i-doit - Open Source-CMDB

Dennis Blümer
synetics GmbH
Humboldtstr. 101
40237 Düsseldorf
Deutschland
dbluemer@i-doit.org

1 Zusammenfassung

Ein gutes IT Service Management (ITSM) ist die Grundlage für einen erfolgreichen und produktiven Betrieb von IT und somit des gesamten Unternehmens. Häufig werden für verschiedene Prozesse, wie das Asset-, Problem- oder Changemanagement unterschiedliche, voneinander unabhängige, Datenquellen genutzt. Dies führt unweigerlich zu dem Problem, dass in vielen Unternehmen kritische Informationen nicht zum gewünschten Zeitpunkt im gewünschten Kontext zur Verfügung stehen. Die Folge sind Inkonsistenzen oder Integritätsverletzungen, die den ordnungsgemäßen Ablauf von Prozessen beeinträchtigen und im schlimmsten Fall den laufenden Betrieb zum Erliegen bringen können.

Um diesem Problem entgegen zu wirken, wurde von der *Central Computing and Telecommunications Agency* (CCTA) die IT Infrastructure Library (ITIL) ausgearbeitet. Diese sollte beschreiben, wie eine IT-Infrastruktur beschaffen sein muss und welche Prozesse benötigt werden, um eine IT möglichst effizient umzusetzen. Unter anderem beinhaltet die ITIL das Konzept der (föderierten) Configuration Management Database (CMDB). In einer CMDB sind alle Mittel und Ressourcen, die zur Leistungserbringung erforderlich sind, dokumentiert und zueinander in Beziehung gesetzt. Sie dient als zentrale Datenbasis für alle Prozesse innerhalb der IT und soll eine einheitliche Gesamtsicht auf den IT-Bestand liefern.

Eine gute CMDB muss flexibel an die Anforderungen eines Unternehmens anpassbar sein. Verschiedene Unternehmen verwalten unterschiedliche Configuration Items. Viele Unternehmen kommen häufig mit einer geringen Detailtiefe aus, während andere Unternehmen teilweise eine port- und kabelgenaue Abbildung ihrer Infrastruktur benötigen. Daher sind viele Implementierungen von CMDBs maßgefertigte Lösungen für ein bestimmtes Unternehmen.

Die Open Source-Lösung i-doit zeichnet sich durch ihre freie Konfigurierbarkeit und Flexibilität aus. Somit kann jedes Unternehmen i-doit seinen Anforderungen an eine Dokumentation anpassen. über die Bereitstellung einer CMDB hinaus, bietet i-doit eine Reihe an Zusatzfunktionen und Modulen, die eine Prozessintegration und ein aussagekräftiges Reporting ermöglichen. Einige dieser Features umfassen beispielsweise die Versionierung von abgelegten Dateien, ein rollenbasiertes Rechtesystem, die Lizenzverwaltung oder die vielfältigen Verknüpfungsmöglichkeiten der Objekte. Vereint werden alle Komponenten in einer Client-Server-Architektur, wobei der Zugriff clientseitig über einen Webbrowser erfolgt. Diese Architektur ist modular aufgebaut, um i-doit auf einfache Weise neue Funktionen hinzufügen zu können. i-doit bietet darüber hinaus um seine CMDB herum Dienste an, die anderen ITSM-Funktionen einen einfachen Zugriff auf die abgebildete Infrastruktur ermöglichen.

Die Firma synetics ist seit 1996 Spezialist für adaptive Infrastrukturlösungen. i-doit entstand im Jahr 2004 aus einem Kundenprojekt. Es sollte die bisherige, teils unstrukturierte, Dokumentation der IT durch eine geeignete CMDB konsolidieren und vereinheitlichen. Seitdem wird i-doit als quelloffenes Projekt kontinuierlich weiterentwickelt, wobei die Aspekte der ITIL praxisorientiert berücksichtigt wer-

den.

Dieser Artikel gibt einen Überblick über die Hauptfunktionalitäten und Module von i-doit. Dabei wird vor allem auf die Aspekte der CMDB und der daran anknüpfenden Prozesse eingegangen.

2 Aufbau der CMDB

Zunächst wird beleuchtet, wie die CMDB in i-doit aufgebaut ist und wie der Einsatz der Software einem Unternehmen einen erheblichen Mehrwert in der Dokumentation der Infrastruktur und im gesamten ITSM verschafft. i-doit versteht sich als Lösung für die ganzheitliche Dokumentation von IT-Systemen. Dies wird durch ein an der CMDB ausgerichteten Framework, das die Anbindung beliebiger weiterer ITSM-Funktionen ermöglicht, erreicht.

Grundsätzlich kann mit i-doit jedes beliebige Objekt dokumentiert werden, unabhängig von seiner Art und Struktur. Dazu steht eine Menge an vordefinierten Objekttypen zur Verfügung, die um weitere Typen ergänzt werden kann. In den folgenden Abschnitten wird der Aufbau der Objekte (Configuration Items) innerhalb der CMDB dargestellt.

2.1 Mandanten

Der Mandant stellt die oberste Ebene innerhalb der Struktur einer zu erfassenden IT-Umgebung dar. Ein Mandant ist ein in sich geschlossenes System, es können aber mehrere Mandanten mittels einer einzelnen i-doit-Installation verwaltet werden. Dazu verwendet i-doit eine Systemdatenbank, die wiederum beliebig viele Mandantendatenbanken verwaltet. Innerhalb der Systemdatenbank sind keine Infrastrukturdaten gespeichert.

Alle Daten, die für die Erfassung einer bestimmten Infrastruktur notwendig sind, werden in einer Mandantendatenbank gehalten. Hier folgen die Daten der folgenden Struktur: Um ein bestimmtes Objekt zu erfassen, wird ihm ein Objekttyp zugewiesen. Die Objekttypen zeichnen sich durch eine Zuordnung zu Kategorien aus. Innerhalb der Kategorien werden die konkreten Ausprägungen zu jedem Objekt als Eigenschaften gespeichert. In den folgenden Abschnittn 2.2 bis 2.4 wird auf diese Strukturierung genauer eingegangen.

2.2 Objekttypen

Mit Hilfe der Objekttypen werden Objekte gleicher Art zusammengefasst, also beispielsweise Server, Clients oder Drucker. Jedes Objekt in der CMDB muss genau einem Objekttypen zugeordnet werden, wonach die Objekte auch gruppiert werden. In i-doit werden eine Reihe vordefinierter Objekttypen mitgeliefert, die die gängigsten IT-Komponenten (Server, Router, Software, etc.) und weitere periphere Informationen (Vertragsdaten, Lizenzen, Netze, etc.) abbilden. Diese Menge kann nach eigenem Bedarf um weitere Objekttypen erweitert werden.

Ein Objekttyp zeichnet sich durch die Zuordnung von Kategorien aus. Darüber hinaus können Objekttypen als so genannte *Containerobjekte* definiert werden. Das ist z.B. bei den mitgelieferten Objekttypen Gebäude, Raum und Schrank der Fall. Somit kann ein Server in einem Schrank stehen, der wiederrum einem Raum und dieser einem Gebäude zugeordnet ist.

2.3 Kategorien

Kategorien dienen der strukturierten Erfassung der für ein Objekt relevanten Daten. Kategorien existieren in i-doit in zwei verschiedenen Varianten: Auf der einen Seite gibt es allgemeine Kategorien, die jedem Objekttypen zugewiesen werden können, auf der anderen Seite gibt es spezifische Kategorien, die speziell für einen Objekttypen gedacht sind. So existiert beispielsweise die allgemeine Kategorie *CPU*, die die Ausprägungen einer oder mehrerer zugehöriger Prozessoren beschreibt. Diese Kategorie

würde man wahrscheinlich dem Objekttypen Server zuordnen wollen, um die zugehörigen Prozessoren zu dokumentieren. Einem Schrank hingegen würde man diese Kategorie eher nicht zuweisen, obwohl dies prinzipiell möglich ist.

2.3.1 Allgemeine Kategorien

Jedem Objekttypen können beliebig viele allgemeine Kategorien zugewiesen werden, je nachdem, wie genau man ein Objekt beschreiben möchte. Meistens werden bestimmte Objekttypen, wie z.B. Server, sehr detailliert beschrieben, wohingegen andere Objekte, wie z.B. Monitore, eher grob beschrieben werden. So kann durch die Wegnahme von Kategorien eine besseren Übersichtlichkeit erreicht werden. Innerhalb allgemeiner Kategorien werden Daten verwaltet, die in der Regel für mehr als nur einen Objekttypen relevant sind.

2.3.2 Spezifische Kategorien

Spezifische Kategorien kommen dort zum Einsatz, wo die durch die allgemeinen Kategorien bereitgestellten Funktionen/Daten nicht ausreichen, das Objekt genau genug zu beschreiben. Jedem Objekttypen wird dabei maximal eine spezifische Kategorie zugewiesen. So besitzt der Objekttyp *Schrank* beispielsweise standardmäßig eine spezielle Kategorie *Schrank*, in der die Anordnung der darin installierten Geräte festgelegt werden kann.

2.4 Objekte

Als Objekte werden in i-doit jegliche Arten von Einträgen bezeichnet, die die Kernelemente der CMDB, die so genannten *Configuration Items*, bilden. So wird z.B. ein einzelner Computer als Objekt bezeichnet. Jedes Objekt gehört zu genau einem Objekttypen und zeichnet sich durch die konkreten Ausprägungen der diesem Objekttypen zugewiesenen Kategorien aus. Alle Änderungen an einem Objekt werden dabei zentral verzeichnet. Um den Lebenszyklus eines Objektes zu beschreiben, werden vier Status unterschieden:

- Unfertig
- Normal
- Archiviert
- Gelöscht

Den Status *Unfertig* erhält ein Objekt unmittelbar nach seiner Erstellung, wenn noch keine Eigenschaften in den Kategorien gespeichert sind, also lediglich der Typ des Objekts bekannt ist. Sobald zumindest der Name festgelegt wurde, erhält das Objekt den Status *Normal*. Objekte können archiviert werden, was zur Folge hat, dass diese nicht mehr bearbeitet oder verknüpft werden kann. Außerdem verschwinden Objekte, die sich im Status *Archiviert* befinden aus den Listenansichten und sind nur noch über die Archivierungssicht einsehbar. Da Objekte prinzipbedingt nicht permanent gelöscht werden dürfen, werden dieses über die Löschfunktion zwar in den Status *Gelöscht* versetzt, sind aber in der Datenbank noch vorhanden. Archivierte oder gelöschte Objekte können durch die Funktion *Wiederherstellen* wieder in den Status *Normal* gebracht werden, nicht aber in den Status *Unfertig*.

2.5 Logbuch

Das Logbuch hält jede Änderung an der CMDB fest. Das betrifft z.B. das Anlegen, Editieren, Löschen oder Wiederherstellen von Objekten und deren Kategorieeinträgen. Zu jedem Logbucheintrag wird die entsprechende Aktion automatisch, unter Angabe von Datum und Uhrzeit, gespeichert. Zusätzlich kann der Benutzer zu jeder Aktion einen eigenen Kommentar angeben, der permanent gespeichert wird. Alles,

was einmal ins Logbuch eingetragen wurde, ist dort unlöschbar gespeichert.

Da im Laufe der Zeit ein Logbuch sehr voll werden kann, was die Datenbanklast erhöhen kann, steht eine Archivierungsfunktion zur Verfügung. Damit können Logbucheinträge eines definierbaren Alters in eine andere Tabelle oder externe Datenbank ausgelagert werden. Dieser Vorgang kann automatisiert werden, um das Logbuch in regelmäßigen Zeitabständen von allein zu archivieren. Die ausgelagerten Einträge werden nur noch auf expliziten Abruf geholt und angezeigt. Ausgelagerte Logbucheinträge können jederzeit wieder ins lokale Logbuch eingespielt werden.

Zusätzlich zu den Änderungen an der CMDB, können im Logbuch objektbezogene Daten aus anderen Datenquellen dokumentiert werden. So beinhaltet der Standardumfang z.B. das Eintragen von Statusdaten aus einer NDO-Datenbank oder Syslog-Meldungen, die sich auf ein bestimmtes Objekt beziehen. Es steht weiterhin eine Schnittstelle zur Verfügung, um beliebige weitere Datenquellen anzugeben, aus denen Daten für die Objekte bezogen und aufbereitet werden können, die ins Logbuch einfließen.

3 Kontakte

Kontakte stellen in i-doit eine spezielle Art von Objekten dar, die genau mit allen anderen Objekten in Beziehung stehen. Innerhalb von i-doit stehen drei Arten von Kontakten zur Verfügung:

- Personen
- Gruppen
- Organisationen

Diese drei Kontakte stellen sich folgendermaßen dar:

Personen können als reine Kontaktinformationen gepflegt werden, enthalten aber auch autorisierte Benutzer von i-doit. Jeder Benutzer von i-doit ist also als eine Person in der CMDB gespeichert, nicht jeder Kontakt muss aber gleichzeitig ein i-doit Benutzer sein. Personen können z.B. einem Objekt als Verantwortlicher zugewiesen werden. Aber auch für *Workflows* (siehe Abschnitt 5) werden Personen als Auftraggeber und -nehmer herangezogen.

Gruppen fassen Personen zu bestimmten Verantwortungsbereichen zusammen. So kann einem Objekt, statt einer einzelnen Person, eine ganze Gruppe als Kontakt zugewiesen werden. Eine Gruppe besitzt drei Parameter. Neben ihrem Namen kann eine Email-Adresse und eine Telefonnummer für Gruppenbenachrichtigungen vergeben werden. Gruppen können nicht Mitglied einer anderen Gruppe werden. Die Zuordnung von Personen zu Gruppen kann entweder über die Person oder über die Gruppe stattfinden.

Organisationen dienen in der Kontaktverwaltung von i-doit dazu, Personen zusammenzufassen. Sie stellen also ein Ordnungselement dar und werden zur Pflege der diesen Personen gemeinsamen Stammdaten genutzt. Im Einzelnen handelt es sich hierbei, neben dem Namen, um die postalische Adresse, zentrale Rufnummern, die Webseite und eine Zuordnung einer Zentrale. Letzteres dient dazu, eigene Zweigstellen oder die eines Lieferanten oder Partners mit einer Zentraladresse zu verknüpfen. Neben der Vermeidung von Redundanzen dienen die Organisationen darüber hinaus als Filterkriterium bei der Auswahl von Kontakten in Objekten oder Workflows.

3.1 Anbindung von LDAP

Anstatt die Benutzer von i-doit im System selbst zu halten, kann i-doit bei der Anmeldung auf ein definiertes LDAP-Verzeichnis zurückgreifen. Nach der Konfiguration eines oder mehrerer LDAP Server

und erfogleichem Verbindungstest können sich Benutzer aus der konfigurierten OU bei i-doit anmelden. Der Benutzername für das Active Directory ist in der Standardeinstellung der sAMAccountName, also der Windows Logon Name. Für Novell und OpenLDAP wird der CN als Benutzername verwendet.

4 Verbindungen und Abhängigkeiten

Eine CMDB zeichnet sich insbesondere dadurch aus, dass sie die vielfältigen Kommunikationskanäle und Interdepedenzen zwischen den dokumentierten Configuration Items abbilden kann. Ohne diese Verknüpfungen würde die CMDB keine brauchbaren Informationen liefern. Deshalb wurde in i-doit großer Wert auf die Möglichkeit gelegt, die Verbindungen zwischen den Objekt sehr detailliert abbilden zu können.

4.1 Verbindungen

Folgende Arten von Verbindungen können in i-doit abgebildet werden:

- Stromnetze
- Datennetze
- Speichernetze
- lokaler Speicher

Stromnetze Um in i-doit Stromnetze darzustellen wird eine Menge verschiedener Stromobjekte angeboten. Zum einen gibt es eigenständige Objekte vom Objekttyp Stromobjekt, zum anderen gibt es untergeordnete Objekte in der Kategorie Verbraucher eines beliebigen Objekttyps der Infrastruktur.

Die eigenständigen Stromobjekte werden wiederum in verschiedene Typen unterteilt. Die untergeordneten Objekte in anderen Objekttypen (z.B. Server) sind dagegen immer vom Typ Verbraucher. Der Stromtyp eines Objektes muss bei Erstellung definiert werden. Jedes Stromobjekt hat in Abhängigkeit seines Typus entweder ein oder mehrere Eingänge (Stecker) oder Ausgänge (Buchsen) oder beides.

Es gibt folgende Typen und Eigenschaften:

- Stromversorger haben Ausgänge.
- USV (Unterbrechungsfreie Stromversorgung) haben Eingänge und Ausgänge.
- Stromverteiler haben Eingänge und Ausgänge.
- Verbraucher haben nur Eingänge.

Die Eingänge und Ausgänge werden genutzt, um Stromobjekte untereinander zu verbinden. Jeder Eingang kann jeweils nur mit einem Ausgang verbunden werden und umgekehrt. Durch diese Verknüpfungen wird ein Stromnetz aufgebaut, dass sich von der Versorgung bis zum Verbraucher zieht. Zu den jeweiligen Stromobjekten können Leistungswerte (Volt, Watt und Ampere) definiert werden. An den Buchsen und Steckern können zusätzlich Sicherungen mit der dazugehörigen Amperezahl eingetragen werden, so dass bis ins Detail dokumentiert werden kann, wieviel (nominalen) Stromverbrauch Objekte haben und ob z.B. die Sicherungen dafür ausreichend sind.

Datennetze Für die Abbildung von Datennetzverbindungen wird in i-doit die Kategorie *Netzwerk* verwendet. Dabei handelt es sich um eine allgemeine Kategorie, die also in gleicher Ausprägung bei Servern, Switches oder Routern zum Einsatz kommt. Um diesen verschiedenen Anwendungsfällen gerecht zu werden und gleichzeitig eine einzelne Zugriffsmethode beizubehalten, wird beim Netzwerkmodell von i-doit zwischen *Interface*, *Port* und *logisches Interface* unterschieden.

Ein *Interface* beschreibt die eigentliche Schnittstelle. Sei sie onBoard oder als PCI-Kartenausführung beim Server oder ein Modul in Router oder Switch. *Ports* können in beliebiger Anzahl auf einem Interface angelegt werden und beschreiben die physikalischen Verbindungen zu einem Datennetz. Ein *logisches Interface* wird wie eine virtuelle Schnittstelle behandelt und kann beliebige Ports des zugrunde liegenden Objekts logisch zusammen fassen.

Mit diesem Modell können nahezu alle netzseitigen Konfigurationen abgebildet werden. Damit wird zwar die Dokumentation einer 1-Port-Netzwerkkarte in einem Server mit der jetzigen i-doit Version etwas aufwendiger, dafür kannn aber nach dem selben Vorgehen auch ein Multi-Slot-Chassis eines Switches dokumentiert werden.

Speichernetze In i-doit werden Speichernetze unterteilt in Storage Area Networks (SAN) und Network Attached Storage (NAS). SANs unterteilen sich wiederum in FibreChannel und IP-basierte (ISCSI und FCIP) Netze. SAN-Systeme bestehen aus mehreren Komponenten. Zum einen gibt es den Objekttyp *SAN*, mit dem die eigentlichen SAN-Speichersysteme mit ihren Festplatten beschrieben werden. Dann gibt es noch den Objekttyp FC-Switch, der mehrere FibreChannel SAN-Objekte miteinander verbindet. Die Clientsysteme, die auf den SAN-Speicher zugreifen, benutzen dazu die Kategorie *Controller* und *FC-Port* unter der Kategorie *Storage*.

lokaler Speicher Wie bei allen Objekten in i-doit kann auch bei lokalem Speicher und Speichernetzen die Detailtiefe vom Benutzer selbst bestimmt werden. In Bezug auf Speicher und Speichernetze heisst dies insbesondere, dass alle im Folgenden dargestellten Abhängigkeiten eingetragen werden können, jedoch nicht eingetragen werden müssen. Mit der Konsequenz, dass man z.B. durchaus eine RAID Konfiguration abbilden kann ohne einen Controller einzutragen, oder man ein logisches Laufwerke eintragen kann ohne eine Festplatte angelegt zu haben. Für die Dokumentation lokalen Speichers stehen die Standardkomponenten *Controller*, *Gerät* und *Laufwerk* zur Verfügung. Zusätzlich stehen die Sonderkomponenten *FC-Port* und *SAN-Pool* bereit.

4.2 Abhängigkeiten

Zwischen den meisten Objekttypen kann man eine Verbindung in i-doit herstellen. Wenn ein bestimmtes Objekt abhängig von einem anderen Objekt ist, soll dies im Groben hervorheben, dass das abhängige Objekt seine Arbeiten nicht mehr verrichten kann, wenn das andere Objekt aus irgend einem Grund ausfällt. Die genaue Semantik der Abhängigkeit bleibt aber der freien Interpretation des Benutzers überlassen. Beispielsweise könnte ein Server nicht mehr funktionieren, wenn er abhängig von einem Stromobjekt (Netzgerät, Steckdose, etc) ist, und dieses Stromobjekt ausfällt. Die Verbindung der Abhängigkeit wird von einem der beiden Objekte ausgehend in der Kategorie *Abhängigkeiten* erstellt, wird dann aber in beiden betreffenden Objekten in der gleichen Kategorie angezeigt. Man muss nur die Richtung entsprechend angeben.

Die abgebildeten Abhägigkeiten dienen als Grundlage für eine Reihe von möglichen Anwendungsbereichen. So sollen in Zukunft z.B. Ausfallszenarien und andere Simulationen nachgebildet werden können, die Aufschlüsse über mögliche Schwachstellen in der Infrastruktur liefern. Im Moment können die Abhägigkeiten beispielsweise dafür genutzt werden, sogenannte *host-* bzw. *service_dependencies* für Nagios zu definieren.

5 Workflows

Das Workflowsystem in i-doit ist ein erweiterbares Modul, welches anhand von so genannten *Workflowtypen* Workflows erzeugen kann. Somit lassen sich einmalige Aufträge, wie aber auch wiederkehrende Checklisten erstellen und an verantwortliche Personen und Objekte zuweisen. Involvierte Personen werden über statusbasierte Benachrichtigungen informiert. Dieser Mechanismus soll unter anderem die Dokumentation und ihre Planung bzw. Zurückverfolgung unterstützen.

Ein Workflow an sich beschreibt eine terminierte vom Benutzer auszuführende Aktion wie beispielsweise das Austauschen des Streamer-Mediums eines speziellen Backups Servers. Jegliche Workflow Zuweisung muss von den entsprechenden Personen akzeptiert werden. Nach erfolgreichem Abschluss kann ein Statusbericht eingereicht werden.

Workflowtypen Der Workflowtyp verhält sich wie eine Schablone und umfasst alle notwendigen Parameter, welche zur Ausführung notwendig sind. Diese Schablone lässt sich über eine Verwaltungsoberfläche bearbeiten. Über diese lassen sich ebenso auch neue Schablonen anlegen. Es stehen zunächst zwei vordefinierte Workflowtypen zur Verfügung:

- *Arbeitsauftrag* – Ein *Arbeitsauftrag* ist eine terminierte von wählbaren Benutzern auszuführende Aktion.
- *Checkliste* – Mit diesem speziellen Workflowtyp können täglich, wöchentlich oder jährlich wiederkehrende Aufträge generiert werden, indem zum reinen Startdatum eine periodische Zeitfolge angegeben und die entsprechende Option der Wiederholung ausgewählt wird. Diese wiederkehrenden Workflows lassen sich ebenfalls terminiert beenden.

6 Module/Schnittstellen

Der modulare Aufbau ist eine der herausragenden Eigenschaften von i-doit. Um das eigentliche Kernmodul CMDB herum sind eine Reihe weiterer Module angelehnt, die den Funktionsumfang von i-doit erweitern. Zum Standardumfang gehören beispielsweise das Workflowmodul aus Abschnitt 5 oder das LDAP-Modul aus Abschnitt 3.1, aber es existieren weitere Module, die dem Kern je nach Bedarf hinzugefügt werden können.

Nagios Eines dieser Module ist das *Nagios-Modul*, über das sich aus der im i-doit dokumentierten Infrastruktur eine fertige Konfiguration für *Nagios*[1] gewinnen lässt. Darüber hinaus können aus NDO-Datenbanken aktuelle und historische Statusdaten zu den Objekten angezeigt bzw. ins Logbuch eingetragen werden, womit die Statushistorie auch im Logbuch dokumentiert wird. Dazu muss allerdings gesagt werden, dass sich i-doit mit diesem Modul nicht als Konfigurationsoberfläche für Nagios versteht. Vielmehr soll aus einer einmalig erstellten Dokumentation mit den relevanten Informationen implizit eine entsprechende Konfiguration gewonnen und von den Details dabei abstrahiert werden.

Report Manager Ein weiteres Modul, das momentan angeboten wird, ist der *Report Manager* zur Erstellung von Reports basierend auf den Inhalten der CMDB.

Import-Schnittstelle Diese dient der automatischen Befüllung der Datenbank anhand von Bestandsdateien im XML-Format, die von *H-Inventory* erstellt wurden. Skripte, die den Bestand eines Systems erfassen, existieren momentan beispielsweise für Windows, Linux, FreeBSD, OpenBSD und Solaris.

[1] siehe www.nagios.org

Webservice-Schnittstelle Für die Zukunft ist eine allgemeine Webschnittstelle geplant, die für andere Anwendungen Informationen aus der CMDB von i-doit liefert. Als Protokoll soll dafür *SOAP* zum Einsatz kommen.

7 Sonstiges

7.1 Rechte

Das Rechtesystem der aktuellen Version von i-doit ist einfach aufgebaut. Innerhalb der Personen kann eine Autorisierung zur Anmeldung festgelegt werden und mit der Zuordnung zu einer der fünf fest vorgegebenen Rollen erlangt der Benutzer definierte Zugriffsrechte. Dabei unterscheidet i-doit die fünf folgenden Rollen:

- *Reader* – Ein Benutzer in dieser Rolle erhält ausschließlich lesende Berechtigung auf die i-doit Daten. Inhalte können eingesehen, aber nicht verändert werden. Auch die Anlage neuer Datensätze oder das Löschen sind nicht möglich.
- *Editor* – Der *Editor* darf bestehende Datensätze lesen und bearbeiten, aber nicht Anlegen oder Löschen.
- *Author* – Der *Author* darf im Gegensatz zum *Editor* auch neue Datensätze anlegen. Aber auch hier besteht kein Recht für das Löschen von Datensätzen.
- *Archivar* – Dieser darf neben den Rechten des *Authors* Datensätze archivieren und damit aus der normalen Ansicht entfernen. Innerhalb der Archivsicht können entsprechende Datensätze durch den Archivar gelöscht werden.
- *Admin* – Vereint alle Rechte der anderen Rollen in sich. Kann darüber hinaus verschiedene Systemeinstellungen ändern und ist in der Lage, gelöschte Datensätze endgültig aus der Datenbank zu entfernen (*Purge*).

7.2 Sprachunabhängigkeit

Um eine einfache Übersetzung der Oberfläche in andere Sprachen zu ermöglichen, greift i-doit auf einen Katalog von Sprachkonstanten zurück. Dies ermöglicht es, die Oberfläche in andere Sprachen zu übersetzen, ohne in den Quelltext einzugreifen. Fär jeglichen Text auf der Oberfläche wird ein Platzhalter definiert, der zur Laufzeit jeweils durch seine Entsprechung in der gewählten Sprache ersetzt wird.

Im Auslieferungszustand sind die Sprachen Deutsch und Englisch voll integriert, wobei weitere Sprachen über einen Sprachkonstanteneditor eingepflegt werden können.

8 Aussicht

Im Folgenden werden einige der zuküftigen Entwicklungen von i-doit dargestellt. Dabei handelt es sich sowohl um Erweiterungen des aktuellen Funtionsumfangs, als auch um Verbesserungen bereits bestehender Funktionen.

Visualisierung Zur besseren Veranschaulichung von bestimmten Konfigurationen ist es oft hilfreich eine visuell aufbereitete Form der Darstellung zu haben. Aktuell ist dies z.B. bei Schränken der Fall, bei denen die enthaltenen Geräte graphisch dargestellt werden und per Drag and Drop verschoben werden können. Geplant ist, dies vor allem auch bei Netzen zu ermöglichen, da eine Visualisierung an dieser Stelle eine häufig gefragte Funktion ist.

Simulationen Bei der Erfassung des Soll-Zustands einer Infrastruktur, stellt sich oft die Frage nach der Sicherheit und Robustheit der Konfiguration. Um Schwachstellen aufdecken zu können, sollen Simulationen z.B. von Ausfällen ermöglicht werden. Auf diese Weise kann ermittelt werden, welche Auswirkungen der Ausfall eines bestimmten Gerätes auf den Rest der Infrastruktur hat und welche Dienste in diesem Falle nicht mehr erbracht werden können.

Verfeinerung des Rechtesystems Die Rollen des aktuellen Rechtesystems fassen Rechte recht grob zusammen. So gibt es momentan keine Strukturen zur Verwaltung von Rechten auf einem einzelnen Objekt oder einer Kategorie. Ziel ist ein feingranulare Rechtevergabe zumindest auf Objektebene.

Gestaltung der Oberfläche i-doit stellt den Anspruch, eine intuitive Benutzerobefläche zu haben. Um diesem Anspruch noch gerechter zu werden, ist eine Umstrukturierung der Oberfläche angedacht. Der Benutzer soll mit weniger Klicks zum Ziel, ein Objekt zu erstellen oder zu verändern, kommen. In diesen Überlegungen spielen insbesondere *Templates* eine zentrale Rolle. Hiermit soll vor allem die Anlage von vielen Objekten vom gleichen Typ und fast gleicher Ausprägung erleichtert werden. So z.B. die Anlage von zehn Servern vom Typ *X*, die alle die gleiche Hardware besitzen.

Webschnittstelle Siehe hierzu Abschnitt 6.