Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

MIGRATE

Verbundprojekt MIGRATE!

Teilvorhaben:

Methoden und Werkzeuge für
energieeffiziente Anwenderinfrastrukturen

Schlussbericht

Zuwendungsempfänger:
Universität Stuttgart

31.03.2015

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Wirtschaft und Energie unter dem Förderkennzeichen 01ME11055 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Projektinformationen

Projektbezeichnung	MIGRATE!: Modelle, Verfahren und Werkzeuge für die Migration in Cloud-basierte energieoptimierte Anwenderinfrastrukturen und deren Management	
Förderkennzeichen	01ME11052 bis 58	
Förderzeitraum	01.10.2011–30.09.2014	
Koordinator	Universität Hohenheim, Stuttgart	
Projekt-URL	http://www.migrate-it2green.de	
Zuwendungsempfänger	Forschungszentrum FZID, Universität Hohenheim, Stuttgart	FZID
	Drees & Sommer Advanced Building Technologies GmbH, Stuttgart	DRESO
	IBM Deutschland Research & Development GmbH, Böblingen	IBM
	Brandenburgischer IT-Dienstleister, Potsdam	ZIT-BB
	Institut für Architektur von Anwendungssystemen, Universität Stuttgart, Stuttgart	IAAS
	Robert Bosch Gesellschaft für medizinische Forschung mbH, Stuttgart	RBK
	Flughafen Stuttgart GmbH, Stuttgart	FSG

Dokumentinformationen

Dokumentbezeichnung	Schlussbericht, Universität Stuttgart
Autoren	Florian Haupt Alexander Nowak Sebastian Wagner
Version	V1.0, 31.03.2015
Status	Final

Inhaltsverzeichnis

I. Übersicht 6
1. Aufgabenstellung 6
2. Voraussetzungen der Projektdurchführung 7
3. Planung und Ablauf des Vorhabens 8
4. Wissenschaftlicher und technischer Stand zu Projektstart 9
5. Zusammenarbeit mit anderen Stellen 11
II. Eingehende Darstellung 12
6. Verwendung der Zuwendung, Ergebnisse und Zielerreichung 12
a. Verwendung und erzielte Ergebnisse 12
1. AP1: Anforderungsanalyse 12
2. AP2: Cloud-basierte energieoptimierte Anwenderinfrastrukturen 22
3. AP3: Migrationsverfahren und -werkzeuge 24
4. AP4: Werkzeuge für energieeffizientes Cloud-Management 52
5. AP5: Modellbildung und Simulation 66
6. AP6: Piloterprobungen 67
7. AP7: Geschäftsmodelle und Dienstleistungen 71
b. Gegenüberstellung der vorgegebenen Ziele 72
7. Wichtigste Positionen des zahlenmäßigen Nachweises 73
8. Notwendigkeit und Angemessenheit der geleisteten Arbeit 73
9. Voraussichtlicher Nutzen 74
a. Wirtschaftlicher Nutzen 74
b. Wissenschaftlicher Nutzen 74
c. Anschlussfähigkeit 75
10. Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen 76
11. Veröffentlichungen der Projektergebnisse 77
a. Bereits erfolgte Veröffentlichungen 77
b. Geplante Veröffentlichungen 79
Literaturverzeichnis 80

Abbildungsverzeichnis

Abbildung 1: Projektablauf im Überblick ... 8
Abbildung 2: Arbeitspakete im Projekt ... 8
Abbildung 3 Unternehmensarchitektur als Schichtenmodell ... 13
Abbildung 4 Beispiel zur Beschreibung der geschäftlichen Schicht ... 18
Abbildung 5 Vorgehensmodell zur Identifikation der vorhandenen Anwenderinfrastruktur ... 19
Abbildung 6 Client-Server-Architektur Maerker ... 20
Abbildung 7 Policy-Erweiterungen der implementierten TOSCA IAs ... 23
Abbildung 8 Phasen des Vorgehensmodells ... 26
Abbildung 9 Phasen des Vorgehensmodells, Ein- und Ausgaben ... 26
Abbildung 10 Phase 1, Übersicht ... 27
Abbildung 11 Portfolio als Entscheidungshilfe ... 28
Abbildung 12 Phase 2, Übersicht ... 29
Abbildung 13 Erstellung von Migrationsszenarien ... 30
Abbildung 14 Auswahl von Migrationsszenarien ... 30
Abbildung 15 Phase 3, Übersicht ... 32
Abbildung 16 Phase 4, Übersicht ... 33
Abbildung 17 Startbildschirm Entscheidungsunterstützungssystem ... 37
Abbildung 18 GUI Mockup Phase 1, Erfassung der AIS Komponenten ... 38
Abbildung 19 GUI-Mockup, Projektverwaltung ... 39
Abbildung 20 GUI-Mockup, Stammdatenverwaltung ... 39
Abbildung 21 Beispielprozess „Urlaubsantrag“ im BPEL Designer ... 40
Abbildung 22 Editor für Softwarekomponenten der logischen Schicht ... 41
Abbildung 23 Formeleditor ... 42
Abbildung 24 Editor für Hardwarekomponenten der physischen Schicht ... 42
Abbildung 25 Cloud Migration unterstützt durch TOSCA ... 43
Abbildung 26 Cloud Service Archive (CSAR) ... 45
Abbildung 27 Lego4TOSCA Properties ... 48
Abbildung 28 Architektur Lego4TOSCA ImplementationArtifact ... 50
Abbildung 29 Architektur OpenTOSCA ... 51
Abbildung 30 Policy Taxonomie für "grüne" Cloud Services ... 54
Abbildung 31 Policy Taxonomie für TOSCA Cloud Services ... 55
Abbildung 32 Lebenszyklus eines Cloud Services und seiner Policies ... 58
Abbildung 33 Architektur des OpenTOSCA Containers ... 59
Abbildung 34 Policy Umsetzung in OpenTOSCA, IA basierter Ansatz ... 61
Abbildung 35 Komponenten für die Implementierung der Green Policy ... 63
Abbildung 36 Durchschnittliches Lastprofil Maerker Anwendung nach Stunden ... 66
Abbildung 37 Maerker Anwendungstopologie ... 68
Abbildung 38 Maerker Build Plan ... 69
Abbildung 39 Maerker Termination Plan ... 70
Abbildung 40: TOSCA Ökosystem ... 75

Tabellenverzeichnis

Tabelle 1: Aufgaben des IAAS im Projekt 9
Tabelle 2: Geschäftliche Sicht - Objekttypen 15
Tabelle 3 Geschäftliche Sicht - Ereignisse 15
Tabelle 4: Geschäftliche Sicht - Beziehungstypen 16
Tabelle 5: Geschäftliche Sicht - Gateways 16
Tabelle 6: Geschäftliche Sicht - Gruppierungen 17
Tabelle 7: Bewertung der untersuchten Policy Sprachen 22
Tabelle 8: Zuordnung von TOSCA-Werkzeugen zu Rollen 71
Tabelle 9: Zielsetzung und Zielerreichung 72
Tabelle 10: Positionen des zahlenmäßigen Nachweises 73

I. Übersicht

1. Aufgabenstellung

Ausgangspunkt für die Forschungsarbeiten des IAAS war die These, dass geeignete Methoden und Werkzeuge den Aufbau und Betrieb energieeffizienter Anwenderinfrastrukturen ermöglichen. Der Forschungsgegenstand war deshalb die IKT-Energieeffizienz insbesondere in kleinen und mittelständischen Betrieben. Der IKT-Einsatz in diesen Unternehmen ist durch eine rasch zunehmende Zahl immer leistungsfähigerer IKT-Systeme gekennzeichnet, entsprechend steigt auch der IKT-induzierte Energiebedarf rasch an und ist dabei abhängig von individuellem Bewusstsein, Prozessen, Demographie, Sozialsystem, Kompetenz, Organisation sowie dem umgebenden Wertschöpfungssystem. Die zuständigen Entscheidungsträger optimieren ihre Prozesse sowie den zugehörigen IKT-Einsatz häufig in lokaler Autonomie und auf Basis klassischer Optimierungsindikatoren. Der Energieverbrauch spielt dabei oft nur eine untergeordnete Rolle. Dies lässt sich auch als Reflexion derzeitig verfügbarer Ansätze ableiten, welche hierfür technologisch bisher keine oder kaum Unterstützung anbieten. Diese Problemstellung erforderte sowohl eine umfassende Betrachtung der zugrundeliegenden Geschäftsprozesse eines Unternehmens als auch die adäquate Anpassung der zugrundeliegenden Technologien, Methoden und Infrastrukturen in Verbindung mit entsprechenden Managementmethoden und -tools.

Der Lösungsansatz des IAAS beruhte auf der Bereitstellung eines übergreifenden Ansatzes zur ganzheitlichen Verbesserung der Energieeffizienz auf Basis umfassender Technologielösungen, welche sowohl die Optimierung bisher verwendeten Infrastrukturen als auch die Optimierung organisatorischen Abläufe einer Organisation betreffen. Entsprechend der Methodik eines Green Business Process Reengineerings mussten dazu die Anforderungen der Zielgruppen auf die Planung, den Entwurf und die Entwicklung von Technologien und Methoden übertragen werden, um neuartige Infrastrukturen, erweitertes Prozessmanagement und Energieeffizienz zu vereinen. Dies wurde insbesondere durch die Analyse und Anpassung organisatorischer Strukturen sowie die Migration der vorhandenen Infrastrukturen in neue, geeignete Infrastrukturen (wie bspw. Cloud Computing) in Verbindung mit geeigneten Managementstrukturen erreicht. Die Adaption von organisatorischen Strukturen umfasst zusätzlich eine energieeffiziente Verteilung der Prozess- und Workflowtopologie auf Basis von ökologischen und ökonomischen Faktoren. Die zielgerichtete Anpassung der bisherigen Technologien und Methoden (inklusive der Berücksichtigung von Standards) war hierbei eine Voraussetzung eines erfolgreichen Lösungsansatzes.

Die Umsetzung und Ausarbeitung der Lösungsansätze diente deshalb der Erreichung mehreren Ziele: (1) Nutzung von Cloud Infrastrukturen zur Verbesserung der Energieeffizienz von IKT unter Berücksichtigung nutzerspezifischer Anforderungen sowie zielgerichteter Integration neuer Infrastrukturen, (2) Identifikation eines umfassenden Konzepts (Technologien, Methoden und Werkzeuge) zur ganzheitlichen Verbesserung der Energieeffizienz inklusive der Identifikation von Patterns zur Unterstützung des Adaptionsprozesses auf „grüne“ Organisationsstrukturen, und (3) Werkzeuge für ein intelligentes Management von energieeffizienten Infrastrukturen zur Unterstützung verschiedener Anforderungsprofile (insb. aus Unternehmen und Gebäudetechnik). Diese Ziele sollen in Zusammenarbeit mit dem Konsortium die Bereitstellung einer energieoptimierten Anwenderinfrastruktur ermöglichen.

2. Voraussetzungen der Projektdurchführung

Das Institut für Architektur von Anwendungssystemen (IAAS) der Universität Stuttgart verfügt neben Kompetenzen im Bereich serviceorientierter Architekturen und Workflows über ein umfangreiches Methodenrepertoire zur Bereitstellung von Cloud Computing Infrastrukturen, sowie um realweltliche Anwenderinfrastrukturen zu modellieren und anschließend automatisiert und individuell konfiguriert in Clouds zu provisionieren. Das IAAS ist mit seinen Forschungsansätzen schon bisher international sehr erfolgreich in der anwendungsorientierten FuE positioniert. Durch das Projekt 4CaaST, gefördert durch das 7. Rahmenwerk der Europäischen Union, verfügt das IAAS beispielsweise über Erfahrungen für das Design, den Betrieb, das Management und den Austausch von Cloud Services und Service-Kompositionen. Die Arbeiten im BMWi Projekt CLOUDCYCLE (01MD11023) ermöglichen Cloud-Betreibern, Dienste bei garantierter Sicherheit und Compliance (d.h. Einhaltung von Gesetzen, Richtlinien und Datenschutzvorgaben) kosteneffizient und skalierbar für Kunden des Mittelstands und der öffentlichen Verwaltung bereitzustellen. Darüber hinaus bestehen durch die Arbeiten im Exzellenzcluster SimTech (EXC 310/1) weitere Erfahrungen in der Bereitstellung von flexiblen Cloud Computing Infrastrukturen und Diensten für die Ausführung von wissenschaftlichen Workflows.

Wichtige Vorarbeiten für MIGRATE! wurden in verschiedenen Bereichen geleistet. Zum einen hatte das IAAS bereits verschiedene Migrationsszenarien durchgeführt. Im Rahmen des Projektes CIMS wurde eine „Informatik Cloud“ entwickelt. Für die Realisierung wurden unterschiedliche Cloud Systeme integriert: Amazon EC2, Eucalyptus und IBM TSAM. In weiteren Projekten wie Sametime3D konnte das IAAS in Zusammenarbeit mit IBM zusätzliche praxisrelevante Erfahrungen bei der Migration des Instant Messaging Clients IBM Lotus Sametime in eine Cloud-Umgebung erlangen. Auch hatte das IAAS im Hinblick auf das Methodenrepertoire, um realweltliche Anwenderinfrastrukturen zu modellieren und anschließend automatisiert und individuell konfiguriert in Clouds zu provisionieren bereits Arbeiten geleistet. Das Cafe-Framework verfügt beispielsweise über Werkzeuge, die Entwickler bei der Softwareentwicklung unterstützen. Dies beinhaltet Werkzeuge (1) zur Modellierung von Anwendungen und deren SLAs, (2) zur Generierung von Workflows und (3) Provisionierungstools, die Anwendungen automatisch in einer Cloud aufsetzen. Mit Enterprise Application Integration as a Service verfügt das IAAS zudem über Methoden und Werkzeuge, Integrationsszenarien mittels Pattern zu modellieren und zu provisionieren.

Auf der anderen Seite verfügt das IAAS, über die Realisierung von Cloud Umgebungen hinaus (Leymann, 2009), über langjährige Erfahrungen im Bereich des Business Process Reengineerings (BPR) im Allgemeinen, sowie erste Erfahrungen in der Anwendung von ökologischen Performance Indikatoren in diesem Bereich. Es verfügt zudem über umfangreiche Erfahrungen und Methodenwissen bezüglich der Überwachung und Analyse von Geschäftsprozessen. Die Arbeit von (Wetzstein, Strauch and Leymann, 2009) zeigt einen Ansatz zur Überwachung von kritischen Performanceindikatoren in automatisierten Geschäftsprozessen. In (Wetzstein et al., 2009) wird darüber hinaus untersucht, wie diese Indikatoren mit Hilfe von Data Mining Techniken (insb. Entscheidungsbäume) analysiert werden können. Ein weiterer Schwerpunkt des IAAS liegt in der Erforschung der Regelkonformität von Geschäftsprozessen (Compliance Management). Auf diesem Gebiet war das IAAS an zwei europäischen Forschungsprojekten beteiligt, welche sowohl die Sicherheit von IT Infrastrukturen zur Laufzeit betrachten (MASTER), als auch modellgetriebene Verfahren zur Regelkonformität in dienst-orientierten Architekturen untersuchen (COMPAS). Die Forschung auf diesem Gebiet hat gezeigt, dass es zahlreiche verschieden Quellen von Anforderungen gibt, sogenannte Compliance Sources. Dazu gehören neben gesetzlichen Vorschriften auch unternehmensinterne Richtlinien, ebenso wie Anforderungen zur Umsetzung einer Green IT (Schleicher et al., 2009). Zur Unterstützung des Compliance

Managements in Geschäftsprozessen wurden sowohl Techniken zur regelkonformen Prozessmodellierung auf Basis konformer Prozessvorlagen (Schleicher et al., 2009), als auch Techniken zur Prozessanalyse auf Basis von Sichten auf Prozesse (Schumm, Leymann and Streule, 2010) entwickelt. Darüber hinaus wurden Architekturen zur Realisierung und Unterstützung von Compliance Management im Bereich des Cloud Computing entworfen (Brandic et al., 2010).

3. Planung und Ablauf des Vorhabens

Im Gesamtvorhaben erfolgte die Entwicklung der Modelle und Verfahren sowie deren Evaluation anhand von vier Migrationsszenarien gemäß eines Engineering-Prozesses, der von der Anforderungs- und Szenariodefinition unter Einbeziehung der IT-Anwender, dem Entwurf der fachlichen Lösungen und der gemeinsamen Softwareinfrastruktur über die prototypische Realisierung und Integration bis zur Evaluation im Labor oder Feld reichte. Der Projektablauf ist anhand der wichtigsten Meilensteine in Abbildung 1 dargestellt.

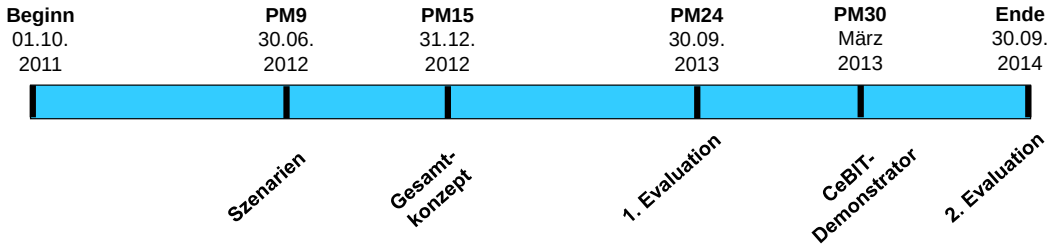


Abbildung 1: Projektablauf im Überblick

Die Projektarbeiten untergliederten sich in acht Arbeitspakete. Die Abbildung 2 zeigt den inhaltlichen Zusammenhang der Arbeitspakete: Die Erhebung der Anforderungen erfolgte im AP1 einschließlich der Definition einer einheitlichen Modellierungssprache für die IT-Anwenderinfrastrukturen. Diese waren Grundlage für die vier methodischen Säulen und Hauptarbeitsgebiete des Gesamtvorhabens – Cloud-basierte Anwenderinfrastrukturen (in AP2), Cloud-Migration (in AP3), Cloud-Management (in AP4) und Modellbildung & Simulation (in AP5). Die gemeinsame Evaluation erfolgte durch Piloterprobungen (AP6). Begleitend zur technischen Entwicklungen wurden die Untersuchungen zu Geschäftsmodellen (AP7) durchgeführt und das Projekt geplant und gesteuert (AP8).

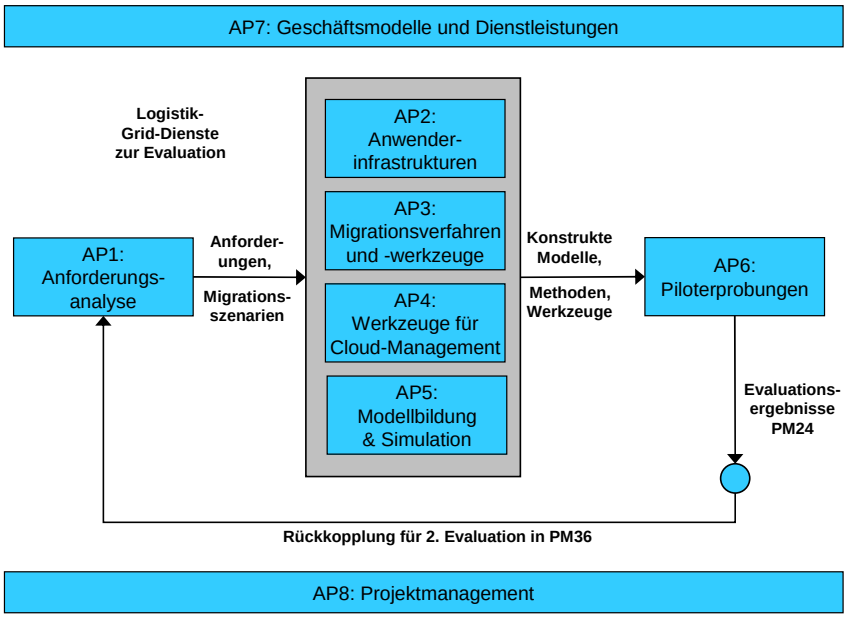


Abbildung 2: Arbeitspakete im Projekt

Der inhaltliche Arbeitsschwerpunkt des IAAS lag in der Analyse energieeffizienter Anwenderinfrastrukturen (AP2), der Bereitstellung von Migrationsverfahren- und Werkzeugen (AP3) sowie der Spezifikation und Implementierung von Cloud Management Werkzeugen (AP4).

Die Aufgaben verteilten sich auf die Arbeitspakete wie in der Tabelle 1 dargestellt.

Arbeitspaket	**Aufgaben des IAAS**
AP1: Anforderungsanalyse	- Begleitung der Anforderungserhebung bei den Anwendungspartnern zur Identifizierung der unterschiedlichen Kundenanforderungen bezüglich Systemumgebung, insbesondere hinsichtlich Verfügbarkeit während der Migration, Skalierbarkeit, Ausfallsicherheit und Datenmigration. - Definition einer Methode für die Beschreibung derzeitiger Anwenderinfrastrukturen
AP2: Cloud-basierte energieoptimierte Anwenderinfrastrukturen	- Entwicklung eines Ansatzes zur Ermittlung des Energiebedarfs von Geschäftsprozessen auf Basis der unterliegenden Infrastruktur/Ressourcen - Identifikation von Green Business Process Patterns zur energieeffizienten Gestaltung von Geschäftsprozessen - Erarbeitung eines Konzepts zur Definition von Policies innerhalb von TOSCA
AP3: Migrationsverfahren und –werkzeuge	- Erarbeitung eines Vorgehensmodell für die Cloud Migration - Implementierung eines Werkzeug zur Unterstützung der Migrationsmethode (OpenTOSCA) - Konzeption und Realisierung von Software Komponenten zur Nutzung in OpenTOSCA
AP4: Werkzeuge für energieeffizientes Cloud-Management	- Erweiterung des TOSCA Containers um Policies - Erweiterung der Anwendungskomponenten hinsichtlich der Realisierung von Green Policies
AP5: Modellbildung und Simulation	- Unterstützung bei der Simulation von Anwendungslasten - Integration der Modelle in OpenTOSCA
AP6: Piloterprobungen	- Piloterprobung der Maerker Anwendung beim Partner ZIT-BB (Definition des TOSCA-Templates sowie Ausführung und energetische Steuerung)
AP7: Geschäftsmodelle und Dienstleistungen	- Einbringung der Erfahrungen des Modell- und Softwareentwicklungsprozesses

Tabelle 1: Aufgaben des IAAS im Projekt

4. Wissenschaftlicher und technischer Stand zu Projektstart

Energieeffizienz in der IT

Vor Projektstart konzentrierten sich Anstrengungen zur Energieeinsparung weitgehend auf die Verbesserung der Energieeffizienz von Einzelkomponenten und auf die Optimierung von Rechenzentren. So konnte die Energieeffizienz von Prozessoren, Speichergeräten, Netzwerktechnik, Stromversorgung und Kühlung erheblich verbessert werden (z.B. durch Measurement and Management Technology von IBM, Insight Control Environment (ICE) von HP, „Best Practices for Creating The Green Data Center“ von Siemens IT Solutions). Erste Studien und Arbeiten zeigten, dass enorme Einsparpotentiale durch veränderte IKT-Architekturen

(Fraunhofer, 2008) und Anpassungen von Prozessabläufen möglich sind. Eine integrative, energieoptimierte Betrachtung von Organisationsstrukturen in Unternehmen und Gebäudekomplexen fanden bis dato jedoch nicht statt.

Cloud Computing

Ein erster vielversprechender Ansatz neuer IKT-Architekturen stellten Cloud-basierte Anwenderinfrastrukturen und energieoptimierte Clouds als Gesamtkonzept dar. Dazu wurden bereits Technologien wie Virtualisierung (z.B. Xen, VMware, Microsoft Hyper-V), Provisionierung (z.B. IBM TPM, Sun N1 Provisioning System), Cloud-Infrastruktur-Management (z.B. IBM TSAM oder Eucalyptus) und Service-Orientierung zur Energieeinsparung eingesetzt (Leymann, 2009). Virtualisierung (Beloglazov and Buyya, 2010) (Dhiman, Marchetti and Rosing, 2009) als Cloud-Basistechnologie stellte dabei den hauptsächlichen Ansatzpunkt zur Energieoptimierung dar. Das Zusammenspiel dieser Komponenten ermöglicht das dynamische und automatische Aufsetzen, Abschalten und Managen von virtuellen Servern. Das IBM China Research Lab (Liu et al., 2009) zeigte mit seiner GreenCloud Architektur erste Optimierungsmöglichkeiten auf dieser Ebene. Allerdings umfasst dies leider keinen vollständigen, übergreifenden Ansatz, denn die Energieeffizienz in Cloud-basierten Anwenderinfrastrukturen wurde häufig nur auf der Ebene von Scheduling Mechanismen betrachtet, bspw. der energiesparenden Platzierung von Virtuellen Maschinen in der Cloud (Li et al., 2009). Weitere Verbesserungsansätze fanden sich im Bereich von Data Centers, Netzwerken und Protokolle sowie Applikationen statt (Choon and Zomaya, 2010). Jedoch bleiben auch hierbei die Geschäftsprozesse und das Nutzerverhalten unberücksichtigt.

Cloud Migration

Die Migration in Cloud-basierte Infrastrukturen wurde vorwiegend ökonomisch motiviert. So konnte gezeigt werden, dass Cloud-Computing in vielen Fällen die Betriebskosten reduziert und die Flexibilität erhöht (Armbrust et al., 2009) (Motahari-Nezhad, Stephenson and Singhal, 2009) (Walker, 2009). Dementsprechend wurden bereits rechtliche Aspekte des Migrationsprozesses (insb. Software-Lizenzrechtliche Fragestellungen) (Cloud Security Alliance, 2009) (Catteddu and Hogben, 2010), die geografische Verteilung von Cloud-Ressourcen (Jaeger et al., 2009) (Joint, Baker and Eccles, 2009) und technische Sicherheitsrisiken (Datensicherheit, Ausfallsicherheit etc.) (Jensen et al., 2009) untersucht. Einflüsse der Migration auf Ebene der Organisation und der Prozesslandschaft wurden bis dato nicht bzw. nur unzureichend betrachtet. Ebenfalls fehlten organisationsübergreifende Ansätze, die für energieeffizientes Cloud-Management unverzichtbar sind. Die wenigen bestehenden Arbeiten fokussierten die Zielinfrastruktur, nicht jedoch die Migration selbst (Khajeh-Hosseini, Greenwood and Sommerville, 2009). Weitere Arbeiten nannten zwar Hindernisse und Integrationsfelder, zeigten aber keine Lösungswege auf (Armbrust et al., 2009) (BITKOM, 2009). Bis dato nicht untersucht wurde die Energieeffizienz vor, während und nach einer Migration. Aus diesem Grund konnte weder die Auswirkung der Migration auf den Gesamtenergiebedarf verlässlich vorhergesagt werden noch konnte eine sinnvolle Auswahl und Priorisierung zu migrierender Anwenderinfrastrukturen getroffen werden.

Cloud Werkzeuge

Neben den theoretischen Ansätzen der Energieoptimierung gab es existierende Werkzeuge zur Unterstützung einzelner Optimierungsaspekte. Diese wiesen jedoch durch einen fehlenden ganzheitlichen Ansatz die gleichen Defizite wie die theoretischen Grundlagen auf. Werkzeuge zur Simulation von IKT-Infrastrukturen bspw. berücksichtigen die optimierte Kühlung von Rechenzentren, differenzierte Lastprofile, etc.; es fehlte jedoch ein übergreifender Simulationsansatz unter Berücksichtigung von IKT (Rechenzentrums- und Anwenderseitig), Geschäftsprozessen und Endanwendern mit unterschiedlichem energieverbrauchsrelevantem Verhalten. Eine Integration in Planung und Management der gesamten Anwenderinfrastruktur

oder gar zur geschäftsprozessübergreifenden Abstimmung war nicht möglich. Existierende Cloud-Management-Werkzeuge (z.B. Amazon CloudWatch, IBM Tivoli, HP OpenView; für Clouds mit eigenen und fremden Ressourcen: RightScale, Kaavo, Zeus, Scalr und Morph) boten die dynamische Bereitstellung von Ressourcen wie Server, Speicherplatz oder Anwendungen sowie das Entfernen nicht mehr benötigter Ressourcen an. Einige Werkzeuge konnten Cloud-basierte Server lediglich wie physische Server verwalten, d.h. weitere Ressourcen nicht dynamisch bereitstellen. Für eine Optimierung der Energieeffizienz der gesamten Anwenderinfrastruktur fehlten aktuellen Cloud-Management-Werkzeugen grundlegende Metriken zur Abbildung und Optimierung der Energieeffizienzparameter. Insbesondere unter Betrachtung der gesamten für den Betrieb benötigten Infrastruktur auf Anbieter- sowie Nutzerseite war eine grundlegende Forschungslücke. Für die Migration von Anwendungen in die Cloud existierten bereits replizierbare Verfahren mit Werkzeug-Unterstützung (z.B. www.cloudswitch.com als rudimentärer Ansatz zur Migration von Virtuellen Maschinen). Ferner boten Hersteller und Verbände auch bereits Whitepaper und Leitfäden für Migrationsvorhaben an (BITKOM, 2009) (HyperStratus, 2009). Auch hier standen Kosten und Technik im Vordergrund; Prozesse, Organisation, Endbenutzerverhalten und rechtliche Aspekte werden nur ansatzweise, energetische Auswirkungen überhaupt nicht betrachtet.

Für die Ermittlung des Grades der Energieeffizienz bestanden bislang keine brauchbaren Ansätze, da sich bestehende Werkzeuge auf eine geringe Menge vordefinierter Metriken (z.B. Prozessorauslastung, Festplattenaktivität) beschränken. Diese Metriken boten allenfalls indirekte Ansatzpunkte zur Steigerung der Energieeffizienz und berücksichtigen diverse „grüne“ Kennzahlen nicht. So ließ sich bspw. die Kennzahl Performance pro Watt einzelner Ressourcen (z.B. im SPECpower-Benchmark) mit existierenden Werkzeugen nicht ermitteln. Für eine umfassende Ende-zu-Ende-Optimierung sind neben physischen Servern zwar weitere Verbraucher (z.B. Kühlanlage) mit einzubeziehen, fanden in den Werkzeugen derzeit jedoch keinerlei Beachtung. Der für eine Gesamtschau relevante Energieverbrauch vielfältiger teils mobiler Nutzer-Endgeräte blieb ebenfalls unberücksichtigt.

5. Zusammenarbeit mit anderen Stellen

Das IAAS arbeitete im Bereich der Entwicklung von TOSCA zur Beschreibung von Cloud Anwendungen sowie bei der Realisierung einer entsprechenden Ausführungsumgebung eng mit dem Projekt CloudCycle (Trusted Cloud, 01MD11023) zusammen. Als Ergebnis konnte eine Laufzeitumgebung (OpenTOSCA) realisiert werden, welche sowohl sicherheitsrelevante als auch ökologische Anforderungen abdecken kann.

Darüber hinaus wurde auch die Integration und Realisierung von verschiedenen Policies der gegebenen Anwendungsbereiche umgesetzt werden.

II. Eingehende Darstellung

6. Verwendung der Zuwendung, Ergebnisse und Zielerreichung

a. Verwendung und erzielte Ergebnisse

Für die Details der Verwendung der Zuwendung und der erzielten Ergebnisse wird im Folgenden auf die einzelnen Arbeitspakete eingegangen.

1. AP1: Anforderungsanalyse

AP1	Anforderungsanalyse	Leitung: IAAS
Ziele (IAAS)	– Analyse derzeitiger Anwenderinfrastrukturen – Erhebung von Anforderungen an Migrationsprozesse (Cloud Migration)	
Ergebnisse (IAAS)	– Methode zur Beschreibung von Anwenderinfrastrukturen – Vorgehensmodell zur Erfassung von – Modelle von Anwenderinfrastrukturen – Anforderungen, insb. im Bereich TOSCA	

AP1.1 Analyse der derzeitigen Anwenderinfrastrukturen

Entwurf einer Methode zur Beschreibung von Anwenderinfrastrukturen

Für die Beschreibung der Anwenderinfrastrukturen wurde zusammen mit dem FZID eine 3-stufige Methode entwickelt. Die Methode basiert auf dem in Abbildung 3 dargestellten mehrschichtigen Modell einer Unternehmensarchitektur. Die einzelnen Schichten werden im Folgenden kurz vorgestellt und ihre Zusammenhänge erläutert.

Die *Geschäftliche Schicht* beschreibt Geschäftsprozesse, deren Aktivitäten durch die in der Logischen Schicht modellierten Softwarekomponenten realisiert werden. Ein Geschäftsprozess besteht dabei im Wesentlichen aus Aktivitäten sowie den Kontroll- oder Datenflussabhängigkeiten zwischen diesen. Er stellt eine strukturierte Beschreibung einer potentiell komplexen Arbeitsaufgabe dar. Der durch einen Geschäftsprozess beschriebenen Ablauforganisation kann immer auch eine Aufbauorganisation zugeordnet werden.

Die *Logische Schicht* beschreibt die auf der Hardware-Infrastruktur betriebenen Softwarekomponenten. Die Beschreibung umfasst zum einen eine Klassifikation der einzelnen Komponenten, beispielsweise als Betriebssystem oder Middleware Komponenten. Zum anderen werden die Beziehungen zwischen den einzelnen Komponenten modelliert wodurch die Erfassung komplexer Anwendungstopologien ermöglicht wird.

Die *Physische Schicht* beschreibt den Aufbau der Hardware-Infrastruktur. Erfasst werden sowohl die vorhandenen Hardwareressourcen als auch die funktionalen Beziehungen zwischen diesen. Relevante Hardwareressourcen sind sowohl die eigentlichen Rechenressourcen, also Server, als auch die zum Betrieb der Server notwendige Infrastruktur wie beispielsweise Netzwerkgeräte, Unterbrechungsfreie Stromversorgung oder Kühlung.

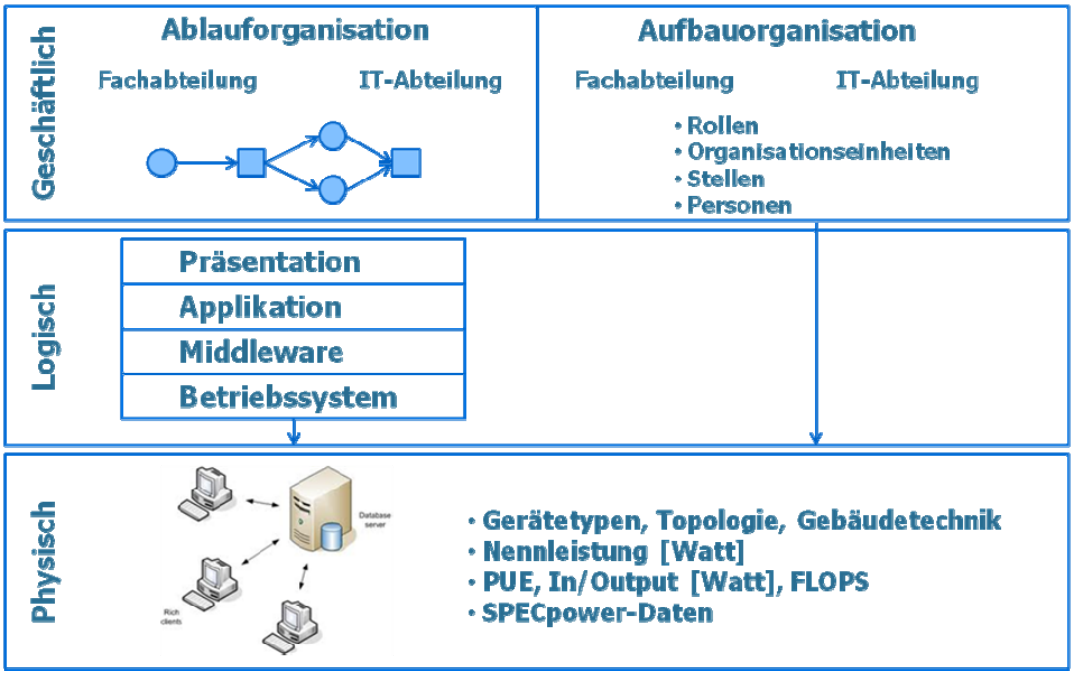


Abbildung 3 Unternehmensarchitektur als Schichtenmodell

Elektrische Energie wird ausschließlich durch Geräte der physischen Schicht verbraucht. Daraus ergibt sich die Notwendigkeit der Abbildung der relevanten Hardwarekomponenten sowie des durch sie verursachten Energieverbrauchs. Eine mögliche Migration bezieht sich dagegen immer auf die logische Schicht. Softwarekomponenten können von ihrer bestehenden Infrastruktur in eine Cloud Infrastruktur migriert werden. Um Migrationsszenarien zu untersuchen ist es damit notwendig, die Logische Schicht ebenfalls zu modellieren. Eine ganzheitliche Betrachtung der Nutzung von IKT Ressourcen wird erst durch die Geschäftliche Schicht ermöglicht. Sie beschreibt die Nutzung von Softwarekomponenten durch Aktivitäten der Geschäftsprozesse.

Eine weiterer Grund für den Entwurf eines mehrschichtigen Modells liegt in der Notwendigkeit, für jede Schicht spezifische Konstrukte vorzusehen. So sind bereits für jede Schicht Methoden, zum Teil standardisiert, verfügbar. Bei Wiederverwendung dieser Methoden verbleibt nur noch die Aufgabe, die Konstrukte schichtenübergreifend in einem Metamodell zu integrieren.

Die einzelnen Schichten der Anwenderinfrastruktur werden zunächst separat modelliert und anschließend zu einem Gesamtmodell zusammengefasst. Aktivitäten der Geschäftlichen Schicht können durch Softwarekomponenten der Logischen Schicht implementiert werden. Dies wird durch Beziehungen zwischen der Geschäftlichen und der Logischen Schicht erreicht. Softwarekomponenten der Logischen Schicht werden wiederum auf Hardware-komponenten der Physischen Schicht ausgeführt bzw. nutzen diese. Demzufolge sind aus der Logischen Schicht heraus Beziehungen zu Komponenten der Physischen Schicht zu definieren.

Die Verknüpfung der Einzelmodelle zu einem Gesamtmodell ermöglicht die ganzheitliche Analyse verschiedener in MIGRATE! relevanter Szenarien. Die Planung einer Migration unter Beachtung energierelevanter Kriterien ist nur dann sinnvoll möglich, wenn entsprechende energierelevante Metriken in der Logischen Schicht verfügbar sind. Der in der Physischen Ebene erfasste Stromverbrauch kann beispielsweise über entsprechend modellierte Beziehungen in die Logisch Ebene propagiert werden um so den Energieverbrauch einzelne Softwarekomponenten zu bestimmen oder zumindest anzunähern. Ausgehend davon kann der Energieverbrauch weiter in die Geschäftliche Ebene propagiert

werden um daraus den Energieverbrauch ganzer Geschäftsprozesse zu bestimmen oder anzunähern.

Wird eine Migration durchgeführt impliziert dies Änderungen in der Logischen Schicht. Diese Änderungen können Auswirkungen auf die Auslastung der darunterliegenden Hardwareressourcen und auf damit verbundene energierelevante Metriken haben. Das vorgestellte mehrschichtige Modell unterstützt die Propagierung von Änderungen innerhalb einer Schicht in weitere Schichten des Modells und ermöglicht damit globale Analysen lokaler Änderungen.

Spezifikation der Methode zur Beschreibung der geschäftlichen Schicht

Die geschäftliche Sicht modelliert die Ablauforganisation, die durch ein oder mehrere interagierende Prozesse realisiert wird (Kollaboration). Die Prozesse modellieren alle Aktivitäten, sowie Ereignisse des zu erfassenden Geschäftsszenarios und deren zeitliche Ausführungsreihenfolge. Des Weiteren werden auch die Rollen bzw. Teilnehmer modelliert, die an der Ausführung des Geschäftsszenarios beteiligt sind. Zur Visualisierung der geschäftlichen Sicht wird eine Teilmenge der BPMN Notation verwendet, deren Semantik in den folgenden Abschnitten beschrieben wird.

Die Aktivitäten der geschäftlichen Schicht können entweder mit oder ohne IT Unterstützung ausgeführt werden. Um den Ressourcen- und Energieverbrauch der Aktivitäten zu bestimmen, werden die Aktivitäten, die mittels IT Unterstützung ausgeführt werden, mit den Komponenten logischen Schicht assoziiert werden. So kann beispielsweise der Energiebedarf einer Aktivität bestimmt werden, die mit Hilfe von einem oder mehreren Softwareartefakten aus der logischen Schicht ausgeführt wird.

Die felgenden Tabellen bescheiben die zur Beschreibung der geschäftlichen Sicht definierten Objekte.

Tabelle 2: Geschäftliche Sicht - Objekttypen

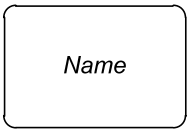
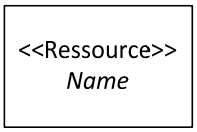
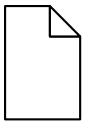
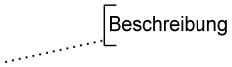
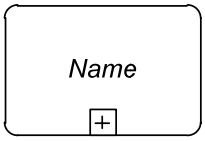
Objektname	Objektbeschreibung	Visuelle Darstellung
Aktivität	Eine Aktivität ist ein Arbeitsschritt innerhalb eines Geschäftsprozesses, der entweder manuell oder aber auch teil- bzw. vollautomatisiert ausgeführt werden kann.	
Ressourcen	ist ein Objekt, welches die von einer Aktivität verwendet Ressourcen repräsentiert	
Datenobjekt	Ein Objekt, welches Informationen enthält, die während der Ausführung eines Prozesses von Aktivitäten erzeugt, manipuliert oder verwendet werden (z.B. ein Dokument). Datenobjekte können auch Informationen repräsentieren, die nicht in elektronischer Form vorliegen, wie z.B. einen Brief.	
Text Annotation	Eine Text Annotation dient zur textuellen Beschreibung eines Objekts der geschäftlichen Schicht.	
Zugeklappter Teilprozess	Ein Teilprozess ist eine Aktivität innerhalb eines Prozesses, weitere Aktivitäten beinhaltet, die aus unterschiedlichen Gründen nicht dargestellt werden sollen (z.B. um die Komplexität zu minimieren).	

Tabelle 3 Geschäftliche Sicht - Ereignisse

Objektname	Objektbeschreibung	Visuelle Darstellung
Start	Definiert den Startpunkt eines Prozesses.	
Ende	Definiert den Endpunkt eines Prozesses.	
Nachricht	Markiert den Nachrichtenempfang bzw. den Nachrichtenversand innerhalb eines Prozesses.	
Zeitliches Ereignis	Ist ein Objekt, welches auf ein bestimmtes zeitliches Ereignis reagiert, z.B. kann mit diesem Ereignis der Prozessablauf für eine bestimmte Zeit pausiert werden.	

Tabelle 4: Geschäftliche Sicht - Beziehungstypen

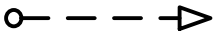
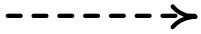
Objektname	Objektbeschreibung	Visuelle Darstellung
Kontrollflusskante	Die Kontrollflusskante bestimmt die Ausführungsreihenfolge von Objekten wie Aktivitäten und Ereignisse.	
Nachrichtenfluss	Der Nachrichtenfluss kennzeichnet den Informationsaustausch zwischen zwei Pools.	
Assoziation	Mit einer Assoziation werden Prozesse, Ereignisse oder Aktivitäten mit einem Datenobjekt verbunden.	

Tabelle 5: Geschäftliche Sicht - Gateways

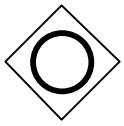
Objektname	Objektbeschreibung	Visuelle Darstellung
Exklusives Gateway	Verzweigung: Abhängig von den Kantenbedingungen genau eine ausgehende Kante aktiviert. Synchronisation: Der Kontrollfluss wartet am Gateway, bis eine eingehende Kante aktiviert wird.	
Paralleles Gateway	Verzweigung: Es werden alle ausgehenden Kanten aktiviert. Synchronisation: Der Kontrollfluss wartet am Gateway, bis alle eingehende Kanten aktiviert sind.	
Inklusives Gateway	Verzweigung: Abhängig von den Kantenbedingungen werden ein oder mehrere ausgehende Kanten aktiviert. Synchronisation: Abhängig von den Kantenbedingungen wartet der Kontrollfluss am Gateway bis ein oder mehrere eingehende Kanten aktiviert sind.	

Tabelle 6: Geschäftliche Sicht - Gruppierungen

Objektname	Objektbeschreibung	Visuelle Darstellung
Pool	Ein Pool ist die graphische Repräsentation eines Teilnehmers einer Kollaboration. Folglich existiert für jeden Teilnehmer ein separater Pool. Ein Pool gruppiert die Objekte, die dem Teilnehmer zugeordnet werden.	
Lane	Lanes werden dazu verwendet, um Aktivitäten zu organisieren und kategorisieren. Sie können innerhalb oder außerhalb von Pools verwendet werden	

Die Ablauforganisation der folgenden Abbildung zeigt Beispielkollaboration zwischen den zwei Teilnehmern *Ambulanz* und *Apotheke*. Der Prozess der Ambulanz wird durch ein zeitliches Ereignis zum Beginn jeder Schicht gestartet. Durch das Parallele Gateway werden die Aktivitäten *Erfasse Medikamentenbestand* und *Erfasse Verbandsmittelbestand* parallel ausgeführt. Wenn der Bestand an Medikamenten ausreichend ist (Gateway-Bedingung *Bestand OK*), wird der Prozess und damit auch die Kollaboration beendet. Falls der Medikamenten- oder Verbandsmittelbestand zu niedrig ist (Gateway-Bedingung *Bestand niedrig*), leitet das Exklusive Gateway den Kontrollfluss zur Aktivität *Bestellung aufgeben* und der Prozess der Ambulanz wird beendet. Diese Aktivität ruft das Nachrichtenereignis des Prozesses des Teilnehmers Apotheke auf. Dieses Ereignis fungiert gleichzeitig als Startpunkt des Prozesses der Apotheke. Die erste Aktivität überprüft, ob die die Apotheke genügend Medikamente bzw. Verbandsmittel vorrätig hat, um die Bestellung der Ambulanz zu erfüllen. Wenn der Bestand ausreichend ist, werden die Medikamente bzw. Verbandsmittel paketiert und versandt. Das Paket wird durch das Datenobjekt *Paket für Ambulanz* repräsentiert.

Für den Fall, dass der Bestand nicht ausreicht, um die Bestellung zu erfüllen (Gateway-Bedingung *Bestand nicht ausreichend*), wird bei einem Großhändler eine entsprechende Bestellung aufgegeben. Die Bestellung umfasst weitere Schritte, wie beispielsweise die Vereinbarung die Durchführung der Bezahlung oder den eigentlich Liefervorgang der Medikamente und Verbandsmittel, die hier der Übersichtlichkeit wegen nicht dargestellt werden. Aus diesem Grund ist die Aktivität als zugeklappter Teilprozess dargestellt. Alle Aktivitäten des Apothekenprozesses, die den Bestand an Medikamenten und Verbandsmitteln verwalten, greifen auf das Bestellsystem zu, das wiederum durch die Topologie der logischen Schicht beschrieben ist.

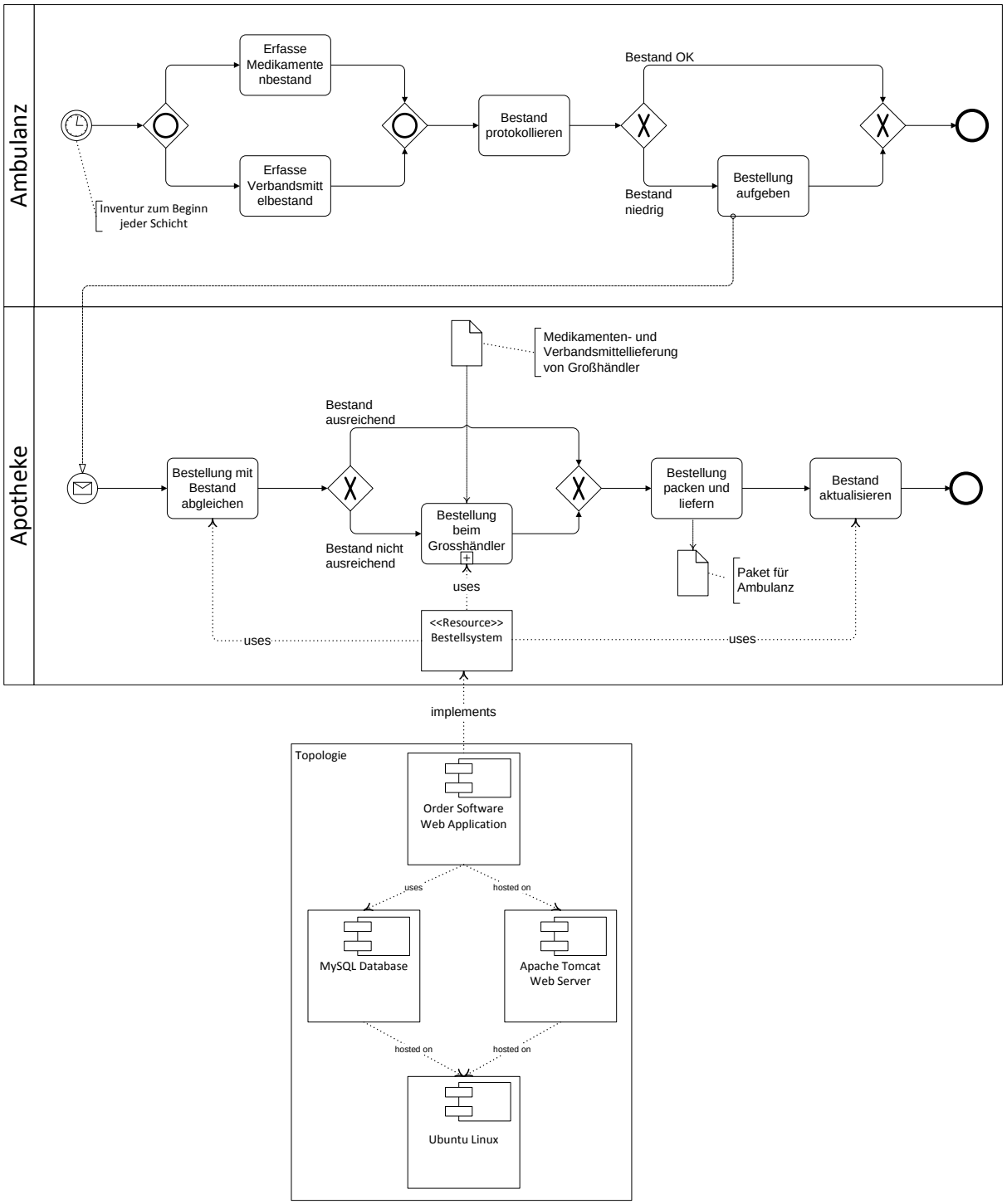


Abbildung 4 Beispiel zur Beschreibung der geschäftlichen Schicht

Erfassung der Migrationsszenarien

Das IAAS hat zusammen mit den Partnern FZID und DRESO eine Analyse der Anwenderinfrastrukturen der Anwendungspartner durchgeführt. Dabei wurden vier verschiedene Migrationsszenarien untersucht:

1. Maerker (Fachanwendung des Brandenburgischen IT-Dienstleisters)
2. Cloud-Client-Computing (Virtuelle Desktops für Mietparteien)
3. GAP IT! (Krankenhausinformationssystem im Robert-Bosch-Krankenhaus)

4. Mainsaver (Instandhaltungssoftware am Flughafen Stuttgart).

Um die Anwenderinfrastruktur eines Unternehmens adäquat zu erfassen, wurde ein entsprechendes und für MiGRATE! gültiges Vorgehensmodell definiert (siehe Abbildung 5). Dieses Modell beschreibt die einzelnen Phasen vom Erstkontakt der Anwendungspartner bis hin zur Identifikation und Auswahl von Migrationskandidaten. Die Realisierung der einzelnen Phasen dieses Modells wird unter anderem durch die Methoden realisiert, welche in den vorhergehenden Kapiteln eingehend erläutert wurden. Dieses Kapitel stellt die einzelnen Phasen des Modells vor und zeigt den Einsatz der verschiedenen Methoden auf.

- In Phase 1 werden die allgemeinen IT-Ziele und Strategien eines Unternehmens erfasst. Diese ermöglichen eine erste Einschätzung, inwieweit die IT-Landschaft eines Unternehmens generelle Migrationsszenarien zulässt.
- Auf Basis dieser Informationen werden in Phase 2 geeignete Methoden zur detaillierteren Erfassung der Anwenderinfrastruktur ausgewählt und gegebenenfalls auf den konkreten Anwendungsfall angepasst.
- Die Anwendung der ausgewählten Methoden aus Phase 2 ermöglicht in Phase 3 die konkrete Erhebung der IT-Anwenderinfrastruktur. Durch die Verwendung der verschiedenen Modelle der Methoden auf der physischen, logischen und geschäftlichen Ebene, lässt sich eine übergreifende „Landkarte“ der IT-Anwenderinfrastruktur erstellen. Die Verwendung geeigneter Werkzeugunterstützung ermöglicht die Dokumentation und erste Analyse der IT-Anwenderinfrastruktur. Die erhobenen Modelle sind zu diesem Zeitpunkt jedoch noch nicht vollständig. Zur Vervollständigung werden weitere Informationen zu Servern, Anwendungen, Prozessen, etc. benötigt.
- Diese Informationen, bspw. Serverauslastungen, Energieverbrauch, etc., werden in Phase 4 gesammelt und aufbereitet.
- Das Ergebnis sind vollständige Modelle, auf denen in Phase 5 eine begründete Auswahl von Migrationskandidaten durchgeführt werden kann.

Phase 6 dient der übergreifenden Dokumentation aller relevanten Ergebnisse.

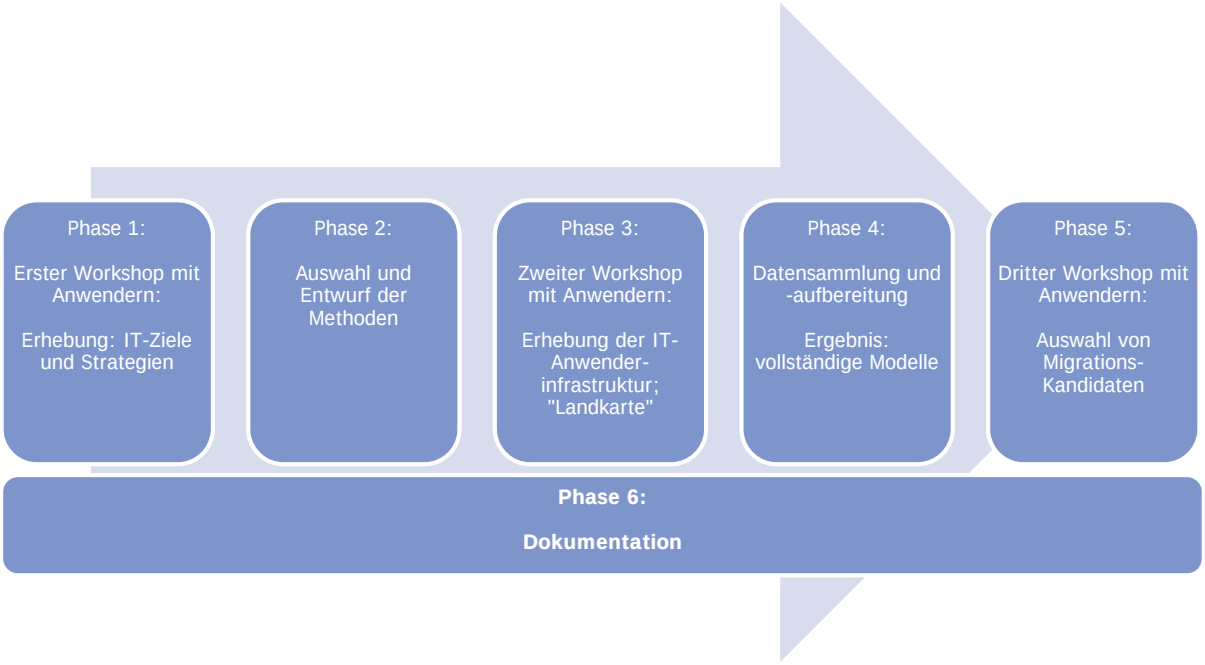


Abbildung 5 Vorgehensmodell zur Identifikation der vorhandenen Anwenderinfrastruktur

Das IAAS war insbesondere an der Erfassung des Migrationsszenarios „Maerker“ beteiligt, da hier ein modularer Aufbau der Anwendung vorlag. Diese modulare Struktur eignet sich besonders zur Migration in eine Cloud-basierte Infrastruktur. Die Anwenderinfrastruktur für Maerker wurde soweit erhoben, als das alle relevanten Informationen für die nachfolgende, formale Beschreibung als so genanntes TOSCA-Template vorliegen.

Der „Maerker“ ist ein Webportal, mit dem es einem Bürger einer Gemeinde ermöglicht wird, Hinweise zu Infrastrukturproblemen an die Gemeinde zu übermitteln. Der zuständige Sachbearbeiter der Gemeinde kann diesem Hinweis dann nachgehen und auf dem Webportal einen Status zur Bearbeitung hinterlassen. Das Maerker-Webportal ist eine Unterseite eines CMS-Systems (SixCMS) und besitzt ein Frontend sowie ein Backend, wie die nachfolgende Abbildung verdeutlicht.

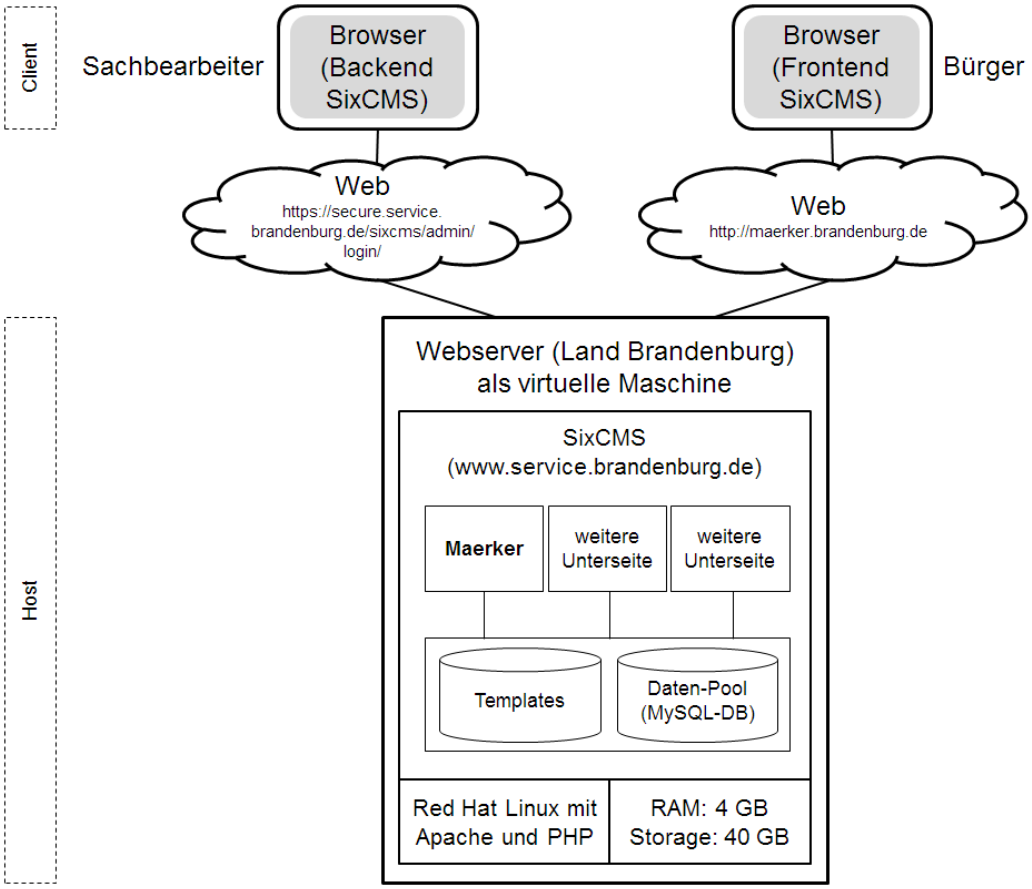


Abbildung 6 Client-Server-Architektur Maerker

Die virtuelle Maschine aus Abbildung 6, d.h. der Webserver des Landes Brandenburg, wird als LAMP-Server betrieben und hostet das SixCMS-System (www.service.brandenburg.de). Dieses SixCMS-System gliedert sich in einige Content- bzw. Unterseiten, wovon eine den Maerker darstellt. Der Maerker ist auf der Client-Seite über das Web via Browser erreichbar. Zum einen kann der Bürger über das Frontend auf den Maerker zugreifen und zum anderen dient das Backend dem Sachbearbeiter als Maerker-Zugang.

AP1.2 Identifikation des Energiebedarfs vorhandener Anwenderinfrastrukturen

Das IAAS hat die Partner DRESO und FZID bei der Identifikation des Energiebedarfs durch die Beschreibung der Anwenderinfrastrukturen unterstützt.

AP1.3 Spezifikation der Anforderungen

TOSCA kann für eines der Hauptprobleme der IT-Infrastruktur der Öffentlichen Verwaltung in Deutschland ein relevantes Werkzeug darstellen: Der hohe Fragmentierungsgrad, welcher mit der föderalen Struktur Deutschlands einhergeht. Konkret setzt die Lösung, welche sich durch TOSCA bietet, auf Abschnitt F – Leistungsfähige IT-Unterstützung – Ziel 18 der nationalen e-Government Strategie[1] an:

Inhalte, Basisdienste, Anwendungen und Infrastruktur lassen sich bündeln und wiederverwenden:

Für e-Government verwendete Inhalte, Basisdienste, Anwendungen und Infrastrukturen werden, soweit möglich, gebündelt und im Rahmen der Wiederverwendbarkeit auch anderen Stellen zur Verfügung gestellt.

- *Der IT-Planungsrat koordiniert die Umsetzung und den Ausbau übergreifender Standards.*
- *Bund, Länder und Kommunen treiben die Wiederverwendung und Bündelung in ihren jeweiligen Zuständigkeitsbereichen und untereinander voran.*
- *Die Wiederverwendung wird praxisnah (z.B. in Modellregionen) erprobt. Durch die abgestimmte Etablierung von Best-Practices in geeigneten Bereichen wird die Grundlage für einen effizienten Transfer geschaffen.*

Es ist absehbar, dass die zukünftige Entwicklung des Cloud-Computing des Öffentlichen Sektors private „Bundesländer-Clouds“ hervorbringen wird. Die föderale Struktur Deutschlands begünstigt eine solche Entwicklung. Dementsprechend wird es zukünftig von hoher Relevanz sein, wie eine Migration einer Cloud-Anwendung bzw. die Schaffung einer neuen Instanz in einer anderen Bundesland-Cloud durchgeführt werden kann. TOSCA, von der Definition her, bietet eine Standardisierung der Portabilität von Cloud-Anwendungen und -Services. Ebenfalls in diesem Modell berücksichtigt sind Anforderungen bzgl. Sicherheit, Governance und IT-Compliance ohne Lock-in-Effekt. Des Weiteren kann TOSCA dazu beitragen, über die Automatisierung der Abläufe, die Kosten einer Cloud-Migration als neue Instanz signifikant zu reduzieren. Dies ist ebenfalls als explizites Ziel der nationalen e-Government Strategie definiert (Zielbereich B, Zielbereich 8 – Die Zusammenarbeit von Bund, Ländern und Kommunen erfolgt regelmäßig über Mittel der IKT).

Das in MIGRATE! ausgewählte Szenario ist ein Versuch, die Möglichkeiten, die sich durch TOSCA bieten, im Sinne der nationalen e-Government Strategie zu nutzen. Bei erfolgreicher Durchführung einer Migration der Maerker-Fachanwendung lässt sich ableiten, dass weitere Anwendungs-Migrationen in funktionell-ähnlicher Form zwischen den zentralen Bundesländer-RZ‘s durchgeführt werden können. Damit würde ein wichtiger Beitrag zur Umsetzung der nationalen e-Government Strategie geleistet.

[1] http://www.cio.bund.de/SharedDocs/Publikationen/DE/Aktuelles/nationale_e_government_strategie_beschluss_20100924_download.pdf?__blob=publicationFile

2. AP2: Cloud-basierte energieoptimierte Anwenderinfrastrukturen

AP2	Cloud-basierte energieoptimierte Anwenderinfrastrukturen	Leitung: FZID
Ziele (IAAS)	– Entwicklung und Spezifikation von Architekturmodellen für Cloud-basierte energieoptimierte Anwenderinfrastrukturen	
Ergebnisse (IAAS)	– Green Policies zur Beschreibung der Anforderungen an die Nachhaltigkeit bzw. Energieeffizienz – Konzept zur Definition von Policies innerhalb von TOSCA – Energieeffizienzeigenschaften untersuchter Cloud-Lösungen und Architekturen – Green Business Process Patterns – Methode zur ökologischen Analyse von Geschäftsprozessen	

Um die Anforderungen an die Nachhaltigkeit bzw. Energieeffizienz für Cloud-Anwendungen darstellen zu können, wurden durch die Definition von „Green Policies“ ein Ansatz zur Beschreibung dieser Anforderungen spezifiziert. Zu Beginn wurde der Stand der Technik durch eine Fachstudie mit dem Titel „Vergleich von Policy Sprachen zur Anwendung im Bereich des Cloud Computings“ erhoben und hierzu bestehende Policy-Sprachen systematisch auf ihre Eignung zur Anwendung im Bereich des Cloud Computing unter besonderer Berücksichtigung energierelevanter Parameter untersucht. Die Anforderungen des Projektes MIGRATE! an eine geeignete Policy-Sprache wurden in die Bewertung der untersuchten Sprachen mit einbezogen. Eine Bewertung der untersuchten Policy Sprachen ist in Tabelle 7 dargestellt. Die Bewertung erfolgte anhand einer normierten Skala von 0 bis 10 und basiert auf einer Gleichgewichtung aller untersuchten Kriterien. Das Bewertungsschema wurde so entworfen, dass einzelne Kriterien und Kriteriengruppen abhängig von konkreten Anforderungen unterschiedlich gewichtet werden können. Die Bewertung wird abhängig von der Gewichtung dynamisch berechnet.

1	WS-Policy	9,38
2	Rei	8,83
3	XACML	8,12
4	PERMIS	7,57
5	SAML	7,40
6	RuleML	5,26
7	P3P	4,80
8	UDDI	4,61
	SecPAL	4,61
9	ODRL	4,48
10	IBM EPAL	4,03

Tabelle 7: Bewertung der untersuchten Policy Sprachen

Die Laufzeitumgebung der Cloud-Anwendung überprüft, ob diese Policies eingehalten werden. Für die Policies wurde eine allgemeine Policy Taxonomie für Cloud Services entwickelt, mit der Policies zu verschiedenen Komponenten bzw. Aspekten der Anwendung zugeordnet werden können (bspw. können Policies für komplette Anwendungen oder nur für Anwendungskomponenten definiert werden). Des Weiteren wurde ein Lebenszyklusmodell für Policies entwickelt, mit dem definiert werden kann, wann bestimmte Policies aktiv sind. So können Policies definiert werden, die nur während des Deployments der Anwendung aktiviert sind, während andere Policies nur zur Laufzeit der Cloud Anwendung relevant sind.

Auf Basis des entwickelten Ansatzes wurde ein Konzept zur Definition von Policies innerhalb von TOSCA erarbeitet. Hierzu wurden die in Arbeitspaket AP3 bereits erarbeiten LeGO4TOSCA (L4T) NodeTypes konzeptionell erweitert. Abbildung 7 zeigt eine Übersicht der entwickelten Architektur. Die gesamten Ergebnisse/Details des Konzepts wurden, in Zusammenarbeit mit dem Projekt *CloudCycle* (TrustedCloud, 01MD11023), auf der OTM2013-Konferenz veröffentlicht und präsentiert (Waizenegger et al., 2013), sowie im Meilensteinbericht M4.1 zusammengefasst.

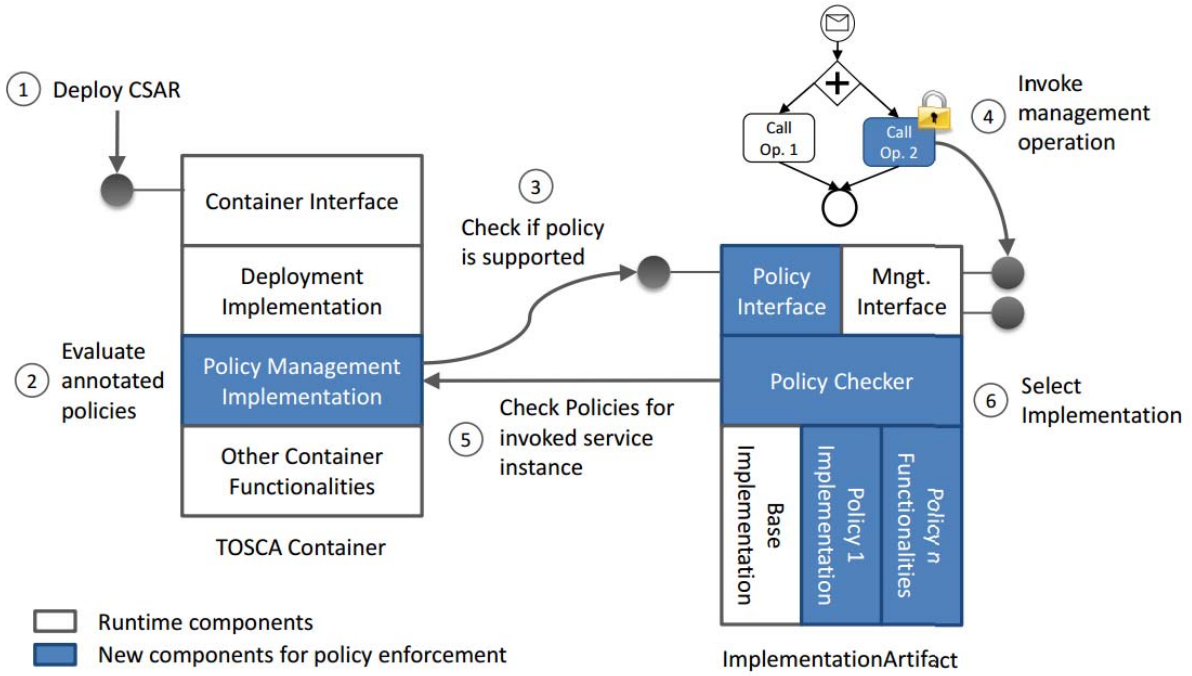


Abbildung 7 Policy-Erweiterungen der implementierten TOSCA IAs

Das Management ökologischer Aspekte in Geschäftsprozessen und Anwenderinfrastrukturen ist sehr vielfältig. Das IAAS hat deshalb zunächst eine Untersuchung durchgeführt, welche die Energieeffizienz verschiedener Cloud-Lösungen und Architekturen vergleicht. Für die einheitliche Beschreibung der Cloud-Lösungen wurde auf bestehende Cloud Patterns zurückgegriffen. Basierend auf diesen Ergebnissen wurde für die logische Schicht, d.h. auf Anwendungsebene, eine Methodik und ein entsprechender Prototyp entwickelt, der die Anwender dabei unterstützt, ihre vorhandene Architektur anhand verschiedener Cloud Eigenschaften zu beschreiben und ihnen anschließend energieeffiziente Cloud Lösungen vorschlägt. Die prototypische Verwendung der Methodik hat gezeigt, dass diese aus Komplexitätsgründen bisher noch nicht uneingeschränkt verwendbar ist und noch weitere Arbeiten zur Verbesserung des Ansatzes notwendig sind.

Neben der Analyse der Infrastruktur wurde im Rahmen der Definition der Migrationsgrundlagen auch ein Konzept zur ökologischen Analyse von Geschäftsprozessen entwickelt. Die visuelle Analyse des Ansatzes soll dabei das schnelle Erfassen des Umweltzustands der in einem Prozess beschriebenen Aktivitäten ermöglichen. Dadurch kann schnell auf ab Abweichungen von definierten Grenzwerten reagiert werden.

Neben der visuellen Darstellung von Prozessmodellen beruht der entwickelte Ansatz auf dem Prinzip der Verantwortungsteilung, d.h. die graphischen Vorlagen, die zu visualisierenden Daten sowie die Visualisierung selbst können unabhängig voneinander beschrieben werden. Dies führt zu einer Verbesserung der Flexibilität, da Änderungen schneller umgesetzt werden können. Die durch die ökologische Analyse gewonnenen Erkenntnisse können innerhalb der Entscheidungsfindung eines Migrationsszenarios hilfreich sein, um bestimmte

Migrationskomponenten (auch auf Ebene der Infrastruktur) detailliert zu bestimmen. Das entwickelte Konzept wurde in (Nowak et al., 2012) publiziert.

Als Datenquelle für die ökologische Analyse der Geschäftsprozesse müssen in Abhängigkeit der vom Prozess verwendeten Ressourcen unterschiedliche Methoden angewendet werden. In der betrachteten Domäne der automatisierten Geschäftsprozesse stellen IT-Systeme die größten Treiber des Umweltzustands dar. Der Energieverbrauch dieser Ressourcen während des Betriebs kann deshalb als größter Treiber des Umweltzustands angesehen werden. In Zusammenarbeit mit dem Partner DRESO wurde deshalb ein Ansatz zur Ermittlung des Energiebedarfs von Geschäftsprozessen auf Basis der unterliegenden Infrastruktur/Ressourcen entwickelt. Die Ergebnisse dieser Arbeit wurden im Rahmen der Konferenz EDOC 2013 veröffentlicht und präsentiert (Nowak et al., 2013).

Im Rahmen der anschließenden energieeffizienten Gestaltung von Geschäftsprozessen und deren ausführenden Ressourcen wurden verschiedene Ergebnisse im Bereich von Green Business Process Patterns erarbeitet. Die in (Nowak et al., 2011) erarbeiteten Patterns fokussieren zunächst explizite Ansätze und Lösungen zur Neugestaltung existierender Geschäftsprozesse. In der weiteren Arbeit wurde auf Basis dieser Erkenntnisse eine generische Methode zur Identifikation von weiteren Patterns entwickelt. Unter Anwendung dieser Methode wurde eine Reihe weiterer Patterns erarbeitet. Dies basieren teilweise auf bereits existierenden Patterns und zeigen, wie diese unter Berücksichtigung von Problemstellungen der Umweltperspektive in dieser neuen Domäne angewendet werden können. Die Ergebnisse dieser Arbeit wurden auf der SOCA2013 Konferenz veröffentlicht und präsentiert (Nowak and Leymann, 2013a). Die komplette Liste der identifizierten Patterns wurde als Technischer Bericht an der Universität Stuttgart veröffentlicht (Nowak and Leymann, 2013b).

3. AP3: Migrationsverfahren und -werkzeuge

AP3	**Migrationsverfahren und -werkzeuge**	**Leitung: IAAS**
Ziele (IAAS)	– Bereitstellung von Migrationsverfahren und –werkzeugen – Entwicklung von Migrationsprozessen	
Ergebnisse (IAAS)	– Vorgehensmodell für die Cloud-Migration – Werkzeugunterstützung – Cloud-Service-Beschreibung mittels TOSCA – Open-Source-Laufzeitumgebung für TOSCA	

AP3.1 Grundlagen der Migrationsentscheidungen

Zusammen mit den Partnern FZID und IBM wurden verschiedene Anforderungen an Cloud Umgebungen und der Migration von Anwendungen in diese Umgebungen identifiziert. Hierzu gehören unter anderem die Beschleunigung der Durchlaufzeiten, die konsequente Automatisierung durch modellbasierte Ansätze, die Eliminierung von Ressourcenverschwendung sowie die Verbindung von Development und Operations (DevOps). Weitere Details zu den Grundlagen können dem Schlussbericht des Partners IBM entnommen werden. Die Umsetzung der einzelnen Anforderungen wird im Rahmen des Migrationsverfahrens in AP3.2 dargestellt.

Über die direkten Anforderungen einer Migration hinaus hat das IAAS auch untersucht, wie die Energieeffizienz von Geschäftsprozessen, welche verteilt in Cloud-Umgebungen ausgeführt werden, verbessert werden kann. Die verteilte Ausführung hat zur Folge, dass

Geschäftsprozesse über Ressourcengrenzen hinweg miteinander kommunizieren. Die auf der IC2E2013-Konferenz vorgestellte Arbeit (Wagner et al., 2013) hat diese Konstellationen untersucht und verschiedene Optimierungsmöglichkeiten aufgezeigt, welche sowohl die ökonomischen als auch die ökologischen Kosten der Ausführung verringern können. Der Kern dieser Arbeit basiert auf der Konsolidierung von Prozesschoreographien und eine damit verbundene interne Optimierung der Prozessausführungsumgebung.

AP3.2 Migrationsverfahren für heterogene Anwenderinfrastrukturen

Die Migration von bestehenden Anwendungen aus dem eigenen Rechenzentrum oder von dem bisherigen RZ-Betreiber in eine Cloud-Umgebung ist für viele Unternehmen eine Herausforderung, die mit der Komplexität der vorhandenen Anwenderinfrastruktur (AIS) zunimmt. Die Grundfragen der Migration sind:

- „Was soll in die Cloud migriert werden?“
- „Wie wird die Migration durchgeführt?“
- „Wie ist die Wirkung auf den Energieverbrauch?“
- „Wie wirtschaftlich ist die Migration?“.

Bisherige Erfahrungen aus Migrationsprojekten zeigen, dass für die Migration ein systematisches Vorgehen notwendig ist, um die Migration zu planen, technisch durchzuführen und den eingetretenen Migrationserfolg in Hinblick auf die Migrationsziele bewerten zu können. Das Forschungsprojekt Migrate! hat sich dieser Herausforderung angenommen und ein Vorgehensmodell entwickelt, das IT-Entscheidungsträger über den gesamten Prozess der Migration modell- und werkzeugbasiert unterstützt.

Vorgehensmodell zur Migration von Anwendungen in die Cloud

Abbildung 8 zeigt die fünf Hauptphasen des Vorgehensmodells:

- Im ersten Schritt (Phase 0) muss der Projektauftrag definiert werden. Dadurch wird der für eine Migration geltende Unternehmensbereich identifiziert und entsprechende Migrationsziele festgelegt. Diese Informationen sind die Voraussetzung für die Identifikation geeigneter Migrationsszenarien sowie möglicher Ziel-Cloud-Umgebungen.
- Phase 1 dient der Erfassung der bestehenden Anwenderinfrastruktur. Das Ziel ist die Identifikation aller relevanten Informationen über die potentiell zu migrierende IT. Dieser Schritt ist erforderlich, um eine Vergleichbarkeit mit möglichen Cloud-Umgebungen sicherzustellen und damit die Entscheidungsfindung zu unterstützen.
- Phase 2 entwickelt Migrationsszenarien. Dabei wird analysiert, welche Cloud-Lösungen sich für den Betrieb der zuvor identifizierten Komponenten eignen. Die Auswahl basiert dabei weitgehend auf den zuvor identifizierten Informationen. Nach der Auswahl eines oder mehrerer geeigneter Migrationsszenarien wird die eigentliche Migration in Phase 3 durchgeführt. Dazu muss ein passendes technisches Migrationsverfahren, welche alle relevanten Aspekte der Migration abdeckt, identifiziert werden.

Nach der Migration muss in Phase 4 die neue Umgebung überwacht werden. Dieser Schritt dient dem Abgleich von Ist-Kennzahlen gegenüber den Plan-Kennzahlen der Migration. Die laufende Überwachung der neuen Umgebung kann wiederum Input für weitere Migrationsvorhaben, auch zwischen verschiedenen Cloud-Umgebungen, bereitstellen.

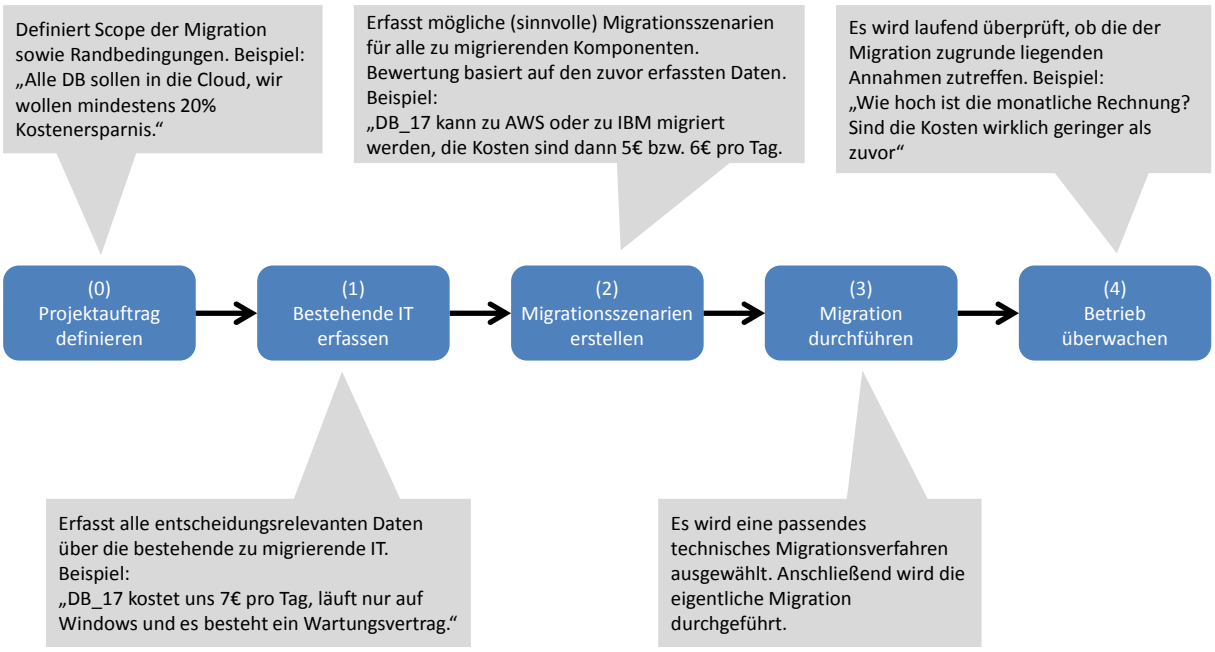


Abbildung 8 Phasen des Vorgehensmodells

Die erweiterte Übersicht in Abbildung 9 zeigt die Phasen inklusive der entsprechenden Ein- und Ausgaben in Form von Modellen der AIS-Komponenten. Ziel hierbei ist es, die Komponenten so zu auszuwählen, dass diejenigen Migrationsszenarien mit dem höchsten potenziellen Nutzen als erste umgesetzt werden. Diese einzelnen Phasen des Vorgehensmodells werden in den folgenden Kapiteln näher beschrieben.

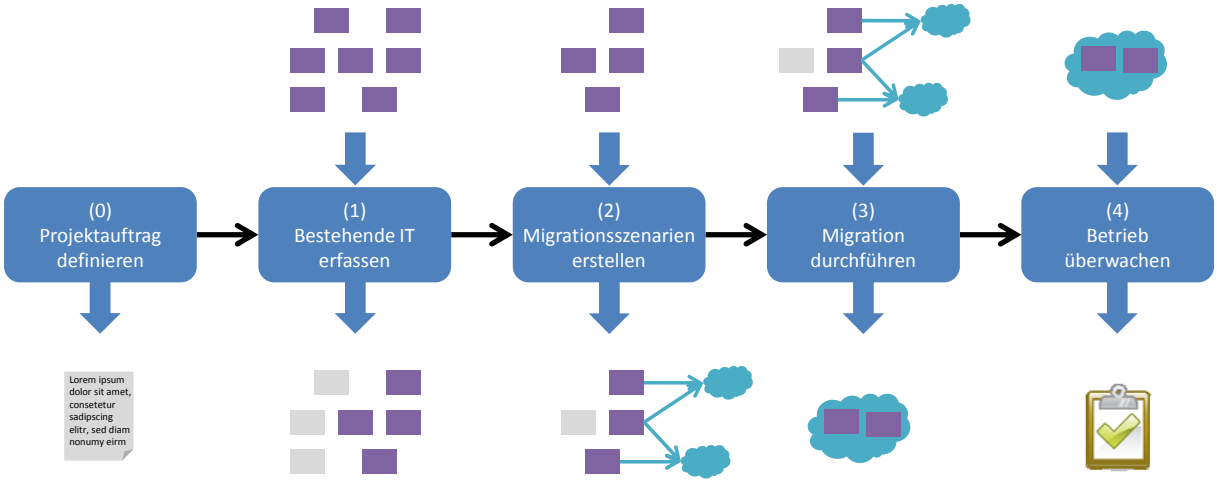


Abbildung 9 Phasen des Vorgehensmodells, Ein- und Ausgaben

Phase 1 – Erfassung der bestehenden Anwenderinfrastruktur

Das Ziel dieser Phase ist die Erfassung der Anwenderinfrastruktur in Bezug auf eine mögliche Migration in eine Cloud-Umgebung. Dabei müssen insbesondere die Vorgaben aus dem Projektauftrag berücksichtigt werden. Die eigentliche Erfassung wird in drei Phasen durchgeführt: (1.1) Erfassung der bestehenden AIS, (1.2) Grobanalyse der erfassten IT und (1.3) Feinanalyse der erfassten Daten. Eine Übersicht dieser Phasen ist in Abbildung 10 zu sehen.

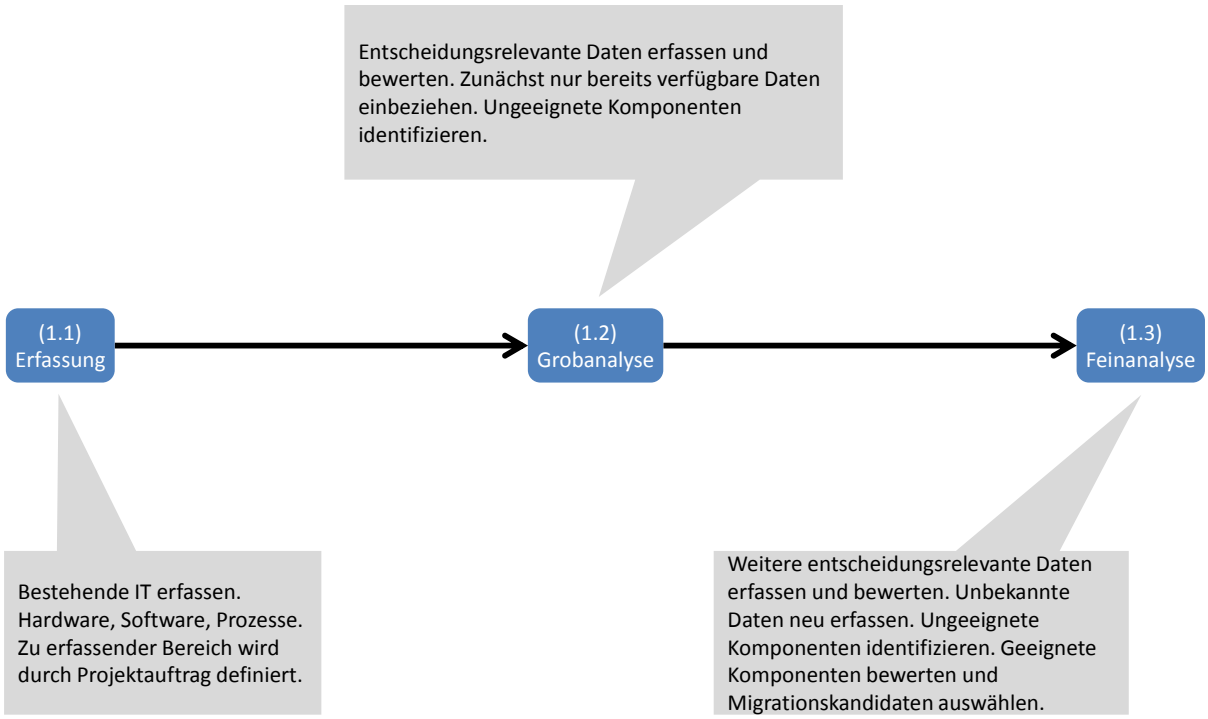


Abbildung 10 Phase 1, Übersicht

Bei der *Erfassung* der AIS in Phase (1.1) wird zunächst der zu relevante Bereich der AIS aus dem Projektauftrag ermittelt. Auf Basis dieser Vorgaben werden die entsprechenden Hardwarekomponenten, die beteiligten Softwarekomponenten und Services, sowie die involvierten Geschäftsprozesse identifiziert und dokumentiert. Dies kann auf Basis einer Identifikation und Auswertung vorhandener Dokumente, der Ergänzung vorhandener Dokumente oder durch eine Neuerfassung erfolgen. Die Dokumentation kann in einem AISM erfolgen (siehe ausführlich Meilensteinbericht M1.1).

Die *Grobanalyse* in Phase (1.2) hat das Ziel, die erstellten Modelle zu erweitern und eine Vorauswahl geeigneter Migrationskandidaten zu treffen. Durch die Analyse der verfügbaren Informationen sollen ungeeignete Kandidaten bereits vorab ausgeschlossen werden können, um Aufwände für weitere Analysen zu vermeiden. Aus diesem Grund sollen in dieser Phase vorranging vorhandene Informationen genutzt werden. Die Kriterien der Analyse umfassen fünf Hauptgruppen und können je nach den im Projektauftrag spezifizierten Zielvorgaben unterschiedliche Ausprägungen oder Gewichtungen besitzen:

Technische Kriterien: Beschreiben die technischen Anforderungen an den Betrieb von Anwendungskomponenten in einer Cloud-Umgebung, wie beispielsweise erforderliche Zugriffsmechanismen.

Rechtliche Kriterien: Beschreiben die rechtlichen Grundlagen für den Betrieb von Anwendungen oder das Speichern von Daten in Cloud-Umgebungen.

Finanzielle Kriterien: Beschreiben alle relevanten finanziellen Aspekte einer Migration, wie beispielsweise die durch die Migration verursachten einmaligen Kosten sowie die im Betrieb entstehenden Kosten bzw. Kostenänderungen.

Energiekriterien: Beschreiben Kriterien bezüglich des derzeitigen Energieverbrauchs oder Auslastungsprofile der betroffenen Hard- und Softwarekomponenten.

Dienstgütekriterien: Beschreiben alle für den spezifizierten Betrieb einer Anwendung relevanten Dienstgütekriterien, wie beispielsweise Verfügbarkeit, Gesamtanzahl der Benutzer, und der zugesicherten Antwortzeiten.

Zur Unterstützung der Analyse können verschiedene Methoden eingesetzt werden. Um das vorhandene Wissen zu strukturieren, können beispielsweise Fragebögen entwickelt oder Interviews und Workshops durchgeführt werden. Sofern vorhanden können auch bereits erfasste Energiemessreihen analysiert werden. Auf Basis der gesammelten Informationen werden anschließend die einzelnen Komponenten bewertet und geeignete Migrationskanndidaten ausgewählt. Als Entscheidungsunterstützung kann hier beispielsweise eine Portfolioanalyse zum Einsatz kommen, siehe Abbildung 11.

Mit Hilfe des Migrationsportfolios werden Komponenten der AIS wie folgt untersucht:

- Auf der horizontalen Achse werden Komponenten hinsichtlich ihrer Energieeffizienz eingeordnet. Im Rahmen der Grobanalyse handelt es sich hier um eine Einordnung entlang des Spektrums gering-hoch.
- Auf der vertikalen Achse werden Komponenten hinsichtlich ihres Beitrages zum IT-Erfolg eingeordnet. Unter IT-Erfolg ist zu verstehen, welche Bedeutung die Komponente für die Erreichung der wirtschaftlichen Ziele des Unternehmens besitzt. Hier erfolgt ebenfalls eine Einschätzung zwischen gering und hoch.
- Zusätzlich kann eine dritte Dimension eingefügt werden: Die Rechteckfläche stellt den Anteil der Komponente am gesamten IT-Energieverbrauch dar. Hier kann zum Teil auf Messdaten zurückgegriffen werden, oder es erfolgt eine Abschätzung.

In Abhängigkeit von drei genannten Kriterien lassen sich somit die Migrationskandidaten mit dem höchsten Optimierungspotential identifizieren.

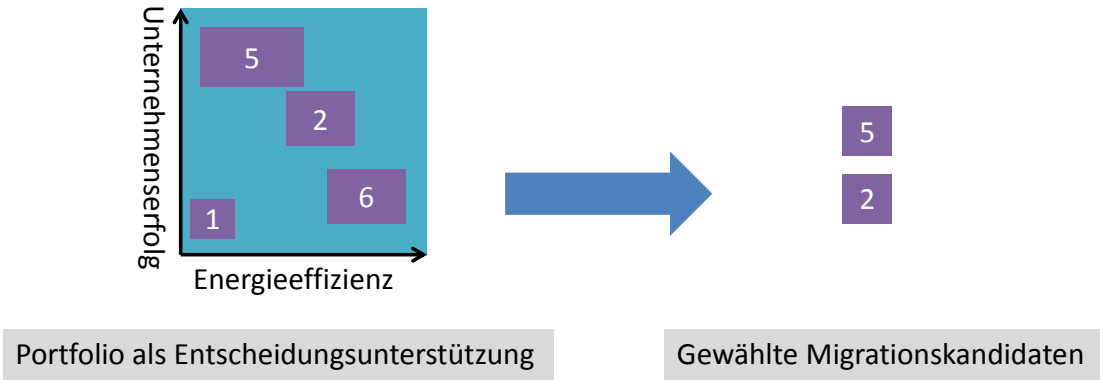


Abbildung 11 Portfolio als Entscheidungshilfe

Die *Feinanalyse* in Phase (1.3) nutzt die Menge an ausgewählten Migrationskandidaten als Ausgangslage für eine weitere Verfeinerung der Modelle sowie der Menge an Migrationskandidaten. Für die identifizierten Kriterien der Grobanalyse werden in diesem Schritt aktiv weitere Informationen erhoben. Die folgende Auflistung zeigt hierzu einige Beispiele:

Technische Kriterien: Beschreiben beispielsweise die Kommunikationsschnittstellen und Protokolle oder Anforderungen an die Laufzeitumgebung, wie Bibliotheken, Plug-Ins, Softwareversionen etc.

Rechtliche Kriterien: Beschreiben beispielsweise Kriterien und Anforderungen zum Datenschutz von personenbezogenen Daten sowie Bestimmungen und Richtlinien bestehender Verträge.

Finanzielle Kriterien: Beschreiben beispielsweise Informationen und Ergebnisse einer Total-Cost-of-Ownership-Analyse, Prozesskosten- oder Investitionsrechnungen.

Energiekriterien: Beschreiben beispielsweise definierte Energiemodelle und erfasste Energiemessreihen sowie die zugehörigen Auslastungsdaten.

Dienstgütekriterien: Beschreiben beispielsweise Monitoring Informationen oder Logfile Auswertungen.

Wie auch bei der Grobanalyse erfolgt auf Basis der erweiterten und verfeinerten Informationen eine Bewertung der ermittelten Migrationsszenarien. Durch den Ausschluss von Komponenten wird die Menge an Migrationskandidaten weiter eingeschränkt. Das Ergebnis der Feinanalyse ist eine Menge von Migrationskandidaten, welche alle für eine potentielle Migration in Frage kommen, d.h. alle oben genannten Kriterien wurden positiv evaluiert.

Phase 2 – Erstellung der Migrationsszenarien

Das Ziel dieser Phase ist die Identifikation einer geeigneten Zielplattform für jeden in Phase 1 ermittelten Migrationskandidaten. Abbildung 12 zeigt hierzu eine Übersicht der drei erforderlichen Schritte: (2.1) Migrationsszenarien erfassen, (2.2) Migrationsszenarien bewerten und (2.3) Migrationsszenarien auswählen.

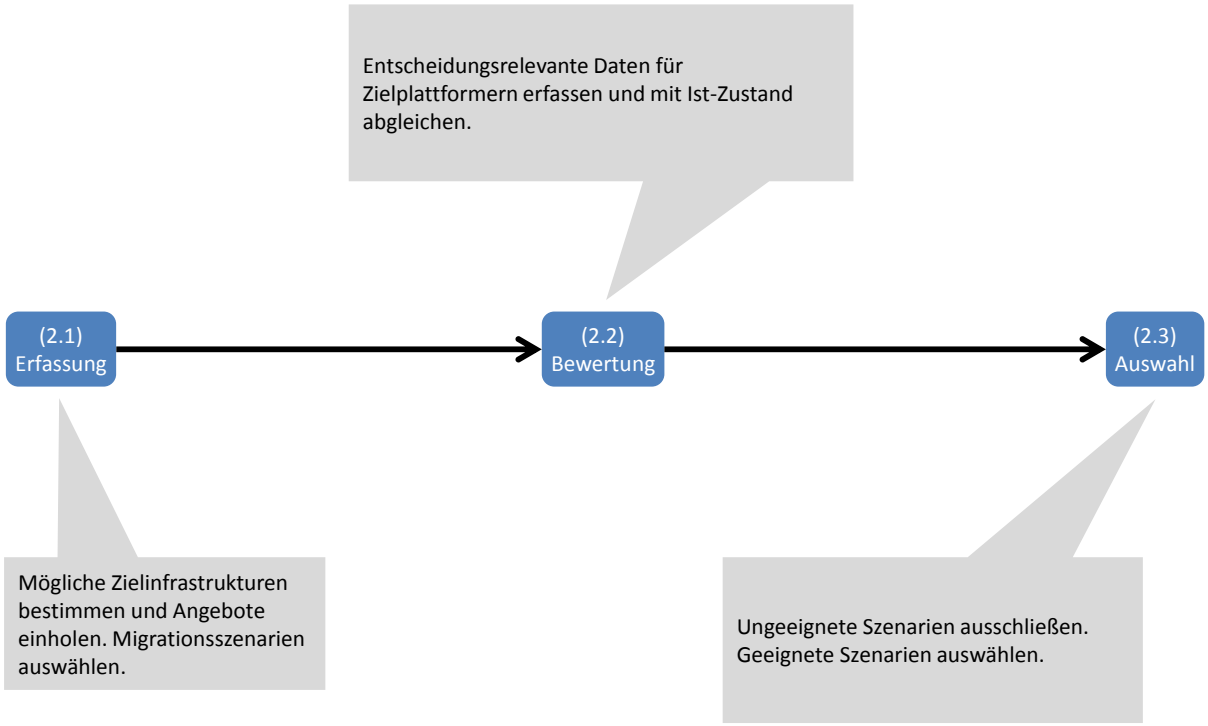


Abbildung 12 Phase 2, Übersicht

Die *Erfassung von Migrationsszenarien* in Phase (2.1) dient dazu, für jeden Migrationskandidaten eine Liste mit potentiellen Zielinfrastrukturen zu bestimmen. Ein Migrationsszenario umfasst dabei einen Migrationskandidaten und eine Zielinfrastruktur in einer Cloud-Umgebung. Für die Bestimmung der Zielinfrastruktur müssen mögliche Cloud-Anbieter bestimmt werden, mögliche Cloud-Diensttypen (*aaS) sowie mögliche Cloud-Bereitstellungstypen (private, public, etc.) bestimmt werden. Zudem müssen für jede Zielplattform die definierten technischen, rechtlichen, finanziellen, Energie und Dienstgüte Kriterien evaluiert werden. Die für eine Auswahl benötigten Informationen werden aus den vorherigen Phasen des Vorgehensmodells bezogen. Abbildung 13 zeigt exemplarisch die Entwicklung verschiedener Migrationsszenarien. Eine konkrete Menge an Migrationsszenarien kann nur auf Basis konkreter Analysen erfolgen.

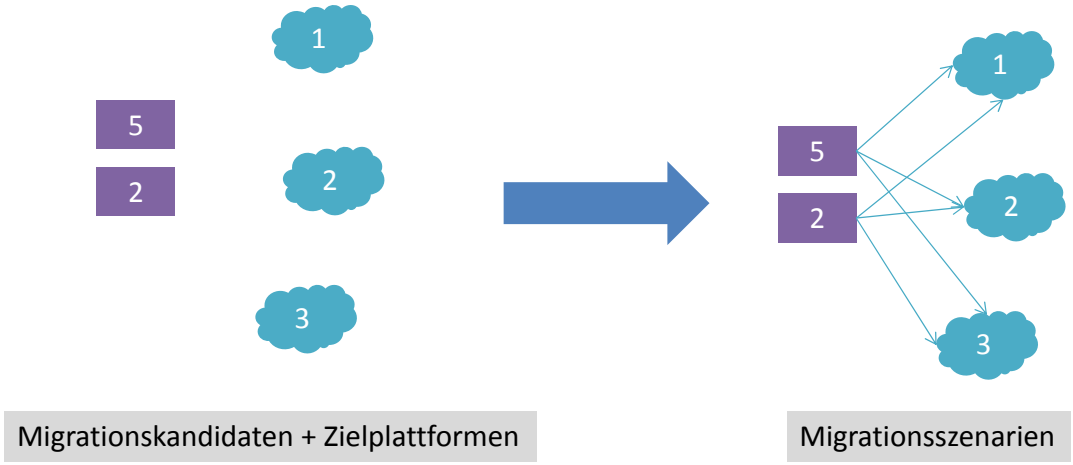


Abbildung 13 Erstellung von Migrationsszenarien

Im Anschluss an die Identifikation der Migrationsszenarien werden diese in Phase (2.2) hinsichtlich der identifizierten Kriterien bewertet. Für jedes Migrationsszenario werden die identifizierten Kriterien mit den Daten und Metriken der Zielplattform sowie dem bestehenden IST-Zustand abgeglichen (siehe Abbildung 14). Auf dieser Basis wird am Ende für jeden Migrationskandidaten genau ein Migrationsszenario ausgewählt und in der folgende Phase technisch umgesetzt.

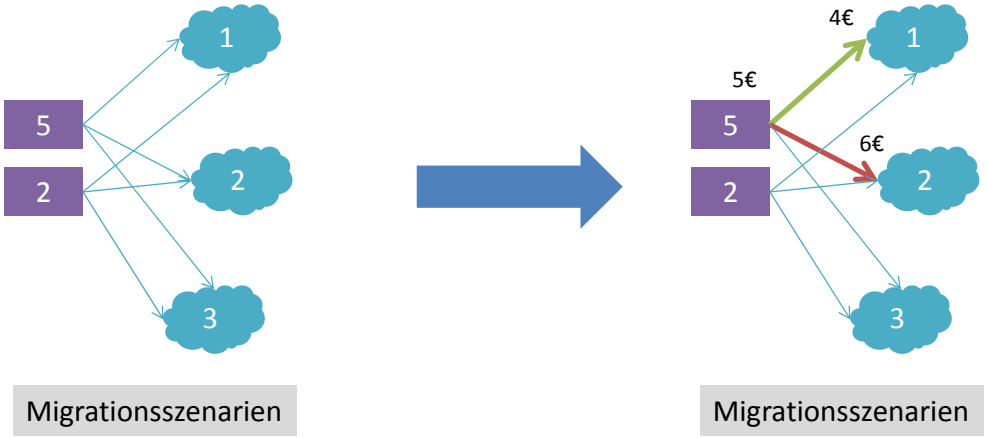


Abbildung 14 Auswahl von Migrationsszenarien

Phase 3 – Planung und Durchführung der Migration

Nachdem in Phase 2 die Migrationskandidaten ausgewählt und einem konkreten Migrationsszenario zugeordnet wurden, wird in dieser Phase die *technische Migration* geplant und durchgeführt. Das heißt, dass am Ende dieser Phase alle Migrationskandidaten in die Cloud migriert wurden sind. Die einzelnen Schritte dieser Phase sind in Abbildung 15 dargestellt. Als erstes werden passende Migrationsverfahren bestimmt, mit denen es möglich ist, den Migrationskandidaten in die Cloud zu migrieren. Im nächsten Schritt werden dann für jedes Migrationsverfahren die finanziellen und energetischen Kosten der Migration ermittelt. Auf Basis der Ergebnisse wird dann ein geeignetes Migrationsverfahren ausgewählt, das dann für den Migrationsschritt verwendet wird.

Die Migrationsverfahren haben unterschiedliche Eigenschaften (Piazaa, 2014). So wird zum Beispiel zwischen Migrationsverfahren unterschieden, mit denen die Migration ohne Ausfallzeit (*Zero-Downtime Migration*) oder mit Ausfallzeit (*Migration with Downtime*) des Migrationskandidaten durchgeführt werden kann. Andere Migrationsverfahren unterscheiden

zusätzlich, ob der komplette Migrationskandidat mit einmal migriert wird (*Forklift Migration*), oder seine Komponenten sukzessive in die Cloud migriert werden (*Hybrid Migration*). Bei der Auswahl der in Frage kommenden Migrationsverfahren muss beachtet werden, dass sie sich sehr stark hinsichtlich Komplexität, Aufwand und damit auch den Kosten für die Migration unterscheiden. Ein weiterer Faktor bei der Auswahl des Migrationsverfahrens ist die Struktur des Migrationskandidaten. Die hybride Migration eignet sich beispielsweise nur für modulare, jedoch nicht für monolithische Anwendungen.

Für jedes ermittelte Migrationsverfahren müssen nun die *Kosten erfasst werden* Dazu werden die Fix- und die Energiekosten ermittelt. Die Fixkosten umfassen u.a. Entwicklungskosten für die Anpassungen am Migrationskandidaten, Datentransferkosten, Anschaffungskosten für neue Hardware oder Software, Transportkosten und Kosten die durch externe Dienstleister entstehen, die die Migration unterstützen. Einige Cloud Anbieter stellen Tools bereit, mit denen Migrationskosten im Voraus berechnet werden können. So gibt es zum Beispiel bei Amazon die Möglichkeit, die Kosten für die Migration von Daten in die Amazon Cloud zu berechnen (Amazon, 2014). Die Migrationskosten sind dann auf den geplanten Nutzungszeitraum der Zielplattform umzulegen, um sie in die Gesamtkostenkalkulation mit einfließen zu lassen. Da auch während der Migrationsdurchführung Energie verbraucht wird (z.B. durch die Migration von Daten), ist der Energieverbrauch ebenfalls zu bestimmen und auf den Nutzungszeitraum der Zielplattform, d.h. auf die energetische Gesamtkostenkalkulation umzurechnen. Dazu ist der potentielle Energieverbrauch entweder auf Basis von Erfahrungswerten oder Simulationen zu ermitteln. Des Weiteren ist auch zu prüfen, ob die Migrationsverfahren die geforderten Dienstgütekriterien erfüllen. So kann es zum Beispiel notwendig sein, dass der Migrationskandidat bzw. seine Daten nur verschlüsselt migriert werden dürfen.

Im nächsten Schritt wird dann das *Migrationsverfahren* für die Migration ausgewählt. Alle Migrationsverfahren, die zu finanziellen bzw. energetischen Migrationskosten führen, die die in Phase 2 ermittelten Gewinne im Betrieb des Migrationskandidaten übersteigen, werden nicht weiter für die Migrationsdurchführung betrachtet. Falls jedes Migrationsverfahren diese Gewinne übersteigt, wird die Migration hier abgebrochen. Unter den Migrationsverfahren, deren Kosten die Gewinne nicht übersteigen, wird das kostengünstigste bzw. energetisch effizienteste ausgewählt. Allerdings können auch hier wieder Dienstgütekriterien bei der Auswahl eine Rolle spielen. So kann zum Beispiel nicht das günstigste Migrationsverfahren ausgewählt werden, sondern dasjenige, welches die kürzeste Migrationsdauer garantiert.

Im letzten Schritt wird die Migration des Migrationskandidaten auf die Zielplattform mittels des ausgewählten Migrationsverfahrens *durchgeführt*. Abhängig vom Modell wird auch entschieden, wann die Quellplattform abgeschaltet wird.

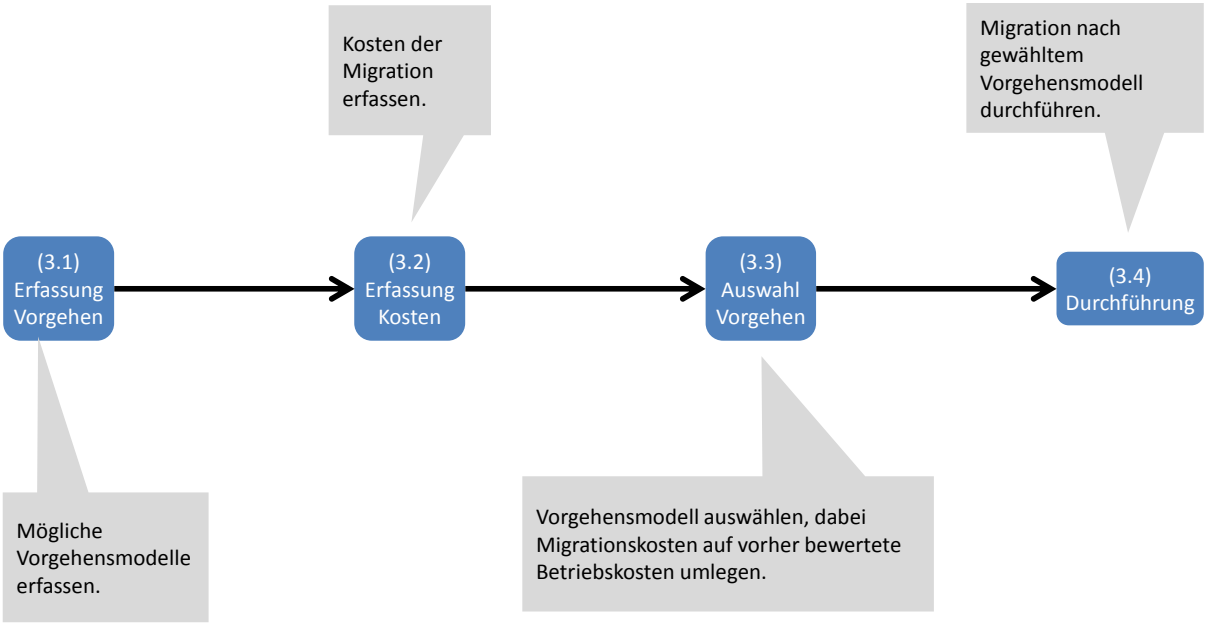


Abbildung 15 Phase 3, Übersicht

Phase 4 – Überwachung

Nachdem die gesamte Anwenderinfrastruktur oder Teile davon auf die entsprechende Zielplattformen migriert wurde, muss nun sichergestellt werden, dass die Migration auch wirklich die gewünschten Ergebnisse erzielt hat. Es muss zum einen überprüft werden, dass die in Phase 2 und 3 antizipierten finanziellen und energetischen Gewinne während des Betriebs der migrierten Anwenderinfrastruktur auch tatsächlich erreicht werden und in wieweit die im Projektauftrag definierten Kriterien (rechtlich, technisch etc.) eingehalten werden. Dazu sind Leistungskennzahlen, d.h. *Key Performance Indicators (KPI)*, zu definieren und von einer geeigneten Messinfrastruktur zu überwachen. Im Falle von Hybrid-Cloud-Szenarien sind dabei sowohl die Quell- als auch die Zielplattform zu überwachen und die gemessenen Daten entsprechend zu aggregieren. Da auch die Eigenschaften der Quell- und Zielplattform Änderungen unterliegen bzw. sie sich bei verschiedenen Auslastungsgraden ändern, ist die Anwenderinfrastruktur während des gesamten Betriebs zu überwachen. Der prinzipielle Ablauf von Phase 4 ist in Abbildung 16 zusammengefasst.

Im ersten Schritt der Überwachung sind alle relevanten KPIs durch Messung zu *erfassen*. So muss beispielsweise während der Laufzeit der Anwenderinfrastruktur permanent der Energieverbrauch gemessen werden. In welcher Häufigkeit das geschieht und welcher Wert bestimmt wird (z.B. Watt pro Stunde, Joule etc.) ist durch den KPI definiert. Ein Beispiel für einen KPI, der ein Dienstgütekriterium repräsentiert, wäre beispielsweise die Antwortzeit der Anwenderinfrastruktur auf Anfragen. Folglich müsste für diesen Indikator der Zeitraum zwischen der Anfrage des Klienten und der Antwort vom Migrationskandidaten gemessen werden.

Im zweiten Schritt werden die Annahmen aus Phase 2 und 3 validiert, indem sie mit den gemessenen KPI-Werten verglichen werden. Wie oben angedeutet, ist die Erfassung und Validierung permanent während des Betriebs der AIS durchzuführen.

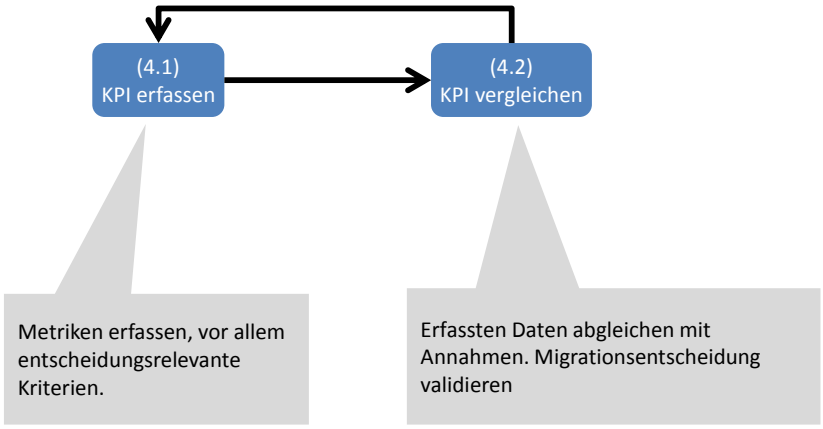


Abbildung 16 Phase 4, Übersicht

Entscheidungsunterstützungssystem auf Basis des Vorgehensmodells

Um das MIGRATE!-Vorgehensmodell softwareseitig nutzbar zu machen, wurde vom Projektpartner IAAS ein Entscheidungsunterstützungssystem entwickelt. Es handelt sich um ein mehrbenutzerfähiges Web- Tool mit persistenter Datenhaltung und einem dynamisch erweiterbaren Satz von Stammdaten. Da sich das Tool momentan noch in Entwicklung befindet, kann es im Folgenden nicht abschließend präsentiert werden. Die wichtigsten Konzepte werden vorgestellt und es wird ein Ausblick auf das Gesamtsystem gegeben.

Anforderungen

Die Entwicklung des Entscheidungsunterstützungssystems basiert auf einem Katalog von Anforderungen. Im Folgenden werden zunächst die wichtigsten funktionalen und anschließend die wichtigsten nichtfunktionalen Anforderungen vorgestellt.

Funktionale Anforderungen

Die funktionalen Anforderungen definieren die fachliche Funktionalität des Tools. Sie sollen zum einen vollumfänglich, zum anderen aber auch möglichst abstrakt sein, Die Funktionalen Anforderungen sollen unabhängig von ihrer konkreten Umsetzung sein, sie sollen keine Technologien vorgeben. Im Folgenden werden die wichtigsten funktionalen Forderungen vorgestellt.

1. Entscheidungsunterstützung

   Eine wesentliche Funktion des Tools ist es, den Benutzer bei der Entscheidungsfindung zu unterstützen. Relevante Daten müssen erfasst und gespeichert werden. Die erfassten Daten sollen dem Benutzer so dargestellt werden, dass sie ihn bei der Entscheidungsfindung unterstützen. Das bedeutet zum Beispiel, dass Daten passend sortiert, nebeneinandergestellt oder hervorgehoben werden. Alle relevanten Daten sollen schnell und intuitiv zugreifbar sein.

2. MIGRATE!-Vorgehensmodell

   Das Tool soll das MIGRATE!-Vorgehensmodell implementieren und den Benutzer bei dessen Durchführung unterstützen.

3. Wiederverwendung des AIS Datenmodells

   In Arbeitspaket 1 des MIGRATE!-Projektes wurde ein Datenmodell sowie ein dazugehöriger Editor zur Erfassung bestehender Anwenderinfrastrukturen entwickelt. Das

Tool soll den Importn der mit diesen Tools modellierten Anwenderinfrastrukturen unterstützen.

4. Persistente Datenspeicherung

   Alle vom Benutzer eingegebenen Daten sollen persistent gespeichert werden.

5. Hilfesystem

   Dem Benutzer soll, wann immer es sinnvoll ist, ein kontextsensitives Hilfesystem zur Verfügung stehen.

6. Projektverwaltung

   Es soll möglich sein, mehrere Projekte zu erstellen und zu verwalten. Ein Projekt entspricht einem Migrationsvorhaben. Es soll private und öffentliche Projekte geben. Private Projekte sind nur für einen Benutzer (dem Ersteller bzw. Besitzer) sichtbar, sie können nur von diesem geöffnet und bearbeitet werden. Öffentliche Projekte sind für alle Benutzer sichtbar. Sie können zu jedem Zeitpunkt nur von maximal einem Benutzer geöffnet oder bearbeitet werden.

7. Projektvariationen

   In bestimmten Abschnitten eines Projektes können Variationen eines Projektes gebildet werden. Variationen repräsentieren alternative Entscheidungen im Vorgehensmodell, die parallel weiter bearbeitet werden. Es soll möglich sein, alternative Projektdurchläufe zu vergleichen.

8. Erweiterbarkeit

   Die Menge der Kriterien Gruppen, der einzelnen Kriterien sowie der möglichen Zielinfrastrukturen soll dynamisch erweiterbar sein. Es sollen jederzeit neue Kriterien Gruppen, Kriterien und Zielinfrastrukturen angelegt werden können.

9. Navigation

   Die Navigation soll jeder klar ersichtlich und verständlich sein. Die Phasen und Schritte des MIGRATE!-Vorgehensmodells sollen klar ersichtlich sein. Der Benutzer soll aktiv durch die einzelnen Schritte geleitet werden. Es soll nur möglich sein, was sinnvoll und erlaubt ist. Beispielsweise sind manche Schritte im Vorgehensmodell verpflichtend, anderer Schritte dürfen übersprungen werden.

10. Projektauftrag

    Der Projektauftrag ist ein zentrales Element, er definiert den Kontext in dem ein Vorhaben ausgeführt wird. Der Projektauftrag soll jederzeit aufrufbar sein, ohne den Benutzer aus dem aktuellen Schritt zu reißen.

11. Mehrbenutzerbetrieb

    Das Tool soll Mehrbenutzerbetrieb unterstützen.

Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen sind unabhängig von den fachlichen Funktionen des Tools. Sie beeinflussen die nichtfunktionalen Eigenschaften des Tools und sind damit ebenso wie die funktionalen Anforderungen entscheidend für die Eignung und Akzeptanz des entwickelten Systems. Im Folgenden werden einige der nichtfunktionalen Anforderungen vorgestellt.

1. Zuverlässigkeit

   Die Software soll reproduzierbar zuverlässig arbeiten. Auf Benutzereingaben soll ordnungsgemäß reagiert werden.

2. Benutzbarkeit

   Die Software soll intuitiv bedienbar sein und dabei aktuellen Konventionen gehorchen. Der Aufwand zur Erlernung der Software soll minimal gehalten werden.

3. Wartbarkeit

   Die Software soll veränderbar sein und damit Fehlerbehebungen, Anpassungen und Erweiterungen ermöglichen.

4. Sicherheit

   Daten sollen verfügbar, integer und sicher abgespeichert werden. Sensible Daten sollen verschlüsselt werden.

Datenmodell

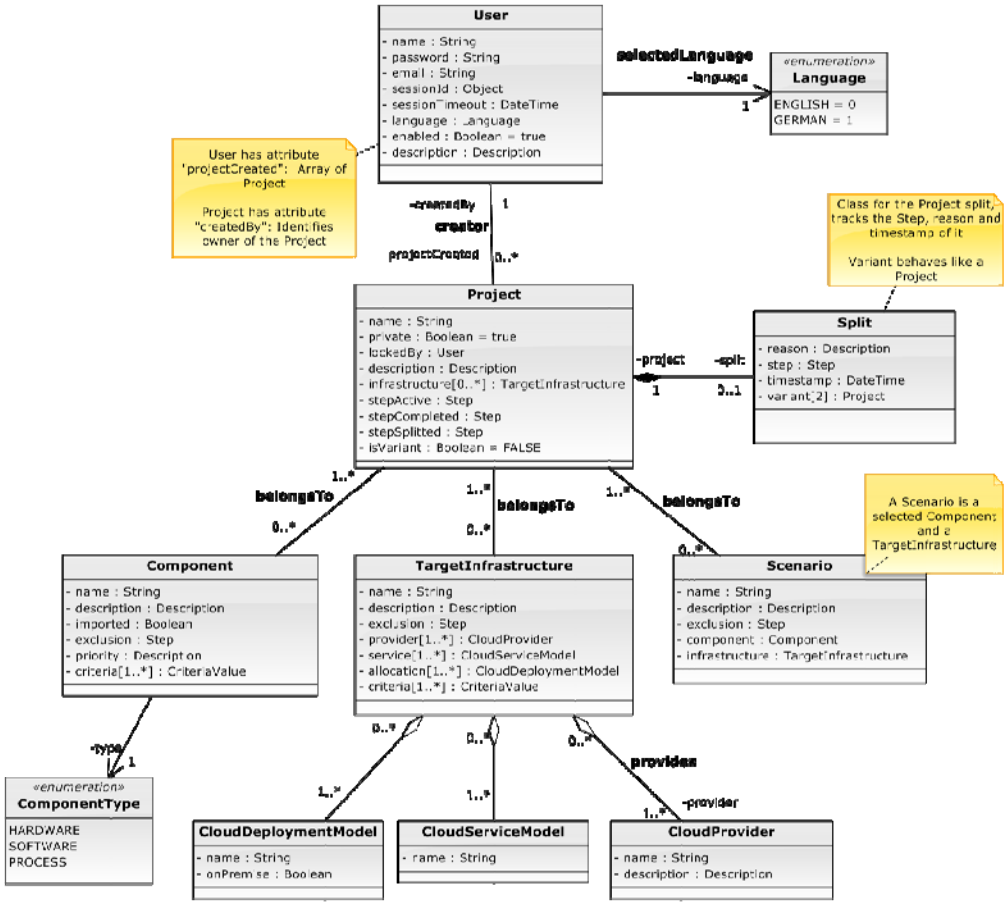
Das Datenmodell wurde auf Basis des Vorgehensmodells sowie der funktionalen und nicht-funktionalen Anforderungen definiert. Aus Übersichtsgründen sind nicht alle Details und Beziehungen dargestellt.

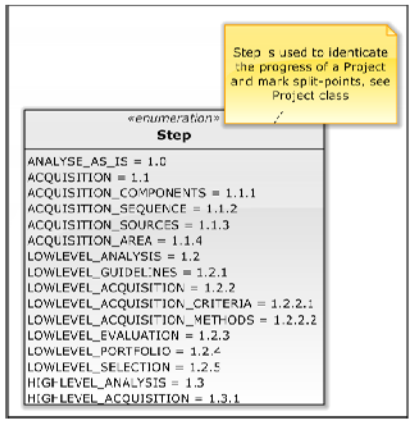
Die Mehrbenutzerfähigkeit wird durch die Entität *User* modelliert. Die zentrale Entität ist *Project*, ein Projekt repräsentiert ein konkretes Migrationsvorhaben. Die Möglichkeit, zu einem bestimmten Zeitpunkt eine Variante eines Projektes zu erstellen wird durch die Beziehung zur Entität *Split* dargestellt. Ein Split beschreibt, in welchem Schritt des Vorgehensmodells eine Variante erstellt wurde. Die Variante selbst wird wiederum durch die Entität Project modelliert, wobei Varianten Projekte erst ab dem Schritt an dem die Variante erstellt wurde mit Daten gefüllt sind.

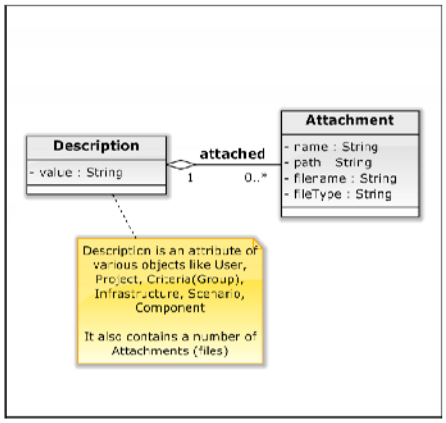
Die grundlegende Datenstruktur eines Projektes umfasst Beziehungen zu den Entitäten *Component*, *TargetInfrastructure* und *Scenario*. Die Entität Component repräsentiert die Komponenten der bestehenden Anwenderinfrastruktur, deren Migration in eine Cloud-Umgebung untersucht werden soll. Die Entität *TargetInfrastructure* modelliert mögliche Zielinfrastrukturen und setzt sich zusammen aus einem Cloud Deployment Modell (*CloudDeploymentModel*), eine Cloud Service Modell (*CloudServiceModel*) und einem Cloud Anbieter (*CloudProvider*). Die Entität *Scenario* stellt Migrationsszenarien dar und verbindet dabei Komponenten und deren mögliche Ziel Infrastrukturen.

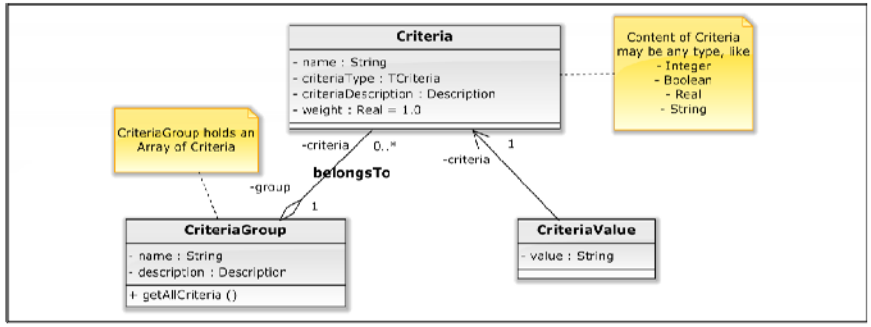
Die Aufzählung *Step* definiert die einzelnen Schritte des Vorgehensmodells. Die definierten Werte werden von diversen anderen Entitäten referenziert. Die Entitäten *Description* und *Attachement* werden ebenfalls von mehreren Entitäten referenziert. Sie ermöglichen es, Textkommentare und Dateien zu Entitäten hinzuzufügen.

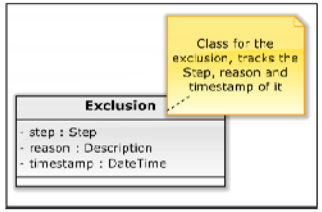
Das vorgestellte Datenmodell wurde mit Hilfe entsprechender Hilfswerkzeuge auf eine Menge von Java Klassen abgebildet und durch ein Persistenz Framework auf eine relationale Datenbankstruktur abgebildet. Dieses modellgetriebene Vorgehen vereinfacht es, Änderungen im Datenmodell vorzunehmen und diese sowohl in den Java Klassen als auch im Datenbankschema nachzuziehen.


User
- name : String
- password : String
- email : String
- sessionId : Object
- sessionTimeout : DateTime
- language : Language
- enabled : Boolean = true
- description : Description
selectedLanguage
-language
1
«enumeration»
Language
ENGLISH = 0
GERMAN = 1
User has attribute "projectCreated": Array of Project
Project has attribute "createdBy": Identifies owner of the Project
-createdBy
1
creator
projectCreated
0..*
Class for the Project split, tracks the Step, reason and timestamp of it
Variant behaves like a Project
Project
- name : String
- private : Boolean = true
- lockedBy : User
- description : Description
- infrastructure[0..*] : TargetInfrastructure
- stepActive : Step
- stepCompleted : Step
- stepSplitted : Step
- isVariant : Boolean = FALSE
-project
1
-split
0..1
Split
- reason : Description
- step : Step
- timestamp : DateTime
- variant[2] : Project
1..*
belongsTo
0..*
A Scenario is a selected Component and a TargetInfrastructure
Component
- name : String
- description : Description
- imported : Boolean
- exclusion : Step
- priority : Description
- criteria[1..*] : CriteriaValue
TargetInfrastructure
- name : String
- description : Description
- exclusion : Step
- provider[1..*] : CloudProvider
- service[1..*] : CloudServiceModel
- allocation[1..*] : CloudDeploymentModel
- criteria[1..*] : CriteriaValue
Scenario
- name : String
- description : Description
- exclusion : Step
- component : Component
- infrastructure : TargetInfrastructure
-type
1
«enumeration»
ComponentType
HARDWARE
SOFTWARE
PROCESS
provides
-provider
CloudDeploymentModel
- name : String
- onPremise : Boolean
CloudServiceModel
- name : String
CloudProvider
- name : String
- description : Description


Step is used to identicate the progress of a Project and mark split-points, see Project class
«enumeration»
Step
ANALYSE_AS_IS = 1.0
ACQUISITION = 1.1
ACQUISITION_COMPONENTS = 1.1.1
ACQUISITION_SEQUENCE = 1.1.2
ACQUISITION_SOURCES = 1.1.3
ACQUISITION_AREA = 1.1.4
LOWLEVEL_ANALYSIS = 1.2
LOWLEVEL_GUIDELINES = 1.2.1
LOWLEVEL_ACQUISITION = 1.2.2
LOWLEVEL_ACQUISITION_CRITERIA = 1.2.2.1
LOWLEVEL_ACQUISITION_METHODS = 1.2.2.2
LOWLEVEL_EVALUATION = 1.2.3
LOWLEVEL_PORTFOLIO = 1.2.4
LOWLEVEL_SELECTION = 1.2.5
HIGHLEVEL_ANALYSIS = 1.3
HIGHLEVEL_ACQUISITION = 1.3.1


Description
- value : String
attached
1
0..*
Attachment
- name : String
- path : String
- filename : String
- fileType : String
Description is an attribute of various objects like User, Project, Criteria(Group), Infrastructure, Scenario, Component
It also contains a number of Attachments (*files)


Criteria
- name : String
- criteriaType : TCriteria
- criteriaDescription : Description
- weight : Real = 1.0
Content of Criteria may be any type, like
- Integer
- Boolean
- Real
- String
CriteriaGroup holds an Array of Criteria
-criteria
0..*
1
-criteria
-group
belongsTo
1
CriteriaGroup
- name : String
- description : Description
+ getAllCriteria ()
CriteriaValue
- value : String


Class for the exclusion, tracks the Step, reason and timestamp of it
Exclusion
- step : Step
- reason : Description
- timestamp : DateTime

Graphische Benutzerschnittstelle

Der Starbildschirm des Entscheidungsunterstützungssystems ist in Abbildung 17 dargestellt. Zentrales Element ist eine schematische Übersichtsgrafik des MIGRATE!-Vorgehensmodells für die Migration bestehender Anwenderinfrastrukturen in eine Cloud-Umgebung. Unterhalb des Willkommenstextes sind die einzelnen Phasen sowie der Projektauftrag noch einmal einzeln dargestellt, dies bildet die Hauptnavigation des Tools. Im oberen rechten Bereich des Bildschirms befinden sich weitere Navigationsschaltflächen. Der Punkt *Dashboard* führt auf den Startbildschirm zurück. Unter *Projects* können Projekte erstellt oder bestehende Projekte geöffnet werden. Unter *Administration* können verschiedene Einstellungen angepasst werden, beispielsweise können neue Benutzer angelegt oder auch die vordefinierte Menge der Kriterien verwaltet werden. Unter *MyAccount* kann der aktuell angemeldete Benutzer Einstellungen wie die Anzeigesprache oder seine Emailadresse bearbeiten.

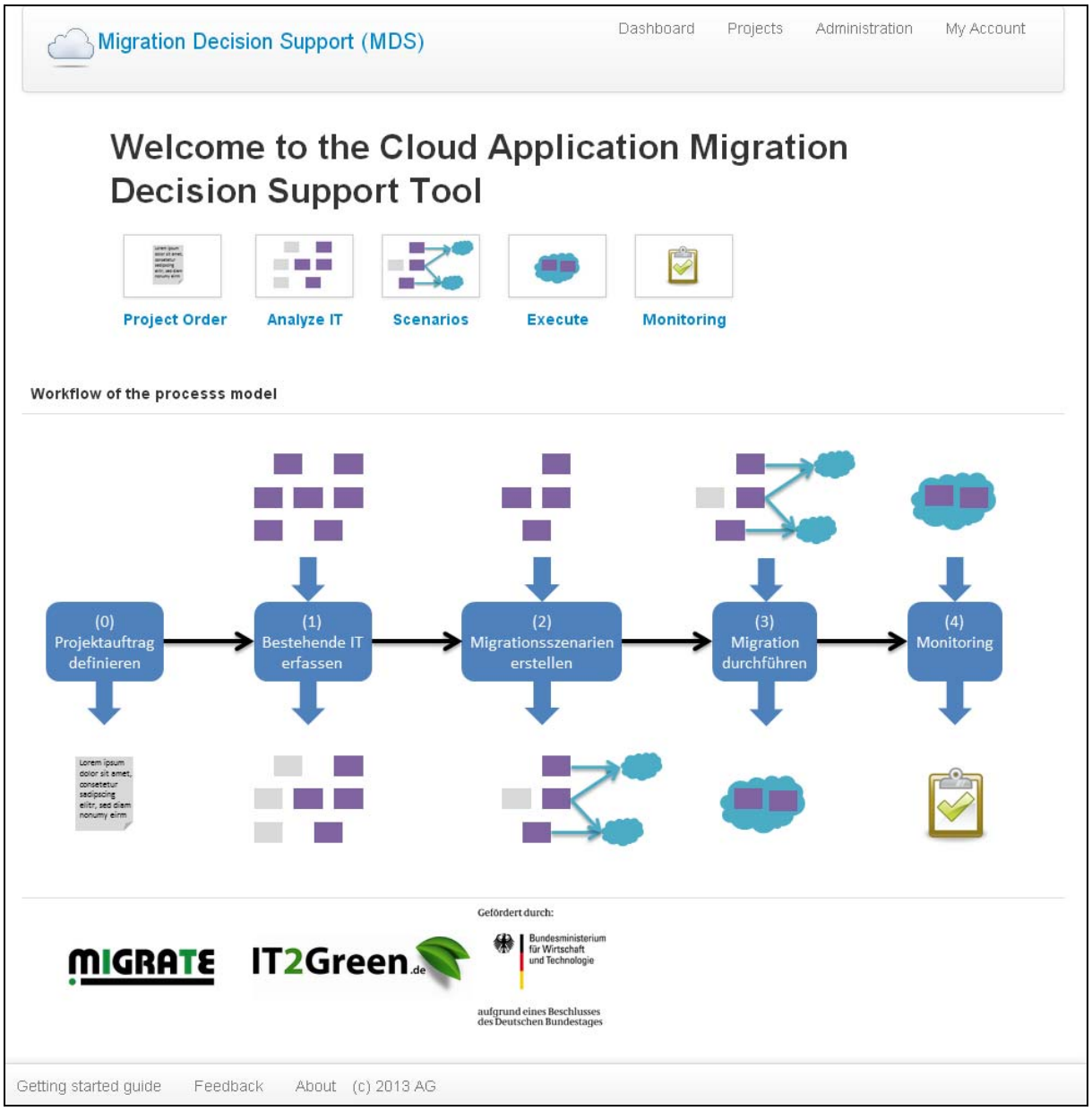


Abbildung 17 Startbildschirm Entscheidungsunterstützungssystem

Während Abbildung 17 ein Screenshot des realen Tools ist, wird der konzeptionelle Aufbau der GUI im Folgenden anhand von Mockups illustriert. Die Erfassung der Komponenten der bestehenden Anwenderinfrastruktur in Phase 1 wird in Abbildung 18 gezeigt. Über die

Schaltfläche *Import* können AIS Modelle des in AP1 entwickelten Modellierungstools importiert werden. Alle erfassten IT Komponenten werden in einer Tabelle dargestellt, einzelne Einträge können hinzugefügt, editiert oder gelöscht werden. Auf der linken Seite des Bildschirms befindet sich die Navigation innerhalb der aktuellen Phasen. Mit den Schaltflächen *Next step* und *Previous step* wird der Benutzer durch die einzelnen Schritte geführt. Die Schaltfläche *Current Project* ist in jedem Schritt sichtbar, sie öffnet den Projektauftrag in einem separaten Popup Fenster.

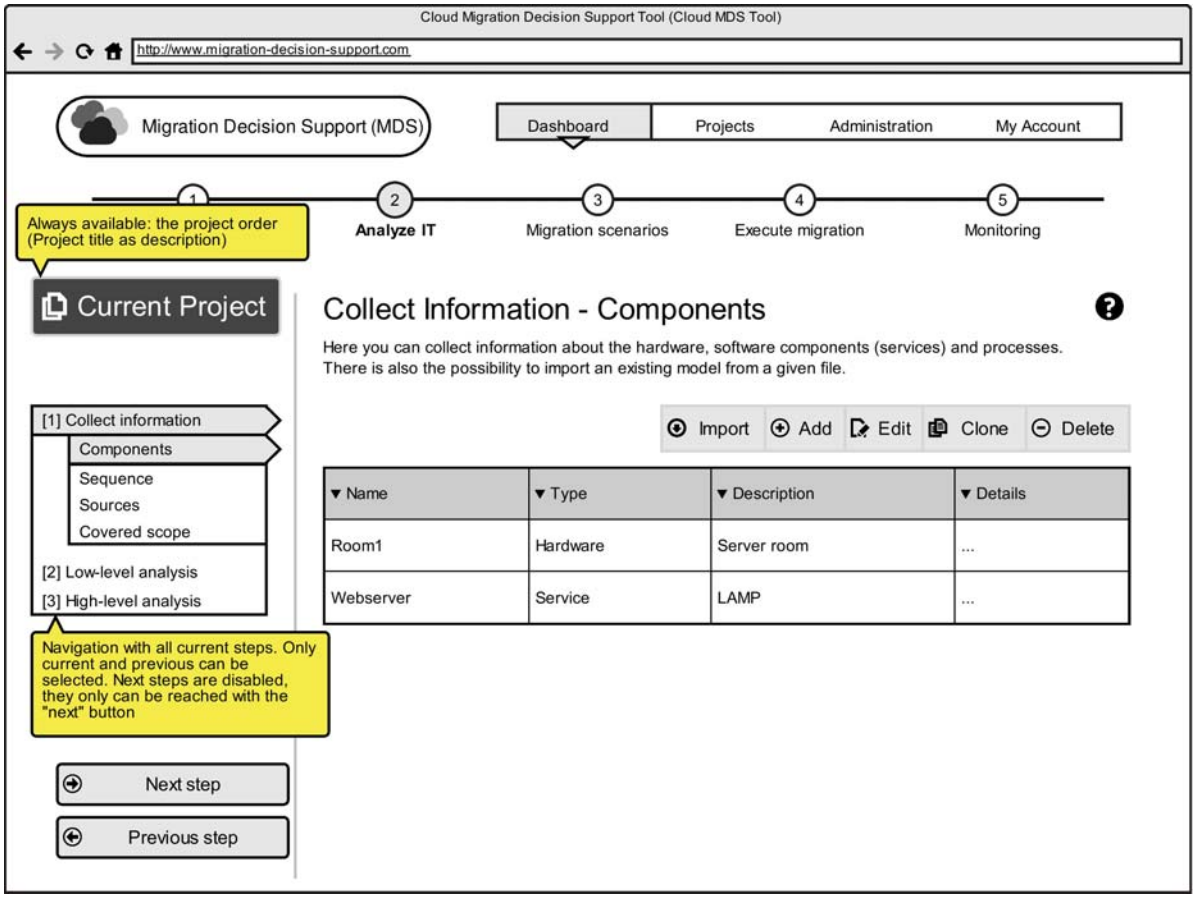


Abbildung 18 GUI Mockup Phase 1, Erfassung der AIS Komponenten

Die Projektverwaltung ist in Abbildung 19 dargestellt, in dieser Maske könne Projekte erstellt, geöffnet und gelöscht werden. Die Möglichkeit, die Stammdaten des Tools bearbeiten und erweitern zu können wird in Abbildung 20 gezeigt. In dieser Maske können zusätzliche Kriterien und Kriterien Gruppen angelegt werden. Kriterien können einen von mehreren vordefinierten Typen besitzen (numerisch, textuell, boolesch).

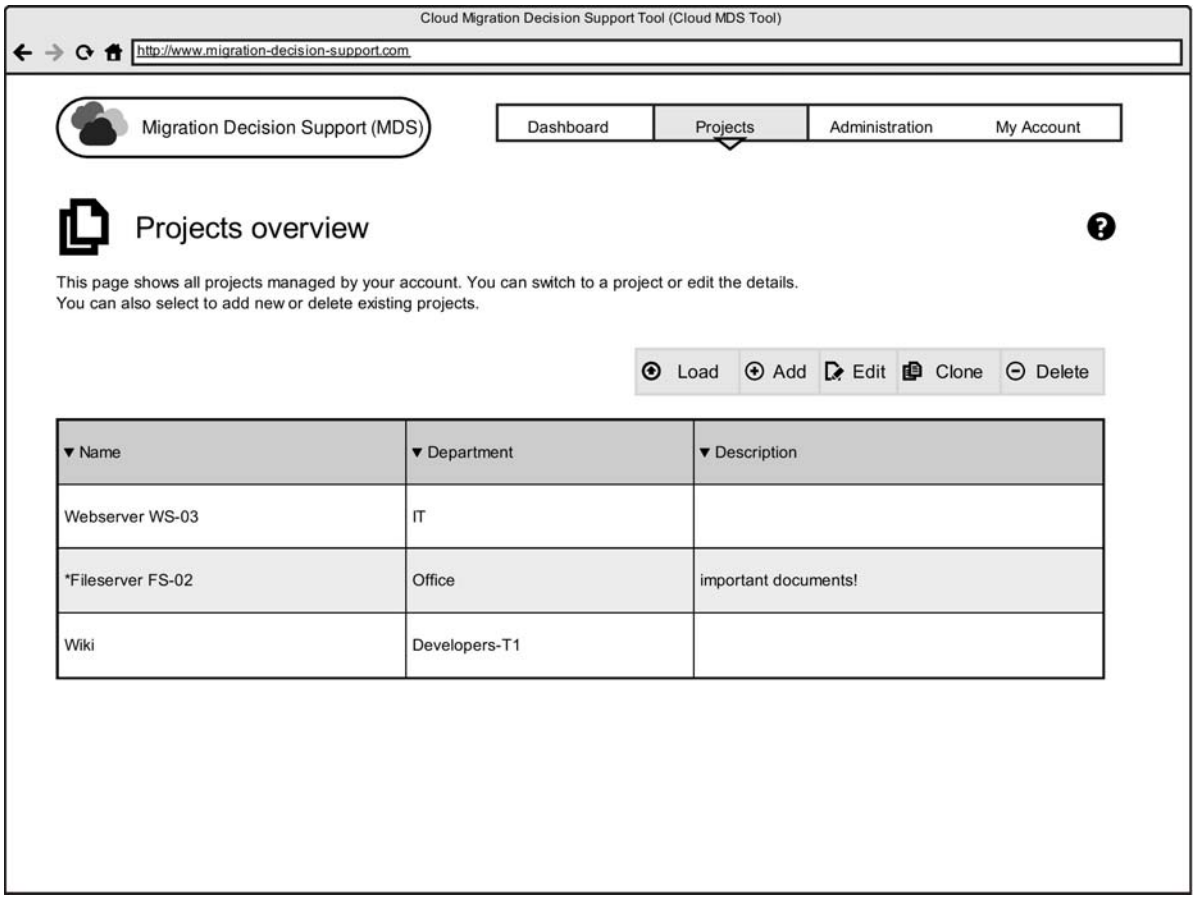


Abbildung 19 GUI-Mockup, Projektverwaltung

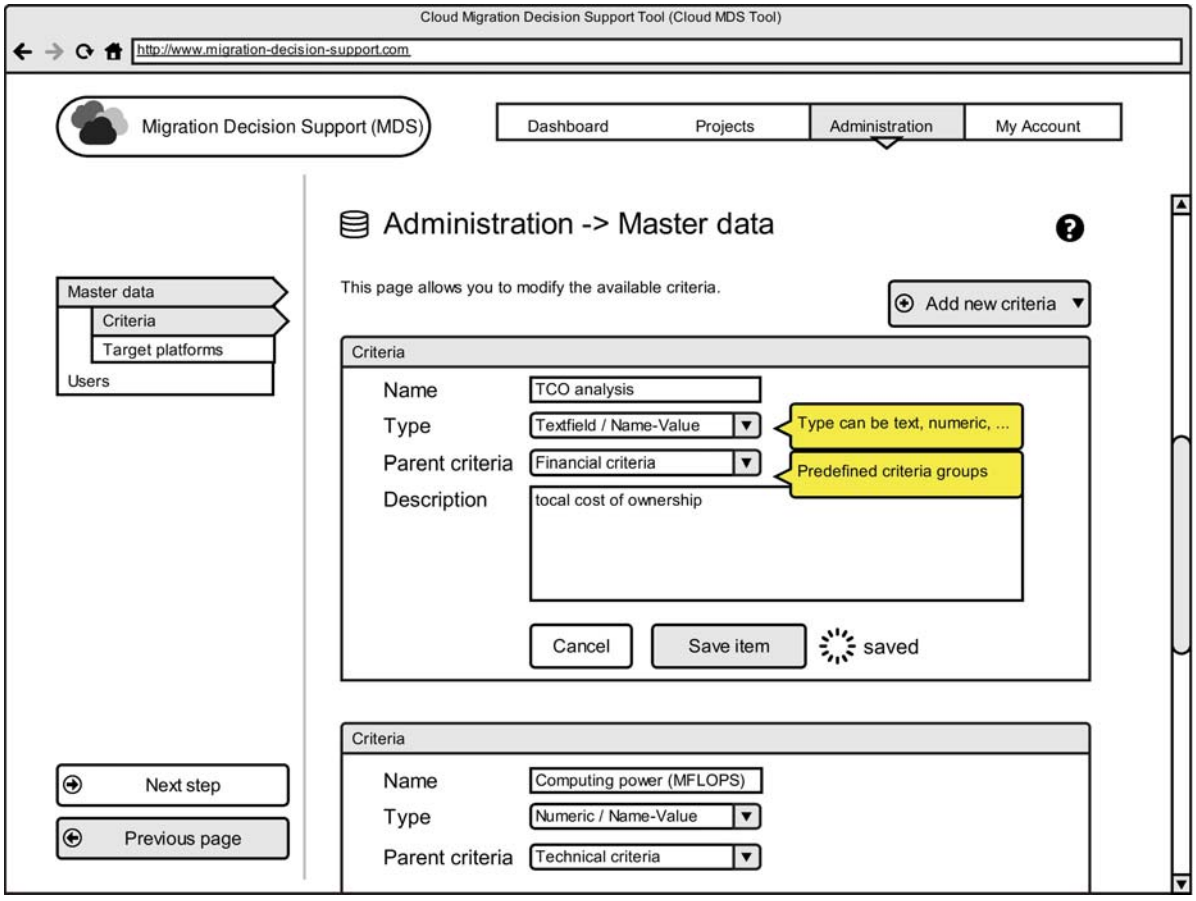


Abbildung 20 GUI-Mockup, Stammdatenverwaltung

Einzelaspekte des Vorgehensmodells: Integration mit Methoden und Tools aus AP1

Um die Anwenderinfrastruktur in Phase 1 des Vorgehensmodells zu ermitteln wurde in AP1 eine entsprechende Methode beschrieben (siehe Meilensteinbericht M1.1). Die Methode unterteilt die Anwenderinfrastruktur in drei aufeinander aufbauende Schichten – die *geschäftliche Schicht*, die *logische Schicht* und die *physische Schicht*. Die geschäftliche Sicht, stellt eine strukturierte Beschreibung der Ablauforganisation dar. Konkret bedeutet dies, dass dort die für die Lösung einer komplexen Arbeitsaufgabe notwendigen Schritte als Prozesse modelliert werden. Unter der geschäftlichen Sicht befindet sich die logische Schicht. Diese Schicht beschreibt die notwendigen Softwaretopologie (Anwendungen, Middleware, Betriebssysteme etc.), um die Aufgaben der geschäftlichen Schicht zu realisieren. Die physische Schicht beschreibt wiederum die Hardware-Infrastruktur (Server, Netzwerkkomponenten) die für den Betrieb der Softwarekomponenten in der logischen Schicht notwendig ist.

Zur Unterstützung der in AP1 entwickelten Methode wurde ein entsprechendes Modellierungstool entwickelt. Das auf Eclipse basierende Tool stellt für jede der drei Schichten einen eigenen Editor zur Verfügung und bietet außerdem die Möglichkeit, den Energieverbrauch von der physischen Schicht über die logische Schicht bis zur geschäftlichen Schicht zu propagieren. Somit kann auch der Energieverbrauch von einzelnen Schritten der Ablauforganisation ermittelt werden.

Die Ablauforganisation der geschäftlichen Schicht wird dabei mit einer erweiterten Version des Eclipse BPEL Designer als BPEL Prozess modelliert, da dieses Tool die notwendigen Modellierungswerkzeuge anbietet, um die Aufgaben und deren Ausführungsreihenfolge zu modellieren. Im Abbildung 21 ist der Beispiel BPEL Prozess „Urlaubsantrag“ dargestellt, der mit dem BPEL Designer erstellt wurde. Der Prozess beschreibt die Schritte (Aktivitäten) die für die Genehmigung eines Urlaubsantrags ausgeführt werden müssen.

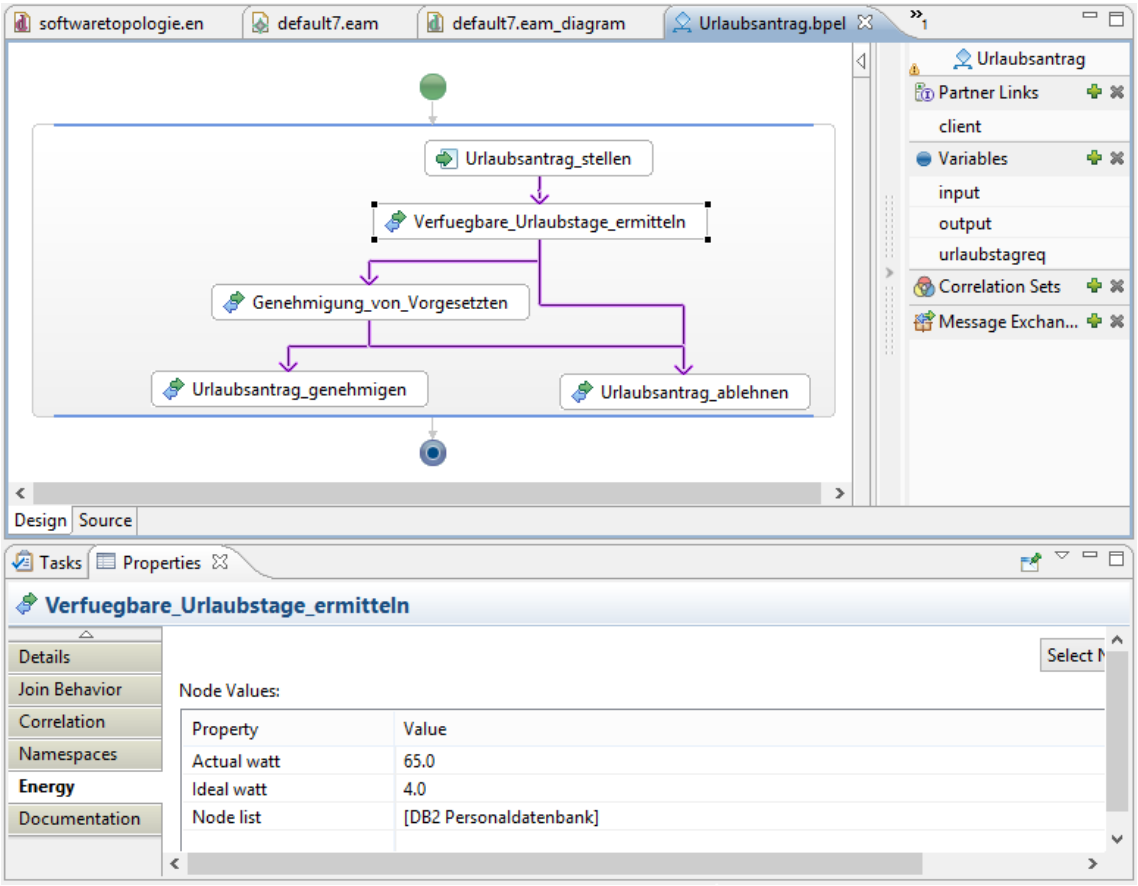


Abbildung 21 Beispielprozess „Urlaubsantrag“ im BPEL Designer

Der BPEL Designer wurde von uns dahingehend erweitert, dass es nun auch möglich ist, den Energieverbrauch der Schritte bzw. Aktivitäten berechnen zu lassen, die mithilfe der Softwarekomponenten der logischen Schicht realisiert werden. So überprüft beispielsweise die Aktivität „Verfuegbare_Urlaubstage_ermitteln“ anhand der Personaldatenbank, ob der urlaubsantragstellende Mitarbeiter noch ausreichend Urlaubstage zur Verfügung hat. Der Energiebedarf pro Datenbankabfrage wird dabei auf Basis einer vorher definierten Formel berechnet (siehe unten).

Abbildung 22 zeigt den Editor, für die Modellierung der logischen Schicht. Bezogen auf das Beispiel, sind das die Softwarekomponenten die den Prozess „Urlaubsantrag“ unterstützen. In Anlehnung an die TOSCA Spezifikation werden im Editor die Softwarekomponenten als *NodeType* und die Beziehungen zwischen ihnen als *RelationshipeTypes* bezeichnet. Die Komponente *Apache Tomcat* ist ein Servlet Container, der die Benutzeroberflächen zu Verfügung stellt, mit denen Urlaubsanträge erstellt und bearbeitet werden können. Über die Benutzeroberfläche kann auch auf die Vertrags- und Urlaubsdaten der Mitarbeiter zugegriffen werden, die in der zentralen *DB2 Personaldatenbank* gespeichert sind. Beide Komponenten sind auf dem Betriebssystem *Windows 2010 Server* installiert. Softwarekomponenten können wiederum mit Komponenten der physischen Schicht assoziiert werden, um so zum einen Abhängigkeiten zwischen den Komponenten zu modellieren und um zum anderen den Energieverbrauch einer Softwarekomponente zu berechnen.

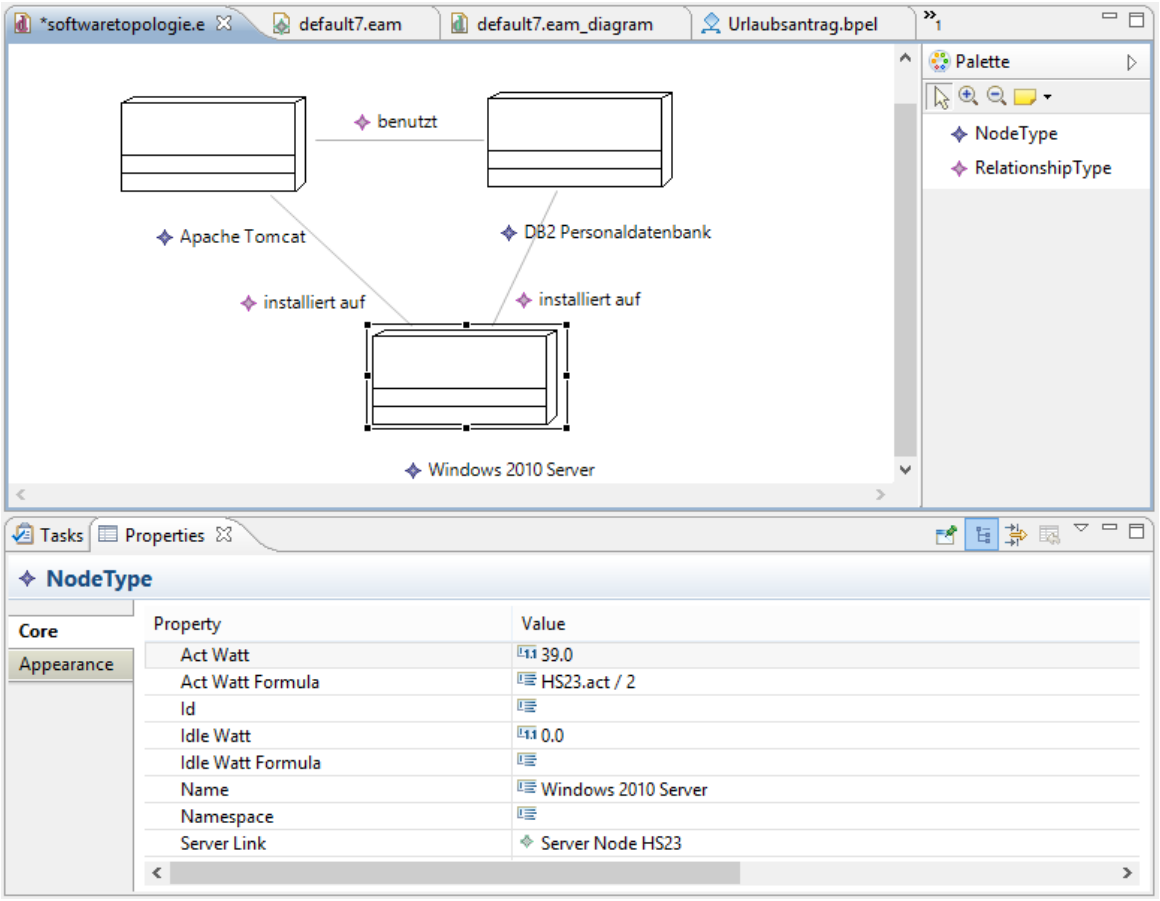


Abbildung 22 Editor für Softwarekomponenten der logischen Schicht

Die Komponente Windows 2010 Server ist beispielsweise auf Blade Server HS23 installiert. Diese Beziehung wurde in der Komponenten-Eigenschaft *Server Link* definiert. Auf Basis des Energieverbrauchs des Servers kann der anteilige Energieverbrauch (*Act Watt*) einer Software-Komponente ebenfalls über eine mathematische Formel berechnet werden. Dazu wurde ein entsprechender Formeleditor entwickelt. Im Formeleditor in Abbildung 23 wurde

zum Beispiel definiert, dass der Energieverbrauch des Windows 2010 Server anteilig die Hälfte der Energie des Blade Servers HS23 verbraucht.

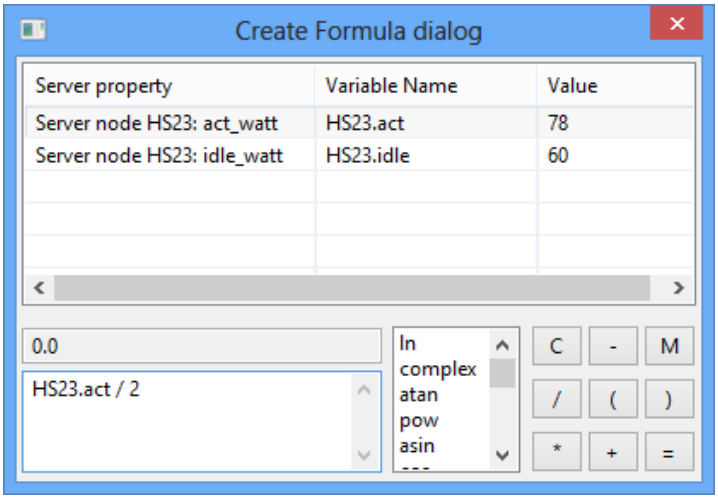


Abbildung 23 Formeleditor

Falls keine mathematischen Formeln zur Verfügung stehen, um den Energieverbrauch anteilig zu berechnen, können die Werte auch direkt angegeben werden.

Der Editor für die physische Schicht ist in Abbildung 24 dargestellt. Neben Servern können hier auch Netzwerkgeräte, Klient-Computer, Komponenten für die unterbrechungsfreie Stromversorgung (USV) und für die Kühlung von Hardware modelliert werden. Um die Übersichtlichkeit zu erhöhen, können die physischen Komponenten lokal gruppiert werden. In Abbildung 24 befinden sich in *Severraum 1* neben den oben erwähnten Blade Server HS23 noch der *Router X45* und die USV *Energy4Ever*. Als Backup-Lösung gibt es zusätzlich noch ein *HS20* Blade Center in *Severraum 2*, das allerdings nicht aktiv ist (Act_Watt 0). Der Energieverbrauch der Komponenten auf dieser Ebene muss manuell angegeben werden. Die Werte können zum Beispiel durch Messungen oder Gerätespezifikationen bestimmt werden.

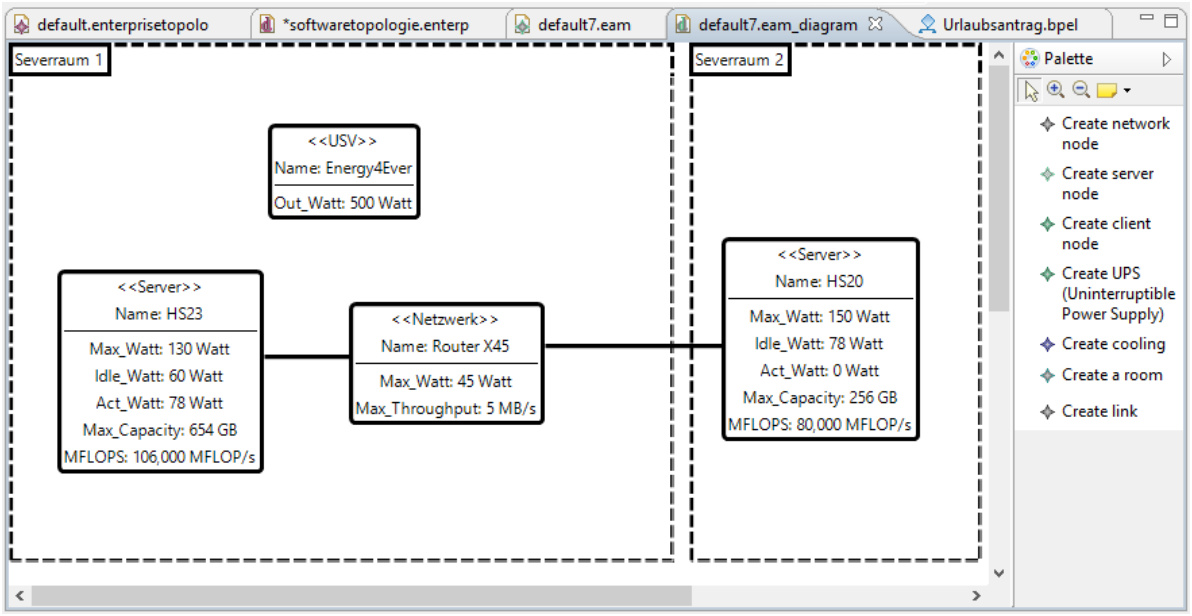


Abbildung 24 Editor für Hardwarekomponenten der physischen Schicht

AP3.3 Definition von Cloud Service Paketierungen

Die ersten drei Phasen des vorgestellten Vorgehensmodells für die Cloud-Migration behandeln vor allem Aspekte der Entscheidungsfindung. Es wird anhand mehrerer funktionaler und nichtfunktionaler Kriterien bestimmt, welche Komponenten in welche Zielinfrastrukturen migriert werden können und sollen. Phase 3 ("Planung und Durchführung der Migration", beschrieben in Kapitel 0) beschreibt die anschließende Durchführung der Migrationsaufgabe. In diesem Kapitel wird vorgestellt, wie TOSCA[2] zusammen mit dafür entwickelten Tools und Komponenten dafür verwendet werden kann, komplexe Anwendungsstrukturen automatisiert und portabel in Cloud-Umgebungen zu provisionieren. Neben der Migration der Anwendung selbst kann TOSCA zusätzlich dazu verwendet werden, die nötigen Verwaltungsprozesse wie beispielsweise Sicherungs-, Upgrade- oder Prüfaufgaben formal zu erfassen und damit die Verwaltung der in der Cloud provisionierten Anwendung zu automatisieren.

Abbildung 25 zeigt den grundlegenden Ablauf der Migration einer Anwendungsstruktur in eine Cloud-Umgebung unterstützt durch TOSCA. Um eine Anwendung automatisiert in eine Cloud-Umgebung provisionieren zu können, wird sie in Schritt 1 zunächst im TOSCA-Format beschrieben. Anwendungsmodelle in TOSCA werden als Graph Struktur beschrieben und bestehen im Wesentlichen aus zwei Modellierungskonstrukten. Die *NodeTemplates* sind die Knoten des Graphen und repräsentieren die einzelnen Komponenten der Anwendungsstruktur. Die Beziehungen zwischen einzelnen Komponenten werden durch die *RelationshipTemplates* dargestellt. Das TOSCA Modell wird auf Basis der bestehenden Anwendungsstruktur und abhängig von der gewählten Zielplattform aufgebaut. Im dargestellten Beispiel wird eine Anwendung mit dem Namen „MyApp" auf einem Tomcat Application Server betrieben der wiederum auf einem Server mit Windows Betriebssystem installiert ist. Das TOSCA-Modell (rechts) ähnelt dem ursprünglichen Modell, wurde jedoch an die Ziel Cloud Infrastruktur angepasst. Der physische Server wurde durch eine virtuelle Maschine (VM) ersetzt, das Windows Betriebssystem wurde durch ein Linux System ersetzt. Die Einführung virtueller Maschinen ist zwingend notwendig (Cloud-Umgebungen sind immer virtualisierte Umgebungen), die Änderung des Betriebssystems ist in diesem Fall optional und kann beispielsweise aus Kostengründen erfolgen (Einsparung von Lizenzkosten).

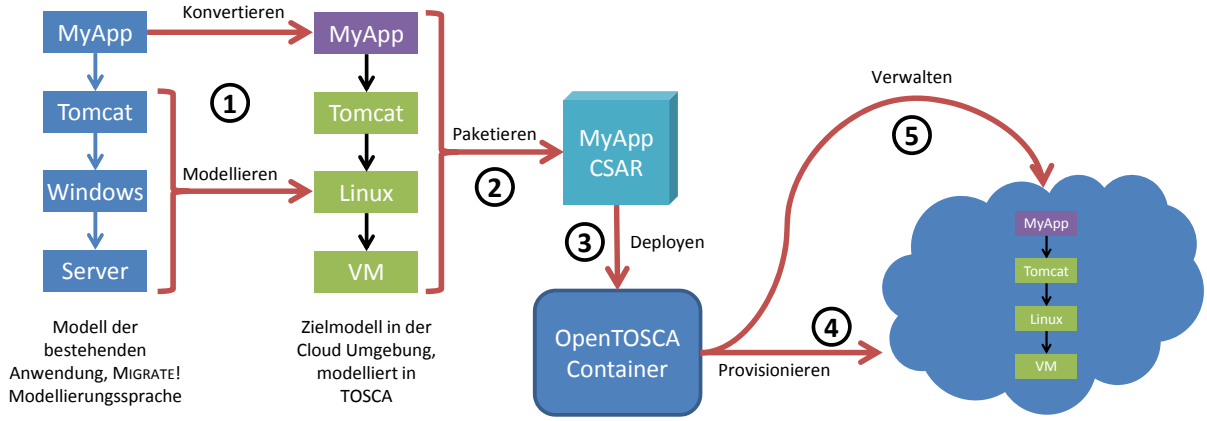


Abbildung 25 Cloud Migration unterstützt durch TOSCA

Die zur Modellierung der Anwendungsstruktur verwendeten Komponenten müssen entweder bereits im TOSCA Format vorliegen oder andernfalls nach Bedarf erstellt werden. Für den Bereich der virtuellen Maschinen, der Betriebssysteme und der Middleware wurden am IAAS im Teilprojekt „Lego4TOSCA" bereits typische Modellierungskomponententypen erstellt. Diese in TOSCA *NodeTypes* genannten Komponenten können zur einfachen Modellierung

[2] http://www.tosca-open.org/

von Anwendungsstrukturen verwendet werden. Spezifische Komponenten, wie im dargestellten Beispiel die Anwendung „MyApp“, müssen weiterhin zunächst in das TOSCA Format überführt werden. Diese bedeutet im Allgemeinen nicht, dass die Anwendung selbst verändert werden muss. Die Anwendung wird vielmehr im TOSCA Format beschrieben. Zu dieser Beschreibung gehören unter anderem die funktionalen Schnittstellen, die zum Deployment nötigen Artefakte, wichtige Eigenschaften der Anwendung und die zur Verfügung gestellte Verwaltungsschnittstellen.

Nachdem die Anwendungsstruktur im TOSCA Format beschrieben wurde, werden in Schritt 2 die erzeugten Modelldaten zusammen mit allen Artefakten, die für das konkrete Deployment der beschriebenen Anwendung nötig sind, zu einem Paket zusammen gefügt. Um beispielsweise die in TOSCA beschriebene Anwendung „MyApp“ deployen zu können, wird neben der Beschreibung der Anwendung auch eine ausführbare Version der Anwendung, in diesem Fall eine WAR Datei (Web Application Archive), benötigt. Weitere typische Artefakt Typen sind VM Images, Skripte, Archivdateien oder auch Verweise auf externe Artefakte. Das von TOSCA spezifizierte Paketformat ist das Cloud Service Archive (CSAR) Format, es wird im Folgenden ausführlich beschrieben.

Als Vorbereitung für die Provisionierung der beschrieben und paketierten Anwendungsstruktur muss das erzeugte Paket in Schritt 3 zunächst auf einem TOSCA Container deployt werden. Ein TOSCA Container ist in der Lage, CSAR Pakete einzulesen und diese so zu verarbeiten, dass die enthaltene Anwendungsstruktur anschließend automatisiert provisioniert und verwaltet werden kann. Der Container liest die TOSCA-Modelldaten ein und verarbeitet alle verwendeten Artefakte. Der *OpenTOSCA* Container ist ein am IAAS entwickelter TOSCA Container[3]. Seine Architektur und Funktionsweise werden unter Punkt AP3.4 detailliert vorgestellt.

Beim Deployment eines CSAR-Pakets auf dem TOSCA Container werden unter anderem die im TOSCA-Modell beschriebenen Pläne deployt und können anschließend ausgeführt werden. Pläne beschreiben im Allgemeinen Verwaltungsaufgaben, welche auf der gesamten Anwendungsstruktur ausgeführt werden. Eine speziellen Verwaltungsaufgabe, das initiale Provisionieren einer Anwendungsstruktur, wird in TOSCA durch einen vordefinierten Plan Typ, den sogenannten *Build Plan*, beschrieben. In Schritt 4 wird dieser Build Plan gestartet um die Anwendungsstruktur in einer Cloud-Umgebung zu provisionieren. Details der Provisionierung können zu diesem Zeitpunkt noch angepasst werden. Voraussetzung dafür ist, dass der Build Plan entsprechende Eingabeparameter anbietet. Ein großer Vorteil der Verwendung von TOSCA ist, dass die erstellten CSAR Pakete im Allgemeinen von jedem TOSCA kompatiblen Container verarbeitet werden können. Zudem definiert TOSCA unterschiedliche Konstrukte, die es erlauben, eine Anwendungsstruktur so zu modellieren, dass sie portabel in unterschiedliche Cloud-Umgebungen provisioniert werden kann. Die nicht-funktionalen Eigenschaften der provisionierten Anwendung können ebenfalls beeinflusst werden, beispielsweise durch die Verwendung von Policies.

Nachdem die Anwendung durch den Build Plan provisioniert wurde ist sie verfügbar und kann verwendet werden. Neben der Migration der Anwendung ist oft auch eine Migration von Daten notwendig. Dieser Schritt kann manuell durchgeführt werden, er kann aber alternativ auch durch einen Plan beschrieben und dann mit Hilfe des TOSCA Containers automatisiert durchgeführt werden. Es muss dabei im Einzelfall entschieden werden, welches Vorgehen geeigneter ist. Der Vorteil der automatisierten Ausführung eines Planes kann durch den Aufwand zur Erstellung dieses Planes wieder aufgehoben werden. Grundsätzlich lohnt sich der Aufwand der Erstellung eines Planes umso mehr, je öfter dieser Plan ausgeführt wird.

[3] http://www.iaas.uni-stuttgart.de/OpenTOSCA/

Auch nach der vollständigen Migration von Anwendung und Daten bietet die Verwendung von TOSCA Vorteile. Typische Verwaltungsaufgaben die regelmäßig anfallen, beispielsweise das Einspielen von Updates oder das Anlegen von Sicherungen, können als Pläne modelliert und dann durch den TOSCA Container automatisiert durchgeführt werden. Dies ist in Abbildung 25 durch Schritt 5 dargestellt.

TOSCA kann, wie im Vorherigen beschrieben, zur Durchführung des Migrationsschrittes in Phase 3 verwendet werden. TOSCA selbst bietet eine formale Sprache zur Modellierung der zu provisionierenden Anwendungsstruktur. CSAR als Paketierungsformat erlaubt es, alle für die Provisionierung benötigen Informationen und Artefakte standardisiert zu paketieren. TOSCA Container wie beispielsweise OpenTOSCA können dann verwendet werden, um die Provisionierung, die Datenmigration sowie die Verwaltung der zu migrierenden Anwendung automatisiert und portabel durchzuführen. Im Folgenden werden einzelne Komponenten und Aspekte des beschriebenen Vorgehens im Detail vorgestellt und diskutiert.

Cloud Service Archive (CSAR)

Das CSAR-Paketierungsformat ist Teil des TOSCA Standards. Ein CSAR-Paket enthält zum einen ein oder mehrere TOSCA Modelle und zum anderen alle zu dessen Instanziierung benötigten Artefakte. Aus technischer Sicht ist ein solches Paket ein einfaches Zip Archiv. Der konzeptionelle Aufbau eines solchen Pakets ist in Abbildung 26 dargestellt.

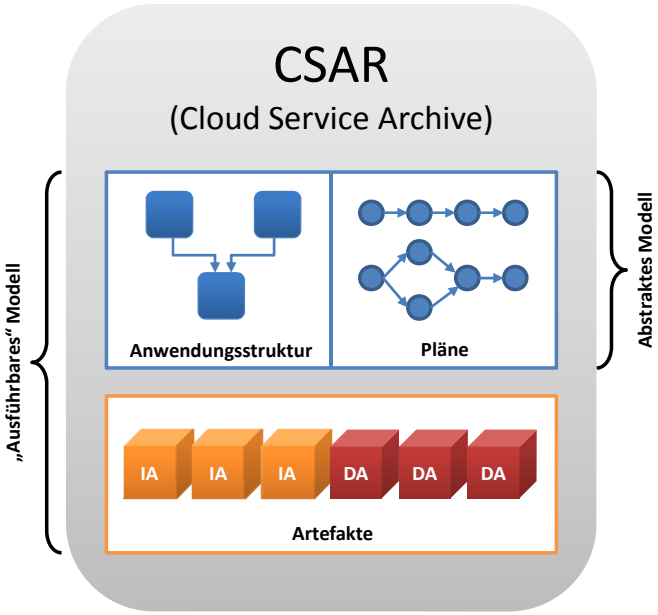


Abbildung 26 Cloud Service Archive (CSAR)

Der TOSCA-Standard schränkt die interne Struktur eines CSAR Paketes nur sehr wenig ein. Vordefiniert ist, dass es im Unterordner /TOSCA-Metadata eine Manifest Datei namens TOSCA.meta geben muss. Diese Datei enthält eine Menge von Name Wert Paaren. Vorgegeben ist, dass die verwendete TOSCA Version, die Version der CSAR Datei selbst und der Name des Autors angegeben werden müssen. Optional kann auf eine TOSCA Modelldate verwiesen werden, die angegebene TOSCA Datei wird bei der weiteren Verarbeitung als bevorzugter Einstiegspunkt gewählt. Die Benennung und Platzierung aller weiteren Artefakte bzw. Dateien ist dem Ersteller des CSAR Pakets überlassen. Alle verwendeten Artefakte werden aus TOSCA Dateien referenziert, über diese Referenzen sind sie interpretierbar und zuordenbar.

Lego4TOSCA NodeTypes

Die Modellierung von TOSCA ServiceTemplates erfolgt unter anderem mit NodeTemplates. Diese sind wiederum von einem bestimmten Typ, den sogenannten NodeTypes. Ein NodeType besteht zum einen aus einem abstrakten TOSCA Modell der jeweils dargestellten Komponenten und zusätzlich aus allen zu deren Instanziierung notwendigen Artefakte. Am IAAS wurde eine Reihe von NodeTypes zur Modellierung von TOSCA Cloud Services entwickelt. Diese NodeTypes sind, sofern sinnvoll, untereinander kombinierbar. Sie stellen Modellierungskonstrukte für einige verbreitete Komponenten typischer Service Topologien dar. Im Folgenden werden die entwickelten NodeTypes kurz vorgestellt. Zu jedem NodeType werden die in TOSCA definierten sogenannten *Requirements* und *Capabilities* aufgezählt. Mit diesen Konstrukten kann angegeben werden, welche Funktionalitäten ein NodeType bietet und welche Funktionalitäten er benötigt. Aus diesen Angaben lässt sich ablesen, welche NodeTypes miteinander kombinierbar sind.

1. Amazon Web Services (AWS) Elastic Compute Cloud (EC2)

   Dieser NodeType repräsentiert Virtuelle Maschinen der AWS EC2 Plattform, einem IaaS Angebot einer Public Cloud Infrastruktur.

   Requirements: ---

   Capabilities: Virtual Machine

2. VMWare VSphere

   Dieser NodeType repräsentiert Virtuelle Maschinen einer VMWare VSphere Plattform. Diese Plattform wird vor allem in Private Cloud-Umgebungen eingesetzt.

   Requirements: ---

   Capabilities: Virtual Machine

3. Ubuntu

   Dieser NodeType repräsentiert ein Linux Betriebssystem der Ubuntu Distribution.

   Requirements: Virtual Machine

   Capabilities: Operating System

4. Windows Server 2012

   Dieser NodeType repräsentiert ein Windows Server Betriebssystem in der Version 2012.

   Requirements: Virtual Machine

   Capabilities: Operating System

5. Tomcat

   Dieser NodeType repräsentiert einen Apache Tomcat Application Server, eine Laufzeitumgebung für Servlets.

   Requirements: Operating System

   Capabilities: Servlet Container

6. Apache Webserver

   Dieser NodeType repräsentiert einen Apache Webserver, einen weit verbreiteten HTTP Web Server.

   Requirements: Operating System

Capabilities: Web Server

7. WSO2 Business Process Server

   Dieser NodeType repräsentiert eine BPEL Workflow Engine.

   Requirements: Operating System

   Capabilities: BPEL Process Engine

8. MySQL

   Dieser NodeType repräsentiert eine relationale SQL Datenbank.

   Requirements: Operating System

   Capabilities: SQL Database

9. SixCMS

   Dieser NodeType repräsentiert das Content Management System SixCMS 8.1

   Requirements: MySQL, Apache Web Server

   Capabilities: SixCMS

Die aufgezählten NodeTypes sind gemäß der angegebenen Requirements und Capabilites miteinander kombinierbar. Die von den NodeTypes angebotenen Management Operationen sind asynchron implementiert, da sie in vielen Fällen potentiell langlaufend sind. Die Implementierung der NodeTypes basiert auf einer generischen Architektur und gemeinsamen Code Basis. Damit wird die Entwicklung weiterer NodeTypes wesentlich vereinfacht.

NodeType Design – Grundlagen

Ziel des NodeType Designs (Haupt et al., 2014) war es, dass sich die sich unterschiedliche NodeTypes einfach, d.h. mit geringem Konfigurationsaufwand, kombinieren lassen, um komplexe Cloud Anwendungen zu erstellen. So ist es beispielsweise mit dem im folgenden beschriebenen NodeType Design möglich, dass, nur der VMWare ESXi NodeType in der Topologie durch AWS EC2 NodeType ersetzt werden muss, um die Cloud Anwendung auf bei Amazon AWS anstatt bei VMWare zu auszuführen.

Die Granularität der angebotenen Managementoperationen stellt außerdem sicher, dass einfache Management Pläne erstellt werden können, da die Komplexität in den Operationen gekapselt ist. Des Weiteren werden die Pläne dadurch vereinfacht, dass die NodeTypes untereinander kommunizieren können.

NodeType Properties

Ein NodeType definiert seine möglichen Eigenschaften (engl. Properties) bzw. Parameter mittels XSD Schema. Wie in Abbildung 27 dargestellt, definiert der VMWare NodeType beispielsweise, ein Property *Size* und ein Property *IP*, mit dem definiert werden kann, auf welchem Internet Port der Tomcat auf Anfragen hört. Die eigentlichen Werte für die Properties werden entweder im NodeTemplate, d.h. während der Modellierung der Cloudanwendungstopologie oder zur Laufzeit der Cloudanwendung von einer Instanz des NodeTemplates (NodeInstance) gesetzt. Abhängig davon, wann die Werte gesetzt werden, unterscheiden wir zwischen drei Arten von Properties, *Konfigurationsproperties*, *ImplementationArtifact-Properties* und *globale Cloudanwendungsproperties*.

Die Werte für Konfigurationsproperties werden während der Modellierung der Cloudanwendungstopologie gesetzt, um ein NodeTemplate zu konfigurieren. Diese Properties werden von den Managementoperationen des zugehörigen ImplementationArtifacts gelesen und beeinflussen deren Ausführung. Das Konfigurationsproperty *size* definiert z.B. die

Leistung der bereitzustellenden VM (Anzahl CPUs, Arbeitsspeicher etc.). So wird die Managementoperation *install*, die eine VM provisioniert, in der Konfiguration *small* beispielsweise nur eine VM mit einer CPU und 4GB Arbeitsspeicher provisionieren, während es in der Konfiguration *large* 16GB Arbeitsspeicher und 4 CPUs wären.

Ein ImplementationArtifact-Property persistiert Daten die von den Managementoperationen der verschiedenen ImplementationArtifacts benötigt werden. Die Daten werden nur von dem ImplementationArtifact geschrieben, das den NodeType implementiert, der das ImplementationArtifact-Property definiert. Gelesen werden dürfen die Daten aber von allen ImplementationArtifacts der Cloudanwendung. So wird das ImplementationArtifact-Property *IP* zum Beispiel von ImplementationArtifact des VMWare NodeTypes geschrieben, sobald die VM provisioniert und somit ihre IP Adresse bekannt ist. Das IA eines auf der VM installierten Tomcats weiß durch diese Property zum Beispiel, unter welcher IP der Tomcat erreichbar ist.

Die *globalen Cloudanwendungsproperties* persistieren Daten, die nicht nur für die Verwaltung einzelner NodeInstances, sondern für die Verwaltung der gesamten Cloudanwendung notwendig sind. Im Gegensatz zu den ImplementationArtifact-Properties werden die Cloudanwendungsproperties von den globalen Managementplänen der Cloudanwendung geschrieben und gelesen. So kann zum Beispiel ein globaler Managementplan periodisch die Antwortzeiten aller Komponenten der Cloudanwendung überprüfen. Die Antwortzeiten für werden dann vom Plan in jeder NodeInstance persistiert. Ein weiterer Managementplan kann auf Basis der Antwortzeiten korrektive Maßnahmen vornehmen, wie zum Beispiel eine weitere VM provisionieren.

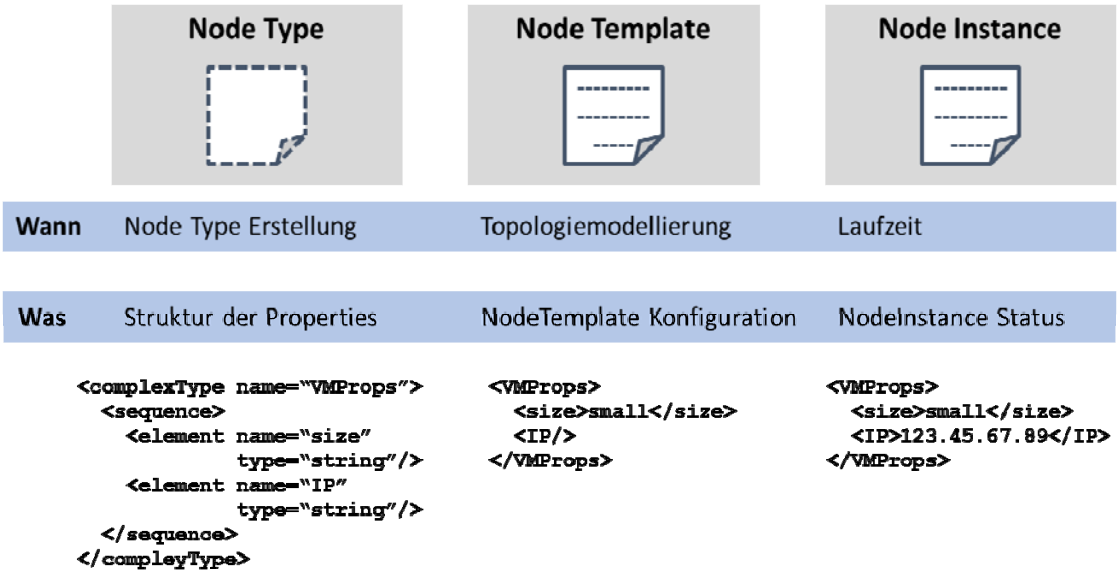


Abbildung 27 Lego4TOSCA Properties

NodeType Interfaces

Jeder NodeType stellt ein oder mehrere Management-Interfaces (Schnittstellen) bereit. Ein Management-Interface besteht aus einer Menge von Operationen, die von den Management-Plänen aufgerufen werden können, um die von dem NodeType repräsentierte Komponente der Cloudanwendung zu administrieren. Üblicherweise stellt jeder NodeType zumindest Operationen bereit, mit denen die Komponente provisioniert, konfiguriert und deprovisioniert werden kann. Um eine komplette Cloudanwendung zu verwalten, rufen die Managementpläne die Operationen der verschiedenen NodeTypes der Topologie in einer definierten Reihenfolge auf. Um zum Beispiel eine Cloudanwendung zu provisionieren, würde ein Managementplan zuerst die Managementoperationen des VM NodeTypes aufrufen, die eine VM provisionieren

und dann die Managementoperationen des Apache Tomcat NodeTypes, die den Apache Tomcat provisionieren.

Jede Managementoperation bietet eine Menge von *funktionalen* Parametern. Die funktionalen Parameter sind NodeType-spezifisch, d.h. dabei handelt es sich um domänen-spezifische Parameter, die konkret für die Provisionierung, Konfiguration etc., der durch die von dem NodeType verwalten Komponente benötigt werden. Dies kann beim Apache Tomcact NodeType zum Beispiel der Port sein, unter dem der Tomcat erreichbar sein soll. Dabei können die funktionalen Parameter auch die in den Konfigurationsproperties festgelegten Werte überschreiben, um zum Beispiel flexibel auf Ausnahmesituationen reagieren zu können. Neben den funktionalen Parametern bietet jede Managementoperation zusätzlich die zwei *technischen* Parameter *node instance id* und *callback address*. Da mehrere Instanzen, d.h. NodeInstances eines NodeTemplates erstellt werden können (z.B. zwei Apache Tomcat NodeInstances vom selben NodeTemplate), muss dem Parameter *node instance id* die eindeutige ID der NodeInstance übergeben werden, auf die die Managementoperation angewandt werden soll. Der Parameter *callback address* wird für die im Folgenden beschriebene asynchrone Kommunikation mit den Managementoperationen benötigt.

Da viele Managementaufgaben langlaufende Operationen sind (Backups können zum Beispiel Stunden dauern), sind die Managementoperationen der Lego4TOSCA NodeTypes als asynchrone Operationen implementiert, d.h. der Aufrufer wird über einen Callback Mechanismus informiert, über das Ergebnis der Managementoperation informiert, sobald diese beendet wurde. Dies hat den Vorteil, dass Verbindungs-Timeouts (z.B. der Standard HTTP Timeout nach 120 Sekunden) bei langlaufenden Operationen vermieden werden und dass der Aufrufer nicht blockiert wird, solange er auf die Antwort von der Managementoperation wartet. Die Rückrufadresse des Aufrufers wird dabei vom durch den oben erwähnten Parameter *callback address* übergeben.

Architektur der Implementation Artifacts

Im Folgenden wird beschrieben, wie die Architektur der Implementierung der NodeTypes, d.h. der ImplementationArtifacts (IAs) die eingangs erwähnten Eigenschaften wie Kombinierbarkeit, Asynchronität etc. realisiert. Dazu werden mittels Abbildung 28 die Abläufe innerhalb eines der IAs anhand der Managementoperation *start* des Apache Tomcat IAs exemplarisch beschrieben.

Ein oder mehrere IAs stellen die Implementierung eines NodeTypes bereit. Jedes IA ist als Web Service implementiert, d.h. es bietet eine Menge bzw. Teilmenge der im NodeType definierten Operationen als plattformunabhängig über WSDL-Schnittstellen an.

In Schritt 1 wird die Managementoperation entweder von einem Managementplan oder einem anderen IA aufgerufen. In Abbildung 28 wird die Operation *start* zum Beispiel von einem Plan aufgerufen. Dieser Operation wird neben dem technischen Parameter *Node Instance ID*, der die Node Instance „myTomcat“ identifiziert, auch der Wert „8000“ für den funktionalen Parameter *debugPort* übergeben. Dadurch, dass dem IA bei jedem Aufruf die Node Instance ID übergeben wird, bleibt es statuslos, d.h. die gleiche Instanz eines IAs kann die Operationen für verschiedene NodeInstances ausführen. Damit ist die Skalierbarkeit der IAs gewährleistet. Da die Managementoperationen asynchron implementiert sind, gibt die Operation, bevor sie mit der eigentlichen Verarbeitung beginnt, sofort eine eindeutige Korrelations-ID zurück (im Beispiel „897uhekfkj“). Diese kann später vom Aufrufer verwendet werden, um das in Schritt 6b zurückgesendeten Ergebnisses den ursprünglichen Aufruf zuordnen zu können.

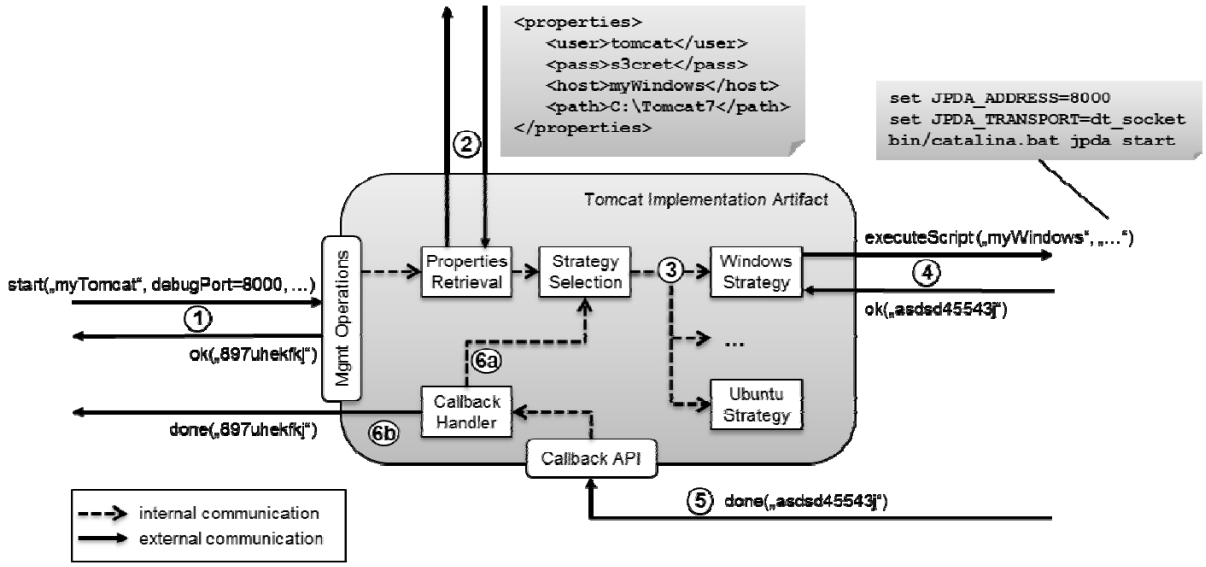


Abbildung 28 Architektur Lego4TOSCA ImplementationArtifact

In Schritt 2 liest die *Properties Retrieval* Komponente die aktuellen Property-Werte für die NodeInstance „myTomcat" aus, wie zum Beispiel das Konfigurationsproperty *user* („tomcat") und das ImplementationArtifact-Property *host* („Windows"). Die Properties Retrieval Komponente nutzt die API des TOSCA Containers, um auf die zuvor vom Cloud- Anwendungsmodellierer bzw. von den IAs gesetzten Properties der NodeInstance zuzugreifen.

Im dritten Schritt bestimmt die *Strategy Selection* Komponente eines IAs, welche Implementierung der aufgerufenen Managementoperation aufgerufen wird. Die aufzurufende Implementierung hängt dabei vom Kontext ab, in dem das IA und die von ihm verwaltete Komponente provisioniert ist. Da der Tomcat zum Beispiel auf einen Windows oder Linux Betriebssystem installiert werden kann, muss die *Strategy Selection* Komponente anhand der Topologie entscheiden, ob es den Start über einen Windows oder Linux NodeType ausführt.

Im 4. Schritt interagiert die *Strategy Selection* Komponente mit dem ausgewählten NodeType bzw. dessen IAs die in den jeweiligen Kontext notwendig sind, um die Managementoperationen auszuführen. So muss in dem Beispiel die *Strategy Selection* Komponente abhängig vom Betriebssystem, auf dem der Tomcat installiert wurde, entweder den Windows NodeType oder den Ubuntu NodeType aufrufen, um die *startup.bat* bzw. die *startup.sh* auszuführen. Die Interaktion mit anderen NodeTypes wird über den TOSCA Container realisiert, d.h. das IA informiert den TOSCA Container darüber, welche Managementoperation eines anderen NodeTypes es aufrufen will und der TOSCA Container übernimmt dann den eigentlichen Aufruf. Dabei gibt die Managementoperationen des von der Strategy Selection Komponente aufgerufenen NodeTypes ebenfalls sofort eine eindeutige Korrelations-ID zurück (im Beispiel „asdsd45543j"), die den Aufruf identifiziert und beginnt dann mit der Verarbeitung.

Das Ergebnis der aufgerufenen Operation wird dann im 5. Schritt vom aufgerufenen IA zum aufrufenden IA zurückgegeben. Dazu bietet jedes IA eine *Callback API* an. Dieser Schnittstelle wird eine Nachricht übergeben, die das Ergebnis der asynchron aufgerufenen Managementoperation, sowie die in Schritt 4 zurück gegebene Korrelations-ID enthält.

Das Ergebnis in der Nachricht wird vom *Callback Handler* über die Korrelations-ID zum in Schritt 4 getätigten Aufruf zugeordnet und verarbeitet. Falls nötig, können über die Strategy Selection Komponente weitere NodeTypes aufgerufen werden (Schritt 6a). Falls keine weiteren Verarbeitungsschritte mehr nötig sind, beendet der Callback Handler wiederum die asynchrone Ausführung der in Schritt 1 aufgerufenen Managementoperation. Dazu sendet er

das Ergebnis nebst der Korrelations-ID an den Aufrufer zurück. In dem Beispiel gibt das IA dem Plan die Information zurück, ob der Tomcat erfolgreich gestartet wurde.

AP3.4 Bereitstellung von Migrationswerkzeugen

Die Verarbeitung von CSAR Paketen erfolgt im Allgemeinen durch sogenannte Container. Am IAAS wurde der TOSCA Container OpenTOSCA entwickelt. Die Architektur des Containers ist vereinfacht in Abbildung 29 dargestellt. Die zentralen Elemente dieser Architektur werden im Folgenden anhand des Prozesses der Verarbeitung eines CSAR Paketes näher vorgestellt.

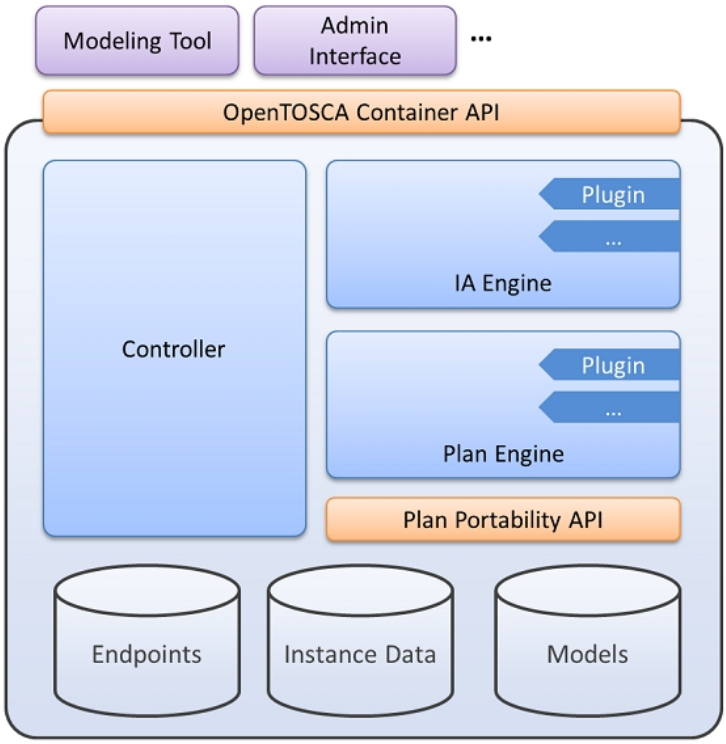


Abbildung 29 Architektur OpenTOSCA

Die Interaktion mit dem OpenTOSCA Container erfolgt über die als REST Service umgesetzte *OpenTOSCA Container API*. Neben der Administration des Containers kann über diese Schnittstelle, beispielsweise direkt aus einem entsprechenden Modellierungstool heraus, eine CSAR Datei hochgeladen werden. Diese Datei wird von der zentralen *Controller* Komponenten angenommen und zunächst entpackt. Anschließend werden die TOSCA Modelldateien eingelesen und in der entsprechenden *Models* Datenbank abgelegt. Nachdem das TOSCA Modell gelesen, validiert und abgelegt wurde werden in den nächsten Schritten die im CSAR Paket enthaltenen Artefakte verarbeitet.

Zur Verarbeitung von ImplementationArtifacts aktiviert die Controller Komponenten die *IA Engine*. Diese Komponente ist durch Plug-Ins erweiterbar. Jedes Plug-In unterstützt einen bestimmten ImplementationArtifact Typ, die Architektur spiegelt somit die Erweiterbarkeit des TOSCA Typsystems wieder. Die IA Engine bestimmt für jedes ImplementationArtifact den Typ und aktiviert das dazu passende Plug-In. Die Plug-Ins sind dann dafür verantwortlich, die übergebenen ImplementationArtifacts auf einer passenden Zielplattform zu deployen. Der OpenTOSCA Container unterstützt momentan das Deployment auf Tomcat Application Servern[4] und auf Axis2[5] Plattformen. Nach dem Deployment eines

[4] http://tomcat.apache.org/

ImplementationArtifacts ist dessen konkreter Endpunkt bekannt, dieser wird anschließend in der Endpoints *Datenbank* abgelegt.

Nach dem Verarbeiten der ImplementationArtifacts stößt die Controller Komponenten die Verarbeitung von Plänen an und aktiviert dazu die *Plan Engine*. Ebenso wie die IA Engine ist diese Komponenten durch Plug-Ins erweiterbar. Der OpenTOSCA Container unterstützt momentan das Deployment von BPEL Plänen auf einer Apache ODE[6] Workflow Engine. Pläne interagieren im Allgemeinen mit den von ImplementationArtifacts angebotenen Managementoperationen. Da die Endpunkte der ImplementationArtifacts zur Modellierungszeit der Pläne nicht bekannt sind, müssen diese vor dem Deployment noch gebunden werden. Dazu werden die Endpunktinformationen aus der Endpoints Datenbank abgegriffen und von den jeweiligen Plan Engine Plug-Ins verarbeitet. Anschließend können die Pläne deployt werden.

In TOSCA können sowohl NodeTemplates als auch RelationshipTemplates Eigenschaften (Properties) besitzen. Diese Eigenschaften können zur Laufzeit für jede Instanz eines Knotens oder einer Beziehung gelesen und geschrieben werden. Diese Daten werden in der *Instance Data* Datenbank verwaltet und sind über die *Plan Portability API* zugreifbar. Diese Schnittstelle soll in Zukunft als Teil des TOSCA Standards ebenfalls standardisiert werden und so die Portabilität von Plänen weiter erhöhen.

Der OpenTOSCA Container wurde als serviceorientierte Architektur entworfen. Die Umsetzung dieser Architektur erfolgte mit OSGi und nutzt *Declarative Services*. Der OpenTOSCA Container stellt somit eine leichtgewichtige, einfach erweiterbare und lose gekoppelte Plattform für die Verarbeitung von CSAR Paketen dar.

4. AP4: Werkzeuge für energieeffizientes Cloud-Management

AP4	**Werkzeuge für energieeffizientes Cloud-Management**	**Leitung: IBM**
Ziele (IAAS)	– Spezifikation und Implementierung von Werkzeugen für energieeffizientes Cloud-Management	
Ergebnisse (IAAS)	– Policy Taxonomie – Policy Taxonomie für „green“ Policies – Policy Taxonomie für TOSCA Cloud Services – Konzeptionelle Erweiterung von OpenTOSCA – Implementierung der green Policies Unterstützung in OpenTOSCA	

AP4.1 Spezifikation der Anforderungen an Cloud-Management-Werkzeuge

Aus den Ergebnissen der vorangegangenen Arbeitspakete wurde die Schlussfolgerung gezogen, dass Energieeffiziente Cloud Services am geeignetsten auf Basis von Cloud Service Templates realisiert werden, welche mit Energieeinsparungs-Richtlinien (Anweisungen an die umsetzende Cloud Infrastruktur) annotiert wurden. In diesem Kapitel wird beschrieben, wie für die Realisierung von Energieeinsparungen bestimmte Richtlinien (Policies) definiert werden können, um damit Vorgaben über die Verwendung von Cloud Services und deren Management festzulegen. Anhand konkreter Szenarien wird untersucht, an welchen Stellen eines Cloud-Lebenszyklus Policies relevant sind und wann eine entsprechende Policy sichergestellt werden muss. Das Dokument betrachtet zudem, wie diese Szenarien auf

[5] http://axis.apache.org/axis2/java/core/

[6] http://ode.apache.org/

TOSCA abgebildet, beschrieben und verwendet werden können. Das Ziel dieses Dokuments ist es, neben dem Verständnis für die in MIGRATE! relevanten Policies (siehe Policy Taxonomie), auch einen Überblick über die entsprechenden Anforderungen an die Umsetzung dieser Policies zu geben.

Policy Taxonomien

Policies stellen, abhängig von dem Akteur der sie sich zu Eigen macht, Anforderungen oder Zusicherungen dar. Policies, die an einem Cloud Service Modell annotierte sind, können beispielsweise Anforderungen an eine ausführende Infrastruktur darstellen. Demgegenüber formulieren die von einer Cloud Infrastruktur bekannt gemachten Policies entsprechende Zusicherungen. Zur Vereinfachung werden im Folgenden, ohne dabei die Allgemeinheit der Aussagen einzuschränken, Policies als Anforderungen interpretiert.

Das Ziel dieses Kapitels ist es, eine systematische und abstrakte Betrachtung der in MIGRATE! verwendeten Policies vorzubereiten. Die systematische Betrachtung von Policies, ihrer Definition sowie ihrer Umsetzung wird durch eine abstrakte Klassifizierung wesentlich vereinfacht. Durch die Einteilung in Klassen und die damit verbundene Abstraktion ist eine umfassende Analyse der betrachteten Policies unabhängig von den konkreten Policy Instanzen möglich. Die Klassifizierung von Policies wird im Folgenden anhand von Taxonomien vorgenommen. Eine Taxonomie definiert eine baumförmige Klassenstruktur. Elemente einer Klasse sind dabei immer auch Elemente aller Oberklassen.

Die Klassifizierung von Policies kann, je nach Betrachtungsweise, nach unterschiedlichen Kriterien erfolgen. Um diesem Umstand Rechnung zu tragen, werden im Folgenden mehrere Policy Taxonomien vorgestellt. Jede Taxonomie klassifiziert Policies nach unterschiedlichen Aspekten. Zwischen einzelnen Taxonomien können wiederum Zusammenhänge bestehen. Die Verknüpfung der einzelnen Taxonomien wird in diesem Kapitel ebenfalls behandelt.

Eine Policy Taxonomie für “grüne” Policies

“Grüne” Policies stellen im Allgemeinen Aussagen über den Umwelteinfluss eines Cloud Services dar. Diese umweltbezogenen Anforderungen können auf unterschiedlichen Ebenen definiert werden. Die in Abbildung 30 dargestellte Taxonomie unterscheidet drei Klassen von “grünen” Policies.

Policies auf *Infrastrukturebene* stellen Forderungen an die Infrastruktur, innerhalb der ein Cloud Service betrieben wird („Infrastructure Level“). Sie stehen in keinem direkten Zusammenhang mit der Struktur des Cloud Services. Im in Abbildung 30 (links) dargestellten Beispiel wird gefordert, dass das zum Betrieb des Cloud Services genutzte Rechenzentrum einen PUE (Power Usage Efficiency) Wert kleiner als 1,3 aufweist. Weitere Beispiele sind die Forderung nach der Verwendung von Ökostrom oder nach beliebigen “grünen” Zertifizierungen. Policies auf Infrastruktur Ebene sind im Allgemeinen sehr grob granular. Ihre konkreten Auswirkungen, also beispielsweise die eingesparte Menge an CO_2, kann nur sehr schwer bis gar nicht angegeben werden.

Policies auf *Implementierungsebene* machen Aussagen darüber, welche Forderungen bei der Instanziierung einzelner Elemente eines Cloud Services zu beachten sind („Implementation Level“). Die einzelnen Elemente eines Cloud Services können Software- oder Hardwarekomponenten darstellen. Zum Zeitpunkt der Instanziierung dieser Modellelemente werden konkrete Implementierungen bzw. Geräte ausgewählt, welche die geforderte Funktionalität implementieren. Die in Abbildung 30 (Mitte)skizzierte Policy fordert beispielsweise, bei der Auswahl von Geräten bestimmte Energieverbrauchskennzahlen zu beachten. Im Gegensatz zu Policies auf Infrastrukturebene sind Policies auf

Implementierungsebene feingranularer. Sie können gezielte Anforderungen an einzelne Komponenten eines Cloud Services formulieren.

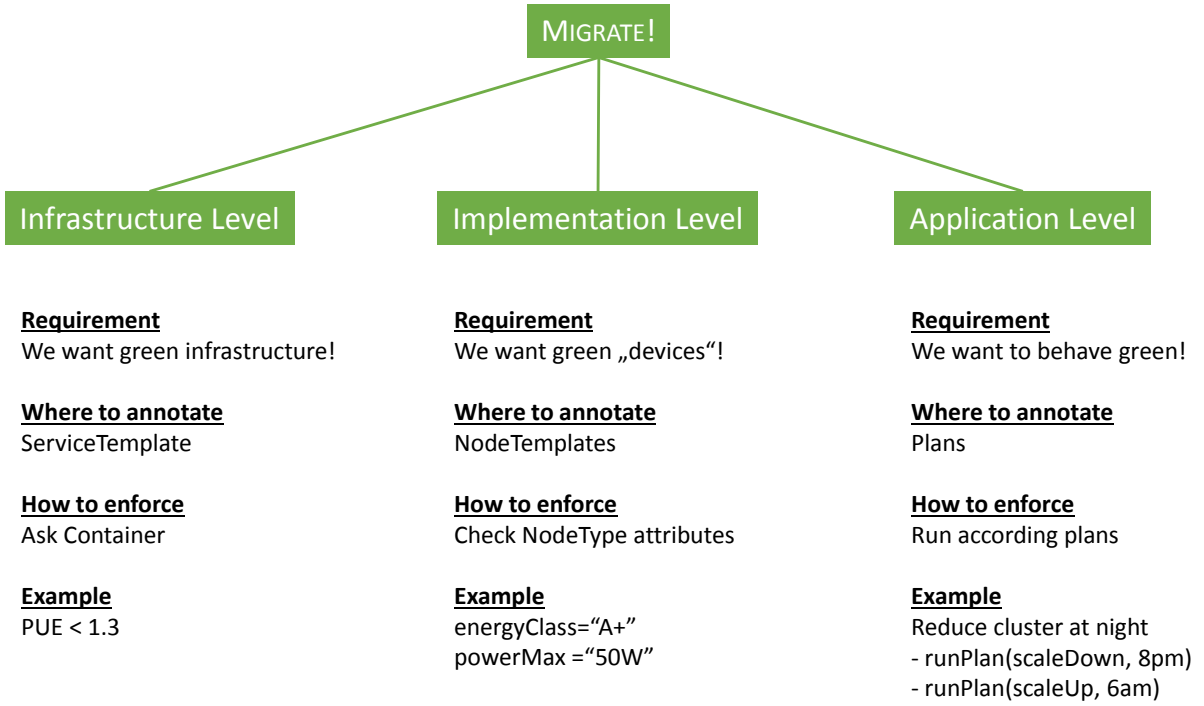


Abbildung 30 Policy Taxonomie für "grüne" Cloud Services

Policies auf *Anwendungsebene* formulieren Forderungen an das nichtfunktionale Laufzeitverhalten eines Cloud Services („Application Level“). Die in Abbildung 30 (rechts) dargestellte Beispiel Policy fordert, dass ein Cluster innerhalb eines vorgegebenen Zeitraums verkleinert wird. Diese Forderung muss durch ein entsprechendes Management des Cloud Services sichergestellt werden.

Die vorgestellten Klassen „grüner“ Policies können unabhängig voneinander verwendet und beliebig kombiniert werden. Der Modellierer eines Cloud Service trägt dabei die Verantwortung, mögliche Wechselwirkungen zwischen den einzelnen Policies zu beachten.

Policy Taxonomie für TOSCA Cloud Services

Eine weitere Klassifizierung von Policies richtet sich danach, auf welchen Aspekt eines Cloud Services sich eine Policy bezieht. Eine Taxonomie für TOSCA Cloud Service Policies ist in Abbildung 31 dargestellt. Sie unterscheidet auf der obersten Ebene zwischen Policies, welche sich auf das Modell eines Cloud Services beziehen („Definitions“) und Policies, welche sich auf das Paket beziehen, in dem diese Modelle und weitere Artefakte enthalten sind („CSAR“, Cloud Service Archive). Der wesentliche Unterschied zwischen diesen Klassen von Policies besteht darin, dass sich CSAR Policies „außerhalb“ des Cloud Service Pakets befinden, wogegen sich alle weiteren Policy Typen „innerhalb“ dieses Pakets befinden. Eine typische Cloud Service Paket Policy kann beispielsweise fordern, dass Cloud Service Pakete nach bestimmten Vorgaben verschlüsselt oder signiert sind. Sind diese Policies verletzt, so kann auf das Cloud Service Paket nicht zugegriffen werden. Policies auf Cloud Service Paket Ebene haben somit Vorrang vor allen weiteren Policies. Sie müssen überprüft und sichergestellt werden, bevor alle weiteren Polcies verarbeitet werden.

Policies, welche an ein beliebiges TOSCA Definitions Dokument annotiert sind, werden zunächst nach ihrem Wirkungsbereich unterschieden. Policies können Anforderungen an den TOSCA Container und dessen Verhalten stellen („Container“), sie sind damit unabhängig von

der eigentlichen Cloud Service Struktur. Daneben können sich Policies aber auch direkt auf die Elemente der Cloud Service Beschreibung beziehen („ServiceTemplate“).

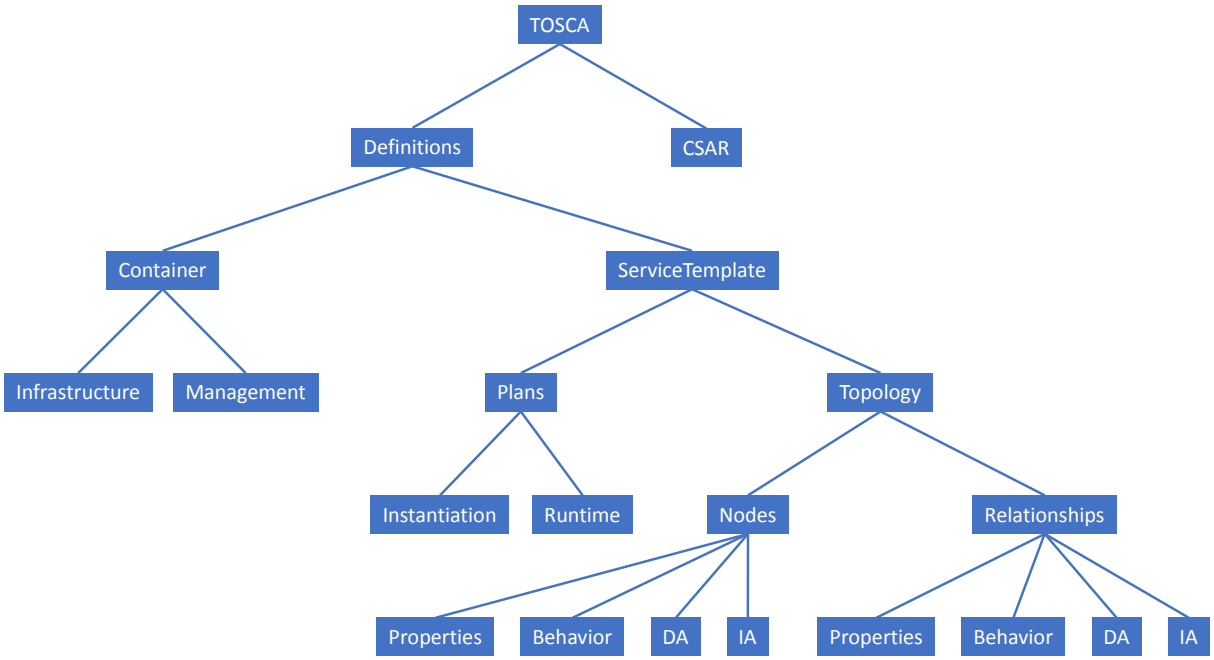


Abbildung 31 Policy Taxonomie für TOSCA Cloud Services

Policies, die sich auf den TOSCA Container beziehen, werden dahingehend unterschieden, ob sie sich auf die Infrastruktur des Containers („Infrastructure“) oder auf die Verwaltung des Containers beziehen („Management“). Anforderungen an die Container Infrastruktur können beispielsweise sein, dass der Container mit einer bestimmten Technik, in einem bestimmten Land oder mit Ökostrom betrieben wird. Beispiele für Management Policies sind, dass das Betriebspersonal bestimmte Qualifikationen besitzen muss oder dass der Betrieb der Container Infrastruktur bestimmte Zertifizierungen erfüllen muss, beispielsweise das Qualitätsmanagement oder Green IT Aspekte betreffend.

TOSCA Modelle beschreiben sowohl die Struktur eines Cloud Services als auch die zur Verwaltung dieses Cloud Services nötigen Prozesse. Dementsprechend werden auch die Policies eines Cloud Services unterschieden in Policies welche sich auf die Struktur, also die Topologie, des Cloud Services beziehen („Topology“) und Policies, welche Anforderungen an die Verwaltung des Cloud Services stellen („Plans“).

In TOSCA wird die Verwaltung eines Cloud Services durch Pläne beschrieben, Prozesse die automatisiert ausgeführt werden können. Policies, welche diese Pläne betreffen, werden dahingehend unterschieden, ob sie sich auf die Instanziierung, also den Start eines Plans beziehen („Instantiation“), oder ob sich die Anforderungen auf die Laufzeit eines Plans beziehen („Runtime“). Bezüglich der Instanziierung eines Plans kann beispielsweise gefordert werden, dass Pläne nur von bestimmten Personen oder Rollen oder nach dem Vier-Augen-Prinzip gestartet werden dürfen. Zur Laufzeit eines Plans kann als Beispiel gefordert werden, dass Pläne nicht beliebige Aktionen ausführen dürfen, sondern dass die Menge der erlaubten Operationen eingeschränkt wird.

Die Struktur eines Cloud Services wird in TOSCA durch eine einfache gerichtete Graph Struktur beschrieben, sie besteht aus Knoten („Nodes“) und Kanten zwischen diesen Knoten („Relationships“). Die Knoten des Graphen repräsentieren die einzelnen Komponenten eines Cloud Services, also beispielsweise virtuelle Maschinen, Betriebssysteme, Middleware- oder beliebige Software-Komponenten. Die Kanten des Graphen stellen die Beziehungen zwischen diesen Komponenten dar, also beispielsweise dass eine Middleware Komponente auf einem

bestimmten Betriebssystem läuft oder dass eine Software Komponenten eine Middleware nutzt. Policies für einen Cloud Service werden demzufolge ebenso danach unterschieden, ob sie sich auf die Komponenten eines Cloud Services beziehen, oder ob sie Aussagen über die Beziehungen zwischen den Komponenten machen.

Sowohl die Komponenten als auch die Beziehungen zwischen diesen werden in TOSCA ähnlich detailliert modelliert. Policies können sich auf verschiedene Aspekte einer Komponente oder einer Beziehung beziehen. Die erste Klasse von Policies bezieht sich auf die Eigenschaften einer Komponenten oder einer Beziehung („Properties“). Eine solche Policy kann beispielsweise die erlaubten Werte für bestimmte Eigenschaften beschränken oder anderen Regeln unterwerfen. Neben Eigenschaften können Komponenten und Beziehungen Operationen definieren, über welche sie verwaltet werden können. Policies können sich auf diese Operationen beziehen („Behavior“) und damit das Verhalten der betroffenen Komponenten beeinflussen. Neben dem Modell einer Komponente oder einer Beziehung werden zusätzliche Artefakte benötigt, um das beschriebene Modell bzw. den beschriebenen Cloud Service konkret instanziieren zu können. In TOSCA wird unterschieden in Implementation Artefakte („IA“) und Deployment Artefakte („DA“). Implementation Artefakte realisieren die von einer Komponenten oder einer Beziehung angebotenen Verwaltungsoperationen, sie können beispielsweise Skripte oder beliebige andere Implementierungen sein. Deployment Artefakte realisieren die Komponente oder Beziehung selbst. Beispiele dafür sind Images für virtuelle Maschinen, ausführbare Programme oder allgemein beliebige Dateien. Policies können dahingehend unterschieden werden, auf welchen Aspekt einer Komponente oder Beziehung sie sich beziehen. Policies mit Bezug auf Eigenschaften oder Verwaltungsoperationen haben oft einschränkenden Charakter. Dies kann ebenso auf Policies zutreffen, die sich auf die Artefakte einer Komponenten oder einer Beziehung beziehen. Zusätzlich können Policies in diesen Fällen aber beispielsweise auch die Auswahl von Artefakten aus mehreren alternativen Artefakten beeinflussen.

AP4.3 Erweiterung der Cloud-Infrastruktur um Energiemanagement-Komponenten

Cloud Service Lebenszyklus

Die Phasen des Lebenszyklus einer Cloud Anwendung von der *Modellierung der Anwendung* bis zur *De-Installation* ist schematisch in Abbildung 32 dargestellt. Im Folgenden Phasen im Kontext von Policies genauer beschrieben.

Initial wird der Aufbau des Cloud Services in der *Phase Modellierung der Anwendung* beschrieben. Dazu wird ein TOSCA Cloud Service Modell erstellt. Das Service Modell ist die Topologie des Cloud Services, in der die Komponenten definiert sind, aus denen der Cloud Service besteht und deren Beziehungen zueinander. Beispielsweise definiert die Topologie für einen Cloud Service, der eine Web Anwendung realisiert, dass die Anwendung aus einem Web Archiv, einen Web Server und einem Linux Betriebssystem besteht. Die Beziehungen definieren wiederum, dass das Web Archiv auf dem Web Server deployed wird und dass der Web Server der auf Linux installiert werden muss. Die eigentliche Topologie wird dabei durch *NodeTemplates* die die Komponenten repräsentieren und *RelationshipTemplates*, die die Beziehungen zwischen den Komponenten darstellen, spezifiziert. Die NodeTemplates sind wiederum konkrete Ausprägungen von abstrakten *NodeTypes* und die *RelationshipTemplates* sind Ausprägungen von *RelationshipTypes*. Folglich müssen während der Modellierung erst die NodeTypes und RelationshipTypes in der Topologie spezifiziert werden. Für die NodeTypes müssen Eigenschaften (Properties), Management-Operationen (Interfaces) definiert werden. Zusätzlich müssen konkrete Implementierungen, für die Business-Funktionalität (z.B. eine ZIP Datei die den Web Server enthält) und deren Management-

Operationen (z.B. ein konkreter Web Service mit dem der Web Server konfiguriert werden kann) ausgewählt werden. Die Implementation- und Deployment-Artefakte müssen zusätzlich die erforderliche Funktionalität zur Verfügung stellen, um die Überwachung und Einhaltung der Policies zu ermöglichen. Allerdings muss dies nicht die komplette Funktionalität für die Policy-Umsetzung sein. Ein Teil kann, je nach Implementierung, auch vom TOSCA Container und von den Plänen bereitgestellt werden. Die Policies des Cloud Services können während der Modellierung auf verschiedenen Ebenen definiert werden. So können Policies beispielsweise für den ganzen Cloud Service oder aber auch nur für eine einzelne Komponente definiert werden. Dabei ist es dem Modellierer überlassen, welche Sprache er zur Modellierung der Policies verwendet. Wenn er jedoch sicher sein will, dass die Policies eingehalten werden, muss er sicher gehen, dass der TOSCA Container die Policy Sprache interpretieren kann.

In der Phase *Paketierung der Anwendung* wird ein *Cloud Service Archiv* (CSAR) erstellt. Dieses Archiv repräsentiert den Cloud Service, d.h. es enthält alle Komponenten, die für den Betrieb des Cloud Services notwendig sind. Dies sind zum einen die Topologie und die Pläne, die in der vorherigen Phase erstellt wurden. Zum anderen besteht das Archiv aus den Deployment Artefakten und den Implementation Artefakten, die zum Betrieb und dem Management des Cloud Services notwendig sind und die nicht von Dritten extern bereitgestellt wurden. Optional können beliebige Meta-Informationen zum CSAR hinzugefügt werden, die zum Beispiel Auskunft über die CSAR Version geben können.

Das erstellte CSAR wird in der Phase *Deployment des Service Pakets* dem TOSCA Container übergeben. Nach der Übergabe kann der Container überprüfen, ob das CSAR Policies enthält, deren Einhaltung vom Container sichergestellt werden muss. So könnten zum Beispiel Policies festlegen, dass der Container nur CSARs bereitstellen darf, die digital signiert wurden. Dann analysiert er die Topologie des Cloud Services und die notwendigen Pläne und Implementation Artefakte werden vom Container bereitgestellt, d.h. betriebsbereit gemacht und aktiviert. Dadurch ist auch die notwendige Infrastruktur vorhanden, um die Policies zu überwachen, die während der Provisionierung des Cloud Services aktiv sind.

In der Phase *Provisionierung des Services* wird der Cloud Service betriebsbereit gemacht. Basierend auf den Informationen in der Topologie bzw. den Plänen werden die Deployment Artefakte des Cloud Services auf der jeweiligen Zielplattform bereitgestellt und mittels der von den Implementation Artefakten angebotenen Management-Operationen automatisch oder semi-automatisch konfiguriert und aktiviert. Nach dem Abschluss der Provisionierung ist der Cloud Service betriebsbereit. Während der Provisionierung wird die Einhaltung der Policies vom Container oder den in der vorherigen Phase bereitgestellten Implementation Artefakte überwacht und sichergestellt.

Während der *Nutzungsphase* nimmt der Cloud Service Nutzeranfragen entgegen und bearbeitet sie. Über die Management-Operationen der Implementation-Artefakte kann der Service verwaltet bzw. umkonfiguriert werden. Neben dem Container und den Implementation-Artefakten tragen hier auch die Deployment Artefakte zur Überwachung und Einhaltung der Policies bei. So kann ein Deployment Artefakt das eine virtuelle Maschine repräsentiert zum Beispiel Informationen zum Energieverbrauch und der Prozessorauslastung sammeln und an den Container oder die Implementation-Artefakte zur Policy-Übewachung senden.

In der *Monitoringphase* wird das Laufzeitverhalten des Cloud Services mittels der Implementation-Artefakte überwacht und u.U. protokolliert. Dabei kann zum Beispiel der korrekte Betrieb des Services überwacht werden, als auch das Nutzerverhalten. Auf Basis dieser Informationen können zukünftige Versionen des Cloud Services entsprechend optimiert

werden. Die Überwachung des Cloud Services geschieht unabhängig von den definierten Policies.

Nach dem Ende der Nutzung des Cloud Services wird er in der De-Installationsphase von der Zielplattform entfernt. Analog zur Provisionierungsphase wird die De-Installation basierend auf der Topologie bzw. den Plänen mittels der Management-Operationen der Implementation-Artefakte durchgeführt. Auch in dieser Phase können Policies aktiv sein, deren Einhaltung durch den Container und den Implementation-Artefakten realisiert wird.

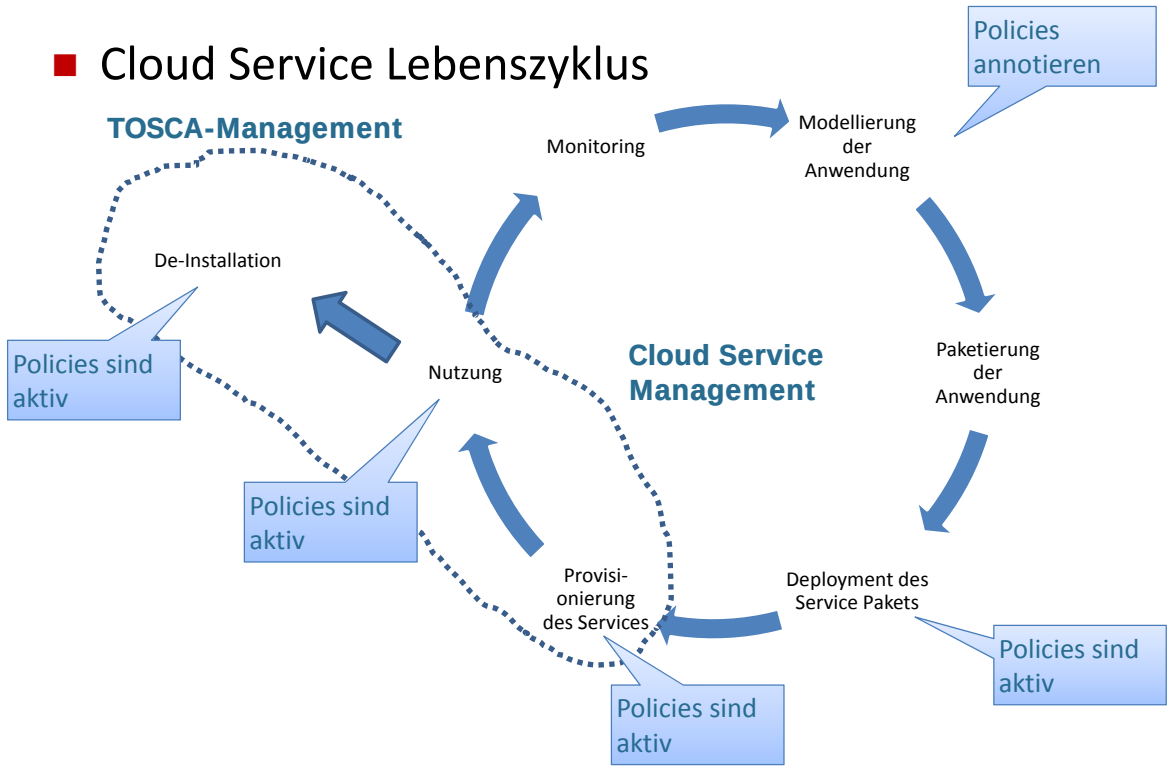


Abbildung 32 Lebenszyklus eines Cloud Services und seiner Policies

AP4.4 Implementierung der entworfenen Cloud-Management-Architektur

TOSCA Container

Das IAAS hat zusammen mit dem Trusted Cloud Projekt CloudCycle (01MD11023) einen TOSCA Container namens OpenTOSCA[7] entwickelt. Die Unterstützung von Policies durch OpenTOSCA wird im Folgenden detailliert betrachtet. Dazu wird zunächst die grundlegende Architektur des Containers beschrieben. Anschließend wird gezeigt, wie diese Architektur so erweitert werden kann, dass sie Policies unterstützt.

Architektur des OpenTOSCA Containers

In diesem Kapitel wird die Architektur des am IAAS entwickelten OpenTOSCA Containers vorgestellt, ohne zunächst auf die Umsetzung von Policies einzugehen. Abbildung 33 gibt einen vereinfachten Überblick über die Architektur von OpenTOSCA. Der Container wurde intern als serviceorientierte Architektur (SOA) auf der Basis von OSGi und deklarativen Services realisiert. Im Folgenden werden die einzelnen Komponenten des OpenTOSCA Containers und ihre Funktionalität näher vorgestellt.

[7] http://www.iaas.uni-stuttgart.de/OpenTOSCA/

Nach außen bietet der Container eine proprietäre, das heißt nicht standardisierte, Schnittstelle an welche als REST API umgesetzt wurde („OpenTOSCA Container API“). Über diese Schnittstelle kann der Container verwaltet sowie seine Kernfunktionalitäten verwendet werden. Klienten dieser Schnittstelle können beispielsweise TOSCA Modellierungstools oder graphische Administrationstool sein. Im OASIS TOSCA Standardisierungskomitee gibt es Bestrebungen, gewisse Funktionalitäten eines TOSCA Containers über eine standardisierte Schnittstelle zur Verfügung zu stellen („Plan Portability API“). Die Standardisierung dieser Schnittstelle soll die Portabilität vor allem von Plänen sicherstellen.

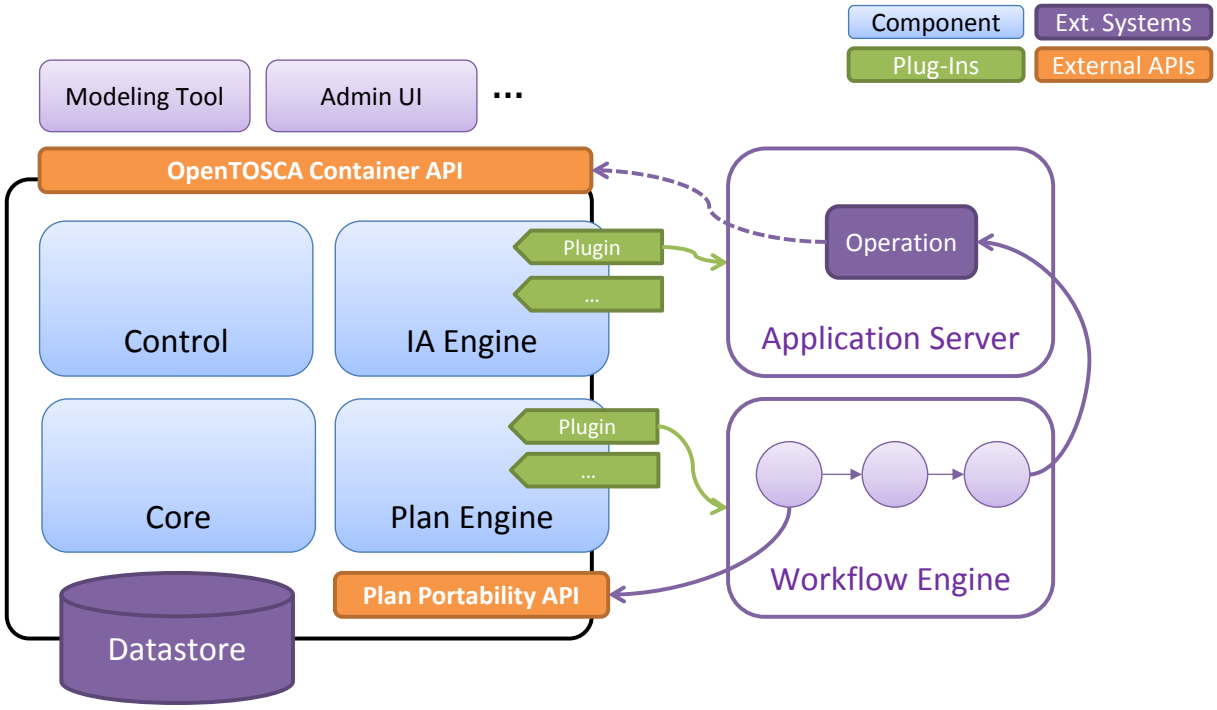


Abbildung 33 Architektur des OpenTOSCA Containers

Intern besteht der OpenTOSCA Container aus vier Hauptkomponenten. Die „Control“ Komponenten nimmt alle Anfragen der äußeren Schnittstellen entgegen und initiiert und überwacht die interne Abarbeitung dieser Anfragen. Sie komponiert und koordiniert die Zusammenarbeit aller weiteren Komponenten und stellt damit eine zentrale Steuerungsinstanz dar. Die „Core“ Komponente bietet eine Menge von Kernfunktionalitäten an, die von allen anderen Komponenten genutzt werden können. Beispiele dafür sind die Ablage und Verwaltung von TOSCA Modelldaten, das Entpacken und Ablegen von CSAR Paketdateien oder die Verwaltung von Endpunktinformationen. Die Speicherung von persistenten Daten erfolgt in einer Datenbank („Datastore“), diese Funktionalität wird von allen Komponenten gemeinsam verwendet. Die „IA Engine“ ist für das Deployment von Implementation Artefakten zuständig. Da es beliebig viele Typen von Implementation Artefakten geben kann, realisiert sie eine erweiterbare Plug-In Architektur. Die IA Engine selbst kann keine Implementation Artefakte deployen, sie bestimmt lediglich ein geeignetes Plug-In und ruft dieses anschließend auf. Der OpenTOSCA Container enthält bereits einige Plug-Ins welche, wie in Abbildung 33 dargestellt, einen externen Application Server nutzen, um Implementation Artefakte zu deployen. Ähnlich wie die IA Engine ist auch die Plan Engine modular aufgebaut. Sie kann mit Hilfe von Plug-Ins Pläne deployen und übernimmt zusätzlich auch das Binden der Pläne an die Endpunkte der verwendeten Implementation Artefakte. Der OpenTOSCA Container beinhaltet bereits ein Plug-In zum Deployment von BPEL Plänen auf eine externe Workflow Engine.

Erweiterungen von OpenTOSCA zur Unterstützung von Policies

Die Unterstützung von Policies in OpenTOSCA kann je nach Phase im Lebenszyklus (siehe Abbildung 32) unterschiedlich realisiert werden.

- Policies, welche für das Deployment des Service Pakets oder der Provisionierung des Services relevant sind, können bereits vor der Verarbeitung in OpenTOSCA entsprechen verarbeitet werden.
- Policies, welche auch zur Laufzeit eines Services sichergestellt werden, müssen entsprechend durch die Laufzeitumgebung abgedeckt werden.

OpenTOSCA bietet prinzipiell zwei Möglichkeiten, Policies zur Laufzeit sicherzustellen: (1) planbasiert und (2) Implementation Artefakt (IA)-basiert. Der planbasierte Ansatz stellt einen imperativen Ansatz dar, welcher spezifische Managementpläne zur Überwachung und Umsetzung der definierten Policies einer Topologie verwendet. Die einzelnen Schritte dieser Managementpläne müssen für jede Policy anhand der spezifischen Semantik definiert werden. Der IA-basierte Ansatz hingegen erweitert die für einen NodeType bestehenden Implementation Artefakte um spezifische Policy-Implementierungen.

Der Vorteil des planbasierten Ansatzes besteht darin, dass damit komplexe Policies auf kompletten Cloud Service Strukturen ausgedrückt werden können. Darin liegt aber auch der Nachteil dieses Ansatzes. Ein Plan bezieht sich immer auf genau eine bestimmte Service Struktur. Policy Pläne sind damit nicht wiederverwendbar, für jeden Service müssen eigene passende Pläne geschrieben werden. Dieser Nachteil kann durch eine modulare und (teilwiese) automatisierte Erstellung von Plänen gemindert werden.

Im IA-basierten Ansatz können Policies nur lokal auf Knotenebene formuliert werden. Komplexe Policies, welche sich auf einen kompletten Service beziehen, sind in diesem Ansatz nicht darstellbar. Dem entgegen steht aber der große Vorteilder Wiederverwendbarkeit. Mit dem IA-basierten Ansatz können Policy fähige NodeTypes erstellt werden. Diese NodeTypes können anschließend in beliebigen Service Beschreibungen verwendet werden. Dieser Vorteil schlägt sich auch in der Modellierung nieder. Der Modellierer eines Cloud Services muss lediglich die „richtigen“ NodeTypes verwenden und ist dann ohne weiteren Aufwand in der Lage, Policies zu benutzen.

Bei der Implementierung von Policies für den OpenTOSCA Container wurde der IA-basierte Ansatz gewählt. Eine Menge von „konventionellen“ NodeTypes wird so erweitert, dass sie Policies unterstützen. Das Ergebnis dieses Ansatzes ist eine Menge wiederverwendbarer Komponenten zur Modellierung Policy fähiger Cloud Services, welche vom OpenTOSCA Container unterstützt werden.

Für die Realisierung des IA-basierten Ansatzes müssen die bestehenden Implementation Artefakte um spezifische Policy Implementierungen erweitert werden. Die bisher bestehenden Implementation Artefakte bleiben funktionell jedoch vollständig erhalten, damit der entsprechende NodeType auch weiterhin ohne Policies verwendet werden kann. Jede Service Management Operation eines Implementations Artefakts besteht deshalb aus zwei alternativen Implementierungen, welche die Realisierung sowohl mit als auch ohne Policy abdecken. Für jeden Aufruf einer Management Operation muss das Implementation Artefakt prüfen, ob eine oder mehrere Policies sichergestellt werden müssen. Auf Basis dieser Prüfung wird anschließend eine konkrete Implementierung ausgewählt.

Die Realisierung dieses Konzepts erfolgt anhand einer zweistufigen Methode. Eine Übersicht ist in Abbildung 34 zu finden. Die erste Stufe umfasst das Deployment einer CSAR Datei und ist in drei Schritten unterteilt: Schritt 1 beinhaltet das Laden einer CSAR, Schritt 2 die entsprechende Verarbeitung innerhalb von OpenTOSCA. Für letzteren Schritt muss die

Abarbeitung einer CSAR Datei dahingehend erweitert werden, dass OpenTOSCA für jede neu geladene TOSCA Topologie Informationen bezüglich der NodeTemplates sowie der annotierten Policies an das korrespondierende IA weiterleitet. In Schritt 3 gibt ein IA dann eine Rückmeldung an OpenTOSCA, ob die an einem NodeTemplate annotierte Policy umgesetzt werden kann oder nicht, d.h. ob eine entsprechende Policy Implementierung vorliegt.

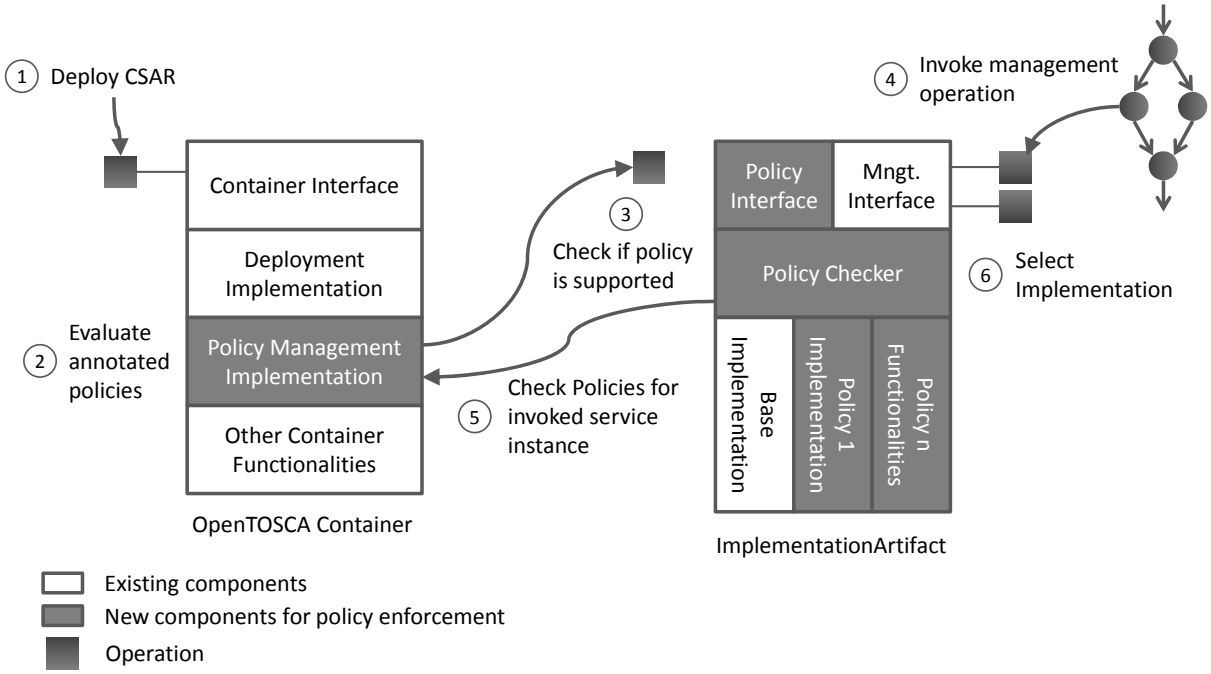


Abbildung 34 Policy Umsetzung in OpenTOSCA, IA basierter Ansatz

Die zweite Stufe beginnt mit Schritt 4 und dem Aufruf einer Managementoperation durch einen Managementplan. Da IAs immer auf Ebene eines NodeTypes definiert werden und nicht für konkrete NodeTemplates, muss das IA in Schritt 5 Informationen bezüglich der an das NodeTemplte annotierten Policies abfragen. Sind für das NodeTemplate, für welches eine Managementoperation aufgerufen wurde, Policies annotiert, wählt das IA in Schritt 6 eine geeignete Implementierung für die Sicherstellung der Policy aus.

Policy Verarbeitung in OpenTOSCA

Nachdem das Konzept der Policy-Verarbeitung abstrakt, d.h. unabhängig von einem konkreten Policy- bzw. TOSCA-Container aufgebaut ist, werden die Konzepte im Folgenden konkret anhand der Green-Policy und des OpenTOSCA Containers beschrieben.

Green Policy

Um Energieffizienz zu gewährleisten, können mit der Green-Policy virtuellen Maschinen (VMs), die in Amazon EC2 oder VMWare provisioniert sind, abhängig von ihrer Auslastung dynamisch Ressourcen zugeteilt werden. Für Amazon EC2 VMs bedeutet das, dass bei hohen Auslastung zum nächst größeren Amazon Instanztyp[8] gewechselt wird bzw. bei einer niedrigen Auslastung zum nächst niedrigeren Instanztyp. Bei VMWare VMs wird die Anzahl der virtuellen CPUs und der Arbeitsspeicher, die einer VM zur Verfügung gestellt werden, entsprechend der Auslastung der VM angepasst, d.h. bei hoher Auslastung werden der VM

[8] http://aws.amazon.com/de/ec2/instance-types/

zusätzliche CPUs zur Verfügung gestellt und bei geringer Auslastung wird die Anzahl der CPUs reduziert.

Aufbau (Schema, Attribute, Beispiele)

Wie eingangs erwähnt, kann die Green-Policy dafür genutzt werden, um die Energieeffizienz von Amazon und VMWare VMs zu verbessern. Folglich kann die Policy für NodeTemplates, die auf den VMWare NodeType oder den Amazon EC2 NodeType basieren definiert werden.

Das XML Schema der Policy ist im Folgenden beschrieben.

```
<xs:complexType name="tGreenPolicyTypeProperties">
        <xs:sequence>
        <xs:element name="PlanDownName" type="xs:string" />
        <xs:element name="PlanUpName" type="xs:string" />
        <xs:element name="MinCPU">
                <xs:simpleType>
                        <xs:restriction base="xs:integer">
                                <xs:minInclusive value="1"/>
                                <xs:maxInclusive value="100"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="MaxCPU">
                <xs:simpleType>
                        <xs:restriction base="xs:integer">
                                <xs:minInclusive value="1"/>
                                <xs:maxInclusive value="100"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="AnalysisInterval">
                <xs:simpleType>
                        <xs:restriction base="xs:integer">
                                <xs:minInclusive value="60"/>
                                <xs:maxInclusive value="86400"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
</xs:sequence>
</xs:complexType>
<xs:element name="GreenPolicyTypeProperties" type="tGreenPolicyTypeProperties"/>
```

Das Element „PlanDownName“ spezifiziert den Namen des BPEL Managementplans, der ausgeführt werden soll, um die einer VM zur Verfügung stehenden Ressourcen zu reduzieren. Der Plan wird nur dann ausgeführt, falls die mittels der im Element „MinCPU“ spezifizierten minimalen CPU-Auslastung unterschritten wird. Analog dazu wird der im Element „PlanUpName“ spezifizierte Managementplan für das Hinzufügen neuer Ressourcen zur VM ausgeführt, wenn die durch „MaxCPU“ definierte durchschnittliche CPU-Auslastung überschritten wurde. Im Element „AnalysisInterval“ kann das Intervall in Sekunden angegeben werden, in dem die Policy Implementierung die durchschnittliche minimale bzw. maximalen CPU Auslastung bestimmt.

Die Green-Policies werden für VMWare und Amazon EC2 folgen dem gleichen Schema, d.h. sie werden auf dieselbe Art und Weise definiert. Nur die Managementpläne, die in den Elementen „PlanDownName“ und „PlanUpName“ angegeben werden können, unterscheiden sich für Amazon EC2 und VMWare VMs. Bei EC2 VMs ist es beispielsweise nötig eine neue

VM anzulegen, um die Ressourcenzuteilung zu anzupassen und die Daten von der alten auf die neue VM zu migrieren. Für VMWare VMs reicht es hingegen aus, dass der Plan die Maschinen auszuschaltet, die Ressourcenkonfiguration anpasst und dann die Maschinen wieder hochfährt. Ein Beispiel für eine Green-Policy, die dem oben gezeigten Policy-Schema entspricht, ist im Folgenden „Policy Template“ beschrieben.

```
<PolicyTemplate type="greenPolicy:GreenPolicyType" id="GreenPolicyTemplate">
        <Properties>
        <greenPolicy:GreenPolicyTypeProperties>

<greenPolicy:PlanDownName>IncreaseVMWareUbuntuInstance</greenPolicy:PlanDownName>
        <greenPolicy:PlanUpName>DecreaseVMWareUbuntuInstance</greenPolicy:PlanUpName>
                <greenPolicy:MinCPU>50</greenPolicy:MinCPU>
                <greenPolicy:MaxCPU>90</greenPolicy:MaxCPU>
                <greenPolicy:AnalysisInterval>600</greenPolicy:AnalysisInterval>
        </greenPolicy:GreenPolicyTypeProperties>
</Properties>
</PolicyTemplate>
```

Entwurf

Um die Green-Policies zu interpretieren und zu implementieren, wurden die bestehenden VMWare und Amazon EC2 Implementation Artefakte erweitert und basierend auf den Policy-Konzepten eine entsprechende Monitoring Umgebung aufgesetzt. Außerdem wurde der OpenTOSCA Container um eine API erweitert, so dass durch die Policy Engine auf die im Policy Template definierten Daten zugegriffen werden kann. Zudem kann die Policy Engine dadurch die entsprechenden Managementpläne ausführen. Basierend auf den abstrakt beschriebenen Konzepten zur Verarbeitung von Policies, wurde die in Abbildung 35 dargestellte Umgebung zur Umsetzung der Green Policies erstellt.

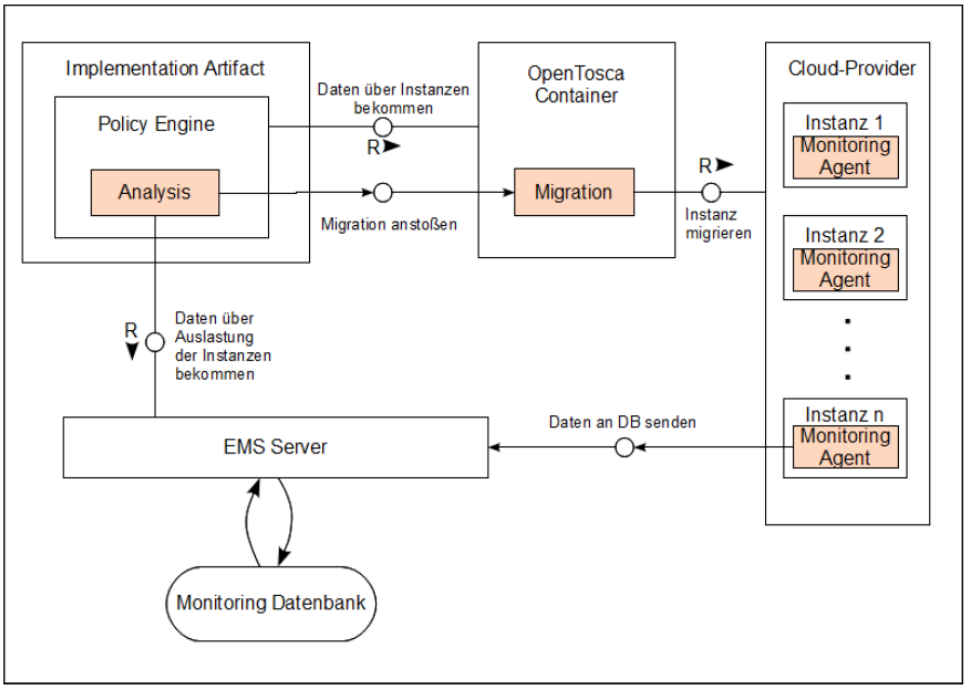


Abbildung 35 Komponenten für die Implementierung der Green Policy

Zur Umsetzung der Green-Policy wurde die Policy Engine in die entwickelten Implementation Artefakte integriert, da Teile der Implementierung einer Policy Engine vom jeweiligen Implementation Artefakt abhängig sind. Das heißt, zur Laufzeit existiert für jedes Implementation Artefakt eine separate Policy Engine. Im Falle der Green-Policy eine für das

Amazon EC2 und eine für das VMWare Implementation Artefakt. Jede Policy Engine kann eine beliebige Anzahl von VM Instanzen überwachen. Die Policy Engine ist für die Steuerung der Monitoring Agenten und das Starten der Managementpläne verantwortlich. Das eigentliche Auswerten CPU Auslastungsdaten wird vom EMS Server übernommen. Die Policy-Engine ist so implementiert, dass zukünftig auch andere Leistungsdaten, wie zum Beispiel die Arbeitsspeicherauslastung, erfasst und entsprechend darauf reagiert werden kann.

Das Zusammenspiel der verschiedenen Komponenten ist in den folgenden Use Cases beschrieben.

Monitoring Agent auf VM Installieren	
Ziel:	Der Monitoring Agent soll auf einer zu überwachenden VM installiert werden
Vorbedingung:	Eine VM wurde provisioniert
Nachbedingung:	Der Monitoring Agent ist auf der VM installiert und sendet Informationen über CPU Auslastung
Akteure:	Policy Engine des jeweiligen IAs (Amazon EC2 bzw. VMWare IA)
Regulärer Ablauf:	1. Policy Engine installiert Monitoring Agent auf VM 2. Policy Engine startet Monitoring Agent

Auslastungsdaten Ermitteln	
Ziel:	CPU Auslastung ermitteln
Vorbedingung:	Zeitpunkt zur Datensammlung ist eingetreten
Nachbedingung:	CPU Auslastungsdaten sind in der Monitoring Datenbank des EMS Servers gespeichert Monitoring Agent wartet auf nächsten Zeitpunkt für Datensammlung
Akteure:	Monitoring Agent
Regulärer Ablauf:	1. Monitoring-Agent sammelt die Werte der aktuellen CPU Auslastung 2. Monitoring-Agent sendet diese Werte an die Datenbank des EMS Servers

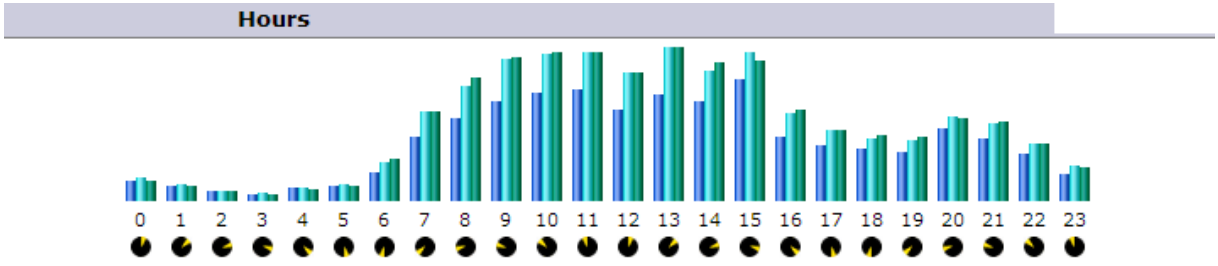
CPU Auslastung Analysieren	
Ziel:	Die Durchschnittliche CPU Auslastung einer VM soll ermittelt und entsprechend darauf durch reagiert werden
Vorbedingung:	Der im Element „AnalysisInterval“ des Policy Templates definierte Analysezeitraum für die durchschnittliche CPU Auslastung ist abgelaufen
Nachbedingung:	Die Policy Engine wartet erneut, bis der im „AnalysisInterval“ definierten Zeitraum abgelaufen ist, um die nächste Analyse zu starten
Akteure:	Policy Engine des jeweiligen IAs, EMS Server, Managementplan
Regulärer Ablauf:	1. Die Policy Engine ermittelt vom EMS Server die durchschnittlichen CPU-Auslastungswerte der VMs 2. Die Policy Engine vergleicht für jede VM, ob die Auslastungswerte in den Grenzwerten liegen, die im Policy Template durch die Elemente „MinCPU“ und „MaxCPU“ definiert wurden.
Sonderfall	2a. Durchschnittliche CPU Auslastung von EC2 VM liegt unter dem in „MinCPU“ definierten Wert und die Policy Engine startet über die API des OpenTOSCA Containers den in „PlanDownName“ spezifizierten Managementplan i. Plan startet neue EC2 VM mit nächst kleineren Instanztypen als die unausgelastete VM ii. Plan migriert Daten von unausgelasteter VM auf die neue VM iii. Plan schaltet unausgelastete VM ab iv. Policy Engine startet Überwachung der neuen VM 2b. Durchschnittliche CPU Auslastung von EC2 VM liegt über dem in „MaxCPU“ definierten Wert und die Policy Engine startet über die API des OpenTOSCA Containers den in „PlanUpName“ spezifizierten Managementplan i. Plan startet neue EC2 VM mit nächst größeren Instanztypen als die ausgelastete VM ii. Plan migriert Daten von ausgelasteter VM auf die neue VM iii. Plan schaltet ausgelastete VM ab iv. Policy Engine startet Überwachung der neuen VM 2c. Durchschnittliche CPU Auslastung von VMWare VM liegt unter dem in „MinCPU“ definierten Wert und die Policy Engine startet über die API des OpenTOSCA Containers den in „PlanDownName“ spezifizierten Managementplan i. Plan stoppt VM ii. Plan reduziert die Anzahl der VM zugewiesenen CPUs (falls der VM mehr als eine CPU zugewiesen ist) iii. Plan startet VM wieder 2d. Durchschnittliche CPU Auslastung von VMWare VM liegt über dem in „MaxCPU“ definierten Wert und die Policy Engine startet über die API des OpenTOSCA Containers den in „PlanUpName“ spezifizierten Managementplan i. Plan stoppt VM ii. Plan erhöht die Anzahl der VM zugewiesenen CPUs iii. Plan startet VM wieder

5. AP5: Modellbildung und Simulation

AP5	Modellbildung und Simulation	Leitung: DRESO
Ziele (IAAS)	– Simulation von Anwenderinfrastrukturen unter Berücksichtigung des Energieverbrauchs	
Ergebnisse (IAAS)	– Lastprofile für Simulation – Bereitstellung und Instrumentierung Lastgenerator	

AP5.1 Definition der Simulationsanforderungen

Der Energieverbrauch der Maerker Anwendung sollte unter realistischen Bedingungen erfasst werden. Deshalb war es nötig, dass die Simulation ein realistisches Auslastungsprofil der Maerker Anwendung erzeugt. Das heißt, die Anzahl und Art der Aufrufe sollten das tatsächliche Nutzerverhalten wiederspiegeln. Dazu wurde auf die Statistikseite der Maerker Anwendung zugegriffen. Diese Seite erfasst alle Zugriffe auf die Maerker Anwendung und bereitet diese Informationen statistisch und grafisch auf. So kann auf die durchschnittliche Anzahl der Zugriffe, die durchschnittliche Menge an übertragenen Daten etc. der letzten 5 Jahre zugegriffen werden. Dabei können verschiedene Zeitintervalle betrachtet werden, wie die durchschnittliche Anzahl der Zugriffe in einem bestimmten Monat, einer bestimmten Woche oder zu einer bestimmten Uhrzeit (wie im Beispiel in Abbildung 36).



Hours	Pages	Hits	Bandwidth	Hours	Pages	Hits	Bandwidth
00	1912	2211	35.31 MB	12	8948	12535	221.55 MB
01	1432	1583	23.60 MB	13	10311	14951	265.44 MB
02	862	949	15.73 MB	14	9685	12658	240.33 MB
03	636	706	11.70 MB	15	11906	14546	242.65 MB
04	1213	1304	17.74 MB	16	6269	8495	158.77 MB
05	1332	1565	25.05 MB	17	5427	6899	122.56 MB
06	2782	3711	71.16 MB	18	5033	6015	113.51 MB
07	6213	8748	154.80 MB	19	4731	5943	109.98 MB
08	8058	11233	213.38 MB	20	7049	8189	143.41 MB
09	9683	13928	250.08 MB	21	6126	7489	136.60 MB
10	10484	14299	259.32 MB	22	4538	5508	98.41 MB
11	10947	14602	259.45 MB	23	2590	3340	58.07 MB

Abbildung 36 Durchschnittliches Lastprofil Maerker Anwendung nach Stunden

Für die Simulation wurde das Lastprofil der Maerker pro Stunde für eine Woche erfasst. Die Art von Zugriffen konnte dem Lastprofil nicht entnommen werden. Deshalb wurden die drei wichtigsten Arten von Zugriffen simuliert, das Anlegen eines Eintrags in der Maerker Anwendung, das Suchen von Einträgen und der Lesezugriff auf Einträge.

Des Weiteren musste geklärt werden, wie automatisch, d.h. ohne Browser, auf die Maerker Anwendung zugegriffen werden kann. Da die Anwendung auf SixCMS als Content Management System aufsetzt und SixCMS eine SOAP-API für Datenzugriffe und Manipulationen anbietet, wurde sich für die Simulation über die API von SixCMS entschieden.

AP5.2 Bereitstellung von Simulationswerkzeugen

Um die im vorherigen Abschnitt definierten Simulationsanforderungen umzusetzen, wurde ein Tool benötigt, dass zum einen fähig ist, Lastprofile zu generieren und zum anderen Zugriffe über die SOAP-API ausführen kann, um auf die Daten der Maerker zuzugreifen bzw. diese zu manipulieren. Die Wahl fiel auf das Lasttest-Tool Load UI 2.6.8[9]. Das Tool bietet einen SOAP Client (sog. SOAPUI Runner), mit dem SOAP-Abfragen vordefiniert werden können. Außerdem bietet es Generatoren und sog. Scheduler, die miteinander verbunden werden, um eine definierte Anzahl an SOAP-Abfragen zu bestimmten Zeiten zu generieren. In den Generatoren wird definiert, wie viele SOAP-Anfragen pro Zeiteinheit gestellt werden. Die Scheduler wiederum erlauben es, dass Intervalle oder bestimmte Zeitpunkte festgelegt werden, an denen die Generatoren und damit die SOAP Anfragen ausgeführt werden.

AP5.3 Simulation von Migrationsszenarien

Das IAAS hat die Simulation der Migrationsszenarien, insbesondere der Anwendungsfall „Maerker“ auf technischer Ebene begleitet. Hierbei wurde sichergestellt, dass Informationen zum Workload erfasst und persistiert wurden, die Kommunikation der Simulationswerkzeuge sichergestellt und die Simulationsergebnisse entsprechend verarbeitet werden.

6. AP6: Piloterprobungen

AP6	**Piloterprobungen**	**Leitung: ZIT-BB**
Ziele (IAAS)	– Erprobung von Migrationsprozessen, Migrationswerkzeugen und erweiterter Cloud-Management-Werkzeuge bei den Anwendergruppen	
Ergebnisse (IAAS)	– Umsetzung des Maerker Use Cases mit TOSCA (Struktur und Management) – Automatisierte Provisionierung des Maerker Use Cases	

AP6.1 Definition der Pilotierungen

Das IAAS hat den Anwendungsfall Maerker als TOSCA-Dienst beschrieben. Dazu wurden die im Rahmen des Projektvorhabens entwickelten NodeTypes verwendet. Maerker selbst ist eine Web-basierte Anwendung, welche es Bürgern erlaubt, die öffentliche Verwaltung über Missstände in ihrer Region zu informieren. Die Beschreibung dieser Anwendung als Cloud Service Archive (CSAR) setzt voraus, dass die in Abbildung 26 gezeigten Bestandteile bereitgestellt werden. Im Folgenden werden die für Maerker entwickelten Bereiche im Detail vorgestellt.

*Anwendungsstruktur***.** Die Anwendungsstruktur der Maerker Anwendung, d.h. die Beschreibung der für den Betrieb notwendigen Anwendungskomponenten und ihrer Relationen, wird in Abbildung 37 gezeigt. Die Modellierung dieser Anwendungsstruktur erfolgte mit Hilfe des Tools *Winery*, welches im Rahmen des Forschungsvorhabens CloudCycle am IAAS entwickelt wurde. Alternativ ist die Modellierung der Anwendungsstruktur auch manuell möglich.

Ausgangsbasis der Topologie ist eine virtuelle Maschine, welche auf einer VMware Infrastruktur betrieben wird. Das Betriebssystem ist durch die Komponente Ubuntu realisiert. Als Middleware wird ein Apache Web Server eingesetzt. Zudem wird die Datenhaltung durch eine MySQL Datenbank umgesetzt. Die Komponente SixCMS repräsentiert das gleichnamige

[9] http://www.loadui.org/

Content Management System und stellt die eigentliche Laufzeitumgebung der Maerker Anwendung bereit.

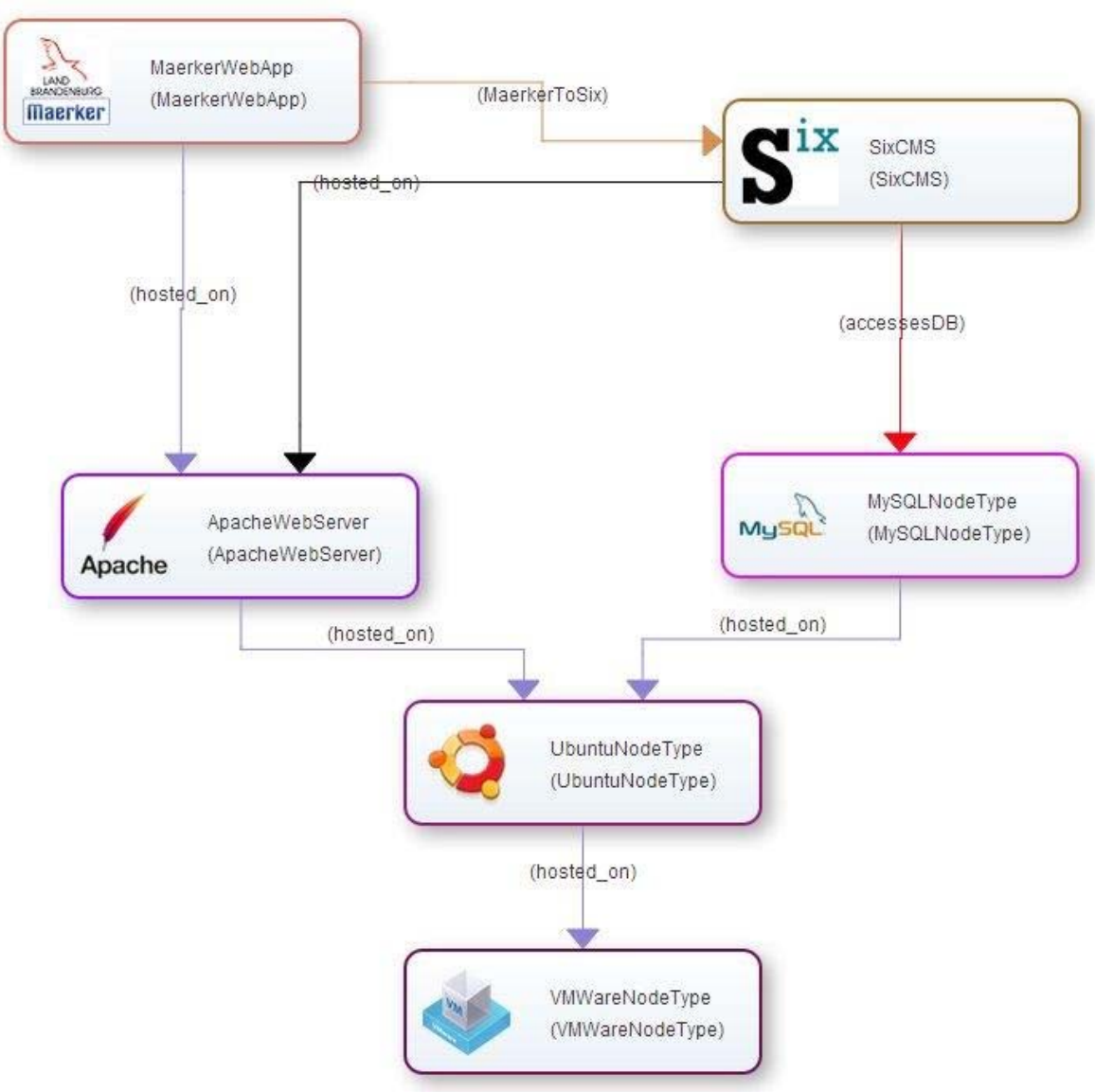


Abbildung 37 Maerker Anwendungstopologie

Pläne. Die in einer CSAR enthaltenen Pläne realisieren verschiedene Managementaufgaben. Im minimalsten Fall ist dies jeweils ein Plan zum Bereitstellen und Terminieren der Anwendung. Für Maerker wurden beide Pläne entwickelt. Abbildung 38 zeigt zunächst den in BPEL realisierten Build Plan, visualisiert im BPEL Eclipse Designer. Der linke Teil der Abbildung zeigt die vorbereitenden Schritte, welche die in den Service Templates der Anwendung spezifizierten Informationen abfragen. Auf Basis dieser Informationen wird im rechten Teil der Abbildung die Operation „InvokeUbuntuInstall“ aufgerufen, welche das Deployment der NodeTypes VMWareNodeType und UbuntuNodeType implementiert. Anschließend werden die weiteren Anwendungskomponenten mittels der spezifizierten NodeTypes nacheinander aufgerufen (Apache Web Server, MySQL Datenbank und SixCMS).

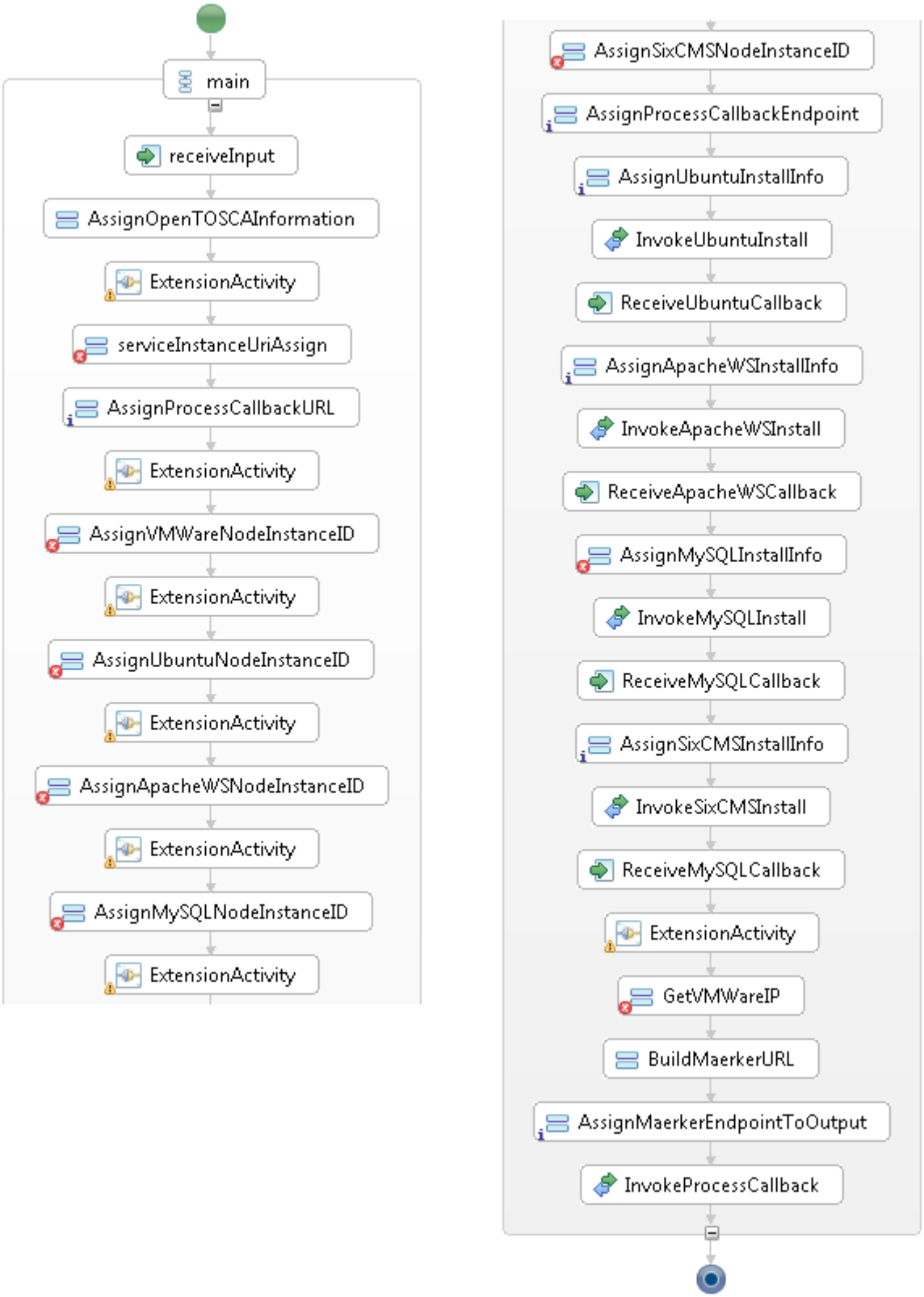


Abbildung 38 Maerker Build Plan

Abbildung 39 zeigt den entsprechenden Termination Plan, welcher eine Instanz der Anwendung terminiert. Hierzu wird zunächst die unterliegende virtuelle Maschine ausgeschalten (*PowerOffVM*). Im nächsten Schritt wird die ausgeschaltete virtuelle Maschine gelöscht. Beide Operationen werden durch das *Implementation Artifact* des VMWareNodeType bereitgestellt.

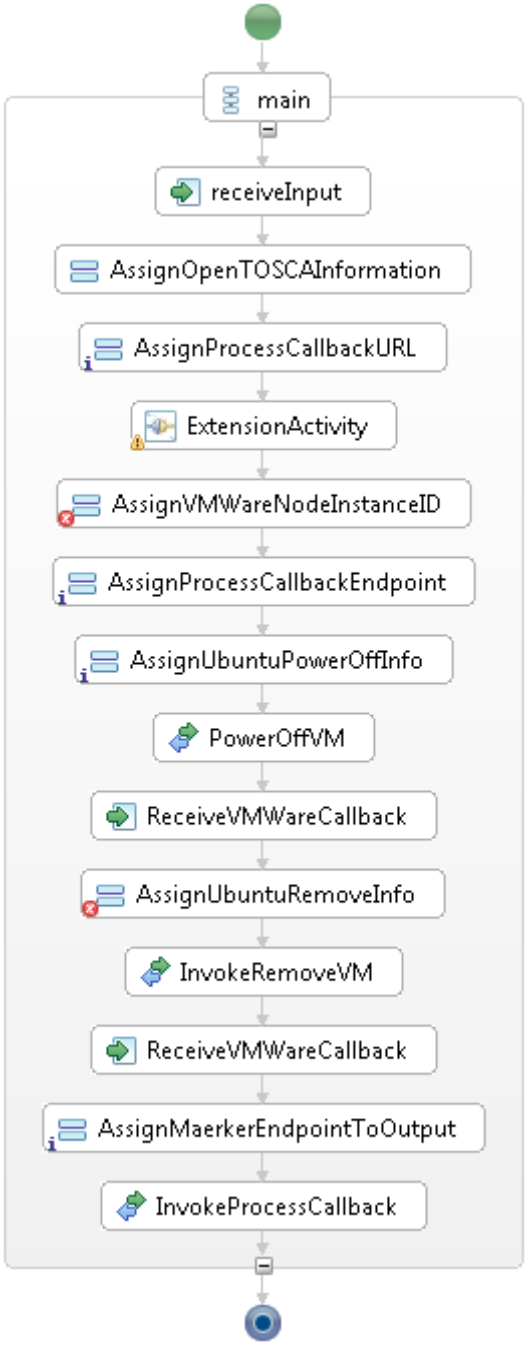


Abbildung 39 Maerker Termination Plan

Artefakte. Die erstellte Maerker CSAR enthält alle für die Bereitstellung und das Management der Anwendung erforderlichen Implementation und Deployment Artifacts. Die Implementation Artifacts der einzelnen NodeTypes sind jeweils als Web Service implementiert. Für die Installation von SixCMS sind zudem verschiedene Deployment Artifacts notwendig, welche sowohl die SixCMS Binaries, als auch erforderliche Konfigurationsdateien enthalten.

AP6.2 Dokumentation der Migrationsprozesse

Die Dokumentation der Migrationsprozesse erfolgte durch die entsprechenden Anwendungspartner. Das IAAS stand in dieser Phase als methodischer und technischer Ansprechpartner zur Verfügung.

7. AP7: Geschäftsmodelle und Dienstleistungen

AP7	Geschäftsmodelle und Dienstleistungen	Leitung: FZID
Ziele (IAAS)	– Begleitung der Entwicklung von Geschäftsmodellen zur Verwertung der Projektergebnisse	
Ergebnisse (IAAS)	– Nutzungspotenziale für TOSCA-Projektergebnisse	

AP7.1 Geschäftsmodelle

Die seitens des IAAS entwickelten Modelle, Methoden und Werkzeuge unterstützen IT-Anwender bei Migrationsentscheidungen hinsichtlich der Nutzung grüner Cloud-Dienste. Zum einen führen verbesserte Entscheidungen zu einem direkten Nutzen für diese Anwender. Zum anderen wohnt den Projektergebnissen ein weitergehendes Nutzungspotenzial inne. Nachfolgend werden diese Potenziale anhand der beiden Ergebnisse „TOSCA-Editor“ und „TOSCA-Container“ aufgezeigt. Hierbei erfolgt die Zuordnung zu generischen Rollen in Cloud-Wertschöpfungssystemen.

Rolle	Nutzungspotenzial
Customer (nutzt Cloud-Dienste)	TOSCA-konforme Modellierung von individuellen AIS, für die es keine Drittlösung gibt (sinnvoll für vergleichsweise große IT-Anwender, die eigene TOSCA-Kompetenz aufbauen und viele Migrationen durchführen). Hinzufügung von individuellen Green-Policies, die aus Erfahrungswissen der AIS-Nutzung gewonnen werden.
Service Provider (bietet grüne Cloud-Dienste als IaaS, PaaS oder SaaS an)	TOSCA-konforme Modellierung von selbst erstellten Applikationen, um diese bei externen Providern betreiben zu lassen (sinnvoll für Applikationen mit vielen Kunden und Betrieb bei vielen, verschiedenen auch wechselnden Providern). Hinzufügung von Default-Green-Policies, die aus Erfahrungswissen der Produktentwicklung/-nutzung gewonnen werden.
Infrastructure Provider (betreibt Infrastruktur und stellt sie Service Providern zur Verfügung)	Betrieb des TOSCA-Containers in der eigenen Infrastruktur, um Applikationen von Dritten automatisiert zu deployen und energetisch zu steuern. Hinzufügung von individuellen Green-Policies, die aus Erfahrungswissen des Infrastruktur-Betriebs gewonnen werden.
Consulting (berät Dritte bei Auswahl und Gestaltung von Cloud-Diensten)	Unterstützung von *Customer* und *Service Provider* bei der TOSCA-konformen Modellierung von AIS bzw. Applikationen. Unterstützung von *Customer*, *Service Provider* und *Infrastructure Provider* bei der Entwicklung von Green-Policies.

Tabelle 8: Zuordnung von TOSCA-Werkzeugen zu Rollen

b. Gegenüberstellung der vorgegebenen Ziele

AP	Zielsetzung	Zielerreichung
1	– Analyse derzeitiger Anwenderinfrastrukturen – Erhebung von Anforderungen an Migrationsprozesse (Cloud Migration)	– Methode zur Beschreibung von Anwenderinfrastrukturen – Vorgehensmodell zur Erfassung von – Modelle von Anwenderinfrastrukturen – Anforderungen, insb. im Bereich TOSCA
2	– Entwicklung und Spezifikation von Architekturmodellen für Cloud-basierte energieoptimierte Anwenderinfrastrukturen	– Green Policies zur Beschreibung der Anforderungen an die Nachhaltigkeit bzw. Energieeffizienz – Konzept zur Definition von Policies innerhalb von TOSCA – Energieeffizienzeigenschaften untersuchter Cloud-Lösungen und Architekturen – Green Business Process Patterns Methode zur ökologischen Analyse von Geschäftsprozessen
3	– Bereitstellung von Migrationsverfahren und –werkzeugen – Entwicklung von Migrationsprozessen	– Vorgehensmodell für die Cloud-Migration – Werkzeugunterstützung – Cloud-Service-Beschreibung mittels TOSCA – Open-Source-Laufzeitumgebung für TOSCA
4	– Spezifikation und Implementierung von Werkzeugen für energieeffizientes Cloud-Management	– Policy Taxonomie – Policy Taxonomie für „green“ Policies – Policy Taxonomie für TOSCA Cloud Services – Konzeptionelle Erweiterung von OpenTOSCA – Implementierung der green Policies Unterstützung in OpenTOSCA
5	– Simulation von Anwenderinfrastrukturen unter Berücksichtigung des Energieverbrauchs	– Lastprofile für Simulation – Bereitstellung und Instrumentierung Lastgenerator
6	– Erprobung von Migrationsprozessen, Migrationswerkzeugen und erweiterter Cloud-Management-Werkzeuge bei den Anwendergruppen	– Umsetzung des Maerker Use Cases mit TOSCA (Struktur und Management) – Automatisierte Provisionierung des Maerker Use Cases
7	– Begleitung der Entwicklung von Geschäftsmodellen zur Verwertung der Projektergebnisse	– Nutzungspotenziale für TOSCA-Projektergebnisse

Tabelle 9: Zielsetzung und Zielerreichung

7. Wichtigste Positionen des zahlenmäßigen Nachweises

Die nachfolgende Tabelle umfasst die wichtigsten Positionen des beiliegenden vollständigen zahlenmäßigen Nachweises:

Position	Betrag
0812 Beschäftigte E12 – E15	513.878,24 €
0822 Beschäftigungsentgelte	36.731,90 €
0843 Sonstige allgemeine Verwaltungsausgaben	745,78 €
0846 Dienstreisen	29.256,26 €
Gesamtausgaben	**580.612,18 €**

Tabelle 10: Positionen des zahlenmäßigen Nachweises

Mittelart 0812: Im Projektverlauf wurden insgesamt 3 verschiedene, wissenschaftliche Mitarbeiter eingesetzt. Die Kompetenzen verteilten sich dabei auf Geschäftsprozess-management, Cloud Architekturen sowie die Entwicklung von Werkzeugen.

Mittelart 0822: Der Einsatz von wissenschaftlichen Hilfskräften war geringer als geplant. Zusätzlich handelte es sich um Studierende ohne ersten wissenschaftlichen Studienabschluss, so dass der tatsächliche Personalkostensatz je Stunde niedriger war als geplant.

Mittelart 0835: -

Mittelart 0843: Hierzu gehören u.a. Softwarelizenzen und Gebühren unter 410 Euro.

Mittelart 0846: Die im Projektverlauf entstandenen Reisekosten untergliedern sich in die folgenden, wichtigsten Anlässe:

- Teilnahme an den regelmäßigen, ein- bis zweitätigen Projekttreffen und Workshops mit Partnern in Stuttgart, Böblingen und Potsdam
- Teilnahme an Sitzungen der IT2Green-Fachgruppen in Berlin
- Teilnahme an den Jahreskonferenz und am Abschlusskongress IT2Green in Berlin
- Teilnahme an wissenschaftlichen Veranstaltungen, darunter Konferenzen zur Vorstellung von Projektpublikationen (siehe Kapitel 11a).

8. Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die Projektergebnisse konnten nur im Projektkonsortium bestehend aus den komplementären Forschungspartnern und den Software- und Anwenderunternehmen erreicht werden. Die Bewältigung des Forschungsaufwandes hätte nicht allein und aus Eigenmitteln bestritten werden können, da Anzahl und Schwierigkeit der hierzu notwendigen wissenschaftlich-technischen Neuentwicklungen und Technologieintegrationen dem entgegenstanden.

Die geleisteten Arbeiten des IAAS erfolgten ausgehend von dem aktuellen Stand des Wissens zu Cloud-Infrastrukturen, Cloud Migration, Cloud-SLA-Management. Zu allen eingesetzten Sprachen, Methoden und Werkzeugen lagen bereits hinreichende Erfahrungen aus verschiedenen Vorarbeiten und Verbundprojekten vor, so dass aufwendige Einarbeitungen entfielen. Es wurden soweit als möglich standardisierte Beschreibungssprachen und quelloffene Basis-Werkzeuge verwendet, um hier potenzielle Risiken zu minimieren. Die Forschungsarbeiten wurden auf die Konzeption und prototypische Realisierung konzentriert.

Im Vordergrund standen die Modell-, Methoden- und Werkzeugentwicklung und deren Evaluation nach Kriterien guter wissenschaftlicher Praxis. Die Modell- und Methodenentwicklung wurde hier auf charakteristische Szenarien beschränkt, die auf einen hohen Generalisierungserfolg hinwiesen. Alle Tätigkeiten wurden gemäß Projektplan durchgeführt und waren hinsichtlich Inhalt und Umfang angemessen. Die Forschungsarbeiten erfolgten derart, dass die Ergebnisse übertragbar sind und in künftige Forschungsarbeiten und in die Ausbildung von Studierenden und Nachwuchswissenschaftlern einfließen können.

9. Voraussichtlicher Nutzen

a. Wirtschaftlicher Nutzen

Entfällt für Universitäten.

b. Wissenschaftlicher Nutzen

Das Vorhaben reiht sich ideal in die langfristige Zielsetzung des IAAS ein, Verfahren für die kundenindividuelle Bereitstellung von Cloud-Anwendungen zu erarbeiten sowie durch Softwarelösungen zu unterstützen. Insbesondere die ökologische Erweiterung des SLA-Managements und damit die Messbarkeit des Umwelteinflusses von Cloud-Anwendungen und deren Prozessen setzen den Grundstein für die Anwendbarkeit der entwickelten Konzepte und Methoden zur effizienten Ressourcenallokation. Die ersten Erfahrungen bei der Erstellung von Cloud-Service-Paketen auf Basis von TOSCA für die Fachanwendung Maerker im Projekt erlauben zudem, die technische Weiterentwicklung des TOSCA-Standards selbst und der TOSCA-basierten Softwarewerkzeuge wie Editoren und TOSCA-Container voranzutreiben, auch in Hinblick auf Usability und Zuverlässigkeit.

Das Zusammenspiel der im Projektvorhaben erarbeiteten Verfahren im Bereich der Paketierung, Auswahl und automatisierten Bereitstellung von Cloud Anwendungen unter Beachtung Energierelevanter Kriterien sowie deren Beitrag zu einem quelloffenen und Standard basierten Cloud-Ökosystem ist in Abbildung 40 dargestellt.

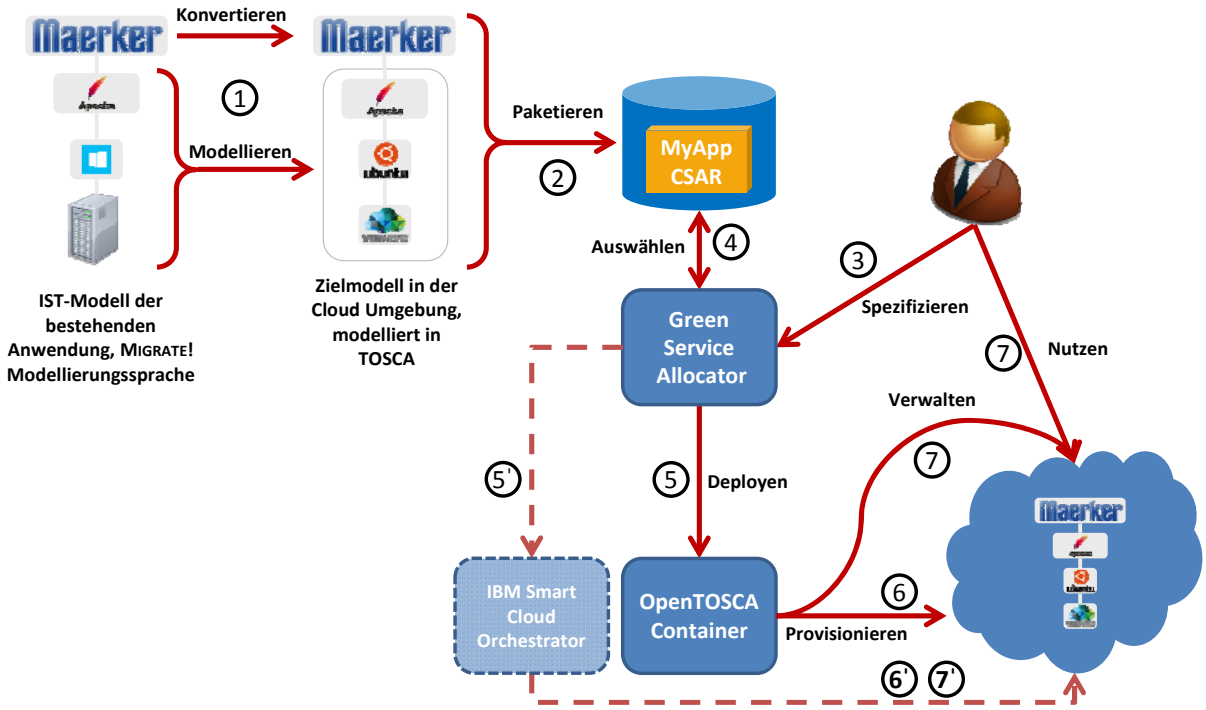


Abbildung 40: TOSCA Ökosystem

Für die Modellierung einer Cloud-Anwendung (Schritt 1) können die im Projektvorhaben entwickelten Lego4TOSCA NodeTypes, standardisierte Bausteine für die Modellierung von TOSCA-Topologien, verwendet werden. Im Rahmen der Modellierung können zudem die definierten Green Policies angewendet werden um damit energierelevante SLA in die Anwendungsbeschreibung zu integrieren, all dies basierend auf dem offenen TOSCA Standard. Das erstellte Anwendungsmodell wird als TOSCA CSAR paketiert und abgelegt (Schritt 2). Für die Spezifikation der Anforderungen und die darauf aufbauende Dienstauswahl integriert sich der vom Projektpartner FZID entwickelte Green Service Allocator in die Toolchain (Schritte 3 und 4). Das automatisierte Deployment (Schritt 5) sowie die automatisierte Provisionierung (Schritt 6) des ausgewählten Cloud Dienstes profitieren von der gewählten TOSCA Standardisierung. Die portable und standardisierte Beschreibung des Cloud Dienstes erlaubt die Nutzung unterschiedlicher TOSCA Container. Die Mitentwickelung des quelloffenen OpenTOSCA Containers im Rahmen des Projektvorhaben leistet einen wesentlichen Beitrag zur Verbreitung des TOSCA Standards und unterstützt seine Anwendung in Forschung und Lehre. Die im Projektvorhaben definierten Green Policies erlauben schlussendlich ein automatisiertes und energiesensitives Management zu Laufzeit (Schritt 7). In ihrer Gesamtheit tragen die im TOSCA Kontext erarbeiteten Beiträge zu einem quelloffenen und Standard basierten Cloud Service Ökosystem bei.

c. Anschlussfähigkeit

Das IAAS vertiefte mit dem Projekt MIGRATE! seine Kenntnisse im Bereich SLA und Policy-abhängiger Provisionierung und Überwachung von Cloud-Diensten. Die erarbeiteten Ergebnisse flossen direkt in die Antragsgestaltung weiterer Forschungsvorhaben sowie in bereits laufende Forschungsvorhaben ein.

Lfd. Nr.	Verwertungsaktivität	Zeithorizont
1	Der Exzellenzcluster SimTech der Universität Stuttgart (EXC 310/1) sowie das vom BMBF geförderte Projekt ECHO (01XZ13023G) nutzen die Ergebnisse um Online Services dynamisch unter Berücksichtigung verschiedener SLAs zu proviosionieren.	Ende 2015
2	Das IAAS führt das BMWi Projekt NEMAR (03ET40188) durch, bei welchem die dynamische und flexible Bereitstellung von Infrastrukturen und Anwendungen unter Berücksichtigung von SLAs eingesetzt werden können, um eine neue Marktrolle zur Bewirtschaftung von Stromnetzen zu unterstützen.	Ende 2016
3	Des Weiteren werden die Erfahrungen des IAAS im Bereich leistungsfähiger Cloud- und Service-Architekturen um Nachhaltigkeitsaspekte erweitert, so dass zukünftig in Architektur-entscheidungen neben Leistungsparametern, wie beispielsweise der Skalierbarkeit, auch Energieeffizienzparameter miteinfließen können.	Ende 2016

10. Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen

Energieeffizienz in der IT

Die Verbesserung der Energieeffizienz in der IT wurde auch in den letzten drei Jahren stetig vorangetrieben. Im Bereich der Energieeffizienz von Einzelkomponenten haben beispielsweise neue Fertigungsverfahren im Bereich von Prozessoren dazu geführt, dass die Energieeffizienz der neusten Prozessoren um ca. 40% verbessert wurde. Die Integration der verschiedenen Einzelansätze wird weiter auch dazu verwendet, die Energieeffizienz gesamter Rechenzentren zu verbessern. Bieswanger et al. (2012) haben beschrieben, wie durch ein intelligentes Zusammenspiel von Management-Komponenten eine lastabhängige, automatisierte Steuerung auf Basis von Echtzeitdaten, bspw. Sensoren, erfolgen kann. Pedram, M. (2012) zeigte die Herausforderungen für die Verbesserung der Energieeffizienz von Rechenzentren auf und beschrieb entsprechende Lösungen. Die Schwerpunkte liegen hier im Bereich der Ressourcenbereitstellung sowie des Energie- und Wärmemanagement in Rechenzentren. Beloglazov et al. (2012) greifen zusätzlich die Anforderungen an Cloud-Rechenzentren auf und beschreibt Heuristiken zur Ressourcenallokation und der damit einhergehenden Verbesserung des energieeffizienten Managements von Rechenzentren.

Cloud Computing

Feller et al. (2012) haben beschrieben, dass existierende Ansätze zur Bereitstellung und Migration von virtuellen Maschinen in vielen Fällen ausschließlich isoliert betrachtet werden. Die Autoren stellen ein System vor, welches diese einzelnen Technologien in ein holistisches Rahmenwerk integriert und somit eine transparente energiesensitive Steuerung von Private Cloud Infrastrukturen unter realen Lastverläufen erlaubt. Borgetto et al. (2012) beschreiben, dass sie in Cloud Computing eine Schlüsseltechnologie auf dem Weg zu nachhaltigen und energieeffizienten IKT Systemen sehen. In diesem Kontext beschreiben sie ein autonomes System sowohl zur Steuerung physischer Maschinen als auch zur Verwaltung und Migration virtueller Maschinen. Das Ziel dieses Ansatzes ist die Erhöhung der Energieeffizienz bei gleichzeitiger Einhaltung aller zugesicherten Dienstgütekriterien (Service Level Agreements, SLA). In (Garg and Buyya, 2012) wird ein umfassender Überblick über Cloud Computing Technologien und ihren Zusammenhang zu Aspekten der Energieeffizienz gegeben. Die Autoren leiten aus dem aktuellen Stand der Technik und Forschung eine Architektur einer

„grünen Cloud“ (Green Cloud Architecture) ab, um somit einen einheitlichen Blickwinkel zu etablieren welcher als eine gemeinsame Basis für Diskussion und Weiterentwicklung der Energieeffizienz im Bereich des Cloud Computing dienen kann.

Cloud Migration

Die Migration in Cloud-basierte Infrastrukturen wird weiterhin vorwiegend ökonomisch motiviert. Zur Unterstützung der technischen Herausforderungen wurden zahlreiche weitere Ansätze entwickelt. Andrikopoulos et al. (2013a) beschreiben, wie Anwendungen für den Betrieb in einer Cloud Umgebung angepasst werden müssen. In (Andrikopoulos, Song and Leymann, 2013) und (Andrikopoulos, Strauch and Leymann, 2013) betrachten sie zudem auf die Unterstützung der Migrationsdurchführung. So wird hier ein Decision Support System vorgestellt, das Nutzer bei der Vorbereitung und Durchführung der Migration einer Anwendung in die Cloud unterstützt. Auch Khajeh-Hosseini et al. (2012) adressieren die Herausforderungen bei der Identifikation von Cloud-fähigen Anwendungen. Ihr entwickeltes Adoption Toolkit unterstützt Entscheidungsträger dabei, Anwendungen für eine Cloud Migration auszuwählen. In Strauch et al. (2014) wird die Migration von Anwendungen am Beispiel von eScience Anwendungen aufgezeigt. Weiterhin nicht untersucht wurde die Energieeffizienz vor, während und nach einer Migration.

Cloud Werkzeuge

Chen et al. (2012) haben ein Modell für den Energieverbrauch in komplexen Cloud Umgebungen vorgestellt. Aufbauend auf diesem Modell wurde ein Werkzeug für die Analyse des Energieverbrauchs von Cloud Systemen entwickelt. Die Integration dieses Werkzeuges in Cloud Infrastrukturen erlaubt, neben der Überwachung des Energieverbrauchs, die Realisierung sowohl statischer als auch dynamischer Optimierungen des Betriebs in Bezug auf den Energieverbrauch. Der von Amazon Web Services (AWS) angebotene Dienst „Cloud Formation“ wurde in den vergangenen Jahren kontinuierlich weiter entwickelt und funktionell aufgewertet. Die ursprüngliche Fähigkeit zur Beschreibung von verteilten Anwendungen basierend auf AWS Ressourcen wurde immer weiter flexibilisiert und ausgebaut. Cloud Formation erlaubt damit die Beschreibung sowie automatische Bereitstellung weitaus heterogener Anwendungsarchitekturen als bisher, ist aber weiterhin auf AWS Cloud Infrastrukturen beschränkt. Im Bereich von Open Source basierten Cloud Systemen spielt das OpenStack System eine zentrale Rolle. Ein Teilprojekt von OpenStack, das Projekt HEAT, verfolgt einen ähnlichen Ansatz wie Amazon Cloud Formation. Bedingt durch die offene und erweiterbare Architektur des OpenStack Systems ist HEAT dabei weitaus generischer und flexibler. Um die Adoption der HEAT Technologie zu fördern wurde bei dessen Entwurf auf Kompatibilität mit dem von AWS Cloud Formation definierten Beschreibungsformat geachtet.

11. Veröffentlichungen der Projektergebnisse

a. Bereits erfolgte Veröffentlichungen

Beiträge in wissenschaftlichen Zeitschriften

1. Nowak, Alexander; Binz, Tobias; Fehling, Christoph; Kopp, Oliver; Leymann, Frank; Wagner, Sebastian: Pattern-driven Green Adaptation of Process-based Applications and their Runtime Infrastructure. In: Computing, Springer Wien, 2012.

Beiträge zu wissenschaftlichen Tagungen:

2. Nowak, Alexander; Breitenbücher, Uwe; Leymann, Frank: Automating Green Patterns to Compensate CO2 Emissions of Cloud-based Business Processes. In: IARIA Xpert Publishing Services (Hrsg): Proceedings of ADVCOMP 2014.
3. Haupt, Florian; Fischer, Markus; Karastoyanova, Dimka; Leymann, Frank; Vukojevic-Haupt, Karolina:Service Composition for REST. In: Proceedings of the 18th IEEE International EDOC Conference (EDOC 2014).
4. Vukojevic-Haupt, Karolina; Haupt, Florian; Karastoyanova, Dimka; Leymann, Frank: Service Selection for On-demand Provisioned Services. In: Proceedings of the 18th IEEE International EDOC Conference (EDOC 2014).
5. Haupt, Florian; Karastoyanova, Dimka; Leymann, Frank; Schroth, Benjamin: A model driven approach for REST compliant services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2014.
6. Haupt, Florian; Leymann, Frank; Nowak, Alexander; Wagner, Sebastian: Lego4TOSCA: Composable Building Blocks for Cloud Applications. In: Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014).
7. Wagner, Sebastian; Kopp, Oliver; Leymann, Frank: Choreography-based Consolidation of Multi-Instance BPEL Processes. In: SciTePress (Hrsg): Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014);Barcelona, Spain, April 3-5, 2014.
8. Nowak, Alexander; Leymann, Frank: Green Enterprise Patterns (to appear). In: Proceedings of the 20th Conference on Pattern Languages of Programs (PLoP), October 23-26, Allerton, IL, USA, 2013.
9. Nowak, Alexander; Leymann, Frank: Green Business Process Patterns - Part II. In: Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013).
10. Nowak, Alexander; Binz, Tobias; Leymann, Frank; Urbach, Nicolas: Determining Power Consumption of Business Processes and their Activities to Enable Green Business Process Reengineering. In: Proceedings of the 17th IEEE International EDOC Conference (EDOC 2013).
11. Wagner, Sebastian; Roller, Dieter; Kopp, Oliver; Unger, Tobias; Leymann, Frank: Performance Optimizations for Interacting Business Processes. In: Proceedings of the first IEEE International Conference on Cloud Engineering (IC2E 2013).
12. Waizenegger, Tim; Wieland, Matthias; Binz, Tobias; Breitenbücher, Uwe; Haupt, Florian; Kopp, Oliver; Leymann, Frank; Mitschang, Bernhard; Nowak, Alexander; Wagner, Sebastian: Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing. In: Meersman, Robert (Hrsg); Panetto, Herve (Hrsg); Dillon, Tharam (Hrsg); Eder, Johann (Hrsg); Bellahsene, Zohra (Hrsg); Ritter, Norbert (Hrsg); De Leenheer, Pieter (Hrsg); Dou Deijing (Hrsg): On the Move to Meaningful Internet Systems: OTM 2013 Conferences.
13. Nowak, Alexander; Karastoyanova, Dimka; Leymann, Frank; Rapoport, Andrej; Schumm, David: Flexible Information Design for Business Process Visualizations. In: Proceedings of the 2012 IEEE International Conference on Service-Oriented Computing and Applications.

14. Wagner, Sebastian; Fehling, Christoph; Karastoyanova, Dimka; Schumm, David: State Propagation-based Monitoring of Business Transactions. In: Proceedings of the 2012 IEEE International Conference on Service-Oriented Computing and Applications.
15. Binz, Tobias; Leymann, Frank; Nowak, Alexander; Schumm, David: Improving the Manageability of Enterprise Topologies Through Segmentation, Graph Transformation, and Analysis Strategies. In: Proceedings of 2012 Enterprise Distributed Object Computing Conference (EDOC).
16. Binz, Tobias; Fehling, Christoph; Leymann, Frank; Nowak, Alexander; Schumm, David: Formalizing the Cloud through Enterprise Topology Graphs. In: Proceedings of 2012 IEEE International Conference on Cloud Computing.
17. Nowak, Alexander; Binz, Tobias; Leymann, Frank; Schleicher, Daniel; Schumm, David; Wagner, Sebastian: Ein Konzept zur Identifikation ökologisch nachhaltiger Verbesserungspotentiale unter Bürgerbeteiligung. In: Tagungsband der Multikonferenz Wirtschaftsinformatik 2012.
18. Nowak, Alexander; Leymann, Frank; Schumm, David: The Differences and Commonalities between Green and Conventional Business Process Management. In: Proceedings of the International Conference on Cloud and Green Computing, CGC 2011.
19. Nowak, Alexander; Leymann, Frank; Schleicher, Daniel; Schumm, David; Wagner, Sebastian: Green Business Process Patterns. In: Proceedings of the 18th Conference on Pattern Languages of Programs, PLoP 2011.

Sonstige:

20. Binz, Tobias; Breitenbücher, Uwe; Haupt, Florian; Kopp, Oliver; Leymann, Frank; Nowak, Alexander; Wagner, Sebastian: OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In: Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC'13), 2013.

b. Geplante Veröffentlichungen

Keine.

Literaturverzeichnis

Amazon (2014) Amazon Web Services Simple Monthly Calculator, http://calculator.s3.amazonaws.com/index.html.

Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. (2013) How to adapt applications for the Cloud environment. In: Computing, Springer, Vol. 95(6), pp. 493-535.

Andrikopoulos, V., Song, Z., Leymann, F. (2013) Supporting the Migration of Applications to the Cloud through a Decision Support System. In: Proceedings of the 6th IEEE International Conference on Cloud Computing (CLOUD 2013), pp. 565-572, IEEE Computer Society, Santa Clara Marriott, CA, USA.

Andrikopoulos, V., Strauch, S., Leymann, F. (2013) Decision Support for Application Migration to the Cloud: Challenges and Vision. In: Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER'13), pp. 149-155, SciTePress, Aachen, Germany.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009) Above the Clouds: A Berkeley View of Cloud Computing. EECS Department, University of California, Berkeley.

Beloglazov, A., Abawajy, J., Buyya, R. (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems, Volume 28, Issue 5, Pages 755-768, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2011.04.017.

Beloglazov, A., Buyya, R. (2010) Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: IEEE International Symposium on Cluster Computing and the Grid, pp. 826-831. Melbourne, VIC, Australia.

Bieswanger, A., Hamann, H.F., and Wehle, H.D. (2014) Energy Efficient Data Center, it - Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik, 54.1: 17-23.

BITKOM (2009) Leitfaden Cloud Computing - Evolution in der Technik, Revolution im Business.

Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J.M., and Brandic I. (2012) Energy-efficient and SLA-aware management of IaaS clouds. In Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy '12). ACM, New York, NY, USA, , Article 25 , 10 pages. http://doi.acm.org/10.1145/2208828.2208853

Brandic, I., Anstett, T., Schumm, D., Leymann, F., Dustdar, S., Konrad, R. (2010) Compliant Cloud Computing (C3): Architecture and Language Support for User-driven Compliance Management in Clouds. In: Proceedings of the 3rd International Conference on Cloud Computing (IEEE Cloud 2010).

Catteddu, D., Hogben, G. (2010) Cloud Computing: benefits, risks and recommendations for information security. In: Communications in Computer and Information Science, p. 17. Springer Berlin Heidelberg.

Chen, F., Schneider, J. G., Yang, Y., Grundy, J., and He, Q. (2012) An energy consumption model and analysis tool for Cloud computing environments. In Green and Sustainable Software (GREENS), 2012 First International Workshop on (pp. 45-50). IEEE.

Choon Lee, Y., Zomaya, A. (2010) Energy efficient utilization of resources in cloud computing systems. In: The Journal of Supercomputing, pp. 1-13. Springer Netherlands.

Cloud Security Alliance (2009) Security Guidance for Critical Areas of Focus in Cloud Computing.

Dhiman, G., Marchetti, G., Rosing, T. (2009) vGreen: A System for Energy Efficient Computing in Virtualized Environments. In: Proceedings of the ISLPED'09. San Francisco, California, USA.

Feller, E., Rohr, C., Margery, D., Morin, C., (2012) Energy Management in IaaS Clouds: A Holistic Approach, Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on , vol., no., pp.204,212, doi: 10.1109/CLOUD.2012.50.

Fraunhofer-Institut für Umwelt-, Sicherheits- und Energietechnik UMSICHT (2008) Studie: Ökologischer Vergleich von PC und Thin Client Arbeitsgeräten.

Garg, S. K., Buyya, R. (2012) Green cloud computing and environmental sustainability. Harnessing Green IT: Principles and Practices, 315-340.

Haupt, F., Leymann, F., Nowak, A., Wagner, S. (2014) Lego4TOSCA: Composable Building Blocks for Cloud Applications, In: Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014).

HyperStratus (2009) Migrating Applications to the Cloud - An Amazon Web Services Case Study, HyperSTratus White Paper, http://hyperstratus.com.

Jaeger, P. T., Lin, J., Grimes, J. M., Simmons, S. N. (2009) Where is the cloud? Geography, economics, environment, and jurisdiction in cloud computing. In: First Monday, Volume 14, Number 5.

Jensen, M., Schwenk, J., Gruschka, N., Lo Iacono, L. (2009) On Technical Security Issues in Cloud Computing. In: Proceedings of the IEEE International Conference on Cloud Computing (CLOUD-II 2009), pp. 109-116. Bangalore, India.

Joint, A., E. Baker, and E. Eccles (2009) Hey, you, get off of that cloud? In: Computer Law & Security Review, Volume 25, Issue 3, pp. 270-274.

Khajeh-Hosseini, A., Greenwood, D., Sommerville, I. (2009) Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. http://arxiv.org/abs/1002.3492v1.

Khajeh‐Hosseini, A., Greenwood, D., Smith, J. W., and Sommerville, I. (2012) The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. Software: Practice and Experience, 42(4), 447-465. http://dx.doi.org/10.1002/spe.1072

Leymann, F. (2009) Cloud Computing: The Next Revolution in IT. In: Dieter Fritsch (ed.), Photogrammetric Week 2009. Stuttgart, Germany.

Li, B., Li, J., Huai, J., Wo, T., Li, Q., Zhong, L. (2009) EnaCloud: An Energy-saving Application Live Placement Approach for Cloud Computing Environments. In Proceedings of the IEEE International Conference on Cloud Computing, pp. 17-24. Bangalore, India.

Liu, L., Wang, H., Liu, X., Jin, X., Bo He, W., Wang, Q. B., Chen, Y. (2009) GreenCloud: A New Architecture for Green Data Center. In: Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session (ICAC-INDST '09), pp. 29-38. New York, NY, USA.

Motahari-Nezhad, H. R., Stephenson, B., and Singhal, S. (2009) Outsourcing business to cloud computing services: Opportunities and challenges. IEEE Internet Computing, 10.

Nowak, A,, Leymann, F., Schleicher, D., Schumm, D., Wagner, S. (2011) Green Business Process Patterns. In: Proceedings of the 18th Conference on Pattern Languages of Programs, PLoP 2011.

Nowak, A.; Karastoyanova, D.; Leymann, F.; Rapoport, A.; Schumm, D. (2012) Flexible Information Design for Business Process Visualizations. In: Proceedings of the 2012 IEEE International Conference on Service-Oriented Computing and Applications.

Nowak, A.; Leymann, F. (2013) Green Business Process Patterns – Part II. In: Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013), Kauai, USA.

Nowak, A.; Leymann, F. (2013) An Overview on Implicit Green Business Process Patterns, Technischer Bericht Nr. 2013/05.

Nowak, A., Binz, T., Leymann, F., Urbach, N. (2013) Determining Power Consumption of Business Processes and their Activities to Enable Green Business Process Reengineering. In: Proceedings of the 17th IEEE International EDOC Conference (EDOC 2013), Vancouver, Kanada, S.259-266.

Pedram, M. (2012) Energy-Efficient Datacenters, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.31, no.10, pp.1465,1484, doi: 10.1109/TCAD.2012.2212898.

Piazaa, J. (2014) White Paper Computing Migration Strategies, http://www.savvis.com/en-us/info_center/documents /hos-20090513-external-whitepaper-computingmigrationstrategies.pdf.

Schleicher, D., Anstett, T., Leymann, F., Mietzner, R. (2009) Maintaining Compliance in Customizable Process Models. In: Proceedings of the 17th International Conference on COOPERATIVE INFORMATION SYSTEMS (CoopIS 2009).

Schumm, D., Leymann, F., Streule, A. (2010) Process Views to Support Compliance Management in Business Processes. In: Proceedings of the 11th International Conference on Electronic Commerce and Web Technologies (ECWeb 2010).

Strauch, S., Andrikopoulos, V., Karastoyanova, D., Vukojevic, K. (2014) Migrating eScience Applications to the Cloud: Methodology and Evaluation. In: Terzo, Olivier; Mossucca, Lorenzo (Editors): Cloud Computing with E-science Applications. CRC Press/Taylor & Francis, 2014.

Wagner, S., Roller, D., Kopp, O., Unger, T., and Leymann, F. (2013) Performance Optimizations for Interacting Business Processes. In Cloud Engineering (IC2E), 2013 IEEE International Conference on (pp. 210-216). IEEE.

Waizenegger, T., Wieland, M., Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., and Wagner, S. (2013) Policy4TOSCA: A policy-aware cloud service provisioning approach to enable secure cloud computing. In On the Move to Meaningful Internet Systems: OTM 2013 Conferences (pp. 360-376). Springer Berlin Heidelberg.

Walker, E. (2009) The Real Cost of a CPU Hour. In: Computer, pp. 35-41.

Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., Leymann, F. (2009) Monitoring and Analyzing Influential Factors of Business Process Performance. In: Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009), pp. 141-150. Auckland, New Zealand.

Wetzstein, B., Strauch, S., Leymann, F. (2009) Measuring Performance Metrics of WS-BPEL Service Compositions. In: Proceedings of the Fifth International Conference on Networking and Services (ICNS 2009), pp. 49-56. Valencia, Spain.