrastrukturkomplexität und Regularien, beispielsweise Compliance-Richtlinien, erforden eine Werkzeugunterstützung des ITSM. Hierfür hat sich das Konzept einer ITIL-basierenden Configuration Management Database (CMDB) als Informationsnexus durchgesetzt. Wir diskutieren den Einsatz einer CMDB im Zusammenspiel mit inter-organisationalen Diensten; das zugrundeliegende Werkzeug bezeichnen wir als ioCMDB. Wir analysieren Anforderungen, die sich bei organisationsübergreifenden Anwendungsfällen an das verwendete Datenmodell ergeben und stellen eine Methodik zur zielorientierten Datenmodellierung vor. Darauf aufbauend stellen wir technische Komponenten zur Realisierung der Datensynchronisation zwischen organisationsinternen CMDBs und einer ioCMDB vor und diskutieren die resultierenden Mehrwerte.

1. Einleitung

Durch die kontinuierlich steigende Komplexität von Informations- und Kommunikationstechnologie in Unternehmen und die immer stärkere Abhängigkeit wichtiger Geschäftsprozesse von der IT bekommen Governance, Risk-Management und Compliance (GRC) eine immer wichtigere Bedeutung. Durch das Business Alignment der IT muss die Ausrichtung der IT-Strategie an die Geschäftsstrategie und damit auch an GRC-Anforderungen gewährleistet werden.

IT-Service-Management (ITSM) Rahmenwerke, wie z.B. die IT Infrastructure Library (ITIL) [7], definieren ITSM-Prozesse, um diese Zielsetzung zu unterstützen. Ein unabhängig von der Größe des Unternehmens essentieller Managementprozess ist hierbei das sogenannte *Service Asset and Configuration Management* (SACM). Das primäre Ziel von SACM ist es, die Geschäfts- und Kundenanforderungen bzgl. des Controllings zu unterstützen. Indem es akkurate Konfigurationsinformationen bereitstellt, dient es weiterhin allen anderen ITSM-Prozessen, z.B. dem Incident Management und dem Change Management. Für diese Aufgaben sieht das SACM als Werkzeug ein sogenanntes *Configuration Management System* (CMS) vor. Dieses beinhaltet als wesentliche

[1] Leibniz-Rechenzentrum, Garching b.München, Germany
[2] Technische Universität München, Germany

Komponente eine integrierte *Configuration Management Database* (CMDB). Mit dieser werden Konfigurationselemente (*Configuration Items*, (CI)) und deren gegenseitige Beziehungen (Relationen) abgebildet. CIs umfassen hierbei Servicekomponenten, Finanzgegenstände (Assets), Release-Pläne, Personen, Server, Anwendungen, Daten usw.

Der Trend zu Outsourcing bzw. kooperativer Diensterbringung, z.B. im Rahmen von Grid- und Cloud-Computing, stellt auch das GRC-Management vor neue Herausforderungen. Bei der betrieblichen Umsetzung ist eine geeignete Werkzeugunterstützung der organisationsübergreifenden ITSM-Prozesse erforderlich. Diese organisationsübergreifenden Aspekte wurden jedoch bisher von ITSM-Rahmenwerken oder auch von Werkzeugherstellern nicht oder nur unzureichend berücksichtigt. Anhand eines Szenarios werden wir im Folgenden motivieren, dass sich für organisationsübergreifend erbrachte IT-Dienste eine CMDB-basierte Werkzeug- und Prozessunterstützung eignet und kostengünstig realisieren lässt. Diesen Ansatz bezeichnen wir als inter-organisationale CMDB (ioCMDB). In Abschnitt 2 zeigen wir, welche Auswirkungen der Einsatz einer ioCMDB auf die Datenmodellierung und das Zusammenspiel mit bestehenden CMDBs hat. Daraus abgeleitete, zur Realisierung notwendige technische Komponenten stellen wir in Abschnitt 2.4 vor; eine Zusammenfassung der bisherigen Ergebnisse und ein Ausblick auf unser weiteres Vorgehen schließen den Artikel ab.

1.1. Kooperativ erbrachte Dienste in multi-domain Umgebungen

In diesem Abschnitt stellen wir exemplarisch ein Szenario für Dienste vor, die von mehreren Organisationen kooperativ erbracht werden, das aus dem Umfeld des Grid- und Cloud-Computings motiviert wurde. Abbildung 1 zeigt ein Schichtenmodell, dessen unterste Ebene die involvierten Organisationen samt ihren Ressourcen und Subservices darstellt. Die Benutzer, die in ihrer Gesamtheit als Community bezeichnet werden, nutzen jedoch nicht die einzelnen Ressourcen, sondern höherwertige inter-organisationale Dienste, die auf Basis einer Kooperation angeboten und z.B. durch Web-Service-Orchestrierung implementiert werden. Die gestrichelten Linien stellen beispielhaft die Abhängigkeit dieser Dienste von den jeweils lokal bereit gestellten Ressourcen dar, die z.B. durch den Einsatz web-basierter Frontends oder dedizierter Middleware zwar für den Benutzer, nicht aber für das integrierte Service Management transparent sind.

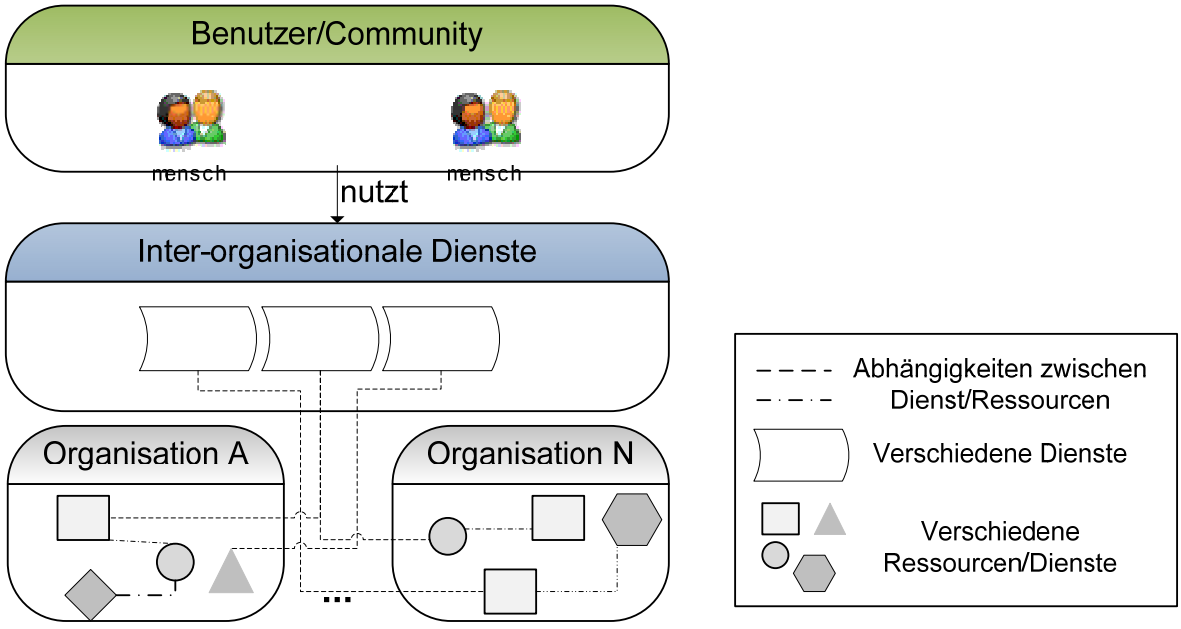

Abbildung 1 Zusammenhang zwischen inter- und intra-organisationalen Diensten

Im Folgenden setzen wir die technischen Herausforderungen bei der transparenten Nutzung von inter-organisationalen Diensten vereinfachend als gelöst voraus (vgl. [3]) und fokusieren auf das ITSM solcher Dienste. So sind etwa im operativen Betrieb im Falle von Störungen Informationen über die gegenseitigen Subservice- und Ressourcenabhängigkeiten notwendig, damit Störungsmeldungen schnellstmöglich an den richtigen Ansprechpartner beim jeweiligen Betreiber weitergeleitet werden können. Für die Planung von Dienstzuteilungen ist es ferner von entscheidendem Vorteil, wenn auf Organisationsebene autonom geplante Änderungen oder Wartungsarbeiten, welche die inter-organisationalen Dienste beeinträchtigen können, bekannt sind und kommuniziert werden können. In der Praxis gibt es allerdings bisher keine Werkzeugunterstützung für ein solches inter-organisationales ITSM (ioITSM). Daraus resultieren in der Praxis oft wesentliche Nachteile, z.B.:

- Das Beheben von Störungen verzögert sich, da das inter-organisationale Incident-Management (ioIncident-Management) erst die betroffene Organisation ermitteln muss.
- Das Planen von Änderungen auf inter-organisationaler Dienstebene ist komplex, da eine aufwendige manuelle Koordination aller beteiligten Diensterbringer notwendig ist.
- Autonom auf Organisationsebene durchgeführte Änderungen können bei mangelnder Berücksichtigung von Abhängigkeiten inter-organisationale Dienste massiv beeinträchtigen.

Im folgenden Abschnitt stellen wir als Lösung dieser Herausforderungen das Konzept einer ioCMDB vor, deren Datenmodellierung den Schwerpunkt dieses Artikels bildet.

1.2. Inter-organisationale CMDB

Abbildung 2 zeigt die Zusammenhänge des Managements von intra- und inter-organisationalen Diensten bzw. Ressourcen. Jede Organisation verwaltet ihre eigenen IT-Ressourcen autonom, wofür in den Organisationen i.A. schon ITSM-Werkzeuge im Einsatz sind. Die Organisationen verwalten auch ihre IT-Services autonom, wofür CMDBs eingesetzt werden können. Zur Unterstützung des ioITSM wird eine ioCMDB eingesetzt, in die, wie im nachfolgenden Abschnitt erläutert wird, auch relevante Informationen aus den jeweiligen CMDBs der Organisationen eingespeist werden.

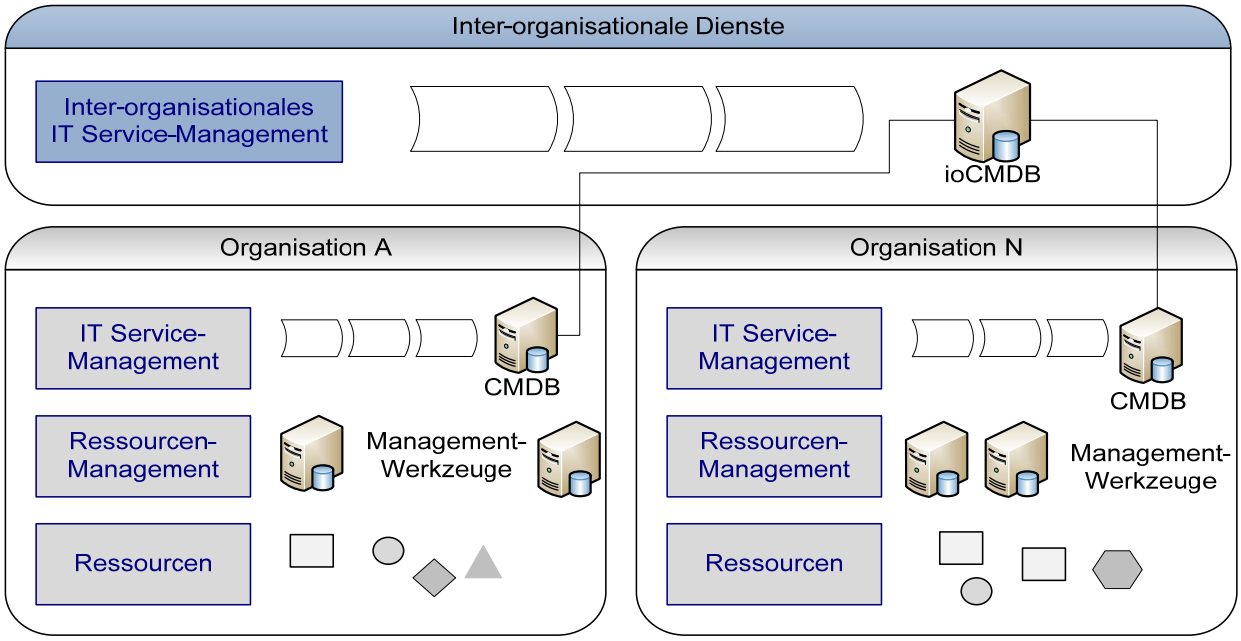

Abbildung 2 Zusammenspiel von intra- und inter-organisationalem IT Service Management

2. Datenmodellierung und Zusammenspiel mit lokalen CMDBs

Bereits bei organisationsinternen CMDB-Implementierungen ist die konkrete Modellierung der CIs und ihrer typisierten Beziehungen keine triviale Aufgabe. Meist wird ein objektorientierter Modellierungsansatz gewählt, der Klassen abzubildender realer Objekte und deren beschreibende Eigenschaften festlegt (vgl. [1], [8]). Dieser wird vorrangig aus Performanzgründen oft auf ein relationales Datenbankschema abgebildet. In der Praxis ist dabei die Entwicklung zu beobachten, dass ebenfalls aus Gründen der Geschwindigkeit analog zu Data Warehousing Verfahren zunehmend denormalisierte Schemata eingesetzt werden [9].

Für die Entwicklung einer ioCMDB ist jedoch insbesondere zu berücksichtigen, dass die bereits vorhandenen CMDBs zwar über standardisierte Schnittstellen zum Zugriff auf die Daten verfügen mögen, es aber kein einheitliches, standardisiertes Datenmodell gibt. Selbst falls entsprechende Standards verabschiedet werden und sich in der Praxis langfristig durchsetzen sollten, müsste jede organisationsübergreifende Lösung flexibel genug sein, um mit bereits vorhandenen, proprietären lokalen Datenmodellen umgehen zu können, um eine Integration zu ermöglichen.

In diesem Abschnitt betrachten wir vier zusammenhängende Aspekte: Zunächst erläutern wir szenarienunabhängig, welche Daten lokaler CMDBs für den Aufbau und Betrieb einer ioCMDB relevant sind. Dann diskutieren wir eine Herangehensweise für das Schema-Design, also die für jedes Szenario spezifische Modellierung der ioCMDB-Daten. Darauf aufbauend zeigen wir den Zusammenhang zwischen lokalen CMDB- und ioCMDB-Daten und schlagen entsprechende Synchronisationsprozesse vor, für die wir eine Komponentenarchitektur zur technischen Umsetzung vorstellen.

2.1. Übernahme relevanter CMDB-Einträge

Prinzipiell könnte der Datenbestand einer ioCMDB zur Unterstützung organisationsübergreifender ITSM-Prozesse mit genau allen dafür nötigen Informationen von Grund auf neu aufgebaut werden. Diese Methodik würde zwar selbst dann noch die Kompatibilität sicherstellen, wenn mindestens eine der beteiligten Organisationen keine lokale CMDB betreibt. Allerdings wäre es in der Praxis jedoch aus offensichtlichen Gründen ineffizient, bereits vorhandene, direkt zugängliche Informationen nicht zu nutzen und dafür in einem separaten System redundant zu akquirieren und zu pflegen.

Zunächst ohne Berücksichtigung der technischen Realisierung und der vom Datenmodell her möglichen syntaktischen Inkompatibilitäten muss deshalb analysiert werden, welche in einer lokalen CMDB vorhandenen Daten in eine ioCMDB übernommen werden sollen. Ausschlaggebend sind hierfür einerseits die Relevanz der Daten für die ioITSM-Prozesse und andererseits aus dem GRC-Management resultierende Sicherheitsaspekte, z.B. der Schutz vor unbefugten Zugriffen durch die anderen Organisationen. Einen Lösungsansatz für den zweiten Aspekt haben wir bereits in einer früheren Arbeit vorgestellt und betrachten den Zugriffsschutz in diesem Artikel als gelöst [5]. Aufgrund der Beschränkung auf relevante Daten und wegen möglicherweise aus Sicherheitsgründen nicht zugänglichen Informationen ist jedoch auf das Problem von Dateninkonsistenzen durch die nur partielle Datenübernahme zu achten, d.h. dass einzelne Attribute von Objekten oder ganze Objekte, auf die sich Relationen zwischen CIs beziehen, fehlen können. Die Vollständigkeit der in einer ioCMDB-Instanz vorliegenden Daten bezüglich der durch ioITSM-Prozesse vorgegebenen Use Cases muss deshalb unbedingt durch einen geeigneten Qualitätsmanagementprozess nachhaltig sichergestellt werden und kann durch Automatismen nur partiell IT-unterstützt werden.

Die Bestimmung der von einer CMDB in eine ioCMDB zu übernehmenden Daten ist szenarienspezifisch, da die CIs und deren Attribute sowie die verwendeten Relationen zwangsweise dienst-, organisations- und kooperationsspezifisch sind. Wir erläutern deshalb im Folgenden anhand einiger ausgewählter Beispiele und bewusst unvollständig, welche häufig gepflegten CIs, Attribute und Relationen zur Kandidatenmenge für die Aufnahme ins ioCMDB-Datenmodell gehören:

- Relevante CIs:
  - Incident Records ausgewählter Zeiträume, wie etwa den „letzten X Wochen". Diese Daten werden vom ioIncident-Management zur Bearbeitung von Anfragen zu aktuellen und jüngst zurückliegenden Vorfällen benötigt, sollen in der ioCMDB jedoch u.a. zur Minimierung von Betriebskosten nicht dauerhaft archiviert werden.
  - Ausgewählte Benutzerobjekte, insbesondere zur organisationsübergreifenden Bereitstellung von Kontaktdaten von Personen, die mit den ioITSM-Prozessen betraut sind oder als lokale ITSM- oder Serviceansprechpartner fungieren.
  - Change Records, speziell auch aus dem lokalen Forward Schedule of Changes (FSC nach ITIL), sofern die Veränderungen organisationsübergreifend relevante Dienste betreffen und beispielsweise zur Planung von Wartungszeiträumen relevant sind.
  - Alle Dienst-/Service- und Hardware-/Server-CIs, die organisationsübergreifend relevante Dienste und Subservices betreffen; hierbei sind neben statischen insbesondere auch dynamische Attribute relevant, die z.B. von Monitoring-Werkzeugen aktualisiert werden und z.B. über Verfügbarkeit und Auslastung informieren.
  - Dienstleistungsvereinbarungen (Service Level Agreements (SLAs)), soweit diese organisationsübergreifend relevante Dienste betreffen und z.B. zur Priorisierung von Incidents im Rahmen des ioIncident-Managements ausgewertet werden müssen.
- Relevante Attribute:
  - Lokale Ansprechpartner bei Service- und Hardware-CIs, sofern diese auch im Rahmen der ioITSM-Prozesse kontaktiert werden sollen.
  - Objekt-Identifikatoren zur eindeutigen Zuordung von korrespondierenden Objekten in der lokalen CMDB und der ioCMDB (sog. Object Association).
  - Zeitstempel der letzten Änderung zur technischen Unterstützung des unten erläuterten technischen Synchronisationsprozesses.
  - Änderungshistorie, die analog zu den o.g. Incident Records auf einen Erfassungszeitraum, der nur die jüngere Vergangenheit betrifft, beschränkt sein sollte.
- Relevante Beziehungen zwischen CIs:
  - Angaben zur Diensthierarchie (Services und deren Subservices bzw. Ressourcen) sowie ggf. weitere relevante Abhängigkeiten zwischen Diensten, um kausale Zusammenhänge z.B. bei Dienstausfällen identifizieren und kommunizieren zu können.
  - Angaben zur Dienstlokalisierung, z.B. „Dienst A läuft aktuell auf Maschine X", die insbesondere zur Fehlerdiagnose beim Einsatz virtueller Maschinen, die dynamisch zwischen verschiedenen Hosts migriert werden können, relevant sind.

Wie bei lokalen CMDBs gilt bei der Umsetzung die Daumenregel, dass eine kleine Menge an gut gepflegten Daten einer möglichst umfassenden Aggregation der vorhandenen Daten im Hinblick auf Nachhaltigkeit und Effizienz der Lösung als auch auf die Betriebskosten vorzuziehen ist.

2.2. Modellierung der ioCMDB CIs und Relationen

Durch die fehlende Datenmodell-Standardisierung bei den organisationsintern eingesetzten CMDBs kommt es bei einer organisationsübergreifenden Betrachtung in der Praxis zu unterschiedlichen CMDB-Schemata, die bezüglich der ausgewählten CIs, Attribute und Relationen zu einem

gemeinsamen ioCMDB-Schema konsolidiert werden müssen. Die dabei als Basis notwendigen Vereinheitlichungen, z.B. bei der Wahl von Attributnamen, die Synonyme und Homonyme eliminieren müssen, sowie die syntaktische Umformung, z.B. von Datumsformaten oder Telefonnummern, werden von einer Vielzahl etablierter Methoden der Datenbankintegration unterstützt und deshalb hier nicht näher betrachtet. Wir fokussieren vielmehr auf die zur Sicherstellung der Interoperabilität notwendigen Schritte und insbesondere zusätzlichen Datenfelder, die sich nur in der ioCMDB, nicht jedoch in den lokalen CMDBs finden. Das Zusammenlegen der Daten aus einer potentiell beliebig großen Menge an Quellen muss durch eine Anreicherung des Datenmodells unterstützt werden, z.B.:

- Zur Vermeidung von Kollisionen im Objektnamensraum (engl.: namespace clashes) muss die Organisationszugehörigkeit, beispielsweise durch ein eindeutiges Präfix pro beteiligter Einrichtung, explizit erfasst werden. Dies betrifft sowohl systeminterne Identifikatoren, z.B. bei zwei CMDBs, die zwei verschiedene Entitäten über denselben Attributwert „id=12345“ identifizieren, als auch Attributwerte, z.B. bei Ortsangaben wie „Gebäude B, Raum 12“. Zur Umsetzung haben sich URN-basierte Ansätze als geeignet erwiesen, z.B. „id=urn:iocmdb:orgA.example.com:12345“.
- Maßeinheiten sollten explizit angegeben werden, beispielsweise falls Strafzahlungen für SLA-Verletzungen in einer CMDB in Dollar und in einer anderen in Euro angegeben werden. Dadurch können Ungenauigkeiten, die sich durch ein wiederholtes Umwandeln in bzw. von einer einheitlichen Maßeinheit ergeben können, vermieden werden; diese sollten bei Bedarf von einer graphischen Managementoberfläche durchgeführt werden.

Darüber hinaus müssen die ioITSM-Prozesse gezielt mit Informationen unterstützt werden, die erst mit der Teilnahme an organisationsübergreifenden Szenarien bedeutsam werden und deshalb u.U. auch nicht implizit aus den Daten der lokalen CMDBs abgeleitet werden können:

- Für die ioITSM-spezifische Kommunikation und Datenweitergabe sind Ansprechpartner zu benennen, die ggf. von den für die organisationsinternen ITSM-Prozesse definierten Personen abweichen.
- Die von ioITSM-spezifischen Rollen wie dem inter-organisationalen Service Desk erfassten Daten müssen geeignet auf ioCMDB-CIs abgebildet werden können, z.B. auf ioIncident Records und ioChange Requests. Die Modellierung der Attribute dieser CIs orientiert sich an ihren organisationsinternen Entsprechungen.
- Alle in der ioCMDB gespeicherten Objekte benötigen Metadaten, die von denen ihrer Ursprungsobjekte separiert sind, beispielsweise den Identifikator des letzten Bearbeiters und den Zeitstempel der letzten Änderung.
- Relationen zwischen den CIs der ioCMDB können organisationsübergreifend sein, d.h. Beziehungen über die Grenzen einer Einrichtung hinaus können in der ioCMDB hinterlegt werden, wohingegen das notwendige Zielobjekt in einer lokalen CMDB fehlen würde.

Bereits hieraus ist ersichtlich, dass die Inbetriebnahme und Pflege einer ioCMDB nicht nur aus dem Zusammenschalten existierender Datenbanken besteht, sondern auch einen initialen und dauerhaften Verwaltungsaufwand für die Pflege dieser zusätzlichen Informationen bedeutet.

2.3. Prozesse zur Datensynchronisation zwischen ioCMDB und lokalen CMDBs

Die Daten in der ioCMDB müssen immer möglichst aktuell gehalten werden, um beispielsweise Fehldiagnosen im ioIncident-Management zu vermeiden, die durch veraltete Daten entstehen würden. Klassische Bulk-Import-Verfahren, die u.a. im Umfeld des Data Warehousing und z.T. im

Rahmen von Business Integration Prozessen eingesetzt werden und dort beispielsweise einmal pro Nacht den gesamten Datenbestand aus dem Quellsystem übernehmen, notwendige Transformationen vornehmen und ins Zielsystem einspeisen, reichen deshalb nicht aus. Nachfolgend skizzieren wir einerseits ein effizienteres Verfahren zum Datenabgleich und diskutieren andererseits die Frage, ob der Datenabgleich zwischen den CMDBs und der ioCMDB im jeweiligen Szenario unidirektional oder bidirektional gestaltet werden sollte; dies beeinflusst insbesondere die Implementierungskosten, wirkt sich im Betrieb aber z.B. auf die notwendigen Maßnahmen zur Sicherstellung einer hohen Datenqualität aus.

Der klassische ETL-Ansatz (Extract, Transform, Load, siehe [6]) steht vor der Schwierigkeit, dass immer der gesamte Datenbestand des Quellsystems nach relevanten Datensätzen, beispielsweise Änderungen seit dem letzten Synchronisationslauf, durchsucht werden muss; durch eine starke Erhöhung der Datenabgleichsfrequenz, beispielsweise alle fünf Minuten, wird zwar die Aktualität der Daten im Zielsystem verbessert, aber das Quellsystem einer unnötig hohen Last ausgesetzt. Insbesondere ist das Zeitintervall, in dem Datenabgleiche erfolgen, fixiert und passt sich nicht dem charakteristischen Nutzungsverhalten (z.B. tagsüber viele Änderungen, nachts nur wenige) an. Auch bei internen Abrechnungsmodellen, in deren Accountingfunktionen die Anzahl ausgeführter Datenbankanfragen eingeht, würden sich die Kosten mit der Datenaktualität vervielfachen.

Aus diesem Grund schlagen wir einen ereignisgesteuerten Synchronisationsmechnismus vor, der durch das *Einfügen*, *Modifizieren* bzw. *Löschen* ganzer Datensätze oder einzelner Attribute ausgelöst wird und die Änderung, sofern sie für das Zielsystem relevant ist, zeitnah an dieses weitergibt. Die technische Umsetzung kann z.B. mit Hilfe von sog. Triggern und Stored Procedures in relationalen und objektorientierten Datenbankmanagementsystemen erfolgen, deren Implementierungskosten sich im Allgemeinen nicht von externen, z.B. JDBC-basierten ETL-Lösungen unterscheiden und durch die unmittelbare Integration in das Datenbanksystem typischerweise eine höhere Performanz aufweisen.

Die technische Schnittstelle zwischen der lokalen CMDB und der ioCMDB besteht somit zunächst aus einem Mechanismus, der die entsprechenden Ereignisse zur Verfügung stellt; darauf baut ein parametrisierbares Filtersystem auf, das für das Zielsystem irrelevante Ereignisse verwirft. Anschließend müssen die Daten bei den Ereignisarten *Einfügen* und *Modifizieren* in das vom Zielsystem benötigte Format gewandelt werden und der Datenbestand im Zielsystem entsprechend aktualisiert werden. Dabei können Ausnahmebehandlungen notwendig werden, beispielsweise falls ein zu modifizierendes Objekt im Zielsystem noch nicht exisiert und somit neu angelegt statt nur verändert werden muss. Bei der Implementierung sind weitere Fehlersituationen, z.B. die temporäre Nichtverfügbarkeit des Zielsystems, zu berücksichtigen.

Die oben diskutierte Bereitstellung von Daten aus den lokalen CMDBs zur Nutzung durch ioCMDB-unterstützte Prozesse impliziert, dass der Abgleich zwischen CMDB und ioCMDB auf jeden Fall in dieser Datenflussrichtung realisiert werden sollte. Die Implementierung eines Datenrückflusses von der ioCMDB in die jeweilige lokale CMDB kann je nach Szenario durchaus mit Synergieeffekten verknüpft sein, die den entstehenden Zusatzaufwand rechtfertigen; im Folgenden werden einige Mehrwerte dieses Use Cases skizziert.

Bei den Datenrückflüssen ist prinzipiell zwischen den Nutzdaten, z.B. modifizierten Attributen von ioCMDB-CIs, und den Metadaten zu unterscheiden; zum Umfang der Metadaten zählen wir hier über den Dublin Core Standard [4], der z.B. Zeitstempel und Identifikator des letzten Bearbeiters umfasst, hinausgehend insbesondere auch Protokolldaten. Informationen bezüglich Benutzeraktivi-

täten anderer Einrichtungen auf ioCMDB-Daten aus lokalen CMDBs können zur Optimierung des ioCMDB-Datenbestands ausgewertet werden. Solche Aktivitäten sind z.B. Schreifzugriffe, Einzel-, Such- oder Batchoperationen und deren Kontext, z.B. im Rahmen einer ioIncident-Bearbeitung,. Außerdem bildet diese Informationen auch die Grundlage für die Integration in Auditierungs- und Risk-Management-Prozesse, die die ioCMDB nicht nur als isoliertes System betrachten.

Bei der Übernahme von Nutzdaten aus der ioCMDB in eine CMDB sind einerseits aktualisierte Attribute von denjenigen CIs zu berücksichtigen, die ursprünglich aus dieser CMDB übernommen wurden. Zum anderen können auch in der ioCMDB neu angelegte CIs relevant sein, beispielsweise ioIncident-Records, deren Impact-Analyse Auswirkungen auf CIs ermittelt hat, die genau einer der beteiligten Organisationen zugeordnet werden können. Die Übernahme dieser Daten hängt jedoch auch davon ab, ob und wie stark die bereits organisationsintern vorhandenen ITSM-Prozesse so angepasst werden können, dass sie auch vom ioCMDB-Datenbestand Gebrauch machen. Sofern die verwendeten ITSM-Werkzeuge direkt an diese zusätzliche Datenquelle angebunden werden können, sollte der Mehraufwand für eine redundante Datenspeicherung vermieden werden. In diesem Fall ist jedoch mindestens auf organisatorischer Ebene zu klären, wie die ggf. notwendige (Langzeit-) Archivierung des ioCMDB-Datenbestands gehandhabt wird, um den jeweils aktuellen ioCMDB-Datenbestand möglichst schlank halten zu können.

2.4. Technische Komponenten zur Implementierung der Datensynchronisation

Abbildung 3 zeigt ein einfaches Beispiel der für die Umsetzung des oben beschriebenen Synchronisationsprozesses benötigten Systemkomponenten. Jede Organisation hat eine lokale CMDB. Diese kann gegebenenfalls aus mehreren einzelnen Datenbanken oder den Repositories der vorhandenen Managementwerkzeuge bestehen, die intern zu einer sog. CMDB-Federation (CMDBf), wie sie vom CMDBf-Komitee spezifiziert wurde, zusammengeführt werden [2]. Die Kommunikation zwischen ITSM-Werkzeugen und der CMDB erfolgt typischerweise über JDBC oder Web Services und ist verallgemeinernd als Enterprise Service Bus (ESB) dargestellt. Nachrichten über Änderungen am CMDB-Datenbestand werden ebenfalls über den ESB an die neue, nachfolgend spezifizierte Komponente, die wir als ioCMDB-Konnektor bezeichnen, übermittelt. Falls die CMDB keine geeigneten Triggermechanismen unterstützt, um jeweils nur die Änderungen am Datenbestand zu melden, muss dem ioCMDB-Konnektor ein Wrapper vorgeschaltet werden, der Änderungen am CMDB-Datenbestand regelmäßig z.B. anhand von Objektzeitstempeln ermittelt und als Events an den ioCMDB-Konnektor weitergibt; wie oben erläutert wurde, sollte dies jedoch aus Lastgründen nur eine Fallback-Lösung darstellen.

Die Verarbeitung der Daten im ioCMDB-Konnektor findet in mehreren Stufen statt. Zunächst werden Änderungen mittels eines Filters auf ihre Relevanz überprüft. Der Filter arbeitet regelbasiert, wobei umfangreichere Regelsätze zu sog. Policies zusammengefasst werden, die wie Policies für andere Dienste in einem Enterprise Policy Repository hinterlegt sind; jede Filterregel entscheidet anhand der Metadaten (z.B. CI eines bestimmten Typs) und der Attributwerte (z.B. Serverhardware, die einem organisationsübergreifend relevanten Dienst zugeordnet ist), ob die Änderung ans Zielsystem propagiert werden soll. Sofern mehrere Regeln für dasselbe Quellobjekt zutreffen, muss das Verhalten im Konfliktfall, d.h. bei einander widersprechenden Regeln, definiert werden. Für die Abbildung der Regeln auf ein maschinenlesbares Format stehen diverse, meist XML-basierte Policysprachen zur Auswahl, die hier nicht näher diskutiert werden.

Im zweiten Schritt muss bestimmt werden, welches Objekt im Zielsystem von der aktuellen Operation betroffen ist. Dies umfasst einerseits die Transformation des lokalen Objektidentifikators

in denjenigen des Zielsystems, wofür wie oben erläutert z.B. Präfixe zum Einsatz kommen können; andererseits muss das Verhalten definiert werden, falls beispielsweise ein zu modifizierendes oder zu löschendes Objekt im Zielsystem nicht vorhanden ist. Auch bei diesen Konfigurationsparametern handelt es sich um Regelwerke, die im Policy Repository hinterlegt werden können.

Daran anschließend sind in einem dritten Schritt sowohl Attributnamen als auch -werte in das Format des Zielsystems umzuwandeln. Wir schlagen hierfür den Einsatz eines XSLT-Prozessors vor. XSLT ist einerseits ein Standard zur Transformation von XML-basierten Daten, so dass die Implementierungskosten sowohl für den Datenkonverter als auch die szenarien-spezifischen Transformationsregeln gering sind, da z.B. Open Source Komponenten wie Xalan [10] eingesetzt werden können; andererseits ermöglicht eine XML-basierte Ausgabe die dynamische Ergänzung zusätzlicher Attribute im Zielsystem, beispielsweise wie oben diskutiert zur expliziten Angabe von Maß- und Kosteneinheiten. Dadurch können aufwendige Schema-Erweiterungen der lokalen CMDBs einfach vermieden werden.

Schließlich werden die konvertierten Daten ins Zielsystem geschrieben bzw. ggf. die Löschoperation ausgeführt. Die Kommunikation erfolgt in unserem ioCMDB-Modell über Web Services; dabei muss konnektoren-seitig ein Puffer implementiert werden, der Änderungen bei temporärer Nichterreichbarkeit des Zielsystems zwischenspeichert. Exakt dieselben Schritte sind – mit entsprechend angepassten Parametern – auch bei Datenrückflüssen aus der ioCMDB in eine lokale CMDB zu durchlaufen. Ein Initialabgleich zwischen CMDB und ioCMDB kann angestoßen werden, indem für jedes Objekt im Quellsystem entweder ein Änderungsevent erzeugt oder eine triviale Änderung, z.B. ein Eintrag ins Historienattribut, durchgeführt wird. Die web-service-basierte Kommunikation mit der ioCMDB sollte prinzipiell verschlüsselt erfolgen, da sensible Daten i.d.R. via Internet übertragen werden, und kann durch zusätzliche Security-Standardmechanismen wie Firewalls dem Bedarf entsprechend abgesichert werden.

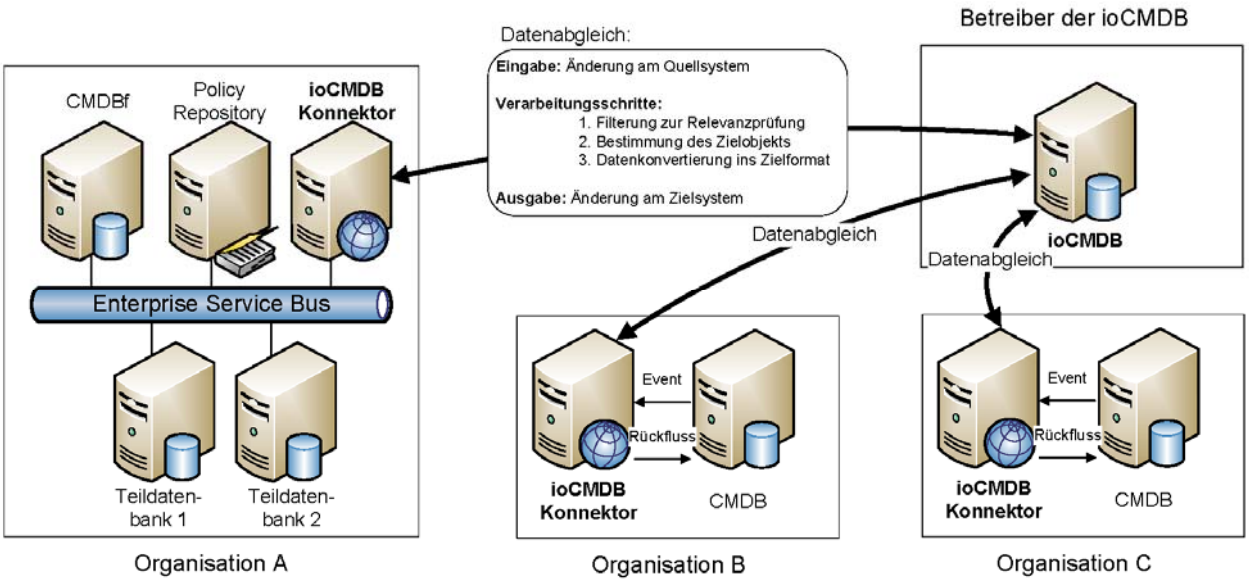

Abbildung 3 Einsatz der ioCMDB-Konnektoren in einem einfachen Szenario

3. Zusammenfassung und Ausblick

Dienste, die auf dem Zusammenwirken mehrerer Organisationen basieren, benötigen zur Umsetzung der komplexen Anforderungen des GRC-Managements eine gezielte Werkzeugunterstützung. In Anlehnung an die ITIL-basierte CMDB setzen wir hierzu eine inter-organisationale

CMDB (ioCMDB) ein. Wir haben ferner gezeigt, dass beim Deployment einer ioCMDB besonderes Augenmerk auf die Datenmodellierung zu richten ist, um für jedes Szenario eine kostengünstige und nachhaltige Lösung umzusetzen. Dabei ist explizit vorgesehen, dass die ioCMDB auch zusätzliche CIs aufnehmen kann, die in den lokalen CMDBs nicht vorhanden sind. Diesem Aspekt muss beim Schema-Design Rechnung getragen werden, wofür wir eine grundlegende Herangehensweise vorgestellt haben. Für den Austausch von Informationen zwischen CMDBs und ioCMDBs haben wir einen Prozess der Datensynchronisation vorgeschlagen, der Event-orientiert auf Basis von Web Services umgesetzt werden kann. Schließlich haben wir die notwendigen technischen Komponenten erläutert und dabei insbesondere den ioCMDB-Konnektor und seinen internen Workflow zur XSLT-basierten Verarbeitung der CI-Daten vertieft.

Im Rahmen einer formalen Kooperation mit einer der Organisationen, die führend am CMDBf-Komitee [2] beteiligt ist, streben wir an, unsere Konzepte zum organisationsübergreifenden CMDB-Einsatz in die aktuellen Standardisierungsbemühungen einfließen zu lassen. Parallel dazu wird an einer prototypischen Implementierung gearbeitet, mit der sich eine bestehende CMDB für den Einsatz als ioCMDB aufrüsten lässt.

Danksagung:
Die Autoren danken der IntegraTUM-Projektgruppe und dem Munich Network Management (MNM) Team für fruchtbare Diskussionen und Anregungen zu diesem Beitrag. IntegraTUM ist das Projekt der TUM zur Schaffung einer nahtlosen und benutzerfreundlichen IuKInfrastruktur, das von der DFG unter Vertragsnummer WGI 554 975 gefördert und vom Vizepräsidenten und CIO der TUM, Prof. Dr. Arndt Bode, geleitet wird. Das MNM-Team ist eine Forschungsgruppe mit Mitgliedern an der LMU, der TUM, der Universität der Bundeswehr München und dem Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften unter Leitung von Prof. Dr. Heinz-Gerd Hegering.

4. Literaturangaben

[1] BMC Software, Best Practices for Modeling Business Entities in the BMC Atrium CMDB, Verfügbar online unter: www.bmc.com/products/documents/20/52/62052/62052.pdf, 2008.

[2] CARLISLE, F. et. al. CMDB Federation (CMDBf) - Committee Draft. Technical report, BMC Software, CA, Fujitsu, Hewlett-Packard, IBM, Microsoft, Januar 2008. http://cmdbf.org/.

[3] FOSTER, I., et. al. The Open Grid Services Architecture, Version 1.5. Informational Document, Global Grid Forum (GGF), July 24, 2006.

[4] HILLMANN, D. Using Dublin Core. Verfügbar online unter http://dublincore.org/documents/usageguide/ . 2005

[5] HOMMEL, W. und KNITTL, S., An Inter–Organizational Configuration Management Database as Key Enabler for Future IT Service Management Processes , In *eChallenges 2008*, 2008, Stockholm, Schweden, Oktober, 2008

[6] KIMBALL, R. CAERTA, J. The Data Warehouse ETL Toolkit. Indianapolis, IN: Wiley. 2004

[7] MACFARLANE, Shirley. Service Transition, Itil, Version 3. Edinburgh: Stationery Office, 2007.

[8] $N(i)^2$. $N(i)^2$ - MULTI-DOMAIN CMDB SYSTEM – White Paper, Verfügbar online unter http://ni2.com/, 2008.

[9] SHIN, S. K. und SANDERS, G. L. 2006. Denormalization strategies for data retrieval from data warehouses. Decis. Support Syst. 42, 1 (Oct. 2006), 267-282.

[10] The Apache Xalan Project. Verfügbar online unter http://xalan.apache.org/. Zugriff am 20.07.2008.