INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Visualisierung von Managementinformationen im Cloudcomputing

Lisa Brodner

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Visualisierung von Managementinformationen im Cloudcomputing

Lisa Brodner

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Dipl.-Inform. Silvia Knittl Dr. David Schmitz Dr. Josef Homolka
Abgabetermin:	30. September 2011

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. September 2011

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Die IT-Infrastruktur der Technischen Universität München (TUM) wird ergänzt durch Cloud-Dienste, die von unterschiedlichen Organisationen angeboten werden. Inhalt einer vorhergegangenen Projektarbeit war die automatisierte Erfassung und Zusammenführung von Configuration-Management-Daten über bestehende Organisationsgrenzen hinweg. Ziel dieser Arbeit war es, den beteiligten Organisationen einen umfassenden Überblick zu verschaffen und das IT-Management in Planung und Fehlersuche zu unterstützen.

Als Weiterführung dieses Ansatzes wird in der vorliegenden Arbeit ein Konzept entworfen, wie aus den Inhalten dieser organisationsübergreifenden Configuration Management Database automatisiert Visualisierungen erstellt werden können. Die Visualisierungen heben die Zugehörigkeit der Daten zu verschiedenen Organisationen hervor und erleichtern das Erkunden des großen Datenbestandes. Das Konzept umfasst die Datenanalyse, die Auswahl der zu visualisierenden Daten, die Auswahl des Visualisierungswerkzeuges, die Transformation der Daten sowie spezifische Anpassungen der Visualisierungen.

Eine prototypische Umsetzung des Konzepts erfolgt anhand von Konfigurationsdaten des Leibniz-Rechenzentrums und des Physik-Departments der TUM, die einen Teil der Netzinfrastruktur der TUM abbilden. Das JavaScript InfoVis Toolkit dient hier als Visualisierungswerkzeug.

Abschließend werden ausgehend von der prototypischen Umsetzung Aussagen bezüglich der Anwendung und Erweiterbarkeit des Konzepts getroffen. Außerdem werden mögliche Weiterentwicklungen vorgeschlagen.

Abstract

The IT-Infrastructure of the Technische Universität München (TUM) includes various cloud services. Those services are provided by different organizations. Previous work was focused on automatic acquisition and consolidation of Configuration Management Data across these organizational borders. The goal was to simplify planning and locating of errors by providing an overview of consolidated data for the network operator, support, and IT management in general.

This paper continues this approach, presenting a concept to automatically visualize existing data sets of an inter-organizational Configuration Management Database. The visualizations emphasize the origin of the data and make it easier to navigate through large amounts of data. The concept includes analysis and selection of the data, selection of the visualization tool, transformation of the data and options to customize the result.

The feasibility of the concept is shown by implementing a prototype which uses configuration data of the Leibniz-Rechenzentrum and the Physics Department of the TUM. The data represents a part of the network infrastructure of the TUM. The JavaScript InfoVis Toolkit is used as visualization tool.

Finally, the concept is assessed by analyzing the prototypical implementation. The paper also discusses possible extensions and proposes future development steps.

Inhaltsverzeichnis

1 Einführung

In diesem Kapitel werden die Motivation für diese Arbeit und deren Aufgabenstellung dargestellt. Als Grundlage wird zunächst die IT-Infrastruktur des Münchner Hochschulumfeldes betrachtet. Dann werden Arbeiten vorgestellt, die mit der hier vorliegenden im Zusammenhang stehen. Dazu gehört eine Projektarbeit, deren Zusammenführung verschiedener Datensammlungen die Grundlage für den praktischen Teil der vorliegenden Arbeit liefert. Es folgt die Aufgabenstellung und ein Überblick über die Vorgehensweise.

1.1 Motivation

Um Infrastruktur jeder Art möglichst reibungslos zu nutzen und zu betreiben, muss diese hinreichend dokumentiert werden. Im hier betrachteten Fall handelt es sich um IT-Infrastruktur der Münchner Hochschulen, im Besonderen der TU München. Diese Infrastruktur wird nicht einheitlich selbst betrieben oder von einem einzigen Anbieter zur Verfügung gestellt und gewartet. Die TUM nutzt Cloud-Services, so dass sich die Infrastruktur aus Einzelbestandteilen zusammensetzt, die von unterschiedlichen Organisationseinheiten und Dienstleistern betrieben werden.

Wenn die Infrastruktur sich aus mehreren verantwortlichen Organisationen und aus vielen Komponenten zusammensetzt, ist es wünschenswert, auf eine zentrale Komponenten-Verwaltung zugreifen zu können. Diese bietet Zugang zur (theoretisch) gesamten Datenmenge, wobei mit steigender Größe die Übersichtlichkeit erschwert wird. Eine Visualisierung der Datenmenge erleichtert in solchen Fällen die Interaktion und Benutzbarkeit des Systems.

1.2 Münchner Hybrid-Cloud

Um IT-Dienste effizient anbieten und verwalten zu können, wird meist eine Configuration Management Database (CMDB) geführt, in die Configuration Items (CIs) eingetragen und über welche sie verwaltet werden. Bei CIs handelt es sich z.B. um Hardware, Software, Räume oder auch Personen - kurz: alle Komponenten, die nötig sind, um einen IT-Dienst erfolgreich leisten zu können [RH07].

Dieses 'Inventar' bildet die Grundlage für das Configuration Management (CM), die Verwaltung der CIs. Die hier zusammengetragenen Informationen werden z.B. verwendet um schneller Fehler beseitigen zu können. Die Meldung von Problemen und deren Behebung findet am Service-Desk statt. Hier ist der Kontaktpunkt zwischen dem Dienstleister und seinem Kunden und von ihm aus wird weiteres Vorgehen initiiert.

Das Rahmenwerk für diese Organisationsform des Managements bildet die Information Technology Infrastructure Library (ITIL), ein Katalog für Best Practices im IT-Service-Management [iti].

Das Leibniz-Rechenzentrum (LRZ) als IT-Dienstleister bietet Kunden die Möglichkeit IT-Dienste zu nutzen, ohne sie selbst von Grund auf bereitstellen und verwalten zu müssen.

Zu den Verantwortlichkeiten des LRZ gehört das Münchner Wissenschaftsnetz (MWN) mit allein mehr als 60 Standorten und über 75.000 angeschlossenen Endgeräten. Dieses Netz verbindet unter anderem die Münchener Hochschulen untereinander und mit dem weltweiten Internet [lrz, lrz08]. Es sind auch Standorte angebunden, die vor Ort selbstständig betrieben werden, so z.B. die Fakultät Informatik der TU München; andere Standorte sind verpflichtet einen Netzverantwortlichen zu stellen. Dieser dient als Ansprechpartner für das LRZ und übernimmt die lokale Verwaltung [lrz10b]. Damit ist der Betrieb des MWN verteilt.

Das Kernnetz wird vom LRZ verwaltet. Dies geschieht mit Hilfe eigener Infrastruktur, dokumentiert in der Netzdokumentation. Die Verantwortung für die lokalen Netze in den an das MWN angeschlossenen Organisationen liegt großteils vor Ort, wo jeweils eigene Managementdaten-Sammlungen für die Verwaltung entstanden sind. Damit kommen die Organisationen unter anderem der Verpflichtung der Dokumentation der eigenen angeschlossenen Netze nach [lrz10a].

Die TU München nutzt neben der eigenen IT-Infrastruktur die Dienstleistungen des LRZ, aber auch Dienste anderer Anbieter wie z.B. Wikispaces [wikc]. Damit ergibt sich für die gesammelte Infrastruktur der TUM die Form einer sogenannten Hybrid-Cloud, eine Mischung aus eigener, klassischer' IT-Infrastruktur und der Nutzung von Cloud-Computing [KL10, Web09]. Ein Ausschnitt dieser Struktur ist abgebildet in Abbildung 1.1.

"Cloud-Computing bezeichnet eine Form der bedarfsgerechten und flexiblen Nutzung von IT-Leistungen, die über das Internet bereit gestellt und nutzungsabhängig abgerechnet werden" [KL10] (nach [Web09, MG11]), was für die vom LRZ angebotenen und genutzten Dienste zutrifft. Die Datengrundlage dieser Arbeit bildet einen Teilbereich der Hybrid-Cloud der TUM ab. Es handelt sich dabei insbesondere um Netzkomponenten und abhängige Infrastruktur, die vom LRZ und dem Physik-Department der TU München betrieben werden.

1.3 Verwandte Arbeiten

Die folgenden Arbeiten sind entweder Vorarbeiten oder befassen sich auch mit dem Themengebiet der Visualisierung zur Unterstützung des IT-Managements.

1.3.1 Datenerfassung für organisationsübergreifende CMDB

Im Rahmen der Projektarbeit "Evaluation des TUM-Trouble Ticket Systems als Ersatz für bestehende lokale Konfigurationsmanagementdatenbanken" wurde von Wenzhe Xia eine CMDB entworfen, die auf dem Open Ticket Request System (OTRS [BBB+11]) aufsetzt. Diese vereint exemplarisch Konfigurationsdaten aus der LRZ-eigenen Netzdoku und der lokalen Managementdatensammlung des Physik-Departments der TU München in einer organisationsübergreifenden Datenbank [Xia10]. In der zugehörigen Implementierung besteht die Möglichkeit über das OTRS-Interface auf die einzelnen Komponenten und ihre Beziehungen zu anderen Komponenten zuzugreifen. Wie eine solche Darstellung in OTRS aussieht, zeigt Abbildung 1.2. Hierbei handelt es sich um eine Auflistung der verfügbaren Informationen zu einem gegebenen Item und der Abhängigkeiten zu anderen Items. Diese Darstellung ist textbasiert und nicht weiter visuell aufbereitet. Für das schnelle Einfinden in komplexere Installationen ist diese Darstellung nur bedingt geeignet.

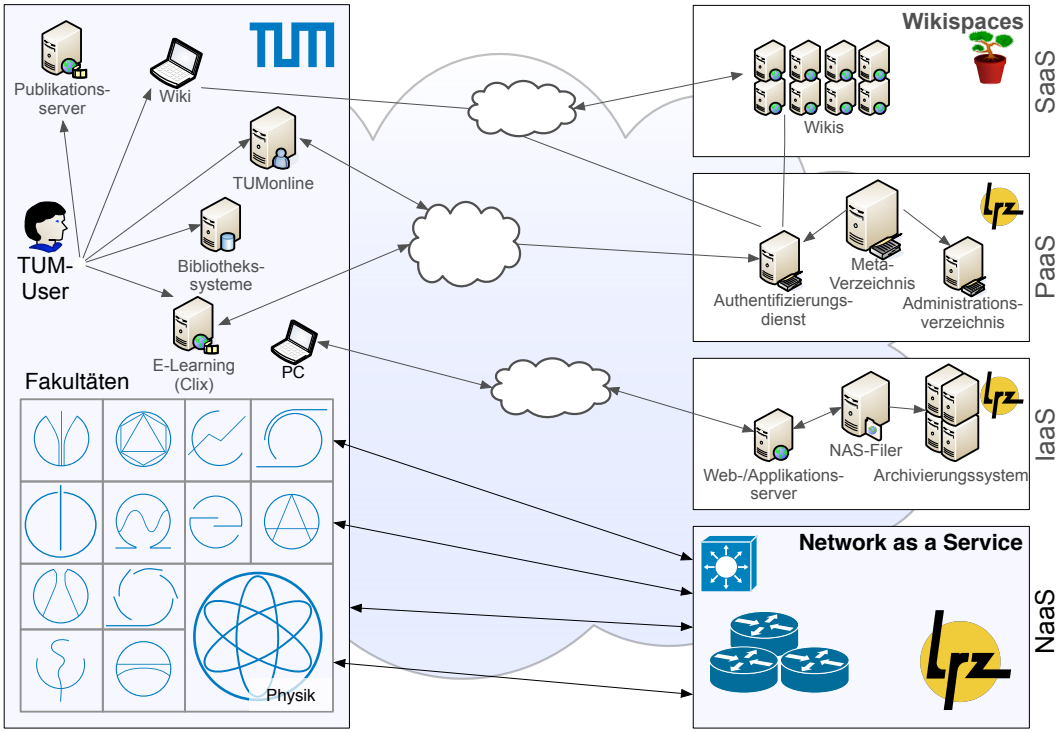

Abbildung 1.1: Hybrid-Cloud an der TUM [KL10], erweitert um Network as a Service (NaaS)

1.3.2 Visualisierung im IT-Management

Menschen, die im IT-Management beschäftigt sind, also Störungen nachgehen, Probleme beheben oder den Ausbau der Infrastruktur planen, müssen in der Lage sein, für sie relevante Daten in einer vorhandenen CMDB aufzufinden. Wenn der digitale Datenbestand dem Fragenden nicht so vertraut ist, wie z.B. den Netzverantwortlichen ihr eigenes Netz geläufig ist, fällt der Zugang schwer und das Ermitteln bestimmter Informationen wird behindert.

Daten-Visualisierung ist eine Möglichkeit, Zusammenhänge aufzuzeigen, Überblick zu verschaffen und damit einen Datenbestand leichter erfass- und nutzbar zu machen [CMS99]. Die so aufbereiteten Daten unterstützen beispielsweise die Arbeit am Help-Desk. Eine verbesserte Darstellung der vorhandenen Infrastruktur ist auch bei der Planung von Erweiterungen oder der Bewertung der vorhandenen Architektur hilfreich. Zudem lassen sich Auswirkungen von Veränderungen im aktuellen System so besser beurteilen. Dadurch wird die Vorbereitung einer möglichst störungsfreien Umsetzung von Anpassungen und von Ausbauten am aktuellen System erleichtert (Change Management).

Visualisierung hat konkret aus oben genannten Gründen beispielsweise in Form von Softwarekarten im Projekt “Modeling and Documentation of IntegraTUM’s System Landscape” als Teil der Dokumentation für das Infrastructure Management des Enterprise Architecture Management Anwendung gefunden [CDPT09]. Bei den hier angewendeten Pattern ist Visualisierung in Form von sogenannten Viewpoint-Patterns ein fester Bestandteil der Herangehensweise an die Dokumentation von Systemen [Ern09]. Die erstellten Karten wurden

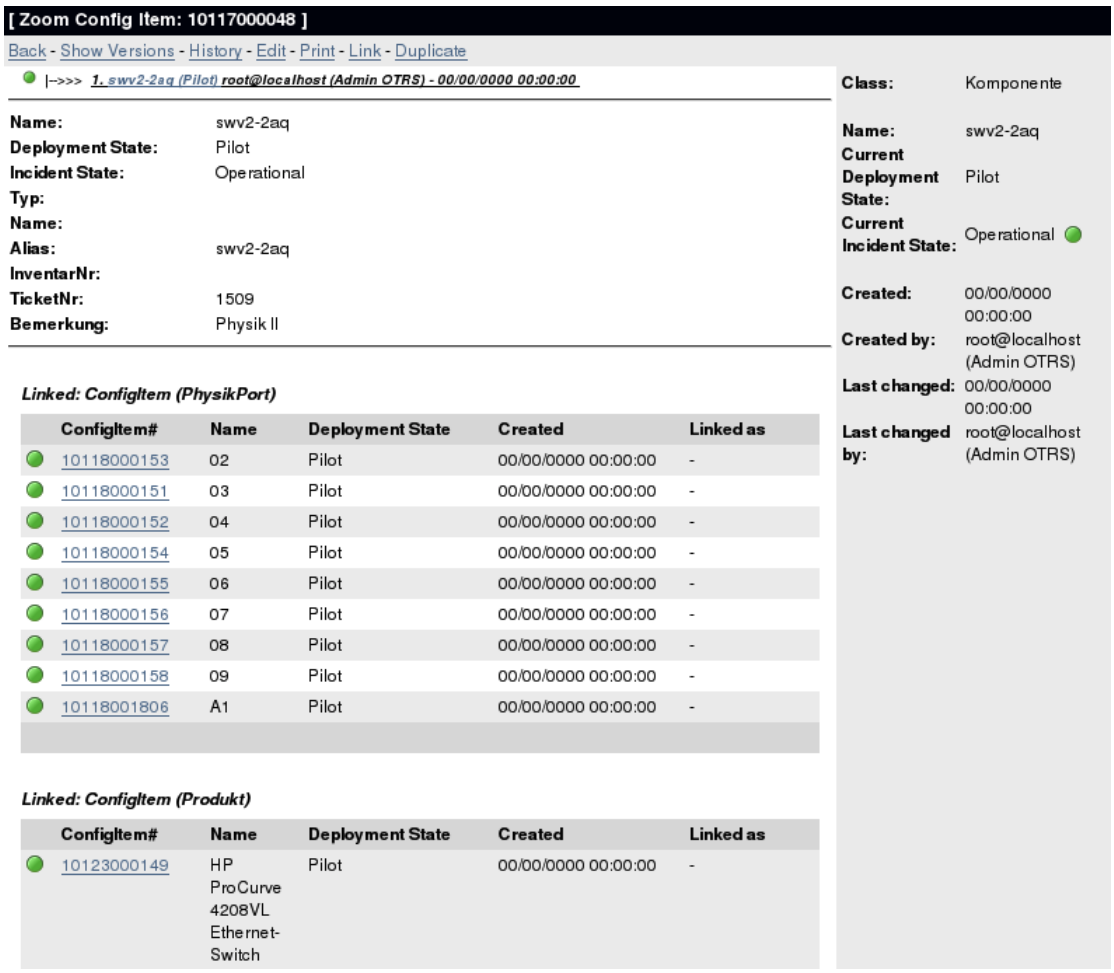

Abbildung 1.2: OTRS: Detailansicht eines Switches aus dem Datensatz von [Xia10]

nicht automatisiert aus dem Datenmodell, sondern mittels Visio von Hand erzeugt. Diese Visualisierungen sind nur über einen gewissen Zeitraum hinweg brauchbar, da die Daten veralten und das Nachpflegen von Hand aufwändig und fehleranfällig ist.

Die Arbeit "Analyse der Anwendbarkeit multimedialer Visualisierungs- und Interaktionstechniken im IT Service Management" beschäftigt sich mit Visualisierung im Zusammenhang mit IT-Management. Der Schwerpunkt ist dort anders gelagert, nämlich auf der Aufbereitung von Dokumenten und Verbesserung von Schulungsunterlagen, sowie des Prozessmanagements [Zei10].

Die vorliegende Arbeit kombiniert die Visualisierung von Daten mit einer automatisierten Aufbereitung und dem Schwerpunkt des Configuration Managements.

1.3.3 Datengrundlage

Die Daten, die von Whenze Xia während seiner Projektarbeit zusammengetragen wurden, bilden die Grundlage für eine praktische Durchführung des in dieser Arbeit entwickelten Konzepts. Hierbei handelt es sich um Auszüge aus der Netzdoku und der TUM-Physik-Datenbank von 2009, die auf die Datenbankstruktur des damals aktuellen OTRS angepasst

und in diese importiert wurden. Es finden sich darunter Informationen zu Gebäuden, Instituten, Komponenten, Personen und Produkten, außerdem Beziehungen zwischen diesen. Die abgebildeten Beziehungen werden in den Abschnitt 2.2.3 und 3.3 genauer beschrieben, die CI-Typen in Abschnitt 2.2.2.

OTRS ist ein Software-System mit dessen Hilfe die gesamte Bandbreite von Kundenbetreung über Verwaltung und Problembehebung anhand von Tickets organisiert werden kann. Diese Tickets können über die OTRS-Benutzeroberfläche in das System eingetragen werden. Ebenso können CIs zum System hinzugefügt und mit passenden Tickets verknüpft werden, wenn man die ITSM-Erweiterung für OTRS installiert. Über unterschiedliche Benutzeroberflächen für Agenten und Benutzer können die so erstellten Tickets und Items eingesehen und von Nutzern mit entsprechenden Rechten bearbeitet werden.

Die Datenhaltung für OTRS findet in einer MySQL-Datenbank statt [BBB+11, mys]. Auf diese kann auch direkt, d.h. ohne Umweg über die Benutzeroberfläche, zugegriffen werden. Für die Zusammenführung der unterschiedlichen CMDBs und für die Abbildung aller diesbezüglichen Informationen wurde das Datenbank-Schema den eigenen Ansprüchen entsprechend angepasst. Damit weicht die vorliegende Datenbank von der eines frisch installierten OTRS mit ITSM-Erweiterung ab. Es wurden eigene CI-Typ-Definitionen und Beziehungen hinzugefügt und das Schema einer Tabelle um zwei Felder erweitert. Diese werden in Abschnitt 2.2.4 beschrieben.

OTRS hat sich inzwischen zu einer Software-Plattform weiterentwickelt, auf der z.B. OTRS ITSM aufsetzt. Darin ist neben dem Ticket-System für den Service-Desk auch eine dedizierte CMDB enthalten [BBB+10, BBB+11]. Von einer Migration auf das aktuelle OTRS (Help Desk) und OTRS ITSM wurde abgesehen, da dann auch die durch [Xia10] erweiterten Datenbank-Felder sowie Veränderungen der Datenbankstruktur übertragen werden müssten. Für diese Arbeit werden stattdessen die durch [Xia10] gewonnenen Daten unverändert genutzt, ohne sie auf eine aktuelle Version zu migrieren oder neu zu importieren. Der Schwerpunkt der vorliegenden Arbeit liegt nicht auf der Gewinnung und Aufbereitung der Daten.

1.4 Aufgabenstellung

Die folgenden zwei Absätze sind Zitate aus der Ausschreibung.

Hintergrund und Motivation: Die Bereitstellung von Dienstleistungen erfolgt immer häufiger in Kooperation verschiedener Dienstleister. Ein neuer Trend hierfür ist das sogenannte Cloudcomputing.

Ziel der Arbeit: In dieser Arbeit soll das Management von kooperierenden Dienstleistern durch eine geeignete Visualisierungslösung unterstützt werden. Betrachtet wird diese Aufgabe konkret am Beispiel der Zusammenarbeit zwischen der Technischen Universität München und dem Leibniz-Rechenzentrum. Diese arbeiten zusammen, um Dienste für Studenten bereitzustellen, wie etwa Internetzugänge oder Dateispeicherdienste. Für das Management dieser Dienste ist an der TUM bereits ein Werkzeug im Einsatz, OTRS - Open Ticket Request System (www.otrs.org). Dieses beinhaltet Informationen über die Zusammensetzung der angebotenen Dienste. Leider liegen diese Informationen nicht in visueller Form vor. Aus diesem Grund ist es die Aufgabe dieser Bachelorarbeit das vorhandene Informationsmodell grafisch zu visualisieren. Zum Einsatz soll hierfür

ein Werkzeug kommen, das gemeinsam mit den Betreuern festgelegt wird (zur Auswahl stehen: http://www.deepamehta.de/ , http://www.w3.org/2001/11/IsaViz/) Ziel der Aufgabe ist es, eine geeignete Schnittstelle zwischen OTRS und diesem Tool zu definieren und prototypisch zu implementieren.

Im Laufe der Besprechungen zur Themenfestlegung und Werkzeugauswahl wurde die Aufgabe noch weiter erläutert und die Vorarbeit von Whenze Xia vorgestellt. Außerdem wurde ein anderes Werkzeug gewählt, das JavaScript InfoVis Toolkit (JIT). Dieses wird in Kapitel 2.3.1 vorgestellt.

Das Ziel dieser Arbeit besteht darin, ein Konzept zu erstellen, das beschreibt, welche Schritte durchlaufen werden müssen, um einen bestehenden Datenbestand automatisiert zu visualisieren. Zur Illustration wird dieses Konzept praktisch umgesetzt. Dabei wird eine Schnittstelle zwischen der vorhandenen OTRS-Datenbank und dem JIT konzipiert und prototypisch verwirklicht.

1.5 Vorgehensweise

Zunächst werden die Anforderungen an die angestrebten Visualisierungen gesammelt. Davon ausgehend wird ein Konzept für die weitere Vorgehensweise entwickelt, mit Hilfe derer man automatisiert Visualisierungen aus Management-Daten erstellen kann:

1. Analyse des Datenbestandes
2. Auswahl eines Visualisierungswerkzeuges
3. Erstellung von Visualisierungen

Bei der Untersuchung des Datenbestandes werden sowohl Struktur als auch der konkret vorhandene Inhalt betrachtet. Dabei findet auch eine Auswahl statt, welche Daten visualisiert werden.

Die Erstellung von Visualisierungen umfasst mehrere Schritte. Die Daten werden in ein Format überführt, das den Ansprüchen des Visualisierungswerkzeuges entspricht, und dann als Visualisierungen dargestellt. Die einzelnen Schritte sind dabei:

1. Abfrage der Daten
2. Erstellen von Datenstrukturen, die Zusammenhänge und Beziehungen abbilden
3. Hinzufügen von Darstellungsinformation
4. Definition des Interaktionsverhaltens der Visualisierung
5. Erstellung der Visualisierung

Die Struktur dieser Arbeit folgt diesem Vorgehen. Kapitel 2 stellt die Anforderungen an das Visualisierungswerkzeug vor, dann wird der Datenbestand analysiert und das gewählte Visualisierungswerkzeug vorgestellt. Kapitel 3 deckt die Erstellung von Visualisierungen ab. Die jeweiligen Ausführungen sind begleitet von praktischen Überlegungen, die sich auf die prototypische Umsetzung beziehen. Kapitel 4 fasst die Arbeit zusammen und nimmt eine Bewertung des Konzeptes vor. Außerdem werden Vorschläge für mögliche Weiterentwicklungen gemacht.

2 Datenanalyse und Werkzeugauswahl

Im Folgenden bezeichnet:

Visualisierung Eine konkrete Visualisierung, die einen bestimmten Sachverhalt abbildet. Sie ist durch einen eindeutigen Namen, z.B. ‘Ort’, gekennzeichnet. Über diesen Namen findet die Zuordnung zu bestimmten Komponenten, zu den abgebildeten Beziehungen, zur Visualisierungsart statt.

Visualisierungsart, Visualisierungstyp Eine Form der Visualisierung. Hier zumeist Formen, die vom JIT unterstützt werden.

Um Visualisierungen erstellen zu können, muss zunächst der Verwendungszweck dieser klar sein.

Es sollen für einen Management-Datenbestand automatisiert Visualisierungen erstellt werden. Diese Visualisierungen sollen das IT-Management unterstützen und insbesondere den Cloud-Charakter der Daten hervorheben, der sich dadurch ergibt, dass die Daten von unterschiedlichen Organisationen stammen und ein gemeinsam betriebenes Netz abbilden.

Um diese Anforderungen zu erfüllen, wurde ein Konzept entwickelt. Dieses lässt sich in zwei Phasen aufteilen:

Vorarbeit Arbeiten, die einmalig durchgeführt werden müssen.

Umsetzung Schritte, die für jede Visualisierung durchlaufen werden müssen.

Dieses Kapitel beschreibt die Phase ‘Vorarbeit’ (Abbildung 2.1).

Nachdem die Anforderungen an die Visualisierungen aufgestellt wurden und das angestrebte Ziel klar ist, werden im ersten Schritt die zu visualisierenden Daten auf ihre Eigenheiten hin untersucht. Dabei werden sowohl die konkreten Daten als auch die Zusammenhänge zwischen diesen betrachtet. Ausgehend von den vorhandenen Daten und den Anforderungen an die Visualisierung wird dann ein Werkzeug gewählt. Ob es den Anforderungen gerecht wird, ist Teil des Evaluierungsprozesses.

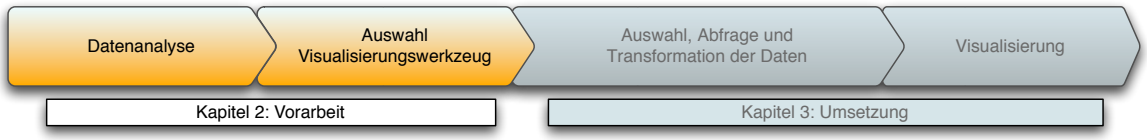

Abbildung 2.1: Konzept: Übersicht Kapitel 2

2.1 Anforderungen an ein Visualisierungswerkzeug

Die folgenden Punkte ergeben sich aus den Anforderungen an die gewünschten Visualisierungen und technischen Überlegungen. Diese wurden in Interviews mit den Betreuern erarbeitet und ausgewählt.

2.1.1 Möglichkeit der Interaktion

Die Visualisierung soll keine reine Grafik sein. Der Betrachter soll durch das Auswählen eines Teils der Visualisierung genauere Information erhalten können. Wie genau dies umgesetzt wird, ob z.B. über dynamisches Anzeigen von Detail-Information, den Verweis auf eine andere Ansicht oder das Öffnen eines neuen Fensters, ist nicht festgelegt. Auch die Rückkehr zu einer allgemeineren Ansicht soll einfach möglich sein.

2.1.2 Unterschiedliche Ansichten und Anpassungsfähigkeit

Es soll die Möglichkeit bestehen, die Darstellungsform anzupassen an die Kriterien, nach denen visualisiert werden soll, und an die Daten, die dargestellt werden sollen. Das Werkzeug soll also verschiedene Visualisierungstypen anbieten, besonders die Möglichkeit einer hierarchischen Ansicht und der Darstellung von Netzen. Die Darstellung von Assoziationen und einzelnen Komponenten soll individualisierbar sein z.B. durch Farbe, Beschriftung oder Form, so dass eine Anpassung der Visualisierung nach organisatorischen Merkmalen realisierbar ist. Es soll z.B. möglich sein, CIs nach Herkunft farblich zu markieren, also beispielsweise die der TUM blau und die des LRZ gelb, damit die Hybrid-Cloud Struktur leichter erkennbar ist.

2.1.3 Erforderliche Konvertierung

Die Überführung der bestehenden Daten in ein für das Werkzeug passendes Format soll mit geringem Aufwand geschehen. Nach dem initialen Aufsetzen soll die Konvertierung möglichst automatisiert ablaufen. Dafür muss betrachtet werden, welche Umwandlung und Aufbereitung der Daten nötig ist.

2.1.4 Systemvoraussetzungen

Die Voraussetzungen, um das Werkzeug nutzen zu können, sollen möglichst gering sein. Dafür kann in Betracht gezogen werden, ob das Werkzeug auf verschiedenen Plattformen zugänglich oder leicht anzupassen ist, welche Infrastruktur benötigt wird, welches Betriebssystem, welche speziellen Programme und zusätzlichen Installationen, um es nutzen zu können.

2.2 Datenanalyse

OTRS kann im Backend unterschiedliche Datenbanken verwenden. Bei der Datenbank aus der Vorarbeit handelt es sich um eine MySQL-Datenbank [BBB+11].

Mit Hilfe der durch [Xia10] gegebenen Dokumentation und der Datenbank selbst fand eine Einarbeitung in die Daten und ihre Struktur statt. Hierbei wurde festgestellt, dass die tatsächliche Datenbankstruktur Abweichungen zur dokumentierten Datenbankstruktur aufweist. Dabei handelt es sich z.B. um Abweichungen in der Tabellenbenennung. Die Tabelle

`link_object2`/`link_object` der Dokumentation entspricht beispielsweise in der Datenbank der Tabelle `link_relation`.

2.2.1 Tabellenstruktur

Tabelle 2.1 zeigt eine Liste aller vorhandenen Tabellen in der Datenbank `otrs`. Die dunkelgrau hinterlegten Zellen entsprechen den Tabellen, die für die Visualisierungen von Interesse sind. Hellgrau hinterlegte Zellen zeigen Tabellen an, die zu den Informationen für ein CI beitragen, aber nicht direkt in Visualisierungen eingehen. Bei den restlichen Tabellen handelt es sich um Daten für andere Features von OTRS, die hier aktuell keine Anwendung finden.

article	article_attachment	article_flag
article_plain	article_search	article_sender_type
article_type	auto_response	auto_response_type
cip_allocate	configitem	configitem_counter
configitem_definition	configitem_version	customer_company
customer_preferences	customer_user	follow_up_possible
general_catalog	generic_agent_jobs	group_customer_user
group_role	group_user	groups
imexport_format	imexport_mapping	imexport_mapping_format
imexport_mapping_object	imexport_object	imexport_search
imexport_template	link_object	link_relation
link_state	link_type	mail_account
notification_event	notification_event_item	notifications
package_repository	personal_queues	postmaster_filter
process_id	queue	queue_auto_response
queue_preferences	queue_standard_response	role_user
roles	salutation	search_profile
service	service_customer_user	service_preferences
service_sla	sessions	signature
sla	sla_preferences	standard_attachment
standard_response	standard_response_attachment	support_bench_test
system_address	theme	ticket
ticket_history	ticket_history_type	ticket_index
ticket_lock_index	ticket_lock_type	ticket_loop_protection
ticket_priority	ticket_state	ticket_state_type
ticket_type	ticket_watcher	time_accounting
user_preferences	users	valid
web_upload_cache	xml_storage	-

Tabelle 2.1: CMDB: Tabellen der OTRS-Datenbank

2.2.2 Configuration Item-Typen

In der Datenbank werden die in Tabelle 2.2 aufgelisteten CI-Typen abgebildet. In Klammern stehende Typen wurden angelegt, aber es befinden sich keine CIs dieser Klasse in der Datenbank. Bei allen genannten Typen handelt es sich um eigens für [Xia10] erstellte und nicht um standardmäßig in OTRS enthaltene CI-Typen.

ClassID	Klassenname	Bezeichnet
109	(Account)	Account
110	(Organisztion)	Organisation
111	(Gelaende)	Gelände
112	Person	Person
113	Institut	Institut
114	Arbeitsgruppe	Arbeitsgruppe
115	Gebauede	Gebäude
116	Raum	Raum
117	Komponente	Komponente
118	PhysikPort	Physikalischer Port
119	LogikPort	Logischer Port
120	VLAN	VLAN
121	SubNetz	Subnetz
122	SubsubNetz	Teilnetz eines Subnetzes
123	Produkt	Produkt
124	(Netzdienst)	Netzdienst
125	(Dokumentation)	Dokumentation

Tabelle 2.2: CMDB: In der Datenbank abgebildete CI-Typen

2.2.3 Beziehungen zwischen Configuration Items

Abbildung 2.2 und Tabelle 2.3 zeigen die Beziehungen zwischen den unterschiedlichen CI-Typen. Diese Beziehungen bilden die Grundlage für die Visualisierungen, da sich so die Zusammenhänge zwischen unterschiedlichen CIs ergeben. Die Pfeile geben die Richtung der Beziehung an. Der Beginn des Pfeils gibt das Quell-Item an, die Pfeilspitze endet am Ziel-Item. Alle abgebildeten Beziehungen haben eine eindeutige Richtung. Es gibt für alle Beziehungen auch die inversen Entsprechungen, diese sind in der Datenbank nicht zusätzlich abgebildet. Ein Beispiel für eine Beziehung ist 'RaumKomponente', wobei ein Raum das Quell-Item ist, und eine Komponente das Ziel-Item. Diese Beziehung bildet die Tatsache ab, dass ein Raum eine bestimmte Komponente beherbergt. Die inverse Beziehung wäre 'KomponenteRaum', die abbildet, dass eine Komponente sich in einem bestimmten Raum befindet. Die in Klammern stehende Beziehung 'PatchPort' ist in der Datenbank als mögliche Beziehung angelegt, wird aber nie verwendet.

Wie man der Tabelle 2.3 entnehmen kann, gibt es vereinzelt mehrere Beziehungen, die den Zusammenhang zwischen denselben CI-Typen abbilden, z.B. verbinden 'RaumKomponente' und 'RaumPatch' beide 'Raum' und 'Komponente'. Dabei handelt es sich nur um Beziehun-

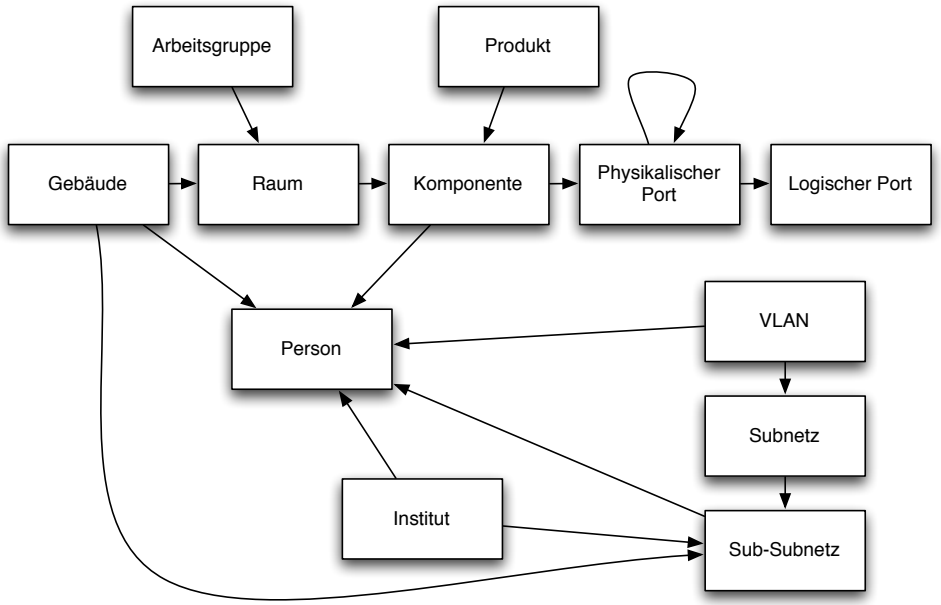

Abbildung 2.2: CMDB: Beziehungen zwischen CI-Typen: Übersicht

ID	Bezeichnung	Verbindet	
8	InstitutPerson	Institut	Person
12	ProduktKomponente	Produkt	Komponente
13	SubnetzToPart	Subnetz	SubsubNetz
14	PhysikPortToLoggikPort	PhysikPort	LogikPort
15	GebaeudePerson	Gebauede	Person
16	KomponentePerson	Komponente	Person
17	InstitutSubnetzpart	Institut	SubsubNetz
18	GebaeudeSubnetzpart	Gebaeude	SubsubNetz
19	SubnetzpartPerson	SubsubNetz	Person
20	VLANPerson	VLAN	Person
21	VLANSubnetz	VALN	SubNetz
22	GebaeudeRaum	Gebaeude	Raum
23	ArbeitsgruppeRaum	Arbeitsgruppe	Raum
9	RaumKomponente	Raum	Komponente
24	RaumPatch		
27	RaumDose		
10	KomponentePort	Komponente	PhysikPort
25	(PatchPort)		
26	SwitchPort		
11	PortPort	PhysikPort	PhysikPort

Tabelle 2.3: CMDB: In der Datenbank abgebildete CI-Verbindungen

gen, die Komponenten betreffen. Hier findet eine Unterscheidung zwischen unterschiedlichen Komponenten-Typen nicht über spezielle CI-Typen, sondern über die jeweils verknüpfende Beziehung statt. Dies erschwert das Erkennen der Komponenten und die Zuordnung von Beziehungen zu bestimmten CI-Typen. Man könnte dieses Problem beheben, indem man z.B. statt unterschiedlichen Beziehungen weitere Komponenten-Typen einführt.

Nach der Aufgabenbeschreibung in [Xia10] wurde davon ausgegangen, dass eine Verknüpfung der Daten von TUM-Physik und LRZ stattgefunden hat. Diese Verknüpfung wäre geschehen anhand von Port-zu-Port-Verbindungen zwischen Switches und Patches. Diese Daten sind jedoch nicht in der Datenbank enthalten. Auch eine Verschmelzung von CIs, die in beiden Datenbanken vorhanden sind, fand nicht statt, d.h. es entspricht z.B. sowohl das CI mit der ID `10117002413` (Herkunft:Physik-Datenbank) als auch das mit der ID `10117000048` (Herkunft:Netzdoku) dem realen Switch 'swv1-2aq'. An einem solchen Fall ist ersichtlich, dass man aus der Herkunftsdatenbank keinen Rückschluss auf die zuständige Organisation treffen kann. Eine weitere Betrachtung der verschiedenen Beziehungen erfolgt in Abschnitt 3.3.

2.2.4 Abweichungen von Standard-OTRS mit ITSM

Um den Import aus mehreren CMDBs in der Ziel-Datenbank abzubilden, wurde diese in [Xia10] erweitert. In der Tabelle `configitem` wurden zwei Felder hinzugefügt:

- `datasource` - Herkunftsdatenbank dieses CIs, ist NETZDOKU oder PHYSIK
- `fremd_id` - ID dieses CIs in seiner Herkunftsdatenbank

Diese Felder gehören nicht zum Standard-Tabellen-Schema von OTRS und finden dementsprechend auch in der OTRS-Benutzeroberfläche keine Anzeige. Für OTRS selbst ist die Tatsache, dass es sich um eine aus mehreren CMDBs vereinigte Datenbank handelt, unsichtbar. Es lässt sich darauf nur über die direkte Interaktion mit der Datenbank zugreifen. Bei diesen Zusatzinformationen handelt es sich um einen wichtigen Bestandteil der geplanten Visualisierungen, da z.B. die Information, zu welcher Organisation oder welchem Zuständigkeitsbereich ein CI gehört, gekennzeichnet sein soll.

Bei der Einarbeitung in die Daten hat sich gezeigt, dass nur bei einem Teil der CIs eine `datasource` eingetragen ist (bei 16591 von 24127 bzw. 68% der CIs fehlt diese Angabe). Da alle Items eine `fremd_id` haben, ist es aber möglich, über das Schema der Fremd-ID Rückschlüsse auf die Herkunft der Daten zu ziehen und diesen Mangel zu beheben.

Tabelle 2.4 listet nach Klassen geordnet auf, ob eine Datasource vorhanden ist oder nicht. Falls es CIs der entsprechenden Klasse gibt, bei denen keine Datasource eingetragen ist, wird ein Beispiel für die Form der Fremd-ID gegeben und die Herkunftsdatenbank angegeben, der man diese ID-Form oder diesen CI-Typ zuordnen kann. Diese Zuweisung geschieht aufgrund von Beispielen aus [Xia10] und den dortigen Ausführungen über das Mapping der CI-Typen aus Netzdoku und der Physik-Datenbank auf die CI-Typen in der Ziel-CMDB.

Bei dieser Auflistung sind 16 CI-Einträge ausgenommen, bei denen es sich um Test-Einträge der unterschiedlichen Klassen handelt. Diese enthalten weder `datasource` noch `fremd_id` und werden als nicht zu eigentlichen Datenbestand gehörig betrachtet.

Bei den CIs, deren `fremd_id` fehlt, wurde dieses Feld beim Import der Daten aus den Ursprungsdatenbanken nicht gesetzt. Es handelt sich dabei um ein Versäumnis der Import-

ClassID	Klasse	Datasoure	Beispiel Fremd-ID	Herkunft
112	Person	nicht vorhanden	PE_276	NETZDOKU
113	Institut	nicht vorhanden	IN_S3P0	NETZDOKU
114	Arbeitsgruppe	vorhanden	-	-
115	Gebaeude	teilweise vorhanden	GB_C	NETZDOKU
116	Raum	teilweise vorhanden	PHY-RAUM-3012	PHYSIK
117	Komponente	nicht vorhanden	PHY-DOSE-3333-01	PHYSIK
			KO_532	NETZDOKU
118	PhysikPort	vorhanden	-	-
119	LogikPort	nicht vorhanden	LP6942	NETZDOKU
120	VLAN	nicht vorhanden	VL112	NETZDOKU
121	SubNetz	nicht vorhanden	SN_365	NETZDOKU
122	SubsubNetz	nicht vorhanden	SS_36	NETZDOKU
123	Prodoukt	nicht vorhanden	PR_13	NETZDOKU

Tabelle 2.4: CMDB: Zuordnung zu einer Herkunftsdatenbank bei fehlender Datasource

Skripte. Die meisten CI-Typen, deren `datasource` fehlt, sind CIs, die sich der Netzdoku zuordnen lassen.

Es wäre möglich mit Hilfe der in Tabelle 2.4 gegebenen Information die Tabelle `configitem` von Hand auszubessern. Das geschieht hier nicht, um Fälle zu berücksichtigen, in denen nur lesender Zugriff auf die Datenbank besteht. Statt den ursprünglichen Datenbestand zu verändern, wird ein weiterer Zwischenschritt in die Datenumwandlung - von Datenbank zu Visualisierung - eingefügt, in dem die fehlenden `datasource`-Einträge bei Bedarf gesetzt werden.

2.2.5 Detailinformationen zu einem Configuration Item

Die Tabelle 2.5 zeigt eine Auswahl der Daten, die für ein CI abgespeichert sind. Für ein bestimmtes CI können in den folgenden Datenbanktabellen jeweils mehrere Einträge vorhanden sein:

- `configitem_version` speichert vergangene und aktuelle Versionen
- `link_relation` speichert direkte Verbindungen mit anderen Items
- `xml_storage` speichert klassenspezifische Informationen

Die mit * versehenen `configitem.id` verweisen vermutlich auf `configitem_version.id`. Das kann alleine anhand der Datenbank und diesen Datensatz nicht überprüft werden.

In der Datenbank ist auch gespeichert, wann ein Objekt angelegt und zuletzt verändert wurde und von welchem Anwender. Diese Information ist bei den vorliegenden Daten allerdings wenig aussagekräftig, da der gesamte Import von einem einzelnen Nutzer erstellt wurde und Erstellungs- und Veränderungsdaten nicht mit bedeutungsvollen Werten gesetzt wurden. Aus diesem Grund werden diese Informationen im Folgenden nicht verwendet. Ebenso nicht benutzt wird die Information aus der Tabelle `link_object`. Diese beschreibt den Typ der

Tabelle	Feld	Beispiel	Verweist auf	Bedeutung
configitem	id	4776	-	DB-CI-ID
	configitem_number	10117000048	-	OTRS-ID
	class_id	117	general_catalog.id	Typ des CIs
	last_version_id	4776	configitem_version.id	Aktuelle Version des CIs
	datasource	(NETZDOKU)	-	Herkunftssystem
	fremd_id	KO_222	-	ID des CI im Herkunftssystem
	cur_depl_state_id	40 (entspricht operational)	general_catalog.id	Aktueller DeploymetState
	cur_inci_state_id	11 (entspricht Pilot)	general_catalog.id	Aktueller IncidentState
configitem_version	id	4776	-	Versions-ID
	configitem_id	4776	configitem.id	Zugehöriges CI (ID)
	name	swv2-2aq	-	Beschreibung des CIs
	depl_state_id	40	general_catalog.id	DeploymetState der Version
	inci_state_id	11	general_catalog.id	IncidentState der Version
general_catalog	id	117	-	General-Catalog-ID
	general_catalog_class	ITSM::ConfigItem::Class	-	Klassentyp des Catalog-Items
	name	Komponente	-	Klassenname des Catalog-Items
link_relation	source_key	21478 (‘HP ProCurve 4208VL Ethernet-Switch’)	configitem.id*	Quell-CI-ID
	target_key	4776	configitem.id*	Ziel-CI-ID
	type_id	12	link_type.id	Beziehungstyp
	link_id	11456-222	-	Eindeutige Bezeichnung der Beziehung
	state_id	1 (entspricht valid)	link_state.id	Status der Beziehung
link_type	id	12	-	Link-Type-ID
	name	ProduktKomponente	-	Name des Link-Types
xml_storage	xml_key	4776	configitem.id*	Zugehöriges CI (ID)
	xml_content_key	[...]‘TicketNr’[...]	-	Key, zu dem Information gespeichert wurde
	xml_content_value	1509	-	Wert

Tabelle 2.5: CMDB: Beispiel: Daten eines Configuration Items - Auszug

Items, zwischen denen eine Beziehung besteht; er ist für alle Items gleich, 'ITSMConfigItem', und kann deswegen weggelassen werden.

2.3 Auswahl eines Visualisierungswerkzeuges

Abhängig von den Anforderungen, die an die Visualisierungen gestellt werden und dem Zweck, den sie erfüllen sollen, wird ein Visualisierungswerkzeug gewählt.

Durch die Datenanalyse ergibt sich, dass keine Verknüpfungen zwischen den CIs der verschiedenen Organisationen in der Datenbank abgebildet sind. Daher beschränkt sich die praktische Umsetzung auf hierarchische Daten, ohne Schwerpunkt auf Vernetzung zwischen Organisationen. Die Anforderungen an das Visualisierungswerkzeug berücksichtigen dennoch die zugrundeliegende Annahme, dass es sich um einen verknüpften Datenbestand handelt, um einen Einsatz auch bei aufgebesserten Daten zu ermöglichen.

2.3.1 JavaScript InfoVis Toolkit

Für diese Arbeit wurde das JavaScript InfoVis Toolkit [GB11] als Visualisierungswerkzeug gewählt.

Das JIT ist ein Framework, das die Visualisierung von Daten mit Hilfe der folgenden Technologien ermöglicht:

JavaScript Skriptsprache, die es ermöglicht Webseiten zu manipulieren und dynamisch clientseitig zu erstellen [ecm11, wika]

JSON JavaScript Object Notation - Datenaustauschformat, das ursprünglich aus JavaScript/ECMAScript entstanden ist [jso, Cro06]

HTML Hypertext Markup Language - Sprache zu Erstellung von strukturierten Dokumenten, zumeist für die Anzeige im Web [w3cg, w3cc]

DOM Document Object Model - Schnittstelle für Programmiersprachen, um den Inhalt von Dokumenten, z.B. HTML-Seiten, manipulieren zu können [w3ce, w3cf]

SVG Scalable Vector Graphics - skriptbare Sprache für die Erstellung zweidimensionaler Grafik [w3ch, w3cd]

Canvas Teil der HTML5 Spezifikation, bietet Bitmap-Canvas für das dynamische Zeichnen und Anzeigen von zweidimensionaler Grafik [w3ca]

CSS Cascading Style Sheets - Beschreibungssprache für das Layout und die Gestaltung von strukturierten Dokumenten [w3cb, w3cc]

Visualisierungen können mit dem JIT interaktiv gestaltet und individuell angepasst werden. Das Toolkit bietet unterschiedliche Visualisierungsarten [GB11] an. Beispiele dafür kann man in Abbildung 2.3 sehen [thea, theb]. Zu jeder dieser Visualisierungsarten gibt es Klassen, deren Objekte in einem ausgewiesenen Teil einer Webseite eine Visualisierung rendern können. Abhängig von der gewählten Visualisierungsart unterscheiden sich das Datenformat und die Parameter, die eingestellt werden können. Bei jeder Visualisierung können sowohl bei als auch nach der Erstellung alle Eigenschaften justiert werden. Eine solche Eigenschaft ist z.B.

bei Visualisierungen von Bäumen, wie viele Ebenen des Baumes angezeigt werden oder wo sich die Baumwurzel befindet. Es lässt sich einstellen, ob Interaktion mit der Visualisierung möglich ist. Außerdem kann man Event-Handler übergeben, die festlegen, was passiert, wenn z.B. ein Knoten mit der Maus angeklickt wird.

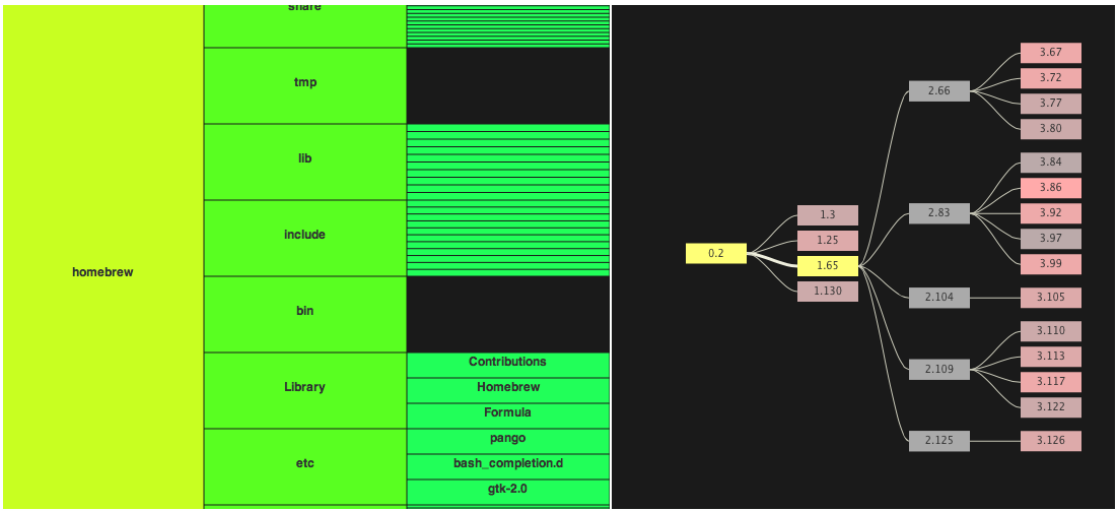

Abbildung 2.3: JIT: Demo-Visualisierungen: Icicle (links) und SpaceTree (rechts) [GB11]

2.3.2 Eingabe-Datenformat

Um durch das JIT dargestellt werden zu können, müssen die zu visualisierenden Daten als speziell strukturiertes JSON vorliegen [GB10]. Die Abbildungen 2.4, 2.5 und 2.6 zeigen die JSON-Varianten. Mit diesen lassen sich sowohl hierarchische Daten als auch Netze abbilden. Der Unterschied zwischen den Formaten in Abbildung 2.5 und 2.6 besteht darin, dass bei letzterem zusätzliche Informationen über die Verknüpfung gespeichert werden können. Für jede dieser Varianten gibt es mehrere Visualisierungsarten. Man kann also dieselben Daten an verschiedene Objekte übergeben, so dass diese unterschiedlich dargestellt werden.

Die praktische Umsetzung beschränkt sich auf Ausschnitte aus dem Datensatz, die sich als Bäume darstellen lassen und auf eine der Visualisierungsarten des JITs, nämlich die Icicle-Visalisierung. Diese Visualisierung bietet zum einen einen guten Überblick über die Baumstruktur der Daten, zum anderen ist die Zugehörigkeit zu einem bestimmten Item durch die räumliche Gestaltung klar gegeben. Der längste zusammenhängende, zyklenfreie Ausschnitt aus den im Beispieldatensatz abgebildeten Beziehungen bezeichnet ‘ist Teil von’ und ‘befindet sich in’-Beziehungen, die damit klar abgebildet werden.

```
{
    "id" : "aUniqueIdentifier",
    "name" : "usually a nodes name",
    "data" : {
        "some key" : "some value",
        "some other key" : "some other value"
    },
    "children" : [
        *other nodes or empty*
    ]
}
```

Abbildung 2.4: JIT: JSON-Datenformat für einen Baum [GB10]

```
[
    {
        "id" : "aUniqueIdentifier",
        "name" : "usually a nodes name",
        "data" : {
            "some key" : "some value",
            "some other key" : "some other value"
        },
        "adjacencies" : ["anotherUniqueIdentifier",
                         "yetAnotherUniqueIdentifier",
                        *etc*]
    },
    *other nodes go here*
]
```

Abbildung 2.5: JIT: JSON-Datenformat für einen einfachen Graph [GB10]

```
[
    {
        "id" : "aUniqueIdentifier",
        "name" : "usually a nodes name",
        "data" : {
            "some key" : "some value",
            "some other key" : "some other value"
        },
        "adjacencies" : [
            {
                "nodeTo" : "aNodeId",
                "data": { *put whatever you want here* }
            },
            *other adjacencies go here*
        ]
    },
    *other nodes go here*
]
```

Abbildung 2.6: JIT: JSON-Datenformat für einen Graph [GB10]

3 Erstellung von Visualisierungen

Dieses Kapitel beschreibt die Phase ‘Umsetzung’ (Abbildung 3.1).

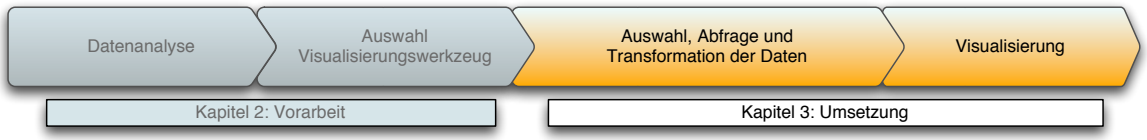

Abbildung 3.1: Konzept: Übersicht Kapitel 3

Die Daten werden abgefragt, den Anforderungen des Visualisierungswerkzeuges entsprechend aufbereitet und dargestellt.

Die allgemeinen Überlegungen zum Datensatz und wie dieser Datensatz z.B. als Baum abgebildet werden kann, gelten unabhängig vom gewählten Werkzeug. Die Transformation der Daten hin zu einem bestimmen Zielformat muss jeweils speziell an das Werkzeug angepasst sein. Außerdem wird durch die Wahl des Visualisierungwerkzeuges bestimmt, welche Möglichkeiten es gibt, die Visualisierungen zu verändern.

In dieser Arbeit wurde das JIT als Visualisierungwerkzeug gewählt. Um die vorhandenen Daten JIT-gerecht auszugeben, werden, neben den oben bereits genannten, zusätzlich folgende Technologien verwendet:

SQL Structured Query Language - Anfragesprache für relationale Datenbanken, hier eingebetet in PHP verwendet [BMW05, wikb]

PHP PHP: Hypertext Preprocessor - Serverseitig ausgeführte Skriptsprache, hier verwendet für MySQL-Abfragen und die Weiterverarbeitung der abgefragten Daten zu JSON [phpa, phpb]

jQuery JavaScript-Bibliothek, hier verwendet für eine vereinfachte Handhabung von HTML-Dokumenten und HTTP-Requests [jqu]

Die Umsetzung folgt den Schritten aus Abbildung 3.2, konkret bedeutet das:

1. Die Daten zu einem bestimmten CI werden dynamisch aus der Datenbank abgefragt; dies geschieht mit der in PHP integrierten MySQL-Unterstützung.

2. In PHP-Skripten werden Array-Strukturen erstellt, die dem Ziel-JSON entsprechen. Diese enthalten die angeforderten Informationen über das CI. Dabei handelt es sich zum einen über die Detail-Informationen zu diesem Item, zum anderen um verschiedene Beziehungsgraphen des Items. Die so erstellten Arrays werden mit der ab PHP 5.2.0 verfügbaren Funktion `json_encode` JSON notiert ausgegeben.

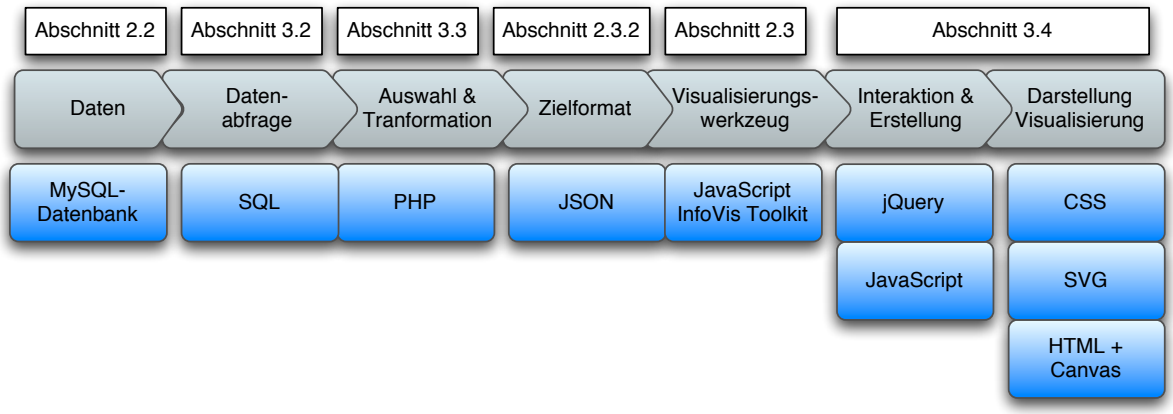

Abbildung 3.2: Konzept: Von der Datengrundlage zur Visualisierung

3. Einem Visualisierungs-Skript, das den JIT nutzt, wird dieses JSON übergeben und es stellt in einem ausgezeichneten Bereich einer HTML-Seite den entsprechenden Graph dar.

Wenn man die Visualisierungen für ein bestimmtes Item anfordert, werden im Hintergrund folgende Aspekte abgefragt:

- Allgemeine Informationen wie Bezeichnung, CI-Typ, Herkunftsdatenbank
- Klassenspezifische Informationen wie Hersteller oder E-Mail-Adresse
- Direkte Beziehungen zu anderen CIs
- Verfügbare Ansichtsarten auf dieses Item
- Graph-Daten für dieses Item pro Ansichtsart

Die erfragten Daten werden dann auf der Webseite dargestellt. Wie diese Abfragen im Einzelnen geschehen und auf welchen Annahmen sie aufsetzen, wird in den folgenden Abschnitten beschrieben.

Zur Veranschaulichung werden in diesem Kapitel die einzelnen Schritte an einer Komponente und deren Beziehungen gezeigt. Dabei handelt es sich um den bereits in Abbildung 1.2 und Tabelle 2.5 verwendeten Switch ‘swv1-2aq’.

3.1 Technische Vorbereitung

Für eine lauffähige OTRS-Installation werden im Grunde folgende Dinge vorausgesetzt [BBB+11]:

- Webserver
- Datenbank
- Angepasste Perl-Umgebung

Nachdem mehrere Nutzer auf denselben Datenbestand zugreifen können müssen, um OTRS sinnvoll in einer Mehr-Personen-Umgebung nutzen zu können, muss diese Installation zumindest in einem lokalen Netzwerk zentral verfügbar sein, um auf die Webapplikation als Nutzer zugreifen zu können.

Um den praktischen Teil dieser Arbeit erstellen und testen zu können, braucht man keine lauffähige Version von OTRS, da diese Umsetzung nicht umfassend in OTRS integriert ist, sondern nur auf demselben Datenbestand aufsetzt. Die Umwandlung und Neudarstellung geschieht also nicht zwischen OTRS und dem zughörigen Datenbestand, sondern vielmehr zwischen dem Datenbestand und dem Browser direkt.

Da es sich aber bei OTRS um eine Webapplikation handelt, die über den Browser benutzt wird und eine Server-Infrastruktur voraussetzt, sind die technischen Voraussetzungen für Visualisierungen im Browser bereits geschaffen.

Die Voraussetzungen, die erfüllt werden mussten, um ohne konkrete OTRS-Installation die Visualisierung programmieren und darstellen zu können, werden nachfolgend vorgestellt.

3.1.1 Export der bestehenden Datenbank aus der OTRS-Installation

Die Daten werden mit den MySQL eigenen Boardmitteln - `mysqldump` - in eine Textdatei exportiert. In dieser Datei sind Datensätze mit fehlerhafter Zeichenkodierung vorhanden. Hierbei handelt es sich nicht um einen neu aufgetretenen Fehler beim Export aus der CMDB, sondern um Altlasten aus vorhergegangenen Export- oder Importvorgängen. Das ergibt sich aus der Tatsache, dass diese Fehler schon in den XML-Dateien enthalten sind, die aus den Herkunftsdatenbanken exportiert wurden. Es handelt sich um Fehl-Kodierungen von Umlauten und Eszett. Diese Datensätze sind in der CMDB fehler- und verlustbehaftet abgespeichert, so dass es nicht möglich ist, sie bei dieser Datenbewegung einfach per Skript zu korrigieren. Abbildung 3.3 zeigt einen Abschnitt aus einer XML-Datei, die beim Export aus der Netzdoku entstanden ist.

3.1.2 Aufsetzen einer funktionsfähigen, lokalen Entwicklungsumgebung

Auf der Entwicklungsmaschine wird als lokale Webserver-Lösung XAMPP 1.7.3 aufgesetzt, da es sich hierbei um eine alleine lauffähige Version handelt ohne Abhängigkeiten zu bereits bestehenden Server-Installationen wie dem standardmäßig vorhandenen Apache bei Mac OS X [xam].

Der zuvor erstellte MySQL-Dump wird in die lokale Datenbank importiert und ist darüber vollständig abruf- und anzeigbar. Außerdem wird, um das Testen zu erleichtern, das Verzeichnis mit den verarbeitenden Skripten in der Apache-Konfigurationsdatei auf eine kurze

```
<CIInstitut>
    <InstitutIntgID>IN_A1F0</InstitutIntgID>
    <InstitutName>Kommission zur vergleichenden ArchÃ¯Â¿Â½ologie
            rÃ¯Â¿Â½mischer Alpen- und DonaulÃ¯Â¿Â½nder</InstitutName>
    <Kennzeichen>A1F0</Kennzeichen>
    <Chef>Herr Thomas Kraus</Chef>
    <Anschrift>Alfons-Goppel-Str. 11, 80539 MÃ¯Â¿Â½nchen</Anschrift>
    <Telephone>+49-89-23031-1274</Telephone>
    <Address>Alfons-Goppel-Str. 11, 80539 MÃ¯Â¿Â½nchen</Address>
    <OrganizationName>Bayerische Akademie
            der Wissenschaften</OrganizationName>
</CIInstitut>
```

Abbildung 3.3: Ausschnitt aus einer Netzdoku-Export-Datei mit Enkodierungsfehlern

URL abgebildet - http://localhost/otrsvis/. In dem entsprechenden Verzeichnis werden alle Skripte, Stylesheets und HTML-Dateien erstellt.

Die so umgezogenen Daten sollen nun abgefragt, umgewandelt und dargestellt werden.

3.2 Datenbankanfragen

Die Interaktion mit der Datenbank geschieht über PHP. Mit der integrierten MySQL-Unterstützung können direkt SQL-Anfragen an die Datenbank gestellt werden, deren Ergebnisse in Form von Arrays zurückgegeben werden. Die in diesen Arrays enthaltenen Informationen können weiterverarbeitet werden und JIT-gerecht abgebildet werden.

Alle Informationen zu einem bestimmten CI können über die zu diesem Item gehörige ID abgefragt werden. Da eine Integration mit OTRS angestrebt wird, wird als nach aussen sichtbare ID die OTRS-eigene ID verwendet. Diese entspricht nicht der Datenbank-ID. Eine Zuordnung zwischen diesen IDs geschieht über die folgende SQL-Query:

```
SELECT id FROM configitem WHERE configitem_number = '$otrsID'.
```

Für `$otrsID` wird die OTRS-ID des Items eingesetzt.

Um ein CI umfassend darzustellen, werden unterschiedliche Informationen abgefragt. In den nächsten Anschnitten folgt, um welche es sich dabei im Einzelnen handelt.

Die jeweils genannten SQL-Queries entsprechen nicht immer ganz genau den im Quelltext verwendeten, sie wurden teilweise gekürzt oder der Übersichtlichkeit halber zusammengefasst.

3.2.1 Abfrage allgemeiner Details

Es gibt Daten, die zu jedem CI vorliegen, z.B. Bezeichnung, Typ und Herkunft des Items. Diese sind über mehrere Tabellen verteilt (vgl. Tabelle 2.5) und müssen zusammengetragen werden. Die folgende Query fragt den Großteil der vorhandenen Informationen zu einem Item ab:

```
SELECT configitem_number, datasource, fremd_id, configitem_version.name,
        class_id, general_catalog.name, incident.name, deployment.name
FROM configitem_version, configitem, general_catalog,
        general_catalog as incident, general_catalog as deployment
WHERE configitem.id = '$id'
 AND last_version_id = configitem_version.id
 AND class_id = general_catalog.id
 AND cur_depl_state_id = deployment.id
 AND cur_inci_state_id = incident.id
```

Für `$id` wird die Datenbank-ID des Items eingesetzt.

Diese Daten werden in Tabellenform als Zusatzinformation zu den Visualisierungen angezeigt. Um ein Item als JIT-Node darzustellen, wird nur ein Teil dieser Information verwendet. Dafür wird eine verkürzte Version dieser Query benutzt.

3.2.2 Abfrage klassenspezifischer Details

Für jeden CI-Typ ist eine CI-Typ-Definition hinterlegt, die festlegt welche Merkmale zu einem gewissen Typ abgespeichert werden können [BBB+10]. Jede so definierte Detailinformation wird zu einem Item als XML-Hash in der Tabelle `xml_storage` abgespeichert [otr]. Dieser Hash ist für alle Items ähnlich, da alle verwendeten Typ-Definitionen aus gleich tief verschachtelten Arrays bestehen. Damit ist es möglich die Informationen ohne weiteres Parsen der Definitionen oder der XML-Hashes abzufragen und dann menschenlesbar abzubilden:

```
SELECT xml_content_key, xml_content_value
FROM xml_storage
WHERE xml_key = '$id'
 AND xml_content_key LIKE "[1]{'Version'}[1]{'%'}[1]{'Content'}"
```

Für `$id` wird die Datenbank-ID des Items eingesetzt.

Um z.B. aus `[1]{'Version'}[1]{'TicketNr'}[1]{'Content'}` den Merkmal-Schlüssel `TicketNr` zu machen, wird in der abfragenden PHP-Funktion ein regulärer Ausdruck

```
#\[1\]{'Version'}\[1\]{'(.*)'}\[1\]{'Content'}#
```

verwendet. Diese Herangehensweise geht von Typ-Definitionen dieser speziellen Struktur aus und ist damit in der Lage, beliebige Typ-Merkmale abzubilden. Sobald eine solche Definition komplexer würde oder die Benennung der Merkmale weniger menschenlesbar, müsste die Abfrage der typ-spezifischen Details angepasst werden.

3.2.3 Abfrage direkter Beziehungen mit anderen CIs

Die direkten Verbindungen mit anderen Items lassen sich über die Tabelle `link_relation` abfragen. Dort sind pro Verbindung die IDs von Ziel- und Quell-Item angeben sowie die Art der Beziehung und ob sie z.B. nur eine vorübergehende Verbindung ist. Damit ist es möglich über die ID des Items alle Beziehungen abzufragen, von denen das Item direkt betroffen ist:

```
SELECT source_key, target_key, configitem_number, type_id,
        link_type.name, link_id, link_state.name
FROM link_relation, link_type, link_state, configitem
WHERE (source_key = '$id' OR target_key = '$id')
 AND link_type.id = type_id
 AND state_id = link_state.id
 AND target_key = configitem.id
```

Für `$id` wird die Datenbank-ID des Items eingesetzt.

Diese Anfrage lässt sich auch spezialisiert stellen, indem nach einer bestimmten Beziehung ausgewählt wird.

3.3 Abbildung von Item-Beziehungen

Um einen Überblick über das Umfeld eines bestimmten CIs zu gewinnen, ist der erste Schritt, das direkte Umfeld abzubilden. Dieses ergibt sich durch unmittelbare Beziehungen zu anderen Items. Die Abbildung 3.4 bietet einen Überblick der in der Datenbank abgebildeten Beziehungen. Sie entspricht der Abbildung 2.2, die CIs sind allerdings grob nach Zusammengehörigkeit von oben nach unten angeordnet.

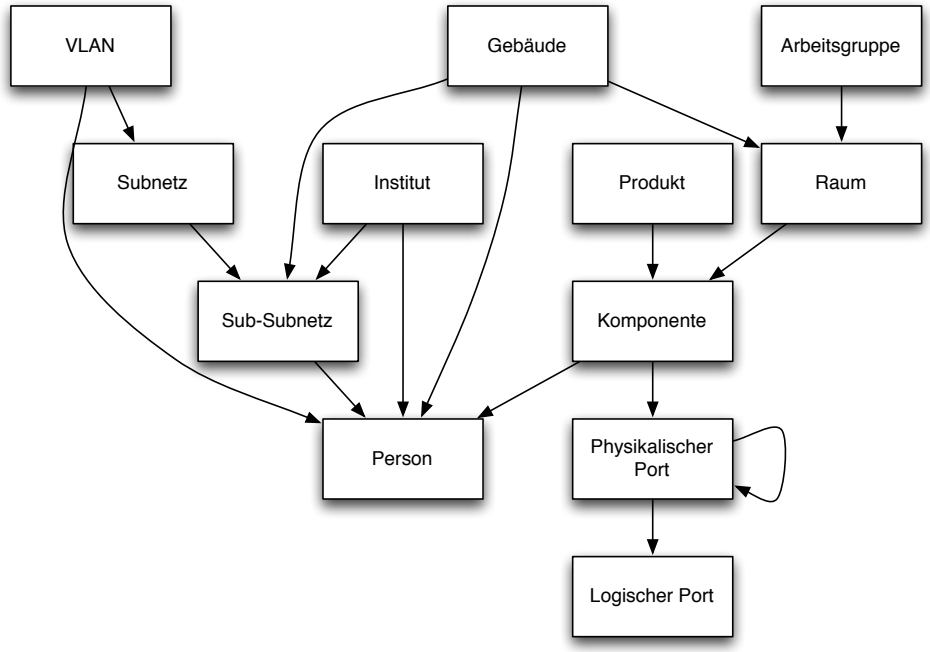

Abbildung 3.4: CMDB: Abgebildete CIs und Beziehungen, hierarchisch dargestellt

3.3.1 Direkte Item-Beziehungen

Da es sich bei den abgebildeten Beziehungen immer um gerichtete Beziehungen handelt, kann man ein Item als Eltern- oder Quell-Item bezeichnen, das Gegenstück als Kind- oder

Ziel-Item. Die Abbildung in der Datenbank erfolgt gerichtet, man kann aber auch, wie in Kapitel 2.2.3 beschrieben, die jeweils umgekehrte Beziehung annehmen.

Im Folgenden werden alle vorhandenen Items und ihre direkten Verknüpfungen zu anderen Items aufgezählt. Für alle direkten Beziehungen gilt, dass man zu einem beliebigen Quell-Item alle durch eine bestimmte Beziehung verbundenen Items auflisten oder als sehr kleinen Baum darstellen kann.

Person Eltern: VLAN, Subsubnetz, Institut, Gebauede, Komponente; Kinder: -;

Institut Eltern: -; Kinder: SubsubNetz, Person;

Arbeitsgruppe Eltern: -; Kinder: Raum;

Gebauede Eltern: -; Kinder: Raum, Person, SubsubNetz;

Raum Eltern: Arbeitsgruppe, Gebaeude; Kinder: Komponente;

Komponente Eltern: Raum, Produkt; Kinder: PhysikPort, Person;

PhysikPort Eltern: Komponente; Kinder: LogikPort; Geschwister: PhysikPort;

LogikPort Eltern: PhysikPort; Kinder: -;

VLAN Eltern: -; Kinder: Subnetz, Person;

Subnetz Eltern: VLAN; Kinder: SubsubNetz, Person;

SubsubNetz Eltern: Subnetz, Gebaeude, Institut; Kinder: Person;

Produkt Eltern: -; Kinder: Komponente;

3.3.2 Zyklenfreie Ausschnitte aus dem Beziehungsgraph

Ausschnitte aus dem Graphen in 3.4 lassen sich hierarchisch darstellen. Die Umsetzung, die hier prototypisch erfolgt, beschränkt sich auf solche Ausschnitte, die sich als Bäume abbilden lassen. Diese umfassen einen Großteil des Datensatzes und dienen damit als gutes Beispiel. Die Beschränkung auf diesen Teilaspekt geschieht, um die Darlegung des Vorgehens übersichtlicher gestalten zu können.

Es ist auch möglich, wie in Kapitel 2.3.1 besprochen, komplexere Beziehungen mit Hilfe des JIT abzubilden. Auch der Datensatz enthält komplexere Beziehungen, aber gerade die Beziehungen, die die Verbindungsstellen zwischen den Organisationen aufzeigen würden, bedürfen einer Nachbesserung im Datensatz.

Logisch zusammenhängende Teile des Beziehungsgraphen werden als einzelne Visualisierungen herausgegriffen. Je nach Augenmerk einer Visualisierung entstehen unterschiedliche Ansichten auf dieselben Daten oder unterschiedliche Ausschnitte aus den Daten eines einzelnen Items.

Eine solche Visualisierung erhält einen eindeutigen Namen und wird einem Array zugeordnet, das den entsprechenden Ausschnitt abbildet. Das geschieht, indem alle beinhalteten CI-Typen dem Beziehungstyp zugeordnet werden, dem man folgen muss, um die Kind-Knoten zu erhalten. Dieses Vorgehen funktioniert nur, da die Beziehungen speziell auf die jeweiligen Items zugeschnitten sind.

Es gibt z.B. für den Switch ‘swv1-2aq’ die Visualisierung ‘Ort’. Abbildung 3.5 zeigt einen kleinen Ausschnitt aus dem dabei erstellten Baum. Die Visualisierung ‘Ort’ bildet Items ab, die in einer ‘ist Teil von’- oder ‘befindet sich in’-Beziehung zueinander stehen.

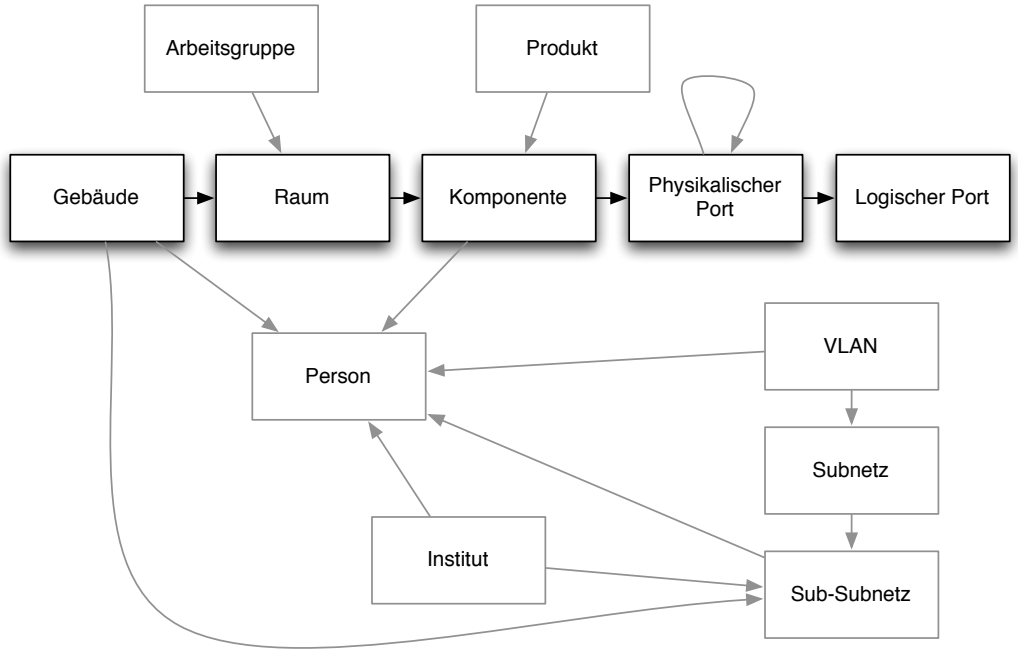

Abbildung 3.5: Beispiel: Beziehungsstrang, der für die Visualisierung ‘Ort’ abgebildet wird

3.3.3 Erstellung eines Beziehungsbaumes

Ausgehend von dem Item, zu dem Visualisierungen angefordert werden, wird dessen ältester Vorfahre ermittelt. Von diesem Item als Wurzel des Baumes ausgehend werden rekursiv jedem Knoten seine Kind-Knoten als `children`-Array hinzugefügt, bis man in den Blättern des Baumes angekommen ist. Die Kind-Knoten werden dem abgebildeten Beziehungsstrang entsprechend ausgewählt. Es gibt zu jeder Visualisierung zwei Arrays, die die Klasse des Items auf eine Folgebeziehung abbilden. Diese Abbildung geschieht in beide Richtungen, d. h., es besteht die Möglichkeit, Kindbeziehungen und Elternbeziehungen abzufragen. So lassen sich hierarchische Beziehungen mit eindeutigen Eltern abfragen.

Jeder Knoten wird abgebildet als ein entsprechend gegliedertes Array analog zu Abbildung 2.4. Abhängig vom Schwerpunkt der Visualisierung lassen sich in das `data`-Array eines jeden Knotens beliebige Zusatzinformationen speichern. Der gesamte Baum ist damit ein über mehrere Ebenen geschachteltes Array. Dieses Array entspricht dem gewählten Teilbaum aus dem Beziehungsgraph für dieses Item. Abbildung 3.6 zeigt einen Ausschnitt aus dem ‘Ort’-Array von ‘swv2-2aq’. Abbildung 3.7 zeigt denselben Ausschnitt in JSON. Nach Generierung der Arrays werden diese an das JIT übergeben.

Die so in JSON abgebildete Baumstruktur bildet die Daten-Grundlage für mehrere der unterschiedlichen Visualisierungstypen des JIT. Man kann also die so abgebildeten Daten unterschiedlich darstellen lassen.

```
Array
(
    [id] => 10115000029
    [name] => TUM, Geb. 5107, Physik II
    [data] => Array (
            [class_name] => Gebaeude
            [datasource] => NETZDOKU
            [$color] => #FED900
    )
    [children] => Array (
            [0] => Array (
                    [id] => 10116000017
                    [name] => TUM, Geb. 5107, Physik II EG
                    [data] => Array (
                            [class_name] => Raum
                            [datasource] => NETZDOKU
                            [$color] => #FED900
                    )
                    [children] => Array (
                            [0] => Array (
                                    [id] => 10117000074
                                    [name] => KAE_KAQ_1
                                    (...)
                            )
                            (...)
                    )
            )
            (...)
    )
)
```

Abbildung 3.6: Beispiel: Ausschnitt aus PHP-Daten für ‘Ort’-Visualisierung von ‘swv2-2aq’

```
{
    "id":"10115000029",
    "name":"TUM, Geb. 5107, Physik II",
    "data":{
        "class_name":"Gebaeude",
        "datasource":"NETZDOKU",
        "$color":"#FED900"
    },
    "children":[
        {
            "id":"10116000017",
            "name":"TUM, Geb. 5107, Physik II EG",
            "data":{
                "class_name":"Raum",
                "datasource":"NETZDOKU",
                "$color":"#FED900"},
            "children":[
                {
                    "id":"10117000074",
                    "name":"KAE_KAQ_1",
                    (...)
                },
                (...)
            ]
        },
        (...)
    ]
}
```

Abbildung 3.7: Beispiel: Ausschnitt aus JSON-Daten für ‘Ort’-Visualisierung von ‘swv2-2aq’

3.4 Anpassung der Visualisierungen

Visualisierungen sollen abhängig von ihrem Inhalt gestaltet werden. Je nach Visualisierungswerkzeug stehen dabei unterschiedliche Möglichkeiten zur Verfügung.

3.4.1 Visuelle Gestaltung

Der Visualisierungstyp wird dadurch bestimmt, wie er in JavaScript einer Visualisierung mit einem bestimmten Namen zugewiesen und dann initialisiert wird. So wird die oben genannte Visualisierung 'Ort' mit einer Icicle-Visualisierung dargestellt. Es wird für jede einzelne Visualisierungsansicht ein Skript erstellt. Es ist in diesem Skript möglich, die Darstellung der Visualisierung feingranular zu bestimmen.

Eine weitere Anpassung der einzelnen Visualisierung kann auf zwei Wegen erfolgen:

- Direkte Einbettung von Darstellungsinformationen in die Daten bei deren Erstellung (PHP)
- Nachträgliche Manipulation vor dem Rendern eines Knotens (JavaScript/JIT)

Schon während der Erstellung des Graphen ist es möglich, Darstellungsinformation zusätzlich einzubetten: Zu jedem Item lassen sich beliebige Zusatzinformationen in einem `data`-Feld speichern, dazu gehören auch Informationen über z.B. die Farbe oder die flächige Ausbreitung einer Knotenrepräsentation. Damit kann ein Großteil der Datenanalyse bereits serverseitig stattfinden. So kann beim Erstellen des Knotens für ein bestimmtes Item gleich die Farbe des Knotens nach Organisationszugehörigkeit gesetzt werden. In einer Darstellung, die die Verknüpfungen zwischen Items direkt enthält, könnte man auch z.B. temporäre Beziehungen hervorheben.

Das Wiederspiegeln des aktuellen Zustandes geschieht dann 'vor Ort', also im ausgeführten JavaScript. Dabei kann es sich z.B. eine farbliche Markierung des aktuell ausgewählten Items handeln. Oder als ein weiterer möglicher Anwendungsfall: Bei Auswahl eines Filters können alle Items hervorgehoben werden, die der Filterregel entsprechen, z.B. solche mit nicht abgeschlossenen Tickets.

3.4.2 Interaktion

Die Ansicht auf die Visualisierung kann entweder durch das Betätigen von Bedienelementen verändert werden oder durch direkte Interaktion via Links- und Rechtsklick. Außerdem kann beim Überfahren der Visualisierung mit der Maus ein Label angezeigt werden. Das Verhalten und die Verfügbarkeit von diesen Interaktionsmöglichkeiten wird vollständig vom erstellenden JavaScript bestimmt.

Abbildung 3.8 zeigt ein Beispiel für solche Bedienelemente. In der Umsetzung hier wurde das Navigieren mit der Maus gewählt. Das Anklicken eines Knotens zeigt auf der erstellten Seite Details zu dem repräsentierten Item an und markiert dieses rot (vgl. Abbildung 3.9), das wiederholte Anklicken eines bereits auswählten Knotens lädt die anderen Visualisierungen zu diesem Knoten nach und macht diesen zur Wurzel des Graphen, in dem man gerade navigiert hat.

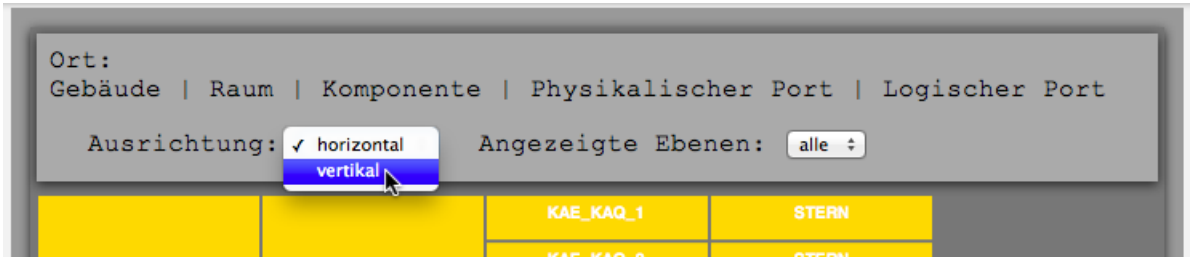

Abbildung 3.8: Beispiel: Bedienelemente zur Veränderung der Visualisierung

3.5 Ablauf der Visualisierung

Es wird eine Anfrage zu einem bestimmten Item gestellt, indem man die URL zu `index.html` mit der OTRS-ID als Parameter aufruft (z.B. http://localhost/otrsvis/?string=10117000048). Nach dem Auslesen des Parameters werden zunächst alle vorhandenen Detail-Informationen abgefragt, dazu gehören sowohl allgemeine Informationen, die zu jedem Item vorhanden sind, als auch spezielle, klassenabhängige Informationen. Dann geschieht die erste Interaktion mit dem mit Visualisierung befassten Teil der Skripte: Es wird angefragt, welche Visualisierungen für dieses Item vorhanden sind. Diese Information ist in den generierenden Skripten fest hinterlegt. Dann werden für jede dieser hier angegebenen Visualisierungen die Graph-Daten abgefragt, die in Form von JSON zurückgeliefert werden. Welche Verknüpfungen hier abgebildet werden, ist ebenso Teil der PHP-Skripte. Abhängig vom Namen der Visualisierung wird der Visualisierungs-Typ ausgewählt und dargestellt. In der jetzigen Umsetzung existiert zu jeder Visualisierung eine eigene JavaScript-Datei, die alle Darstellungs- und Verhaltensinformationen zu dieser speziellen Visualisierung enthält.

Mittels der dargestellten Visualisierungen lässt sich zwischen den Items mit der Maus navigieren. Mit Hilfe von Bedienelementen kann die Darstellung der Visualisierung verändert werden. Außerdem wird eine rudimentäre Legende zu jeder Visualisierung angezeigt.

Abbildung 3.9 zeigt ein Beispiel für eine so generierte Seite. Diese besteht aus den Detail-Informationen zu dem ausgewählten Item auf der linken Seite, einschließlich einer Auflistung der direkten Beziehungen zu anderen Items, und den zu diesem Zeitpunkt verfügbaren Visualisierungen auf der rechten Seite. Der rot markierte Knoten entspricht dem ausgewählten Item.

Die Abbildungen 3.10 zeigt ein Beispiel für die ‘Ort’-Visualisierung. Es ist einfach möglich der gewählten IP-Adresse z.B. die entsprechende Komponente oder den Raum, in dem sich diese befindet, zuzuordnen ohne mehreren Links folgen zu müssen

Abbildung 3.11 zeigt anhand von Dummy-Daten ein Beispiel für den Übergang zwischen Organisationsgrenzen. Der rot hinterlegte Knoten entspricht dem ausgewählten, die gelben lassen sich dem LRZ, die blauen der TUM zuordnen.

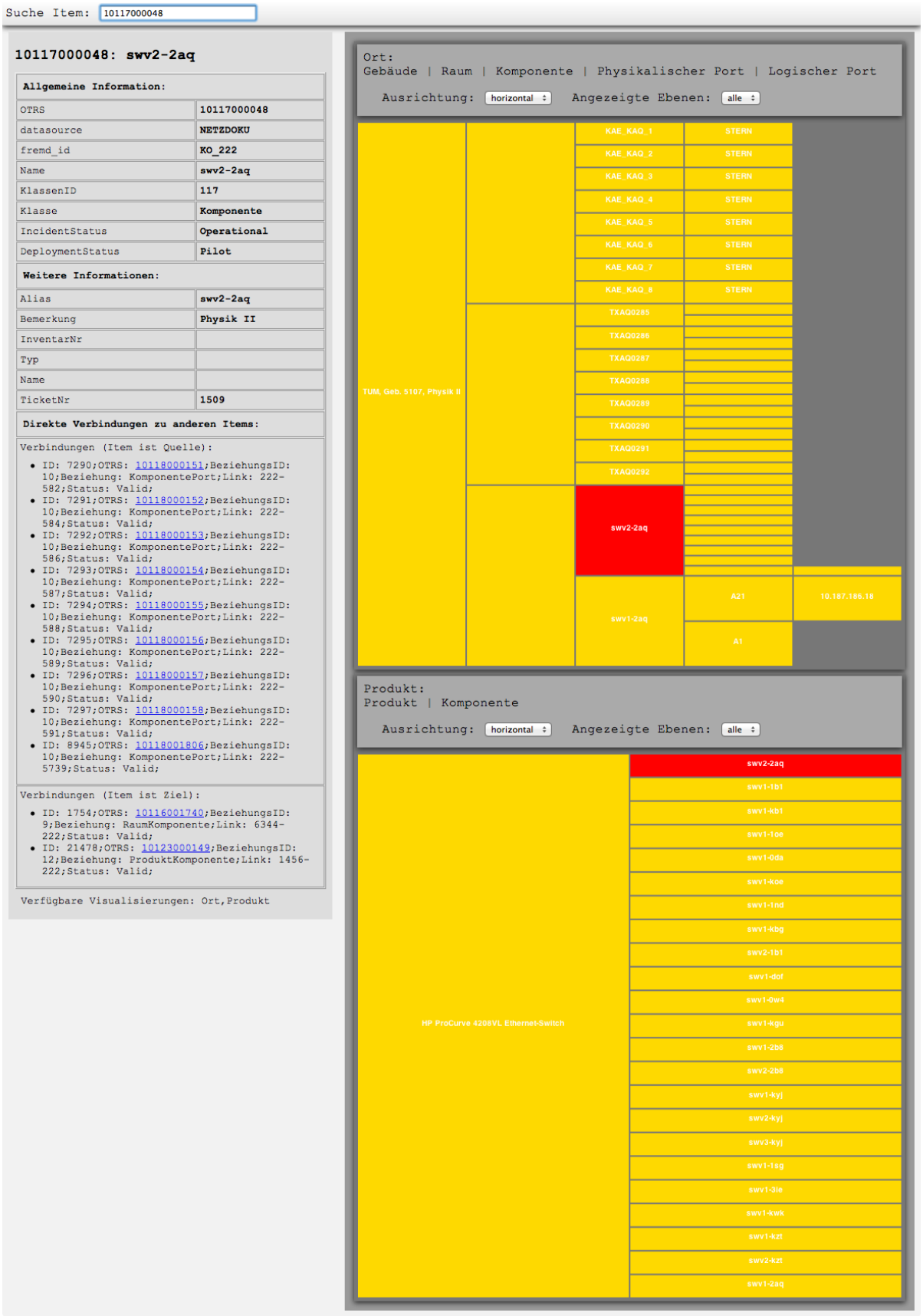

Abbildung 3.9: Beispiel: Gesamte generierte Seite für ‘swv2-2aq’

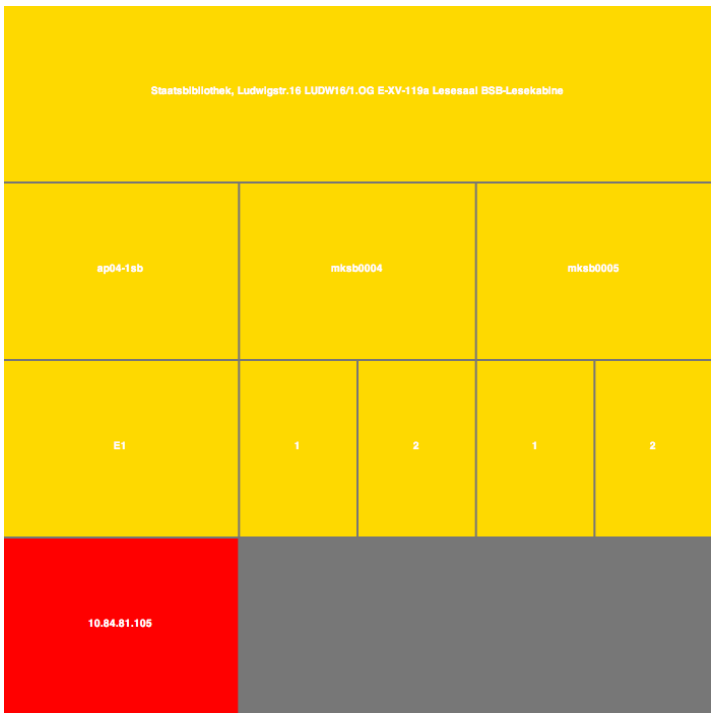

Abbildung 3.10: Beispiel: Zuordnung einer IP zu einem bestimmten Raum

Abbildung 3.11: Beispiel mit Dummy-Daten: Übergang zwischen Organisationen

4 Fazit und Ausblick

In dieser Arbeit wurde ein Konzept für die automatisierte Erstellung von Visualisierungen vorgestellt, um das IT-Management zu unterstützen. Das Konzept umfasst die Analyse und Auswahl der zu visualisierenden Daten, die Wahl des Visualisierungswerkzeuges, die Transformation der Daten hin zu einem Format, das für das Visualisierungswerkzeug geeignet ist, sowie Ansatzpunkte für die Anpassung der resultierenden Visualisierung. Es erfolgte eine prototypische Umsetzung des Konzeptes.

Es wurden folgende Kriterien für die gewünschten Visualisierungen aufgestellt:

- Interaktive Visualisierungen, nicht statische Bilder
- Unterschiedliche Visualisierungstypen mit der Möglichkeit diese für einen bestimmten Anwendungsfall nach eigenen Vorstellungen zu verändern
- Geringer Aufwand bei der Erstellung von Datensätzen, die dem Eingabeformat des Visualisierungswerkzeuges entsprechen und Automatisierung dieses Vorganges

Von diesen ausgehend wurde sich zunächst in den Beispieldatensatz eingearbeitet und die Informationen ausgewählt, die für die angestrebte Visualisierung von Cloud-Managementdaten besonders von Interesse waren. Dann erfolgte eine Vorstellung des ausgewählten Visualisierungswerkzeuges. Anschließend wurden in mehreren Einzelschritten die Daten in das Eingabeformat des Visualisierungswerkzeuges umgewandelt und mit dessen Hilfe dargestellt. Dabei wurde Software entwickelt, die ausgehend von einer OTRS-CI-ID automatisiert Informationen zu diesem Item abfragt und die dafür verfügbaren Visualisierungen anzeigt. Es wurde eine hinreichende Teilmenge der Daten und ein Ausschnitt der möglichen Visualisierungen berücksichtigt, anhand derer die Tragfähigkeit des Konzepts nachgewiesen werden kann.

4.1 Bewertung

Im Rahmen dieser Arbeit wurden anhand des dafür entworfenen Konzeptes Visualisierungen erstellt, die den gestellten Anforderungen gerecht werden.

4.1.1 Bewertung Konzept

Die entworfene Vorgehensweise hat sich als funktional erwiesen. Das Herangehen in den Schritten Datenanalyse, Einarbeitung in das Werkzeug und dann Erstellung der Visualisierungen (vgl. Abbildung 3.1) lässt sich auf beliebige Datensätze übertragen, ebenso wie auf andere Visualisierungswerkzeuge.

Die konkrete Umsetzung erfüllt die Anforderungen, die in 2.1 an sie gestellt wurden:

- Es besteht die Möglichkeit der Interaktion mit den Visualisierungen. Sowohl aktive (z.B. Klicken) als auch passive (z.B. Hovern) Interaktion ist möglich.

- Das Visualisierungswerkzeug bietet verschiedene Visualisierungsarten.
- Die Visualisierungen können nach eigenen Kriterien angepasst werden, z.B. ein eigenes Farbschema oder Logos für die Herkunft von Daten.
- Die Konvertierung der Daten lässt sich automatisieren. Die Aufbereitung der Daten beinhaltet hauptsächlich die logische Anordnung nach den Anforderungen einer bestimmten Visualisierung.

Die erstellten Visualisierungen erleichtern das Navigieren in der organisationsübergreifenden CMDB und ermöglichen damit, dass man sich schneller in dieser zurechtfindet. Das Auffinden des Raumes, in dem sich eine Komponente mit einer bestimmten IP-Adresse befindet, ist beispielsweise leichter (vgl. Abbildung 3.10).

Das Erstellen der Visualisierungen wird auch zukünftig funktionieren, wenn weitere Daten in der organisationsübergreifenden CMDB ergänzt werden. Das Hinzufügen von weiteren Organisationen, so z.B. anderer Departments der TUM, ist nur mit einer geringfügigen Anpassung der PHP-Skripte verbunden.

Zu diesem Zeitpunkt wird nur ein Teil der möglichen Visualisierungen erzeugt. Diese können also noch erweitert werden. Wenn andere Daten in Form von z.B. anderen CI-Typen zu der CMDB hinzugefügt würden, können dafür weitere Visualisierungen erstellt werden. Die Erweiterung der Visualisierungsskripte ist leicht möglich.

4.1.2 Bewertung Datenbasis

Wie in Abschnitt 2.2 erwähnt, wurde in folgenden Punkten Verbesserungspotential für die Daten identifiziert:

- Die Zeichenkodierung der Daten ist teilweise fehlerhaft.
- Die Information zur Herkunft einzelner Datensätze ist nur teilweise vorhanden und nicht ausreichend für eine eindeutige Zuordnung von Verantwortlichkeitsbereichen.
- Die Information zur Herkunft von Daten und über deren Bezeichnung in der Herkunftsdatenbank ist nicht Teil der extra neu erstellten OTRS-CI-Typ-Definitionen.
- Die vorhandenen CI-Typen sind so allgemein gehalten, dass sich nicht immer erschließen lässt, um welche Art von Komponente es sich handelt und der Typ indirekt über die verknüpften Beziehungen oder über die Produktbezeichnung ermittelt werden muss, falls diese vorhanden sind.
- Es wurden nicht die in OTRS ITSM enthaltenen, umfangreichen CI-Klassen genutzt und die zugehörigen Beziehungen.
- Die Daten wurden nicht zusammengeführt, so dass zum einen Duplikate einzelner CIs mit unterschiedlichen Informationen enthalten sind, zum anderen keinerlei Verknüpfung zwischen den Datensätzen der unterschiedlichen Organisationen besteht.

Die Daten eigenen sich trotzdem zur Erstellung von Visualisierungen. Gerade mit Schwerpunkt auf Cloud-Computing wäre es jedoch sinnvoll, die Daten aufzubereiten, bevor das Erstellen weiterer Visualisierungen verfolgt wird.

4.1.3 Bewertung Visualisierungswerkzeug

Das JIT hat geringe Systemvoraussetzungen und lässt sich plattformunabhängig verwenden. Dadurch, dass es als Skript im Browser läuft, lässt es sich gut neben OTRS betreiben, da auf dieses auch per Web-Ansicht zugegriffen wird.

Man kann mit Hilfe des JIT Visualisierungen erstellen. Es handelt sich bei diesem Toolkit um kein spezialisiertes Werkzeug für die Darstellung von z.B. Netzkomponenten, die integrierten Visualisierungstypen umfassen aber Bäume und Netze und decken damit die meisten Anwendungsfälle ab.

Die Performanz der erstellten Visualisierungen ist stark davon abhängig, ob die Visualisierungen mit Canvas gezeichnet oder als formatiertes HTML in das DOM eingefügt werden. Visualisierungen auf DOM-Basis lassen sich leichter individuell anpassen, weil JIT dafür dedizierte Methoden zur Verfügung stellt. Die Performanz der Webseite bricht aber schon bei relativ kleinen Datenmengen ein. Canvas-basierte Visualisierungen reagieren auch bei großen Datenmengen problemlos, die Anpassung der Darstellung ist aber aufwändiger.

Bei der Darstellung von großen, häufig unterteilten Datenmengen stößt das Toolkit mit der ausgewählten Visualisierung an seine Grenzen. Ein Beispiel dafür zeigt Abbildung 4.1, in der die Namen der CIs nicht im Knoten angezeigt werden. Durch die Wahl anderer Visualsierungsarten lässt sich das allerdings umgehen.

Abbildung 4.1: JIT: Eingeschränkte Darstellung der Knoten bei größeren Datenmengen

Wenn man die zugrundeliegende Datenstruktur eines Baumes oder eines beliebigen anderen Graphen für das JIT erstellt hat, ist es sehr einfach von einer Visualisierungsart auf eine andere zu wechseln, die das selbe Datenformat erwartet.

4.2 Ausblick

Im Folgenden werden Möglichkeiten für zukünftige Arbeiten vorgestellt.

4.2.1 Erweiterung der Prototypen

Da es sich nur um eine prototypische Umsetzung handelt, bietet sie viele Ansatzpunkte zur Verbesserung und Erweiterung. Man kann z.B. weitere Visualisierungen zu dem hier entwickelten Prototypen hinzufügen, insbesondere solche, die auf Graphen basieren. Außerdem

lässt sich die Darstellung noch weiter anpassen und verbessern, so könnte man z.B. die Legende erweitern oder die Interaktion touch-freundlich gestalten.

4.2.2 Integration des Prototypen mit OTRS

Die Verbindung von OTRS und den Visualisierungen besteht im Augenblick nur über die Verwendung des selben Datenbestandes und den Zugriff auf bestimmte CIs über dieselbe ID. Eine bessere Integration mit OTRS wäre wünschenswert. Dabei könnte man als ersten Schritt z.B. die URL zum Aufruf der Visualisierungen eines Items in die Detail-Anzeige von OTRS einbinden. Alternativ könnte man auch die Visualisierungen selbst in die Detail-Ansicht integrieren.

4.2.3 Verbesserung der Datengrundlage

Die in Abschnitt 4.1.2 genannten Probleme der Datenbasis bieten großes Potential für Verbesserungen dieser. Beispielsweise kann ein Abgleich der Daten gegen die jeweiligen Herkunfsdatenbanken mittels Regressionstests zu einer konstanteren Datenqualität führen.

4.2.4 Erweiterung der Datengrundlage

Die Integration von organisationsübergreifenden CMDBs kann noch fortgeführt werden, indem man Datensätze anderer Organisationen (oder Organisationseinheiten) hinzufügt und integriert. Man könnte beispielsweise die Konfigurationsdatensammlungen der anderen Departments der TU München oder weitere relevante CIs aus CMDBs von anderen Cloudanbietern, deren Dienste die TUM nutzt, importieren.

4.2.5 Einsatz im Live-Betrieb

Sowohl die Vorarbeit von [Xia10] als auch diese Arbeit befassen sich mit der Erstellung von Konzepten. Beide Arbeiten finden keine konkrete Anwendung in der Praxis, haben aber in Form von Prototypen die Anwendbarkeit dieser Konzepte demonstriert. Nach den oben genannten Verbesserungen ist auch eine Umsetzung und Inbetriebnahme in der Praxis denkbar.

4.3 Danksagung

Vielen Dank an meine Betreuer Silvia Knittl und Dr. David Schmitz für Beistand und Geduld, Stefan Loidl vom Leibniz-Rechenzentrum für eine Einführung in die Netzdoku und die Arbeit des LRZs, Dr. Josef Homolka von der Physik der TU München für Einblicke in die Aufgaben eines Netzverantwortlichen und die Hilfestellungen und Kommentare zu dieser Arbeit.

Verzeichnisstruktur der beiliegenden CD

- /
  - brod11
    - Dokumentation — Ausarbeitung
      - Latex
        - Bib — Bibliografie
        - Bilder — Eingebettete Bilder
        - Titel — Titelblatt
          - Code — Eingebetteter Quellcode
      - PDF
    - Folien — Vortragsfolien
    - Quellen — Verwendete Quellen
      - Bibliotheken — JavaScript-Bibliotheken
        - JIT
        - jQuery
      - Bilder
      - Papers — Online verfügbare Papers
      - Programme — Installer, Disk-Images
        - MacOS-10-7
        - OpenSuse-11-4
        - VorarbeitXia — Aus der Vorarbeit verfügbare Daten
          - Anhang — Quellcode und Export-Ergebnisse
          - SQL-Dump — Dump der OTRS-Datenbank
    - Sourcen — Selbst erstellte Dateien
      - Screenshots — Screenshots
      - Visualisierungen — Quellcode für die Erstellung der Visualisierungen
        - htdocs — Root-Verzeichnis
          - css — Stylesheets
          - images — Statische Bilder
          - js — JavaScript-Code
            - vis — Einzelne Visualisierungen
          - jslib — JavaScript-Bibliotheken
          - json — Beispieldaten
          - php — PHP-Code
      - Omnigraffle — Omnigraffle-Datein

Abkürzungsverzeichnis

CI Configuration Item

CM Configuration Management

CMDB Configuration Management Database

CSS Cascading Style Sheets

DOM Document Object Model

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IaaS Cloud Infrastructure as a Service

IP Internet Protocol

IT Information Technology

ITIL Information Technology Infrastructure Library

ITSM IT-Service-Management

JIT JavaScript InfoVis Toolkit

JSON JavaScript Object Notation

LMU Ludwig-Maximilians-Universität München

LRZ Leibniz-Rechenzentrum

MWN Münchner Wissenschaftsnetz

NaaS Network as a Service

OTRS Open Ticket Request System

PaaS Cloud Platform as a Service

PHP PHP: Hypertext Preprocessor

SaaS Cloud Software as a Service

SQL Structured Query Language

SVG Scalable Vector Graphics

TU Technische Universität

TUM Technische Universität München

URL Uniform Resource Locator

VLAN Virtual Local Area Network

XML Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BBB+10] Bakker, René, Hauke Böttcher, Jens Bothe, Udo Bretz, Martin Edenhofer, Manuel Hecht, Christopher Kuhn, André Mindermann, Henning Oschwald, Thomas Raith, Stefan Rother, Burchard Steinbild, Marco Romann und Werner Siebecke: *OTRS::ITSM 2.0 - Grundlagen*, 2010. Abrufbar unter `http://doc.otrs.org/itsm/2.0/de/html/`.

[BBB+11] Bakker, René, Stefan Bedorf, Michiel Beijen, Shawn Beasley, Hauke Böttcher, Jens Bothe, Udo Bretz, Martin Edenhofer, Carlos Javier García, Martin Gruner, Manuel Hecht, Christopher Kuhn, André Mindermann, Marc Nilius, Elva María Novoa, Henning Oschwald, Martha Elia Pascual, Thomas Raith, Carlos Fernando Rodríguez, Stefan Rother, Burchard Steinbild und Daniel Zamorano: *OTRS 3.0 - Administrator-Handbuch*, 2011. Abrufbar unter `http://doc.otrs.org/3.0/de/html/`.

[BMW05] Bruns, Kai und Klaus Meyer-Wegener: *Taschenbuch der Medieninformatik*. Fachbuchverlag Leipzig, 2005.

[CDPT09] Ciechanowicz, David, Philipp Donie, Tomislav Pravidur und René Tiede: *Modeling and Documentation of IntegraTUM's System Landscape*. Projektarbeit, Technische Universität München, München, 2009.

[CMS99] Card, Stuart K., Jock D. Mackinlay und Ben Shneiderman: *Readings in Information Visualization - Using Vision to Think*. Morgan Kaufmann Publishers Inc., 1999.

[Cro06] Crockford, Douglas: *RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON)*. `http://www.ietf.org/rfc/rfc4627.txt`, Juli 2006.

[ecm11] *ECMA International - Standard ECMA-262, 5.1 Edition, ECMAScript Language Specification*, jun 2011. Abrufbar unter `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf`.

[Ern09] Ernst, Alexander M.: *Enterprise Architecture Management Patterns*, feb 2009. Abrufbar unter `http://wwwmatthes.in.tum.de/file/Projects/EAMPC/Download/Er08a.pdf`.

[GB10] Garcia Belmonte, Nicolas: *JavaScript InfoVis Toolkit - Loader.js*. `http://thejit.org/static/v20/Docs/files/Loader/Loader-js.html` (Letzter Zugriff am 10.09.2011), 2010.

[GB11] Garcia Belmonte, Nicolas: *JavaScript InfoVis Toolkit*. `http://thejit.org/` (Letzter Zugriff am 10.09.2011), 2011.

[iti] *Official ITIL® Website*. http://www.itil-officialsite.com/ (Letzter Zugriff am 24.08.2011).

[jqu] *jQuery: The Write Less, Do More, JavaScript Library*. http://jquery.com/ (Letzter Zugriff am 10.09.2011).

[jso] *JSON - JavaScript Object Notation*. http://www.json.org/ (Letzter Zugriff am 10.09.2011).

[KL10] Knittl, Silvia und Albert Lauchner: *Hybrid-Cloud an der Technischen Universität München - Auswirkungen auf das IT-Management*. In: *Neue Wertschöpfungsmodelle und Dienste durch Cloud Computing - Workshop im Rahmen der 40. Jahrestagung der Gesellschaft für Informatik*, Leipzig, September 2010. Gesellschaft für Informatik e.V. (GI).

[lrz] *Das Leibniz-Rechenzentrum - IT-Dienstleistungen (Letzter Zugriff am 24.08.2011)*. Abrufbar unter http://www.lrz.de/wir/Einfuehrung-LRZ.pdf.

[lrz08] *Flyer: Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften*, Dezember 2008. Abrufbar unter http://www.lrz.de/wir/lrz-flyer/de/faltblatt.pdf (Letzter Zugriff am 24.08.2011).

[lrz10a] *Das Münchner Wissenschaftsnetz (MWN) - Konzepte, Dienste, Infrastrukturen, Management*, April 2010. Abrufbar unter http://www.lrz.de/services/netz/mwn-netzkonzept/MWN-Netzkonzept-2010.pdf.

[lrz10b] *LRZ: Netzverantwortliche in den Instituten*. http://www.lrz.de/services/netz/netzverantwortliche/ (Letzter Zugriff am 24.08.2011), Dezember 2010.

[MG11] Mell, Peter und Timothy Grance: *SP 800-145 - The NIST Definition of Cloud Computing (Draft)*. Abrufbar unter http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf, Januar 2011.

[mys] *MySQL :: The world's most popular open source database*. http://www.mysql.com/ (Letzter Zugriff am 10.09.2011).

[otr] *OTRS: Developer API*. http://dev.otrs.org/ (Letzter Zugriff am 10.09.2011).

[phpa] *PHP: Hypertext Preprocessor*. http://www.php.net/ (Letzter Zugriff am 10.09.2011).

[phpb] *PHP: What can PHP do? - Manual*. http://www.php.net/manual/en/intro-whatcando.php (Letzter Zugriff am 28.08.2011).

[RH07] Rance, Stuart und Ashley Hanna: *ITIL V3 Glossary v01*, May 2007. Abrufbar unter http://www.itil-officialsite.com/ (Letzter Zugriff am 24.08.2011).

[thea] *Icicle - Icicle tree with limited levels shown*. http://thejit.org/static/v20/Jit/Examples/Icicle/example2.html/ (Letzter Zugriff am 10.09.2011).

[theb] *Spcaetree - Tree Animation.* `http://thejit.org/static/v20/Jit/Examples/Spacetree/example1.html` (Letzter Zugriff am 10.09.2011).

[w3ca] *4.8.11 The canvas element - HTML5.* `http://www.w3.org/TR/html5/the-canvas-element.html` (Letzter Zugriff am 26.08.2011).

[w3cb] *Cascading Style Sheets.* `http://www.w3.org/Style/CSS/` (Letzter Zugriff am 10.09.2011).

[w3cc] *HTML & CSS - W3C.* `http://www.w3.org/standards/webdesign/htmlcss` (Letzter Zugriff am 10.09.2011).

[w3cd] *Scalable Vector Graphics (SVG) 1.1 (Second Edition).* `http://www.w3.org/TR/SVG/` (Letzter Zugriff am 10.09.2011).

[w3ce] *Scripting and Ajax - W3C.* `http://www.w3.org/standards/webdesign/script` (Letzter Zugriff am 10.09.2011).

[w3cf] *W3C Document Object Model.* `http://www.w3.org/DOM/` (Letzter Zugriff am 10.09.2011).

[w3cg] *W3C HTML.* `http://www.w3.org/html/` (Letzter Zugriff am 10.09.2011).

[w3ch] *W3C SVG Working Group.* `http://www.w3.org/Graphics/SVG/` (Letzter Zugriff am 10.09.2011).

[Web09] Weber, Mathias: *BITKOM Leitfaden: Cloud Computing - Evolution in der Technik, Revolution im Business*, Oktober 2009. Abrufbar unter `http://www.bitkom.org/files/documents/BITKOM-Leitfaden-CloudComputing_Web.pdf`.

[wika] *JavaScript - Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=JavaScript&oldid=446843608` (Letzter Zugriff am 10.09.2011).

[wikb] *SQL - Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=SQL&oldid=447137961` (Letzter Zugriff am 10.09.2011).

[wikc] *Wikis for Everyone - Wikispaces.* `http://www.wikispaces.com/` (Letzter Zugriff am 10.09.2011).

[xam] *apache friends - xampp.* `http://www.apachefriends.org/en/xampp.html` (Letzter Zugriff am 10.09.2011).

[Xia10] Xia, Wenzhe: *Evaluation des TUM-Trouble Ticket Systems als Ersatz für bestehende lokale Konfigurationsmanagementdatenbanken.* Projektarbeit, Ludwig-Maximilians-Universität, München, Januar 2010. Abrufbar unter `http://www.nm.ifi.lmu.de/pub/Fopras/xia10/PDF-Version/xia10.pdf`.

[Zei10] Zeiser, Jennifer: *Analyse der Anwendbarkeit multimedialer Visualisierungs- und Interaktionstechniken im IT Service Management.* Bachelorarbeit, Ludwig-Maximilians-Universität, München, Dezember 2010.