Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 171

Analyse existierender visueller Notationen zur Modellierung von Anwendungstopologien und deren Integration mit Prozessnotationen

Lars-Alexander Albrecht, Rene Trefft,
Michael Zimmermann

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Uwe Breitenbücher
Beginn am:	2013-01-15
Beendet am:	2013-07-17
CR-Nummer:	H.1.2

Kurzfassung

Die Modellierung von Topologien, bestehend aus vielen einzelnen Komponenten und deren Beziehungen zueinander, sowie deren Management ist ein großes Problem im Bereich von komplexen Cloud-Anwendungen. Die Topologie und Orchestration Specification for Cloud Applications (TOSCA) hat sich diesem Problem angenommen und einen Standard zur Beschreibung von Cloud-Anwendungen geschaffen.

Mit Vino4TOSCA wurde eine wohldefinierte und auf etablierten Usability-Forschungen basierende visuelle Notation für TOSCA entworfen. Allerdings ermöglicht Vino4TOSCA bisher lediglich die Darstellung von Topologien. Aus diesem Grund wird in dieser Arbeit eine auf Vino4TOSCA aufbauende visuelle Notation erarbeitet, welche sowohl die Modellierung von Topologien als auch der dazugehörigen Managementplänen in einem Diagramm ermöglicht.

Inhaltsverzeichnis

Abkürzungsverzeichnis

BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CS	Committee Specification
DA	Deployment Artifact
EPK	Ereignisgesteuerte Prozesskette
ER-Diagramm	Entity-Relationship-Diagramm
FMC	Fundamental Modeling Concepts
FOPL	First-Order Predicate Logic
GWorkflowDL	Generic Workflow Description Language
HIPO-Diagramm	Hierarchy plus Input-Process-Output-Diagramm
IaaS	Infrastructure as a Service
IA	Implementation Artifact
IAAS	Institut für Architektur von Anwendungssystemen
IBM	International Business Machines Corporation
IT	Informationstechnik
NIST	National Institute of Standards and Technology
OMG	Object Management Group
PaaS	Platform as a Service
RPC	Remote Procedure Call
SaaS	Software as a Service
SCA	Service Component Architecture
SE	Software Engineering
SOA	Service Oriented Architecture

SPIKES	Structured Plans for Improving Knowledge Transfer in Engineering of Systems
TOSCA	Topology and Orchestration Specification for Cloud Applications
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Vino4TOSCA	Visual Notation for TOSCA
VM	Virtuelle Maschine
WAR	Web Archive
WD	Working Draft
XML	Extensible Markup Language

Abbildungsverzeichnis

1 Einleitung

Die Installation und Wartung von Anwendungen auf Servern ist für ein IT-Unternehmen in der Regel eine aufwändige Aufgabe. In den letzten Jahren hat sich ein Trend entwickelt, entsprechende Anwendungen in die Cloud auszulagern. Dadurch entfällt die kostenintensive Bereitstellung und Wartung von Hard- und Software, die für die Ausführung der Anwendung erforderlich ist. Stattdessen werden diese Tätigkeiten vom Cloud-Anbieter übernommen, der eine sichere und zuverlässige Infrastruktur bereitstellt. Die Abrechnung erfolgt nutzungsbasiert, Aktualisierungen der Infrastruktur erfolgen automatisch und Größenänderungen in jede Richtung stellen kein Problem dar.

Die Topology and Orchestration Specification for Cloud Applications (TOSCA) setzt bei diesem Trend an und definiert eine portable und interoperable Sprache zur Beschreibung einer Cloud-Anwendung durch ihre Topologie und deren Management mittels Plänen. Ein Plan ist ein Geschäftsprozess, deren Notation nicht durch TOSCA vorgegeben wird. Stattdessen sollen bereits existierende Standards, insbesondere die Business Process Model and Notation (BPMN) und Business Process Execution Language (BPEL), eingesetzt werden. Mittels TOSCA soll die Migration einer Cloud-Anwendung zu einem anderen Anbieter deutlich vereinfacht werden. Insbesondere jedoch kann das Management der Anwendung in jeder Umgebung automatisiert werden. [TOS13]

Mit der Visual Notation for TOSCA (Vino4TOSCA) wurde eine visuelle Notation entworfen, mit der die Topologie einer TOSCA-Anwendung grafisch dargestellt werden kann (siehe Abschnitt 4.1.1). Generell sind visuelle Notationen einfacher zu erlernen als textuelle Notationen. Sie ermöglichen eine schnelle und effektive Kommunikation[1] von Informationen, wohingegen eine textuelle Notation auf eine vollständige Wiedergabe von Informationen ausgerichtet ist.
Dem Entwurf von Vino4TOSCA ist eine umfangreiche Anforderungsanalyse vorausgegangen, in der u. a. Usability eine wichtige Rolle spielte. Dies stellt eine Besonderheit dar, da bei den meisten, weiteren visuellen Notationen lediglich die Semantik die Schlüsselanforderung im Entwurfsprozess darstellte. Die Pläne einer TOSCA-Anwendung können in einem Vino4TOSCA-Diagramm nicht dargestellt werden,

[1] Die Effektivität einer visuellen Notation resultiert aus der leistungsfähigen und hoch parallelisierten visuellen Informationsverarbeitung des menschlichen Gehirns.

obwohl das Management eine zentrale Rolle in einem TOSCA-Modell einnimmt. [BBK+12a]

In dieser Fachstudie wird eine visuelle Notation entworfen, mit der die Topologie und Pläne eines TOSCA-Modells zusammen (integriert) dargestellt werden können. Zunächst werden in Kapitel 2 Begriffe eingeführt, die in dieser Arbeit benötigt werden. Analog zu Vino4TOSCA wird eine Anforderungsanalyse durchgeführt, auf die in Kapitel 3 eingegangen wird. Im Anschluss werden in Kapitel 4 bereits existierende visuelle Notationen für Anwendungstopologien (u. a. Vino4TOSCA) und Geschäftsprozesse (u. a. BPMN) vorgestellt und analysiert. Auf Grundlage dieser Analyse und den aufgestellten Anforderungen wird die visuelle Notation entworfen, die in Kapitel 5 beschrieben wird. Abschließend wird in Kapitel 6 die Arbeit zusammenfasst und Anregungen für zukünftige Arbeiten gegeben, die thematisch mit dieser Arbeit in Zusammenhang stehen.

2 Grundlagen

In diesem Kapitel sollen Begrifflichkeiten erläutert werden, die für das Verständnis dieser Arbeit relevant sind. Cloud Computing bildet die Basis, sodass auf diesen Begriff zuerst eingegangen wird.

2.1 Cloud Computing

Momentan existiert keine allgemeingültige Definition für Cloud-Computing. Im wissenschaftlichen Bereich wird jedoch meist die Definition der Standardisierungsstelle NIST (National Institute of Standards and Technology), die 2009 veröffentlicht wurde, verwendet:

> Cloud Computing ist ein Modell, das es erlaubt bei Bedarf, jederzeit und überall bequem über ein Netzwerk auf einen geteilten Pool von konfigurierbaren Rechnerressourcen (z. B. Netze, Server, Speichersysteme, Anwendungen und Dienste) zuzugreifen, die schnell und mit minimalem Managementaufwand oder geringer Serviceprovider-Interaktion zur Verfügung gestellt werden können. [MG11]

Durch Cloud-Computing sollen also IT-Ressourcen effizient in und über Netzwerke zur Verfügung gestellt werden [MÏ2]. Die virtuelle Ort, an dem sich diese befinden, bezeichnet man dabei als Cloud[2] (deutsch „Wolke“).

IT-Ressourcen werden als Dienste zur Verfügung gestellt. Generell unterscheidet man zwischen folgenden Servicemodellen:

- Infrastructure as a Service (IaaS) stellt Datenspeicher, Netzwerkkapazität und Rechenleistung zur Verfügung. Auf virtuellen Recheninstanzen können z. B. Betriebssysteme mit Anwendungen installiert werden. [Bun]
- Platform as a Service (PaaS) stellt eine Ausführungsumgebung für Anwendungen bereit. Auf diese kann mittels standardisierter Schnittstellen zugegriffen und Anwendungen installiert werden. Ein Zugriff auf die Infrastruktur ist nicht möglich. [MG11]

[2] Der Begriff der Cloud kommt daher, dass für einen Nutzer der IT-Ressourcen die zugrunde liegende Infrastruktur verborgen bleibt.

- Software as a Service (SaaS) stellt eine Software zur Verfügung, auf die in der Regel über einen Webbrowser zugegriffen werden kann. [Bun]

Cloud-Dienste werden üblicherweise von Anbietern bereitgestellt, die sich auf dem Gebiet des Cloud-Computing spezialisiert haben, sodass eine hohe Sicherheit und Zuverlässigkeit gewährleistet ist.

2.2 Topology and Orchestration Specification for Cloud Applications (TOSCA)

Die Topology and Orchestration Specification for Cloud Applications (TOSCA) definiert eine Sprache, mit der Cloud-Anwendungen und deren Management portabel und interoperabel beschrieben werden können, d. h. unabhängig von einem konkreten Cloud-Anbieter oder einer Hosting-Technologie. Das Datenformat eines TOSCA-Modells ist XML. [BBK+12a]

In diesem Dokument kommt TOSCA in der Version CS01[3] (vom 2013-03-18) zum Einsatz. Die visuelle Notation, die im Rahmen dieser Arbeit entworfen wird, basiert auf dieser Version.

In einem TOSCA-Modell wird eine Cloud-Anwendung mittels ihrer Struktur beschrieben. Diese wird durch ein Topology Template repräsentiert, das sich aus Node Templates und Relationship Templates zusammensetzt. Ein Node Template stellt eine Komponente der Anwendung dar und ist durch ein Node Type typisiert. Zwei Node Templates können mittels einem Relationship Template, das analog ein Relationship Type referenziert, miteinander verbunden werden. [BBK+12a]

In Node Types und Relationship Types können in erster Linie Schnittstellen mit Management-Operationen definiert werden. In letzterem Konstrukt wird dabei wird Quell- und Zielschnittstellen (Source Interfaces und Target Interfaces) unterschieden. Ein Quellschnittstelle definiert Operationen, die an der Quelle der Relation ausgeführt werden, um die Verbindung zwischen den entsprechenden zwei Node Templates zu realisieren. Analog dazu wird eine Operation einer Zielschnittstelle am Ziel der Relation ausgeführt. Ein Node Type Implementation bzw. Relationship Type Implementation repräsentiert die Implementierung eines referenzierten Node Type bzw. Relationship Type. Hierzu definiert es Implementation Artifacts (IAs), welche die Schnittstellen realisieren. In einer Node Type Implementation können zusätzlich Deployment Artifacts (DAs) angegeben werden, die ein zugehöriges Node Template,

[3] TOSCA Spezifikation Version CS01: http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf

also eine Komponente der Anwendung repräsentieren. Die Definition von DAs ist auch direkt in einem Node Template möglich. [BBK+12a]

Denkbar wäre ein Node Template, das eine Cloud-Anwendung repräsentiert. Das zugehörige Deployment Artifact wäre dann die Distribution der Anwendung (z. B. ZIP-Datei). Das Deployment der Cloud-Anwendung wäre eine mögliche Aufgabe eines Implementation Artifacts (IAs, z. B. WAR-Datei). Ein „hostedOn“-Relationship Template könnte die Cloud-Anwendung mit einem Betriebssystem, das ein weiteres Node Template darstellt, verbinden.

Die Definition eines Artefakts erfolgt in einem Node Template, einer Node Type Implementation und Relationship Type Implementation durch eine Referenz auf ein Artifact Template. Ein Artifact Template spezifiziert ein Artefakt direkt oder durch Referenzen. Ein Referenz ist dabei eine URI, die auf eine Datei oder einen Ordner verweist. In letzterem Falle sind Patterns erlaubt, mit denen Dateien an der Artefakt-Referenz ausgeschlossen werden können. Die Notation, in der Patterns angegeben werden, ist durch TOSCA nicht spezifiziert und kann daher beliebig gewählt werden. Beispielsweise können reguläre Ausdrücke zum Einsatz kommen. Ein Artifact Template ist durch ein Artifact Type typisiert, welches die Menge der zulässigen Artefakte einschränkt. [BBK+12a]

Pläne (Plans) repräsentieren in einem TOSCA-Modell das Management der Cloud-Anwendung. Ein Plan ist ein Modell eines Geschäftsprozesses (Workflow), das Operationen, die durch IAs bereitgestellt sind, zu höherwertigen Management-Operationen kombiniert (orchestriert). Denkbar wäre ein „Build Plan“, der eine Cloud-Anwendung bereitstellt bzw. instanziiert. Analog zu einem Artifact Template kann ein Plan direkt oder mittels einer URI-Referenz definiert werden.

Eine konkrete Sprache für Pläne bzw. Prozesse wird von TOSCA nicht vorgegeben. Stattdessen sollen existierende Standards, insbesondere BPMN und BPEL, eingesetzt werden. [BBK+12a]

Ein Service Template spezifiziert einen Service und deren Management und besteht dazu aus einem Topology Template und Plänen. Nach einer erfolgreichen Ausführung eines Build Plans existiert eine Instanz eines Service Template bzw. deren Topology Template, d. h. ein konkreter Service. Definitions besteht aus Service Templates und den erwähnten Types, die in Service Templates referenziert werden. [BBK+12a]

Alle angesprochenen Konstrukte stellen XML-Elemente dar und bilden nach beschriebener Hierarchie ein gültiges TOSCA Definitions-Dokument. [BBK+12a]

Der Visual Editor for TOSCA (VALESCA) ist ein webbasiertes Modellierungswerkzeug für TOSCA, das u. a. die visuelle Notation Vino4TOSCA (Visual Notation for

TOSCA) unterstützt, mit der Topology Templates grafisch dargestellt werden können [BBK+12b]. Näheres zu dieser Notation in Abschnitt 4.1.1.

3 Anforderungsanalyse

In diesem Kapitel werden Anforderungen genannt und beschrieben, die bei der Definition von Vino4TOSCA 2 in Kapitel 5 berücksichtigt werden sollen. Die Aufstellung von Anforderungen bildet die Basis für die Erstellung einer Notation, die effektiv einsetzbar ist.

Wir kategorisieren die Anforderungen in vier Bereiche.
In Abschnitt 3.1 werden zunächst visuelle Anforderungen behandelt. In der ersten Version von Vino4TOSCA [BBK+12a] (siehe Abschnitt 4.1.1) bildete die Design Theorie „The Physics of Notations“ von Moody [Moo09] die Grundlage für die Entwicklung der visuellen Notation. Moody setzt sich in diesem Dokument mit der physikalischen Wahrnehmung von Notationen im Rahmen der menschlichen Fähigkeiten auseinander. Er beschreibt Prinzipien, welche aus der Theorie sowie empirischen Untersuchungen stammen. Auch bei Vino4TOSCA 2 wird u. a. dieses Dokument für die Bestimmung der visuellen Anforderungen herangezogen.
Im Anschluss folgen in Abschnitt 3.2 Anforderungen, die sicherstellen sollen, dass sich mit der visuellen Notation korrekt TOSCA-Topologien darstellen lassen.
In 3.3 werden Usability-[4] und User Experience[5]-Anforderungen beschrieben. Die Notation soll effektiv und effizient genutzt werden können. Das Nutzungserlebnis soll dabei auch berücksichtigt werden.
Abschließend werden in Abschnitt 3.4 Anforderungen dargelegt, welche die Integration von Prozessnotationen betreffen.

Jede Anforderung, die im Folgenden genannt wird, erhält die Bezeichnung Ax, wobei x für eine Zahl steht, die eindeutig eine bestimmte Anforderung referenziert. In den weiteren Kapiteln kann so einfach auf Anforderungen verwiesen werden.

[4] Benutzbarkeit
[5] Benutzerfreundlichkeit

3.1 Visuelle Anforderungen

Visuelle Anforderungen sollen die kognitive Effektivität[6] steigern, welche die wichtigste Variable für die Bewertung und den Vergleich einer visuellen Notation darstellt [Moo09].

Die semiotische Klarheit (A1) stellt die erste visuelle Anforderung dar. Sie besagt, dass zwischen den semantischen Konstrukten einer visuellen Notation und ihrer grafischen Darstellung eine 1:1 Beziehung existieren muss. Jedes semantische Konstrukt der Notation darf nicht durch mehr als ein grafisches Element ausgedrückt werden können (Symbol Redundanz). Weiterhin dürfen mehrere Konstrukte nicht durch das gleiche grafische Symbol repräsentiert werden (Symbol Überladung). Auch darf es kein grafisches Symbol geben, das keinem semantischen Konstrukt angehört (Symbol Überschuss). Ebenso darf es kein Konstrukt geben, dass keinem grafischen Symbol zugewiesen wurde (Symbol Defizit). [Moo09]
Abbildung 3.1 veranschaulicht die genannten Anomalien der semiotischen Klarheit.

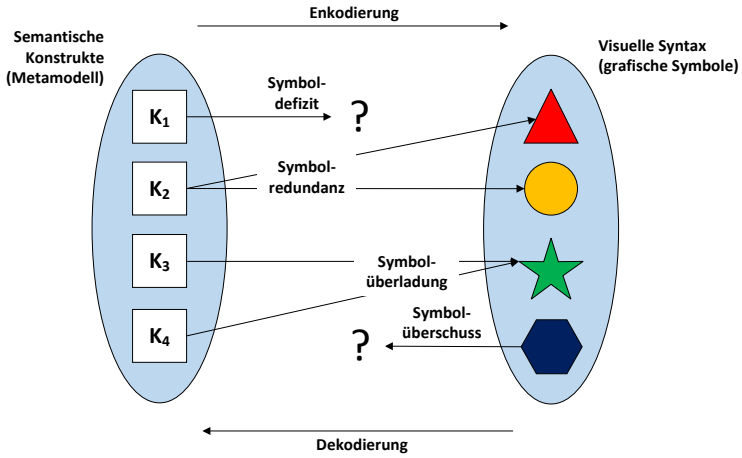

Abbildung 3.1: Anomalien der Semiotischen Klarheit nach [Moo09].

Die differenzierte Wahrnehmung (A2) stellt eine weitere visuelle Anforderung dar. Jedes Element der Notation muss eindeutig unterschieden werden können, damit es zu keinen Missverständnissen kommt. Der Grad an Unterscheidbarkeit ist dabei durch die visuelle Distanz[7] zwischen den grafischen Elementen gegeben. Beispielsweise gibt es in vielen visuellen Notationen zentrale Elemente, die sich verhältnismäßig nur wenig unterscheiden. Insbesondere für Personen, die nicht mit der Domäne der

[6] Die kognitive Effektivität ist nach [LS87] die Geschwindigkeit, Leichtigkeit und Genauigkeit mit der Informationen vom menschlichen Verstand verarbeitet werden können.
[7] Die visuelle Distanz wird bestimmt durch die Anzahl der visuellen Variablen (z. B. Form, Farbe, Schriftgröße oder Schriftstil), in denen sich grafische Elemente unterscheiden und dem Umfang der Unterschiede.

visuellen Notation vertraut sind, ist der kognitive Aufwand zum Verständnis eines Modells hoch, wenn sich grafische Elemente nicht deutlich unterscheiden. [Moo09]

Gemäß der semantischen Transparenz (A3) soll am Aussehen eines grafischen Elements seine Bedeutung erkennbar sein. In [Moo09] wird verdeutlicht, dass dieses Prinzip nicht binär[8] ist, sondern ein Kontinuum darstellt: Man spricht von dem Grad zwischen dem Aussehen und der Bedeutung eines grafischen Elements. Abbildung 3.2 veranschaulicht diesen Zusammenhang.

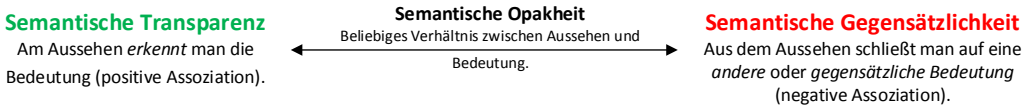

Abbildung 3.2: Semantische Transparenz ist ein Kontinuum nach [Moo09].

Icons sind grafische Symbole, die in der Regel eine starke Bindung zwischen Aussehen und Bedeutung aufweisen und sich daher gut für die Umsetzung der semantischen Transparenz eignen. Zusätzlich fällt das Erlernen und Erinnern an Icons meist leichter als bei Formen. Auch wirken sie visuell ansprechender.

Die semantische Transparenz lässt sich ebenso auf den Zusammenhang von Elementen anwenden. Die Bedeutung eines Elements soll sich aus seiner Lage und Beziehung mit weiteren Elementen ableiten lassen können. Man spricht hierbei von der semantisch transparenten Relation (A4), welche wir explizit als separate Anforderung definieren. [Moo09]

Es sollten Mechanismen vorgesehen werden, um mit Komplexität umzugehen bzw. diese bei Bedarf zu verringern. Man spricht hierbei vom Komplexitätsmanagement (A5). Systeme sollten auf verschiedenen Hierarchieebenen betrachtet (Abstraktion) und in kleinere Teile (Subsysteme) zerlegt werden können (Modularisierung). Die Bildung von sinnvollen Hierarchieebenen ermöglicht „top down understanding“. In [NC99] wurde gezeigt, dass diese Vorgehensweise zum Verständnis von Diagrammen aus dem Bereich des Software Engineerings beiträgt. Weiterhin kann man durch Hierarchiebildung den Interessen unterschiedlicher Leser gerecht werden. Die niedrigste Hierarchieebene (Detaillierungsgrad) sollte dabei zunächst einen Überblick über das gesamte System geben. Die weiteren Ebenen können dann einzelne Komponenten im Detail veranschaulichen. [Moo09]

Eine weitere Anforderung betrifft die Integration von Diagrammen. In der Regel wird ein System nicht nur durch ein einzelnes Diagramm repräsentiert, sondern durch eine

[8] Binär bedeutet in diesem Kontext, dass die Anforderung lediglich erfüllt bzw. nicht erfüllt sein kann.

Vielzahl von Diagrammen unterschiedlicher Typen. Für den Leser stellt das Zusammensetzen der Informationen aus den verschiedenen Diagrammen einen zusätzlichen kognitiven Aufwand dar. Der kognitiven Integration entsprechend sollte eine visuelle Notation Mechanismen vorsehen, welche diesen Aufwand senken (A6). Man unterscheidet dabei zwischen der konzeptuellen Integration, die den Leser unterstützen soll, Informationen aus verschiedenen Diagrammen zu einer kohärenten mentalen Gesamtdarstellung des System zusammenzusetzen und der perzeptuellen Integration, welche die Navigation und Übergänge zwischen den Diagrammen vereinfachen soll. Abbildung 3.3 veranschaulicht diese beiden Begriffe. [Moo09]

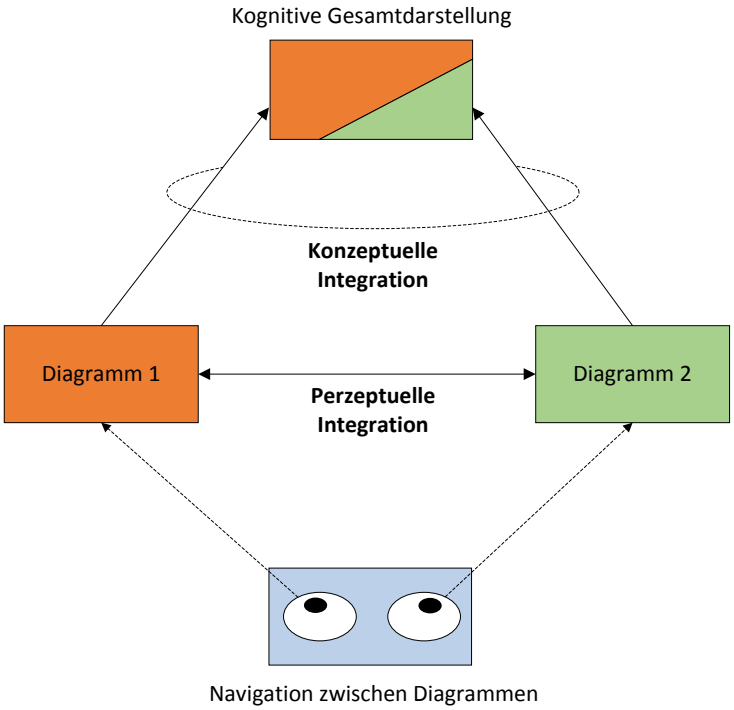

Abbildung 3.3: Kognitive Integration nach [Moo09].

Weiterhin sollte eine visuelle Notation möglichst viele visuelle Variablen (z. B. Form und Farbe) einsetzen. Je mehr Variablen im visuellen Vokabular einer Notation enthalten sind, desto höher ist ihre visuelle Ausdrucksfähigkeit (A7). Klassische Formen wirken dabei am wenigsten kognitiv effektiv [LS87]. Im Gegensatz dazu gehört Farbe zu den kognitiv effektivsten visuellen Variablen, da das menschliche Auge sehr empfindlich auf Farbvariationen reagiert und schnell zwischen Farben unterscheiden kann. Farbe sollte jedoch keinesfalls als einzige Variable zur Unterscheidung von grafischen Elementen eingesetzt werden, da ansonsten Personen mit einer Farbsinnstörung oder Farbfehlsichtigkeit u. U. die Elemente nicht mehr unterscheiden können. Stattdessen sollte Farbe lediglich zur redundanten Kodierung verwendet werden. [Moo09]

Text und Grafik sollten zusammen eingesetzt werden, um Informationen zu übermitteln (kodieren), was als „Dual Coding“ (A8) bezeichnet wird. Grundsätzlich werden textuelle Informationen im verbalen System verarbeitet bzw. gespeichert, bildliche Informationen hingegen im nicht-verbalen bzw. imaginalen System. Falls eine Information textuell als auch grafisch präsentiert wird, werden folglich beide Systeme aktiviert. Es werden Verbindungen zwischen den Systemen aufgebaut, die zu einer deutlichen Steigerung der Merkfähigkeit führen. Anwendung findet dieses Prinzip bspw. bei Annotationen, welche grafische Elemente durch Text ergänzen und in Hybrid-Symbolen, die Grafik und Text in einem grafischen Element vereinigen. [Moo09]

Weiterhin sollte der grafischen Ökonomie entsprechend darauf geachtet werden, dass die Menge der Elemente im visuellen Vokabular noch kognitiv verarbeitet werden kann. Die grafische Komplexität der Notation sollte somit nicht zu hoch sein (A9). Jedes weitere Element im visuellen Vokabular senkt zudem die kognitive Effektivität, was nicht wünschenswert ist. Personen, die mit der Domäne der visuellen Notation nicht vertraut sind, sind insbesondere von der graphischen Komplexität betroffen. Dies haben empirische Studien ergeben ([NC99]), die mit SE-Diagrammen durchgeführt wurden.

Der Symbol-Defizit, den wir bereits im Rahmen der semiotischen Klarheit angesprochen haben, kann auch gezielt dazu eingesetzt werden, die graphische Komplexität zu senken. Die semantischen Konstrukte, die vom Symbol-Defizit betroffen sind (semantische Konstrukte, die kein grafischen Symbol besitzen), müssen in diesem Fall durch Text repräsentiert werden. Dieser Ansatz findet in vielen visuellen Notationen Anwendung (zumindest in einem gewissen Umfang) [Moo09, S. 15].

Auch sollte eine Notation verschiedene visuelle Dialekte anbieten, mit denen Informationen dargestellt werden können. Dadurch können Modelle erzeugt werden, die für verschiedene Aufgaben sowie Leser mit unterschiedlicher Erfahrung passend bzw. optimiert sind. Diese Forderung entspricht der Theorie des „Cognitive Fit“ (A10). [Moo09] Abbildung 3.4 veranschaulicht, durch welche Parameter der „Cognitive Fit“ beeinflusst wird. Durch „Cognitive Fit“ kann insgesamt eine schnellere Problemlösung und eine höhere kognitive Effizienz erreicht werden. Neben dem visuellen Dialekt, der die Basisnotation darstellt, können weitere visuelle Dialekte (Profile) für verschiedene Einsatzbereiche, Aufgaben oder Zielgruppen die Basisnotation erweitern bzw. anpassen. Beispielsweise könnte es ein Whiteboard-Profil geben, das Icons durch Formen ersetzt, die für das händische Skizzieren auf einem Whiteboard gedacht sind. Solche Formen könnten z. B. mit wenigen, schwarzen Linien auskommen und keine ausgefüllten Bereiche besitzen. In den meisten SE-Notationen kommen Profile (leider)

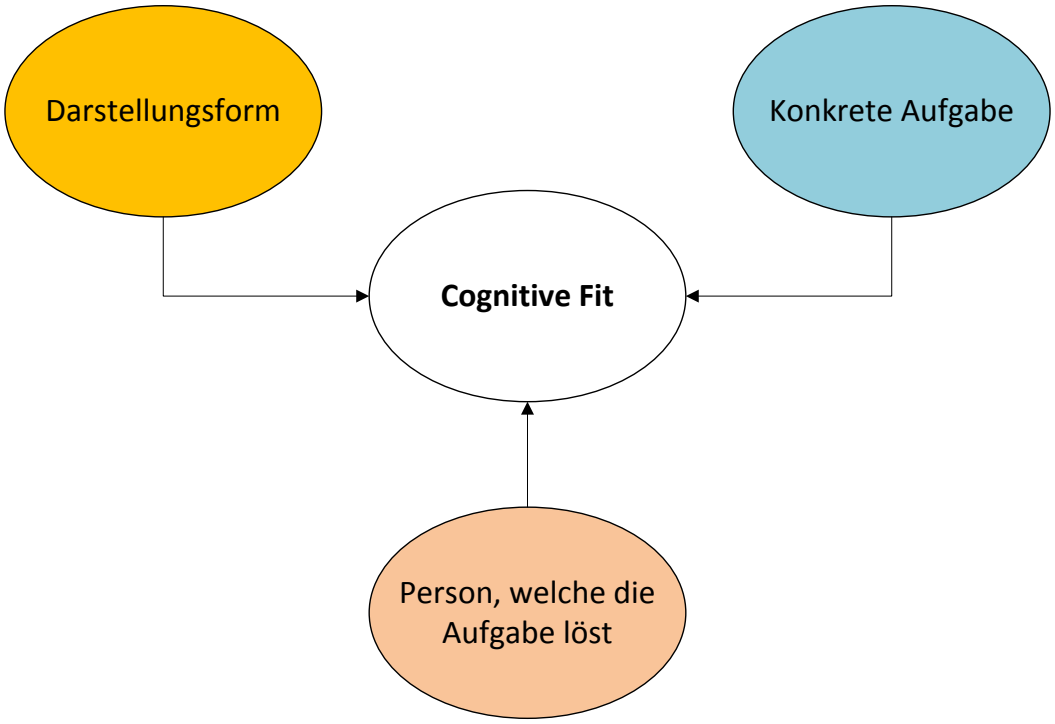

Abbildung 3.4: Einflussfaktoren des „Cognitive Fit“ nach [Moo09].

nicht zum Einsatz. Stattdessen enthält bereits die Basisnotation einfache grafische Elemente, um dem Bedürfnis nach Skizzieren nachzukommen.

Gemäß der Kommunikationstheorie sollten Informationen, die der Modellierer in einem Modell der visuellen Notation hinterlegt, möglichst unverändert zu einem Leser des Modells übertragen werden (A11). Anders ausgedrückt sollten Modelle der visuellen Notation die jeweiligen Informationen eindeutig repräsentieren, sodass Missverständnisse vermieden werden.

Werden Änderungen an einem Modell der visuellen Notation vorgenommen, so sollten sich diese möglichst wenig die Gesamtstruktur auswirken (A12) [PQ06]. Das Hinzufügen eines Knotens bspw. sollte nicht zu grundsätzlichen Änderungen am Modell führen bzw. diese erfordern. Der Aufwand zum (erneuten) Verständnis eines Modells nach einer Änderung soll durch diese Anforderung möglichst gering gehalten werden.

Weiterhin sollte die visuelle Notation von Personen, die mit der Domäne vertraut sind, möglichst schnell erlernt werden können. Alle anderen Personen sollten ebenfalls in einer angemessener Zeit ein Modell der Notation verstanden haben (A13). [PQ06]

Die visuelle Notation sollte Features vorsehen, die einem Leser bei der Navigation durch ein Modell der Notation unterstützen. Dies wird als minimale Desorientierung

(A14) bezeichnet. Beispielsweise könnten hierfür Orientierungspunkte zum Einsatz kommen oder Vorschriften definiert werden, die das Layout bzw. die Struktur betreffen. [PQ06]

Visuelle Metaphern, die eine Notation vorsieht, sollten effektiv und ausdrucksstark sein (A15), sodass sie sinnvoll eingesetzt werden können [PQ06]. Beispiele für visuelle Metaphern bilden Icons oder Farben, die eine bestimmte Information übermitteln sollen.

3.2 TOSCA-spezifische Anforderungen

Die Anforderungen, die im Folgenden dargelegt werden, sollen sicherstellen, dass die semantische Konstrukte der visuellen Notation auf TOSCA zugeschnitten sind.

Node Templates und Relationship Templates eines Topology Templates müssen mit der visuellen Notation dargestellt werden können. Auch sollen Node Types und Relationship Types, die von Node Templates bzw. Relationship Templates referenziert werden, repräsentiert werden können. Die genannten Elemente müssen somit in ein Modell der visuellen Notation überführt werden können. Wir fassen diese Anforderungen unter der Vollständigkeit zusammen (A16).

Die genannten Typ-Elemente sollen dargestellt werden können, da in diesen, die Schnittstellen mit Management-Operationen definiert sind. Die Darstellung von Operationen verdeutlicht, welche Managementrolle ein Node Type einnimmt. Insbesondere jedoch können dadurch Beziehungen zwischen einem Node Type und einem Plan dargestellt werden (siehe A26).

Entsprechend der semantischen Korrektheit (A17) sollte eine gültiges Modell der visuellen Notation ein gültiges Topology Template mit zugehörigen Node Types und Relationship Types repräsentieren [BBK+12a].

Auch sollte die visuelle Notation erweiterbar (A18) sein. Elemente sollten mit Informationen wie z. B. Erläuterungen oder zusätzliche Eigenschaften versehen werden können. [BBK+12a]

TOSCA-Topologien sind in den meisten Fällen umfangreich. Daher sollte die visuelle Notation Mechanismen vorsehen, mit denen Topologien in verschiedenen Detaillierungsgraden dargestellt werden können (A19; siehe auch A5). [BBK+12a]

3.3 Usability- und User Experience-Anforderungen

Die visuelle Notation sollte für ihre Aufgaben optimiert sein (A20). Die Modellierung von Topology Templates, Node Types, Relationship Types, Management-Plänen und den Beziehungen zwischen Node Types bzw. Relationship Types und Plänen (siehe A26) müssen folglich ohne Probleme möglich sein. [BBK+12a]

Ein Modell der visuellen Notation sollte möglichst ohne zusätzliche Informationen, d. h. jene, die nicht aus des Modell stammen, verstanden werden können. Die grafischen Elemente der Notation sollten folglich selbsterklärend sein (A21). Am Aussehen eines Elements sollte auf dessen Bedeutung geschlossen werden können, sofern man mit der Domäne der entsprechenden Notation vertraut ist. [BBK+12a]

Die grafischen Elemente der Notation sollten schnell und einfach zu zeichnen sein (A22), sodass die Notation auch bspw. für ein Whiteboard oder für eine Tafeln geeignet ist. Es bietet sich dabei an, gesonderte Elemente für das Skizzieren zu erstellen, die in einem Profil bereitgestellt werden können (siehe A10). [BBK+12a]

Weiterhin sollten die Elemente visuell ansprechend wirken bzw. den Vorlieben des Menschen entsprechen (A23) [BBK+12a]. Ästhetisch ansprechende Dinge führen zu positiver Emotion, was die Fähigkeit verbessert, Probleme zu lösen [Nor02]. Ein visuell ansprechendes Modell wird von einem Leser in einer kürzeren Zeit verstanden als ein Modell mit selbigem Inhalt, das jedoch weniger ansprechend wirkt. In einer Studie von Bar und Neta [BN06] wurde bspw. gezeigt, dass Personen abgerundete Formen bevorzugen. Auch sind jene Formen leichter zu zeichnen.

3.4 Anforderungen für die Integration visueller Geschäftsprozessnotationen

Die zu entwerfende visuelle Notation soll Prozessnotationen integrieren. Für den Integrationsaspekt werden gesonderte Anforderungen benötigt, auf die im Folgenden eingegangen wird.

Bestehende visuelle Notationen für Geschäftsprozesse bzw. Pläne dürfen für die Integration bzw. gemeinsamen Darstellung mit einem Topology Template (mit zugehörigen Types) nicht verändert werden (A24). Prozessnotationen erfüllen in der Regel jedoch nicht alle Anforderungen, die in den Abschnitten 3.1 und 3.3 dargelegt sind. Daher erlauben wir explizit, dass Prozessnotationen die Anforderungen aus den genannten Bereichen nicht erfüllen müssen. Die eindeutige Unterscheidbarkeit zwischen Topologie (mit Types) und Plänen muss jedoch gegeben sein (A25).

Existiert eine Beziehung zwischen einer Management-Operation, die zu einem Node Type bzw. einem Relationship Type gehört und einer Aktivität eines Plans (ruft die Operation auf), so sollte dies grafisch verdeutlicht werden können (A26).

Entsprechend dem Komplexitätsmanagement, welches wir bereits in 3.1 angesprochen haben, sollten auch für visuelle Prozessnotationen Möglichkeiten zur Abstraktion vorgesehen werden (A27). Einzelne Aktivitäten eines Prozesses, die als nicht relevant für ein Modell bzw. seine beabsichtige Aussage eingestuft werden, sollten ausgeblendet werden können. Auch sollten Prozesse vollständig „zusammengeklappt“ werden können. Diese Anforderung steht im Widerspruch zu A23, wo gefordert wird, dass visuelle Prozessnotationen nicht verändert werden dürfen. Generell erlauben wir Mechanismen zur Abstraktion, sofern entsprechende Möglichkeiten nicht bereits Teil der visuellen Notation des Plans bzw. Prozesses sind.

Die letzte Anforderung dieses Bereichs betrifft die semantische Korrektheit (A28): Der Modellierer entscheidet, ob ein Plan, der zu einem Topology Template gehört[9], in einem Modell der visuellen Notation repräsentiert wird. Falls er sich dafür entscheidet, so darf die Semantik des Plans nicht verändert werden. Beispielsweise dürfen keine Aktivitäten entfernt oder der Ablauf verändert werden. Wird Abstraktion angewendet, so muss dies eindeutig ersichtlich sein.

[9] Die Zusammengehörigkeit ist gegeben, falls das Topology Template und der Plan in selbigen Service Template definiert sind.

4 Analyse existierender visueller Notationen

Als Grundlage für den Entwurf der neuen visuellen Notation sollen in diesem Kapitel existierende visuelle Notationen für Anwendungstopologien und Geschäftsprozesse vorgestellt und analysiert werden. Es soll dabei lediglich auf wichtige und insbesondere standardisierte Notationen aus den genannten Bereichen eingegangen werden. Jede Notation wird durch eine oder mehrere Abbildungen veranschaulicht, die konkrete Beispiele oder visuelle Elemente der Notation (verallgemeinert) darstellen.

In der Analyse (siehe Abschnitt 4.3) wird unter anderem bestimmt, inwieweit sich die Notationen im Bereich der Anwendungstopologien für die Darstellung von TOSCA-Topologien und die Integration von Geschäftsprozessen eignen. Konzeptionelle bzw. Entwurfsentscheidungen, die als passend und brauchbar eingestuft werden, sollen in den Entwurf der neuen Notation einfließen (ggf. mit Anpassungen, sodass die aufgestellten Anforderungen erfüllt sind).

Die visuellen Notationen für Geschäftsprozesse werden im Hinblick auf die Integration mit TOSCA-Topologien untersucht. Eignung und Brauchbarkeit sollen dabei betrachtet werden. Es sollen schließlich Geschäftsprozessnotationen bestimmt werden, die zur Integration mit einer TOSCA-Topologie akzeptiert werden. Entsprechend Anforderung A24 müssen Prozessnotationen die aufgestellten Anforderungen aus den Abschnitten 3.1, 3.2 und 3.3 nicht erfüllen. Aus diesem Grund findet eine entsprechende Überprüfung nicht statt. Die Anforderungen für die Integration von Prozessnotationen (siehe Abschnitt 3.4) werden im Rahmen des Entwurfs der neuen Notation umgesetzt.

4.1 Anwendungstopologien

4.1.1 Vino4TOSCA

TOSCA spezifiziert keine grafischen Elemente für die semantischen Konstrukte der Sprache [TOS13]. Eine entsprechende visuelle Notation wäre jedoch wünschenswert, da damit eine TOSCA-Topologie auf einem einfachen und schnellen Wege repräsentiert bzw. kommuniziert werden kann.

Aus dieser Motivation heraus ist Vino4TOSCA (Visual Notation for TOSCA) entstanden, mit der TOSCA Topology Templates, die aus Node Templates, Relationship Templates und Group Templates bestehen, modelliert werden können. Zu letzterem Element sollte hierbei erwähnt werden, dass dieses in der aktuellen TOSCA Spezifikation CS01 vom 2013-03-18 [TOS13] nicht mehr vorgesehen ist.

Die Basis für den Entwurf der visuellen Notation bildete eine umfangreiche Anforderungsanalyse, bei der neben den üblichen Aspekten wie semantische Korrektheit und Vollständigkeit insbesondere auch die menschliche Wahrnehmung, Usability, Ergonomie und User Experience miteinbezogen wurde. Für visuelle Anforderungen wurde dabei die Design Theorie „The Physics of Notations“ von Moody [Moo09] herangezogen, die auf der Theorie sowie empirischen Studien beruht. [BBK+12a] Viele bekannte visuelle Notationen wie z. B. UML wurden im Gegensatz dazu nicht auf der Grundlage von visuellen Prinzipien entwickelt. Ein expliziter Entwurfsprozess für die visuelle Syntax wurde nicht vorgesehen. Stattdessen wurde der Fokus auf Semantik gelegt. Diese Vorgehensweise führt in der Regel zu einer Notation, die sich durch Defizite in der Usability auszeichnet. Auch Christopher Alexander [Ale64] hat dieses Problem existierender visueller Notation erkannt. Er spricht von einer „unself-conscious design culture“ (auf Deutsch etwa „freie Designkultur“), die auf Instinkt, Imitation und Tradition basiert und sich nicht nach expliziten Designprinzipien richtet. [BBK+12a]

Vino4TOSCA wurde am Institut für Architektur von Anwendungssystemen (IAAS) der Universität Stuttgart entwickelt und im September 2012 veröffentlicht. [BBK+12a]

Im Folgenden soll nun die eigentliche Notation anhand der verschiedenen Shapes (Elemente) erläutert werden.

Ein Node Template Shape (siehe Abbildung 4.1) ist ein abgerundetes Rechteck (durchgezogene Linie), dass ein Node Template repräsentiert. Das zugehörige Node Template kann mittels einem (i) Icon, der (ii) ID bzw. dem (iii) Namen des Node Templates oder über die (iv) ID bzw. dem (v) Namen des zugehörigen Node Type (in Klammern gesetzt) definiert werden, wobei mindestens eine dieser fünf Varianten verwendet werden muss. Eine ID muss grundsätzlich unterstrichen werden. Weiterhin gibt es ein optionales Feld, in dem zusätzliche Informationen hinterlegt werden können (Text oder Bilder). Dieses ist ebenfalls abgerundet und wird hinter dem Haupt-Shape positioniert, sodass die oberen Ecken nicht sichtbar sind. Das Haupt-Shape darf ein Hintergrundbild enthalten, wobei dieses nicht den Icon-Bereich oder den Text überdecken darf. [BBK+12a]

Ein Relationship Template Shape (siehe Abbildung 4.2) repräsentiert ein Relationship Template. Es handelt sich um eine Linie, die an ihren Enden beliebige (kleine) Shapes

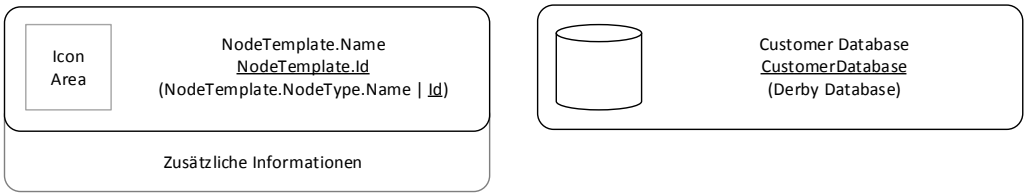

Abbildung 4.1: Node Template Shape mit Beispiel nach [BBK+12a].

besitzen darf, z. B. einen Pfeil. Der Stil der Linie kann frei gewählt werden, wobei sie nicht gestrichelt wie bei einem Visual Group Shape sein darf. Analog zu einem Node Template Shape kann das zugehörige Relationship Template definiert werden. Ein Icon steht über der Linie, falls die Linie horizontal ist, andernfalls (vertikal bzw. diagonal) auf einer beliebigen Seite. Ein Relationship Template Shape verbindet grundsätzlich zwei relationale Elemente. Ein relationales Element ist dabei ein Node Template Shape, Collapsed Group Template Shape oder Collapsed Visual Group Shape, wobei auf die letzteren beiden noch näher eingegangen wird. Ein optionales Feld für zusätzliche Informationen wird unter die Linie gesetzt und berührt die Linie, falls sie horizontal ist. Andernfalls kann sich das Feld an einer beliebigen Seite befinden. [BBK+12a]

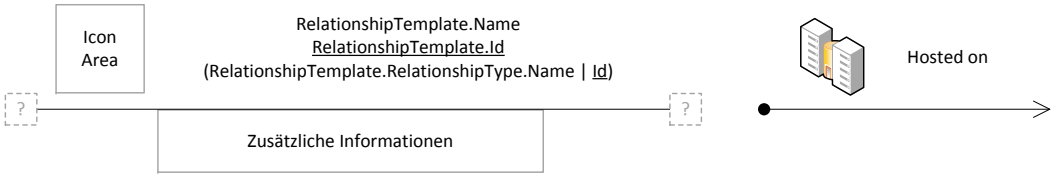

Abbildung 4.2: Relationship Template Shape mit Beispiel nach [BBK+12a].

Ein Group Template kann durch zwei verschiedene Shapes (siehe Abbildung 4.3, links) dargestellt werden. Das Expanded Group Template Shape enthält die Elemente des Group Template. Die Form dieses Shapes kann frei gewählt werden, wobei der Linienstil durchgezogen sein muss. Zur Beschreibung kann ein Icon, eine ID bzw. ein Name definiert werden, wobei mindestens eine dieser drei Möglichkeiten verwendet werden muss. Das Collapsed Group Template Shape dagegen abstrahiert die Elemente des Group Template. Es ist ein Oval, dessen Linie durchgezogen sein muss. Ein kleines Quadrat mit einem Plus-Zeichen symbolisiert die Abstraktion. Auch hier kann ein Icon, eine ID bzw. ein Name zur Beschreibung verwendet werden. [BBK+12a]

Die Visual Group Shapes (siehe Abbildung 4.3, rechts) dienen zum visuellen Gruppieren bzw. Abstrahieren von Elementen, also ohne das Topology Template zu verändern. Die Linien der Shapes sind gestrichelt, im Übrigen entsprechen sie den Shapes des

Group Templates. Visual Group Shapes können auch zur Integration von anderen Diagrammen verwendet werden. [BBK+12a]

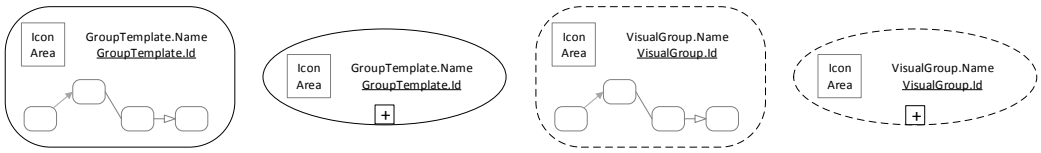

Abbildung 4.3: Expanded / Collapsed Group Template Shapes (links) und Visual Group Shapes (rechts) nach [BBK+12a].

Mit den Visual Relationship Group Shapes (siehe Abbildung 4.4) können Relationship Template Shapes visuell gruppiert bzw. abstrahiert werden. Die Expanded-Variante besteht aus zwei gestrichelten Linien, die relationale Elemente verbinden. Zwischen den Linien muss mindestens ein Relationship Template Shape stehen. Zur Beschreibung kann ein Icon, eine ID bzw. ein Name verwendet werden, die über der oberen Linie stehen müssen. Eine dieser drei Möglichkeiten muss mindestens verwendet werden. Die Collapsed-Variante dagegen abstrahiert Relationship Template Shapes. Es handelt sich um eine gestrichelte Linie zwischen relationalen Elementen, die in der Mitte ein kleines Quadrat mit einem Plus-Zeichen enthält, dass die Abstraktion symbolisiert. Im Übrigen entspricht sie der Expanded-Variante. [BBK+12a]

Grundsätzlich darf bei keinen Group Shapes ein Hintergrundbild gesetzt werden. [BBK+12a]

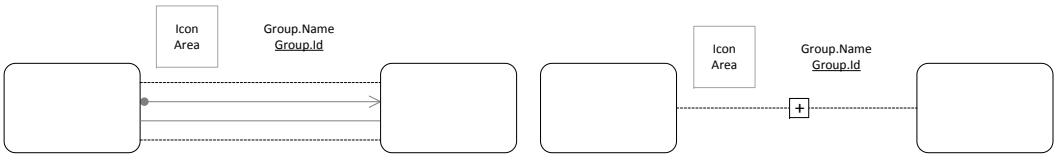

Abbildung 4.4: Expanded / Collapsed Visual Relationship Group Shapes nach [BBK+12a].

Weiterhin können die zulässigen Instanzen von Node Templates sowie Group Templates visualisiert werden (siehe Abbildung 4.5). Dazu wird eine Linie teilweise um das entsprechende Shape gezeichnet und der min-Wert auf die linke Seite, der max-Wert auf die rechte Seite über dem Shape geschrieben. [BBK+12a]

Vino4TOSCA ermöglicht auch die Definition von Profilen. Dabei handelt es sich um domänenspezifische Erweiterungen bzw. Anpassungen der Notation für bestimmte Bedürfnisse oder Fähigkeiten von Benutzern. Ein Aufgabe kann so intuitiv und effektiv gelöst werden. Beispielsweise könnte es ein Whiteboard-Profil geben, welches

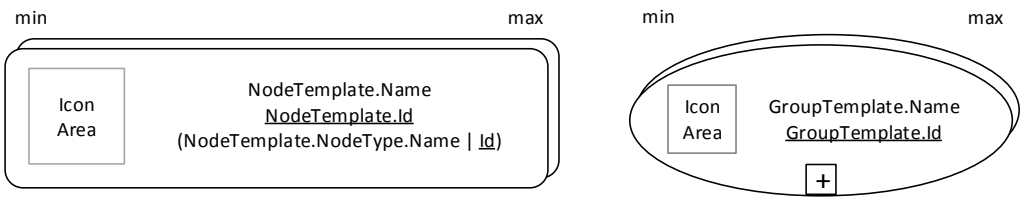

Abbildung 4.5: Zulässige Instanzen von Node Templates (links) und Relationship Templates (rechts) nach [BBK+12a].

definiert, dass alle Linien schwarz sein müssen und keine Icons verwendet werden dürfen. [BBK+12a]

Die Darstellung einer Form, ihre Orientierung und ihr Linienstil darf grundsätzlich nicht verändert werden. Alle weiteren visuellen Variablen sind frei wählbar, sofern sie nicht durch ein Profil fest definiert werden. Ein Profil darf lediglich die visuellen Variablen Farbe, Textur und Größe definieren. [BBK+12a]

Abbildung 4.6 zeigt ein Vino4TOSCA-Diagramm eines Web Shops, der auf einem Apache Tomcat deployed ist und eine Apache Derby Datenbank verwendet. Tomcat läuft auf Ubuntu, dass sich auf einer Amazon EC2 Instanz befindet.

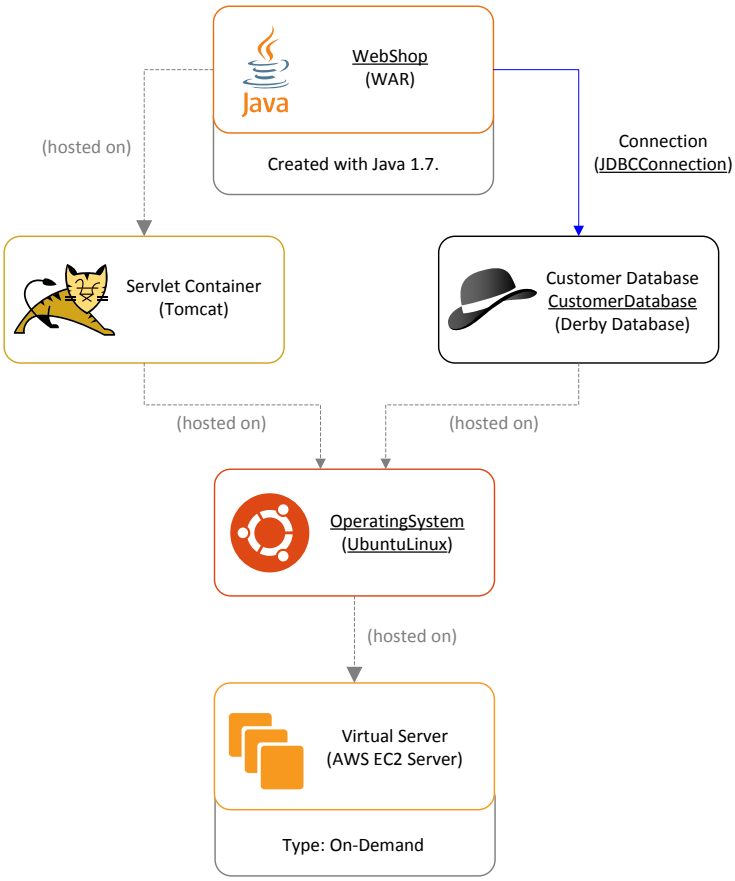

Abbildung 4.6: Vino4TOSCA-Diagramm eines Web Shops nach [BBK+12b].

Vino4TOSCA basiert auf der TOSCA Spezifikation WD07 vom Juli 2012 [BBK+12a]. Wie bereits angesprochen, gibt es in der momentanen Version CS01 keine Group Templates mehr, sodass entsprechende Formen für Group Templates in der zu entwerfenden Notation nicht mehr benötigt werden. Node Types bzw. Relationship Types wurden in WD07 in einem Service Template definiert, das zugleich das Wurzelelement bildete. Nun befinden sich die genannten Elemente im Wurzelelement Definitions. Service Template ist ein Unterelement von Definitions, dass nur noch aus einem Topology Template und Plänen besteht. [TOS13]

4.1.2 Fundamental Modeling Concepts

Die Fundamental Modeling Concepts (FMC) sind eine, zur Modellierung und Darstellung von komplexen Softwaresystemen, konzipierte Methodik. Die Entwicklung begann in den 70er Jahren unter der Leitung von Prof.-Dr. Siegfried Wendt. Anfangs **S**tructured **P**lans for **I**mproving **K**nowledge Transfer in **E**ngineering of **S**ystems – kurz SPIKES – genannt, wurden die Konzepte im Jahr 2001 zu Fundamental Modeling Concepts umbenannt. [FMC]

Grundgedanke der Fundamental Modeling Concepts ist die Möglichkeit einer grafischen Darstellung der konzeptionellen Struktur eines Softwaresystems und deren Informationsverarbeitung und nicht, wie bei vielen anderen üblichen Notationen, die Dokumentation der konkreten Umsetzung. Dadurch soll eine effiziente Kommunikation über Softwaresysteme, auch zwischen verschiedenen projektbeteiligten Gruppen, mit verschiedenen technischen Vorkenntnissen, wie zum Beispiel Software-Architekten und Auftraggebern ermöglicht werden. [Wik13c]

Obwohl sehr auf theoretischen Grundlagen basierend, nutzen bekannte Unternehmen wie SIEMENS, SAP und ALCATEL die Fundamental Modeling Concepts in der Praxis. [FMC]

FMC beinhaltet drei grundlegende Diagrammtypen zur grafischen Darstellung von Softwaresystemen. Dies sind Aufbaudiagramme, Ablaufdiagramme und Wertebereichsdiagramme.

Aufbaudiagramme dienen der Darstellung von Beziehungen verschiedener Systemkomponenten zueinander. Jede Komponente wird der Kategorie Akteur, Kanal oder Speicher zugeordnet. Akteure, dargestellt als eckige Knoten, sind aktive Komponenten. Kanäle, dargestellt durch meist kleine Kreise und Speicher, dargestellt durch Ovale, sind passive Komponenten. Passive Komponenten können nicht mit anderen passiven Komponenten in Beziehung stehen wie auch aktive Komponenten nicht mit anderen aktiven Komponenten in Beziehung stehen können. Dementsprechend können Beziehungen nur zwischen aktiven sowie passiven Komponenten bestehen. Für Kanten zwischen verschiedenen Knoten gilt: Zwischen Akteuren und Speichern müssen Kanten gerichtet sein, zwischen Akteuren und Kanälen können sie dagegen auch ungerichtet sein. Eine Kante von einem Speicher oder einem Kanal zu einem Akteur hat die Bedeutung, dass der Akteur aus dem Speicher liest bzw. von einem Kanal empfängt. Eine entgegengesetzt gerichtete Kante bedeutet, dass der Akteur in den Speicher schreibt beziehungsweise über den Kanal sendet. In beide Richtungen gerichtete Kanten sind nicht vorgesehen, stattdessen müssen zwei entgegengesetzt gerichtete Kanten im Falle einer Beziehung zwischen Speicher und Akteur oder eine ungerichtete Kante zwischen Kanal und Akteur genutzt werden. Kanäle können mit einem „R“ samt Pfeil gekennzeichnet sein und stellen damit einen Request/Response-Kanal dar. Über diesen Kanal kann eine Komponente einen Dienst einer anderen Komponente aufrufen und bekommt eine entsprechende Antwort geliefert. Des Weiteren können Knoten durch die Einbettung in einen anderen Knoten gruppiert und dadurch Gemeinsamkeiten ausgedrückt werden. [Wik13c]

Abbildung 4.7 zeigt ein Beispiel eines Aufbaudiagramms. Es veranschaulicht ein Softwaresystem das es Kunden ermöglicht, über ein Buchungssystem Reisen zu buchen. Kunden können dafür eine Anfrage (Request) an dieses Buchungssystem

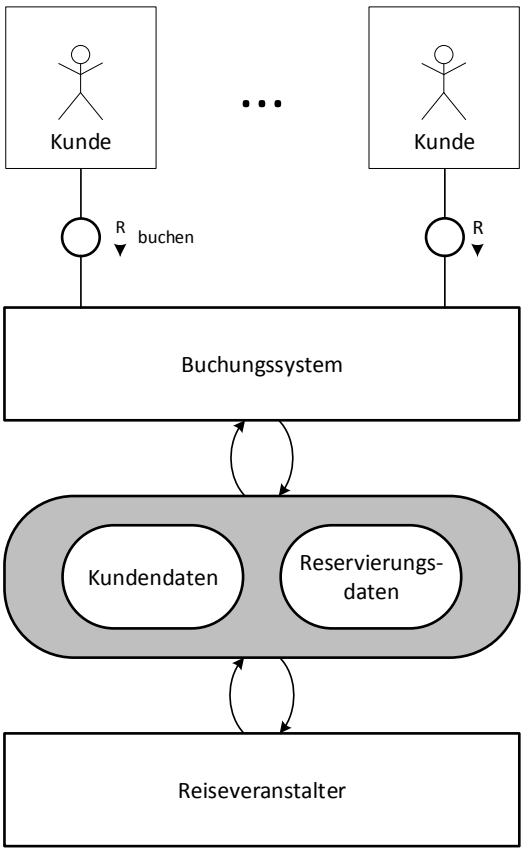

Abbildung 4.7: Beispiel eines FMC-Aufbau-Diagramms nach [Wik13c].

schicken und bekommen nach dessen Bearbeitung eine Response (Antwort) zurück. Das Buchungssystem ist dabei mit zwei Speichern zur Ablage von Kunden- und Reservierungsdaten verbunden, die zur Übersichtlichkeit und zur Reduzierung von Kanälen in einem übergeordneten Speicher gruppiert sind. Die Abbildung zeigt des Weiteren eine Reiseveranstalter-Komponente, die seinerseits Zugriff auf die Kunden- sowie Reservierungsdaten hat.

Ablaufdiagramme werden mit einer erweiterten Art von Petri-Netzen (siehe Abschnitt 4.2.1) dargestellt und stellen den Ablauf innerhalb eines Softwaresystems oder eines Ausschnitts davon dar. Stellen im FMC-Petri-Netz können dabei nur eine Marke aufnehmen, wohingegen mit einem Doppelkreis gekennzeichnete Stellen, unendlich viele Marken aufnehmen können. Des Weiteren können von einer Stelle abgehende Kanten im FMC-Petri-Netz, zur Bestimmung der schaltenden Transition, mit einer Bedingung versehen werden. [Wik13c]

Wertebereichsdiagramme sind im FMC von Entity-Relationship-Diagrammen (siehe Abschnitt 4.1.6) abgeleitet und Beschreiben die im Softwaresystem möglichen Werte-

bereiche. Relationen werden in FMC als eckige Knoten, Entitäten, welche wiederum Attribute enthalten können, als runde Knoten dargestellt. [Wik13c]

Gemeinsam haben alle drei Diagrammtypen, dass sie jeweils eine Klasse mit runden Knoten sowie eine Klasse mit eckigen Knoten enthalten und bipartit sind.

4.1.3 UML-Komponentendiagramm

Komponentendiagramme sind eine der 14 Diagrammarten der UML. Sie werden hauptsächlich zur Visualisierung von Software-Systemen eingesetzt. Hier liegt der Grad der Veranschaulichung auf den einzelnen Komponenten und deren Zusammenhang im größeren System. [Wik13f]

In einem Komponentendiagramm wird eine Komponente durch ein Rechteck (durchgezogene Linie) repräsentiert, das den Stereotyp „«component»“ und den Namen der Komponente enthält. Die Komponenten in einem Komponentendiagramm bilden ein System bzw. Teilsystem. Ein Port kommt zum Einsatz, falls eine Komponente mehrere Schnittstellen besitzt. Es handelt sich dabei um ein kleines Quadrat, dass sich am Rand einer Komponente befindet. Eine Schnittstelle wird durch einen nicht ausgefüllten Kreis repräsentiert, der mittels einer durchgezogenen Linie mit der Komponente verbunden ist, welche die Schnittstelle anbietet bzw. realisiert. Benötigt eine Komponente hingegen eine Schnittstelle, so kann dies durch einen Halbkreis visualisiert werden, der den entsprechenden Kreis bzw. die Schnittstelle umschließt. Hierbei sollte erwähnt werden, dass ein Halbkreis nicht alleine existieren kann, d. h. jede Schnittstelle, die von einer Komponente benötigt wird, muss von einer anderen Komponente angeboten werden. Der umgekehrte Fall, das eine bereitgestellte Schnittstelle nicht genutzt wird, ist hingegen erlaubt. Mittels einer Delegation können die Operationen einer Schnittstelle, die von einer Komponente angeboten wird, zu einer Unterkomponente weitergereicht werden.

Abbildung 4.8 zeigt ein Komponentendiagramm, dass die Komponenten Postbox, Briefkasten und Postverwaltung enthält. Die Postbox nutzt eine Schnittstelle, die von der Postverwaltung zur Verfügung gestellt wird. Die Schnittstelle ist dabei über einen Port mit der Postverwaltung verbunden. Die Komponente Briefkasten stellt eine Schnittstelle bereit, die von der Postverwaltung benötigt wird. Ein (externer) Benutzer nutzt über eine Delegation eine Schnittstelle der Postverwaltung.

Aufgrund des geringen visuellen Vokabulars sind UML-Komponentendiagramme einfach zu lesen, schnell erlernbar und benutzerfreundlich. Systeme können weitestgehend unabhängig von der eingesetzten Programmiersprache dargestellt werden. Weiterhin können Komponenten beliebig verschachtelt werden. Daraus folgt jedoch, dass Komponentendiagramme ab einer bestimmten Komplexität unübersichtlich

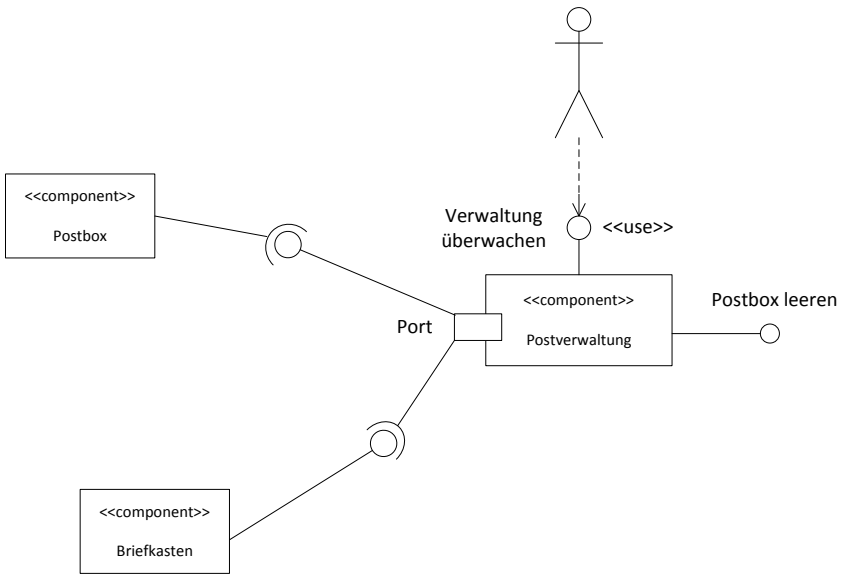

Abbildung 4.8: Beispiel eines UML-Komponentendiagramms.

werden können. Dynamische Daten lassen sich mit einem Komponentendiagramm nicht darstellen.

4.1.4 Acme

Acme ist eine generische Architektur-Sprache, welche zur Beschreibung von Software-Architektur angewendet wird. Gute Notation erlauben es komponentenbasierte Entwürfe der Software zu erstellen, Aussagen über deren Eigenschaften zu treffen und Analyse und System Integration automatisch ablaufen zu lassen. [GMW00]

Acme gehört zur der zweiten Generation von architekturbeschreibenden Sprachen. Acme besitzt Eigenschaften, welche auf den Erfahrungswerten aus anderen architekturbeschreibenden Sprachen gewonnen wurden. Acme ist eine einfache Sprache, dessen grundlegende Elemente, den Architekturentwurf, die natürlichen Erweiterungen unterstützen und komplexere Architektur Erstellung ermöglicht. Acme stellt eine einfache, semantisch erweiterbare Sprache bereit, welche auf eine Menge von Werkzeugen zurückgreifen kann. Diese Werkzeuge unterstützen die Analyse und die Integration unabhängig von den in der Entwicklung verwendeten Werkzeugen. [GMW00]

Acme unterstützt vier Aspekte der Architektur. Der erste Aspekt ist die Struktur. Die Struktur organisiert ein System und dessen Einzelteile. Der zweite Aspekt sind die Eigenschaften, welche von Interesse sind. Diese Eigenschaften stellen Informationen bereit, welche Rückschlüsse über das Verhalten des Systems ermöglichen. Der dritte

Aspekt sind die Einschränkungen. Die Einschränkungen sind Richtlinien, welche beschreiben, wie sich die Architektur über die Zeit verändern kann. Der vierte und letzte Aspekt sind Typen und Stile. Typen und Stile, welche die Klassen und die Familien der Architektur beschreiben. [GMW00]

Die Acme-Struktur besteht aus sieben Kernentitäten: Anhänge, Komponenten, Konnektoren, Systeme, Ports, Rollen, Repräsentationen und Repräsentationen-Karten. [GMW00]

Komponenten repräsentieren berechnende Elemente und die Datenhaltungsschicht des Systems. Eine Komponente kann mehrere Schnittstellen besitzen, mit zugewiesenen Ports interagieren. [GMW00]

Ein Port ist der Interaktionspunkt zwischen einer Komponente und der Systemumgebung. Ports können einfache Schnittstellen und auch komplexere Schnittstellen bereitstellen. [GMW00]

Die Konnektoren repräsentieren die Interaktion zwischen den einzelnen Komponenten. Die Schnittstellen der Konnektoren sind als Rollen definiert. [GMW00]

Rollen definieren einen Teilnehmer einer Interaktion und sind von den Konnektoren abhängig. Es gibt binäre Konnektoren, welche jeweils eine von zwei verschiedenen Rollen annehmen können. Als Beispiel können wir hier einen RPC (Remote Procedure Call) Konnektor betrachten. Hier kann entweder die Rolle des Anrufers oder die Rolle des Angerufenen übernommen werden. Es gibt jedoch auch komplexere Konnektoren für die eine Rolle mehr als zwei Funktionen übernehmen kann. [GMW00]

Das System wird in Acme durch einen definierten Graphen dargestellt. Der Graph besteht zum einen aus Knoten, welche durch Komponenten dargestellt sind. Des weiteren existieren Kanten, welche die Konnektoren repräsentieren. [GMW00]

Anhänge definieren Kanten indem sie Ports mit den zugehörigen Konnektoren der Rollen verbinden. [GMW00]

Die Repräsentationen in Acme (siehe Abbildung 4.9) erlauben das Spezifizieren von Hierarchien der Struktur, Kapselung von Subsystemen, und eine alternative Sicht auf die Struktur. [GMW00]

Mittels der Repräsentationskarten (siehe Abbildung 4.10) können die Verbindungen zwischen verschiedenen Ebenen der Struktur betrachtet werden. [GMW00]

Um Hierarchien in der Architektur darzustellen, erlaubt Acme die Komponenten und Konnektoren in detaillierten Ebenen darzustellen. Diese detaillierten Beschreibungen werden auch als Repräsentationen bezeichnet. Eine Repräsentation kann z. B. dafür genutzt werden, den Datenfluss innerhalb der Schnittstelle zu verfolgen. Die interne

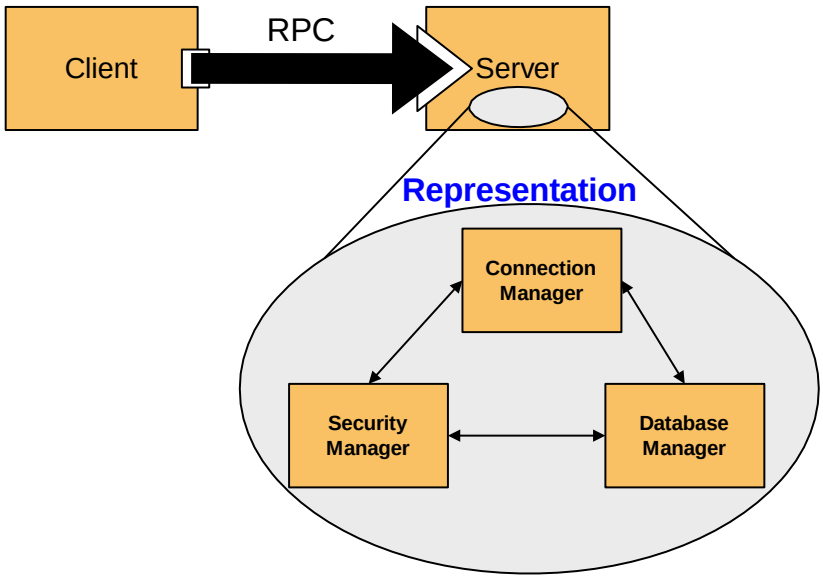

Abbildung 4.9: Beispiel einer Acme-Repräsentation [Ley12].

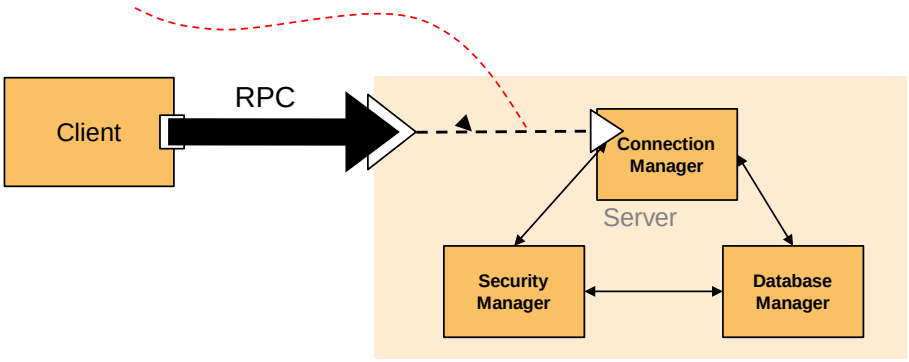

Abbildung 4.10: Beispiel einer Acme-Repräsentationskarte [Ley12].

Korrespondenz auf diesen Ebenen muss ebenfalls dargestellt werden. In Acme wird dies als Repräsentations-Karte „Rep-map“ dargestellt. [GMW00]

Die sieben Entwurfsklassen ermöglichen eine ausreichende Beschreibung der Architektur als Graph aus Komponenten und Konnektoren. [GMW00]

Acme erlaubt das Hinzufügen von Eigenschaften an jede der oben beschriebenen Entitäten um die Architektur noch weiter zu spezifizieren. [GMW00]

Jedoch gibt es auch Kontruktionsbeschränkungen, welche durch die Syntax gegeben sind. Acme verwendet eine beschränkende Sprache, welche auf FOPL (First-Order Predicate Logic) aufgebaut ist. Beschränkungen können dabei mit Entwurfs-Elementen verknüpft werden. Dies ist auf zwei Wegen möglich. Entweder als Invariante oder

als Heuristik. Invarianten stellen Regeln bereit, welche nicht überschritten werden können, wohin eine Heuristik die teilweise Überschreitung dieser Regeln erlaubt. [GMW00]

Acme bietet, wie bereits erwähnt, die Möglichkeit Typen und Stile für die Architektur zu definieren. [GMW00]

Stile erlauben entweder domänenspezifische oder architekturspezifische Beschränkungen zu definieren. Diese sind entweder auf die Typen von Eigenschaften, auf die strukturellen Typen oder auf den Stil, im Allgemeinen anwendbar. Das Annotieren von Eigenschaften ist eine Möglichkeit Annotationen anzuwenden. Die zweite Möglichkeit Annotationen zu verwenden ist, spezielle Typen von Konnektoren, Komponenten, Rollen und Ports zu beschreiben. Stil ist die allgemeine Beschreibung von Eigenschaftstypen und Strukturtypen. Dies wird in Acme auch als Familie bezeichnet. Eine Familie wird durch vier Angaben beschrieben. Eigenschaftenstypen, Strukturtypen, den Beschränkungen und der Standardstruktur. [GMW00]

Acme kann zu verschiedenen Aufgaben verwendet werden: Zur Beschreibung und Analyse der Softwarearchitektur, als Basis um neue Entwurfs- und Analysewerkzeuge zu entwerfen und die Integration von neuen Werkzeugen zu unterstützen. [GMW00]

4.1.5 Service Component Architecture

Service Component Architecture (SCA) ist eine standardisierte Notation um Geschäftsprozesse und die zu Grunde liegende Implementierung darzustellen. Sie basiert auf der Service Oriented Architecture (SOA) und stellt die Geschäftsprozesse serviceorientiert dar. Durch die SCA ist eine getrennte Darstellung von der Geschäftslogik und der Implementierung der Prozesse möglich. [IBM06]

In der SCA existieren drei verschiedene Ebenen. Die erste Ebene ist die Geschäftsintegrationslogik. In dieser Ebene befinden sich die Geschäftsprozesse. Die zweite Ebene wird als Servicekomponenten-Ebene bezeichnet und enthält die Servicekomponenten, in welchen die einzelnen Services (Dienste) implementiert sind. Die dritte Ebene ist die Implementierungsebene. Hier liegt der Programmcode, welcher in den Services verwendet wird. In der SCA wird zudem noch zwischen verschiedenen Artefakten unterschieden. Eine Servicekomponente konfiguriert eine Serviceimplementierung. Die Servicedatenobjekte erweitern die SCA um definierte Services und die Beziehung zwischen den einzelnen Komponenten. Servicequalifikations-Merkmale steuern die Interaktion zwischen dem Service-Client und einem Zielclient. Die SCA besitzt auch Module. Die Einheiten, welche in das Enterprise Archive geschrieben werden, sind von Modulen bestimmt. Aus Leistungsgründen werden die Komponenten in einem Modul zusammengefasst. Die Daten können durch Verweise auf andere Komponenten

übergeben werden. Zuletzt existieren in der SCA Importe und Exporte. Beide definieren die Schnittstellen von Modulen. Importe erlauben den Zugriff auf Services, welche außerhalb des Moduls liegen. Exporte erlauben Komponenten, Services anderen Komponenten zur Verfügung zu stellen. Für die Implementierung der Services einer Komponente können Serviceimplementierungstypen vergeben werden. Da SCA auf verschiedene Geschäftsprozesse angewandt werden kann, wird in Abbildung 4.11 ein allgemeiner Ansatz zur Verdeutlichung gezeigt. [IBM06]

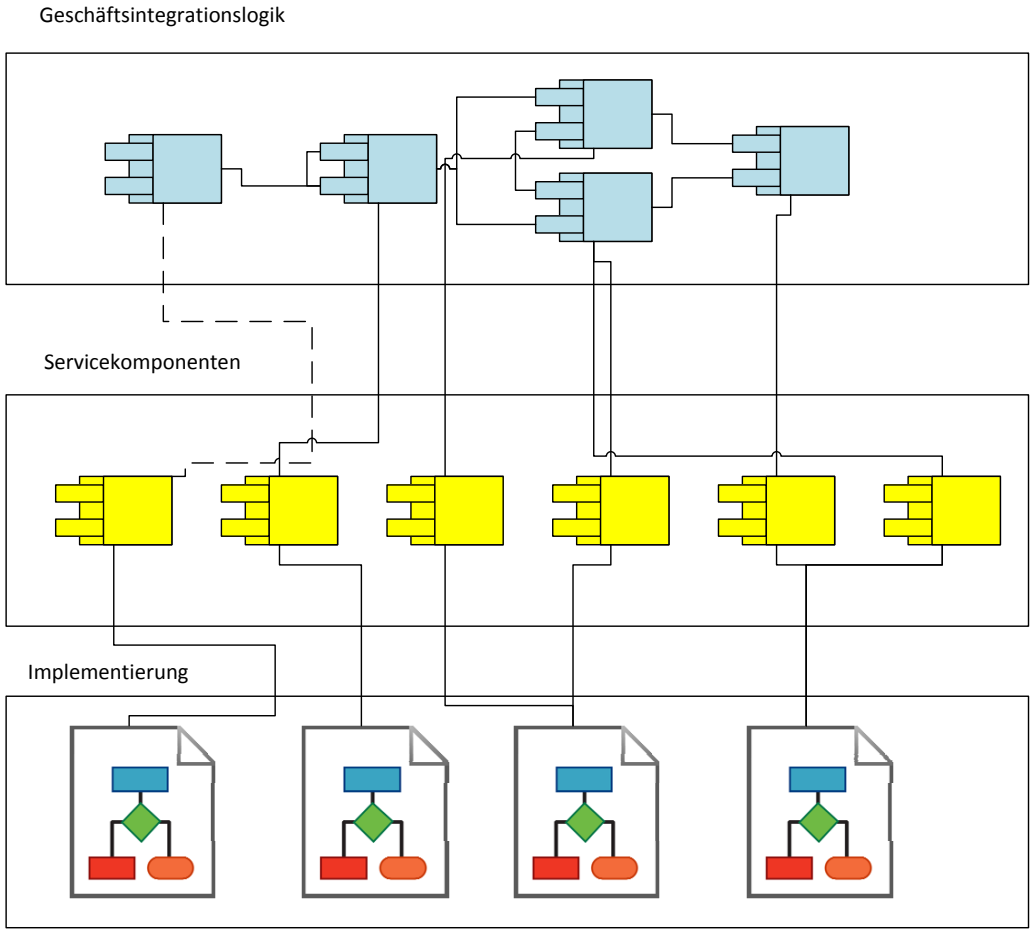

Abbildung 4.11: Beispiel eines SCA-Diagramms nach [IBM06].

4.1.6 ER-Diagramm

Grundsätzlichen dienen ER (Entity Relationship)-Diagramme, im Rahmen der Datenmodellierung, zur Beschreibung eines Ausschnitts der reellen Welt [Wik13a]. Die wohl wichtigste praktische Anwendung ist der Datenbankentwurf. Ein Modell einer Datenbank kann mittels ER-Diagrammen unabhängig von einem konkreten Datenbanksystem wie MySQL dargestellt werden [Che04].

Das visuelle Vokabular besteht lediglich aus drei Elementen. Eine Entität (Entity) wird als Rechteck dargestellt und steht für ein konkretes Objekt. Die Beziehung (Relationship) verknüpft zwei oder mehrere Entitäten miteinander und wird als Raute visualisiert. Eine Eigenschaft (Attribute) wird einer Entität zugewiesen und durch eine Ellipse dargestellt. An den Verknüpfungen kann zudem auch dargestellt werden, wie viele Entitäten von einer anderen Entität ausgehen können. Entweder kann dies durch eine Zahl, oder ein Sternchen festgelegt werden. Sehr häufig findet man die die Zahl 1 oder auch 0..1, d.h. es können null oder eine Entität existieren. Das Sternchen bedeutet, dass beliebig viele Entitäten existieren können. [Che04]

Abbildung 4.12 zeigt zwei ER-Diagramme. Im obigen Diagramm besitzt die Entität Angestellter die Attribute Name und Kürzel. Die Eins neben der Angestellten Entität in Verbindung mit der Beziehung und der Entität Projekt bedeutet, dass ein Angestellter ein Projekt leitet. Hier müssen wir jedoch das Sternchen bei der Entität Projekt betrachten. Daraus folgern wir, dass ein Projektleiter mehrere Projekte leiten kann.
Das zweite Diagramm stellt eine Entität Autor dar, welche zwei Attribute besitzt: Name und Alter. Weiterhin existiert die Entität Buch, welche als Attribute die ISBN und einen Namen besitzt. Hier kann man also als erstes feststellen, das null oder mehr Autoren null oder mehr Bücher verfassen. Im unteren Teil des Diagramms steht vereinfacht, dass ein Verlag einen Namen hat und dieser null oder mehrere Bücher verlegt.

Aufgrund ihres geringen visuellen Vokabulars, sind Entity-Relationship-Diagramme, einfach zu lesen, schnell erlernbar und benutzerfreundlich. Zudem können Sie Daten unabhängig vom Datenmodell darstellen. Wie jedoch auch schon bei den Petri Netzen, sind Entity-Relationship Diagramme ab einer gewissen Größe unübersichtlich und eignen sich dadurch ebenfalls nicht für die Darstellung von Topologien. Ebenso wie Petri Netze lassen sich Entity-Relationship-Diagramme nicht für dynamische Verfahren anwenden. Dies liegt an der Tatsache, das Entity-Relationship-Diagramme keine Operationen besitzen, welche dynamische Modellierung ermöglicht, wodurch ein Entity-Relationship Diagramm nur zur statischen Modellierung eingesetzt werden kann. [Loh05]

4.1.7 HIPO-Diagramm

Ein HIPO (Hierarchy plus Input-Process Output)-Diagramm ermöglicht die Darstellung von Funktionen, die in einem System ausgeführt werden. Jede Funktion wird durch die Eingabe, die Verarbeitung und die Ausgabe beschrieben. HIPO-Diagramme wurden um 1970 von IBM entwickelt. [Wik13e]

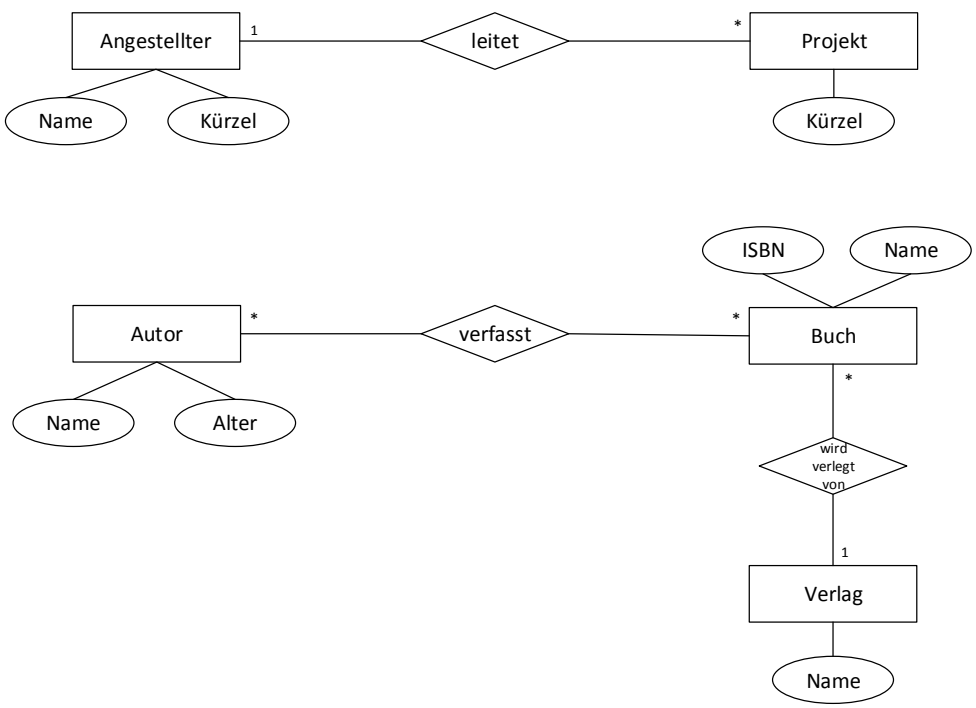

Abbildung 4.12: Beispiel eines Entity-Relationship-Diagramms nach [Wik13a].

Grundsätzlich besteht ein HIPO-Diagramm aus zwei Teilen: Der erste Teil ist die Hierarchie, durch die festgelegt wird, welche Prozesse welche Unterprozesse besitzen. Im zweiten Teil sind die Eingabe, der zuständige Prozess und die resultierende Ausgabe beschrieben. [Wik13e]

Abbildung 4.13 zeigt ein HIPO-Diagramm, in dem die Eingaben „Materialbuchführung“, „Lohnbuchführung“, „Anlagenführung“, „direkt“ und „Debitoren“ in den Prozess „Erfassung“ einfließen. Für einen weiteren Prozess „Gliederung“ existiert die Eingabe „Angabe zu Art der Leistungsabhängigkeit Erfassungsart“. Letzteren Prozess benötigt die Ausgabe des ersten Prozesses, sodass eine Abhängigkeit zwischen den Prozessen besteht.

Abhängigkeiten lassen sich zwischen beliebigen Prozessen definieren, sodass einfache Prozesse zu größeren, komplexeren Prozessen zusammengesetzt werden können.

HIPO-Diagramme erlauben die Darstellung von komplexen Abläufen. Zudem können mittels HIPO-Diagrammen Systeme bis hin zur Funktionsebene strukturiert werden. Zur Bestimmung der Komplexität einer Funktion können HIPO-Diagramme verwendet werden. Jedoch werden HIPO-Diagramme, wie die zuvor vorgestellten Diagrammtypen, auch ab einer gewissen Größe unübersichtlich. Ein weiterer Nachteil ist dass sich nur Programm- und keine Datenflüsse darstellen lassen. [LS78]

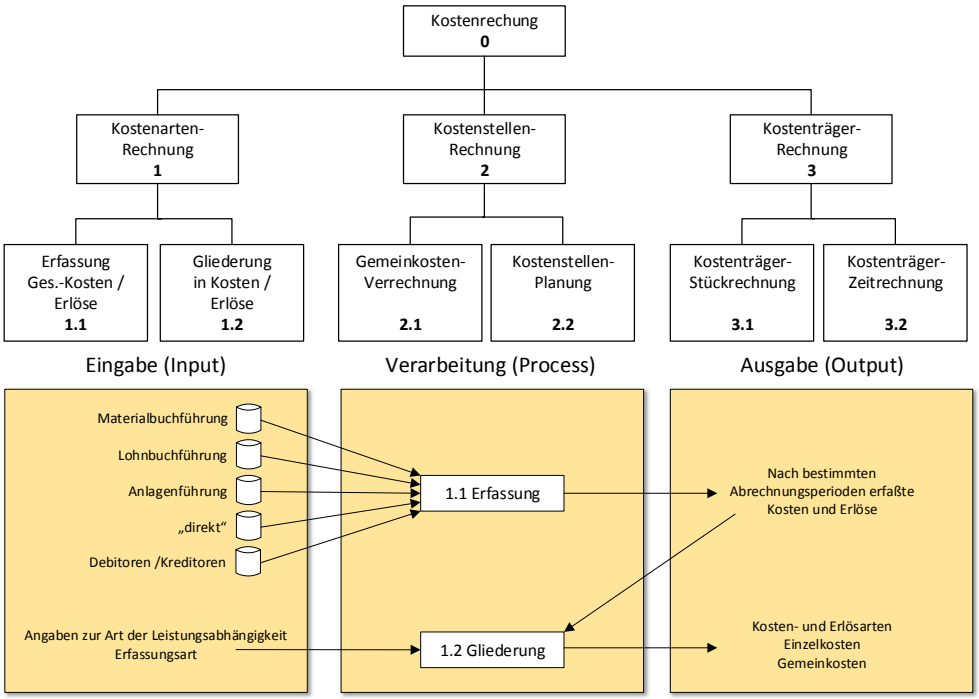

Abbildung 4.13: Beispiel eines HIPO-Diagramms nach [Fri02].

4.2 Geschäftsprozesse

4.2.1 Petri-Netz

Die erste visuelle Notation welche betrachtet werden soll, sind Petri-Netze. Petri Netze dienen zur Analyse, Modellierung, und Simulation von verteilten dynamischen Systemen mit nebenläufigen und nichtdeterministischen Vorgängen. Petri Netze gibt es schon seit den 1960er Jahren und wurden von ihrem Namensgeber, Carl Adam Petri entwickelt. Petri Netze entstanden aus endlichen Automaten und sind aus zwei verschiedenen Arten von Knoten aufgebaut, den Stellen und Transitionen. In Stellen liegen während der Ausführung Tokens. Transitionen verschieben Tokens an andere Stellen. Zwischen Stellen und Transitionen können Verbindungen existieren. Die Verbindungen werden auch als Kanten bezeichnet. Existiert eine Verbindung zwischen einer Stelle und einer Transition kann die Transition ausgeführt werden sobald die benötigte Anzahl der Tokens an ihr anliegt. Dies resultiert daraus, dass zusätzlich an den eingehenden und auch den ausgehenden Kanten einer Transition, Bedingungen erstellt werden können. So kann zum Beispiel eine Transition erst einen Token erhalten wenn in der Stelle aus der der Token kommt zwei Tokens anliegen. Dies ermöglicht die Erzeugung von neuen Tokens aber auch die Reduzierung von schon vorhandenen Tokens.

Abbildung 4.14 zeigt ein Petri-Netz, das aus sechs Stellen und vier Transitionen besteht. Zu Beginn liegen in der Start Stelle und in Kunde am Schalter 1 jeweils ein Token. Die erste Transition welche gefeuert werden kann ist die Kunden Auskunft. Der Token wird aus Kunde am Schalter 1 in Schalter frei und in Kunde informiert gelegt. Kunde am Schalter 1 ist nun nicht mehr belegt. Zum gleichen Zeitpunkt wird aber auch die Transition „Kunde zu Schalter 2“ gefeuert. Dies resultiert aus den belegten Stellen Schalter frei und Start. Nach den ersten beiden Schritten wurden also zwei Transitionen gefeuert und einige Tokens verschoben. Da es in diesem Petri-Netz keinen Fangzustand gibt und die Tokens immer wieder erzeugt werden, kann dieses Petri Netz unendlich viele Zustände durchlaufen.

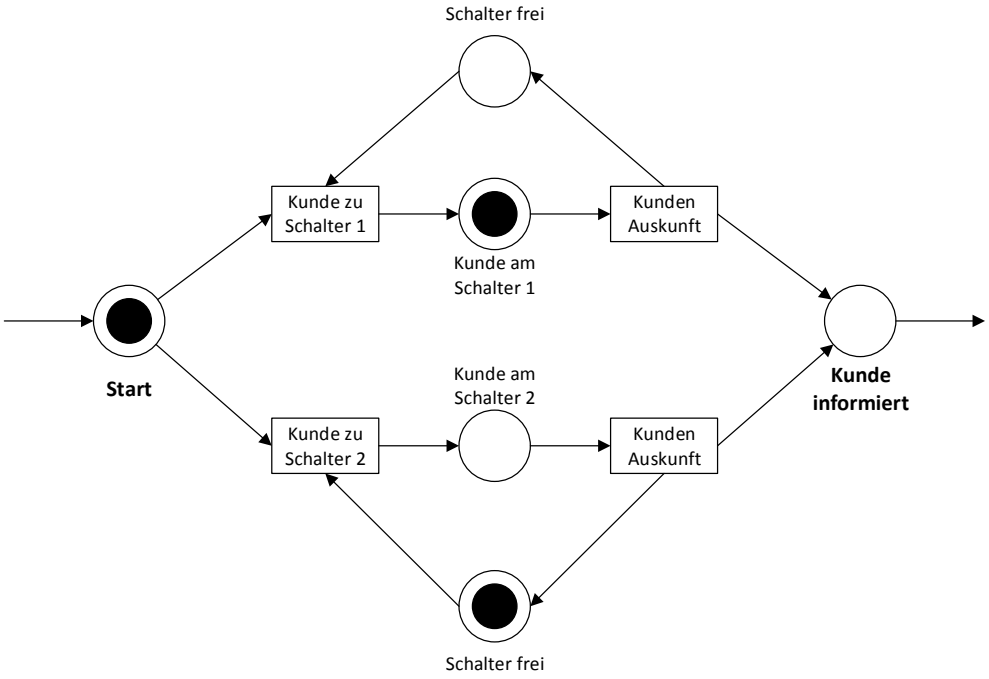

Abbildung 4.14: Beispiel eines Petri-Netzes nach [sof]

.

Petri-Netze ermöglichen komplexere Vorgänge zu beschreiben, obwohl ihre Syntax sehr einfach ist. Durch diese sehr simple Syntax kann die Erstellung von solchen Petri-Netzen schnell erlernt und angewandt werden. Zusätzlich ermöglichen Petri-Netze eine einfache Situationsanalyse durch die Verwendung von Tokens. Der Aufbau eines Petri Netzes erlaubt auch die Modellierung von genauen Abläufen und Vorgängen, welche mit Vor- und Nachbedingungen erstellt werden. Leider existieren für die praktische Anwendung keine einheitlichen Notationen für höhere Petri Netze. Dadurch ist die Erstellung dieser höheren Petri Netze sehr aufwändig und die Analyse dieser oft sehr kompliziert. Petri-Netze sind außerdem statische Strukturen wodurch ihre

Einsatzmöglichkeiten beschränkt sind. Aus Mangel einer einheitlichen Vorgehensweise zur Erstellung von Petri Netzen, können bei der Erstellung eines Petri Netzes schon Probleme auftreten. Zuletzt sind Petri-Netze ab einer gewissen Größe unübersichtlich. Aus diesem Grund eigenen sich Petri Netze nicht zur Darstellung von komplexeren Prozessen, wie sie in Managementplänen und Topologien zu finden sind.

4.2.2 GWorkflowDL

Die nächste Notation die der Analyse unterzogen wird ist die Generic Workflow Description Language (GWorkflowDL). Die grundlegende Sprache der GWorkflowDL ist XML. Die GWorkflowDL eignet sich um Geschäftsprozesse und ausführbare Prozesse zu beschreiben. Die GWorkflowDL basiert auf höheren Petri Netzen und wird vom Frauenhofer Institut für Rechnerarchitektur und Softwaretechnik seit 2001 entwickelt. (Quelle Wiki) Im GWorkflowDL werden die Aktivitäten eines Prozesses durch Transitionen dargestellt. Die Tokens repräsentieren die verwendeten Daten. Neben der Modellierung und der Analyse dient das GWorkflowDL auch zur Überwachung und der Ausführung von Prozessen. Dies resultiert daher, dass sich Kontrollfluss und Datenfluss mittels des GWorkflowDL gut beschreiben lassen. Die GWorkflowDL kann sowohl technische Prozesse als auch abstrakte Geschäftsprozesse abbilden. [Wik13d]

In Abbildung 4.15 ist ein GWorkflowDL-Diagramm dargestellt. Zu Anfang liegen zwei Token, die zu bearbeitenden Daten, in der Aktivität beginn. Die Daten werden nun an den Prozess geliefert, der als Transition dargestellt wird. Nach Durchlauf des Prozesses werden die bearbeiteten Daten in die Aktivitäten ausgabeDaten und wurdeSortiert geschrieben. Hier ist leicht zu erkennen, das Rückmeldung über den Prozess in der darauffolgenden Aktivität gegeben wird.

Wie unschwer erkennbar ist, lassen sich mit dem GWorkflowDL komplexe Prozesse gut modellieren. Meist jedoch ist das Resultat sehr komplex und daher schwer zu analysieren.

4.2.3 Nassi-Shneiderman-Diagramm

Nassi-Shneiderman-Diagramme (Struktogramme) bieten eine weitere Möglichkeit der Darstellung von Daten in graphischen Sichtweisen. Nassi-Shneiderman-Diagramme dienen hauptsächlich der Erstellung von Programmentwürfen.
Nassi-Shneiderman-Diagramme wurden von Isaac Nassi und Ben Schneiderman 1972/73 erfunden. Nassi-Shneiderman-Diagramme zerlegen das Gesamtproblem in mehrere kleinere Teilprobleme. Nassi-Shneiderman-Diagramme werden auch oft in

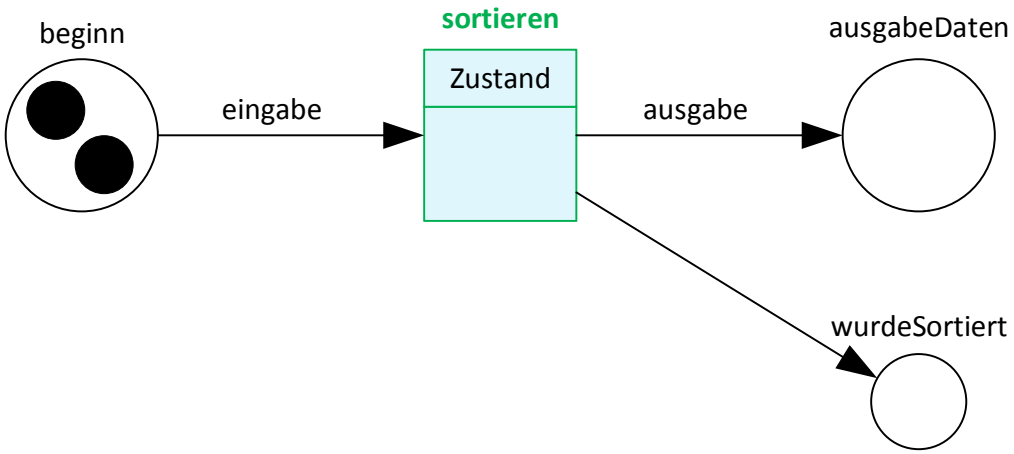

Abbildung 4.15: Beispiel eines GWorkflowDL-Diagramms nach [Fra11].

der Top-Down Entwicklung von Programmen verwendet um an diesen eine Programmanalyse zu ermöglichen. Die Darstellung von Nassi-Shneiderman-Diagramme sind hintereinander geschaltete oder geschachtelte Strukturblöcke. [Wik13g]

Abbildung 4.16 zeigt ein Nassi-Shneiderman-Diagramm. Als erstes wird die Ausgabe auf 2,3 gesetzt. Im nächsten Schritt wird die Variable n mit dem Wert fünf initialisiert. Nun existiert eine Schachtelung. Die nächsten Schritte werden laut dem Diagramm ewig durchlaufen. Wir haben in diesem Beispiel also eine unendliche Schleife. Zu Beginn wird i mit drei initialisiert. Nun folgt eine weitere Schleife. In dieser Schleife wird bei jedem Durchlauf der Wert zwei auf den Wert von i hinzu addiert. Wenn die angegebene Bedingung nicht mehr gilt, wird der nächste Programmschritt ausgeführt. In diesem Schritt wird geprüft ob i*i > n ist. Anhand vom Wahrheitswert wird dann entscheiden in welchem Zweig das Programm weiterlaufen soll.

Anhand des Beispiels konnte man erkennen, dass Nassi-Shneiderman-Diagramme sehr leicht erstellt werden können. Zudem erzwingt ein Nassi-Shneiderman-Diagramm einen disziplinierten Programmablauf zu gestalten. Durch ihre Simplizität sind Nassi-Shneiderman-Diagramme leicht verständlich und schnell auch von Laien erstellbar. Wie aber auch schon bei den vorherigen Diagrammen, können Nassi-Shneiderman-Diagramme durch ihre Größe oder die dargestellten Algorithmen sehr komplex und daher schwer zu lesen werden. [Gab]

4.2.4 Folgeplan und Flussdiagramm

Der nächste Diagramm-Typ der betrachtet werden soll, ist der Folgeplan bzw. die Flussdiagramme. Flussdiagramme werden öfters auch als Programmstrukturpläne

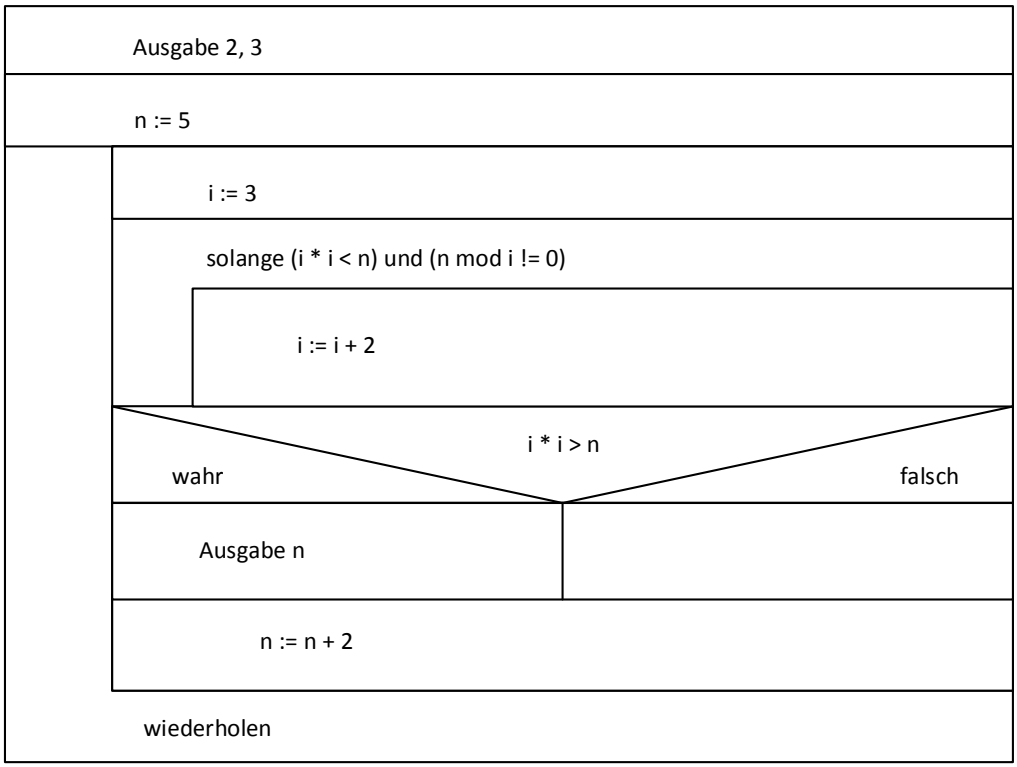

Abbildung 4.16: Beispiel eines Nassi-Shneiderman-Diagramms nach [LR09].

bezeichnet. Ein Flussdiagramm beschreibt einen Algorithmus bzw. die Folge der im Algorithmus abgehandelten Operationen. Ein Flussdiagramm unterstützt somit den Entwickler bei der Programmierung bzw. der Umsetzung des Algorithmus. Flussdiagramme gehören zu einer der Diagrammarten welche genormt sind, wodurch die Arbeit mit ihnen deutlich erleichtert wird. Das Flussdiagramm soll nun anhand eines Beispiels vorgestellt werden. Flussdiagramme bestehen aus Knoten und Kanten. Die Knoten haben jedoch unterschiedliche Formen und Farben, abhängig davon was sie zu bedeuten haben. Kanten stellen den Informationsfluss der Daten dar. In Abbildung 4.17 wird mit dem Startknoten begonnen. Danach folgt die erste Aktivität, „Zum Telefon gehen“. Danach folgt die nächste Aktivität „Hörer abnehmen“. Nun wird durch eine Bedingung, dargestellt durch eine Raute, abgefragt. Falls es ein Ankommender Ruf existiert, wird erst die Aktivität „Gespräch führen“ durchgeführt und danach „Hörer auflegen“. Im Falle, dass die Bedingung „Ankommender Ruf“ falsch ist, folgt die Aktivität „Nummer wählen“. Hiernach folgt die nächste Bedingung. Falls ein Gesprächspartner ans Telefon geht, wird das Gespräch geführt. Ansonsten wird der Hörer aufgelegt. Nach der Aktivität „Hörer auflegen“ wird Stop aufgerufen.

Flussdiagramme bieten gegenüber anderen Diagrammtypen sehr erhebliche Vorteile. Unter anderem besitzen Flussdiagramme ein geringes visuelles Vokabular wodurch

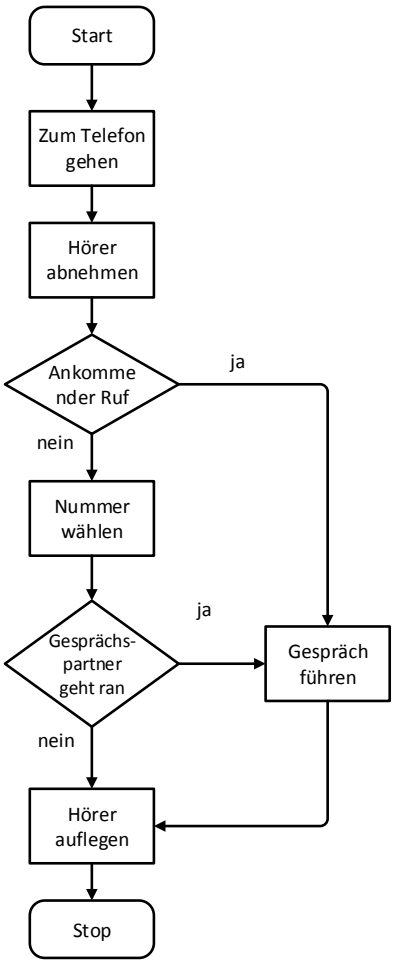

Abbildung 4.17: Beispiel eines Flussdiagramms nach [www].

sie in sehr kurzer Zeit erlernbar sind. Flussdiagramme finden ihre Anwendung nicht nur im IT-Bereich sondern auch in anderen Bereichen wie z.B. Prozessplanung und Durchführungsstrategien. Leider existieren für Flussdiagramme keine Symbole für grundlegende Kontrollstrukturen. Daraus ist es in Flussdiagrammen schwer zu erkennen ob Strukturen geschachtelt sind oder nicht. Flussdiagramme lassen sich leider nicht optimal für objektorientierte Programmierkonzepte einsetzen. Hier eignen sich UML-Diagramme besser.

4.2.5 Datenflussdiagramm

Datenflussdiagramme sind ein weiterer Diagrammtyp, die nun näher betrachtet werden sollen. Das Datenflussdiagramm beschreibt den Fluss von Daten durch das System. Hierzu gehören sowohl die Bereitstellung, die Veränderung und die Verwendung der Daten. Datenflussdiagramme lassen sich auch zur Darstellung von

Prozessen und Tätigkeiten verwenden. Des Weiteren sind Datenflussdiagramme ebenso wie Folgepläne genormt (DIN66001). Datenflussdiagramme besitzen vier verschiedene graphische Elemente. Das erste Element ist der Datenspeicher. Dieser wird mittels zwei paralleler Linien dargestellt, zwischen welchen der Name des Speichers angegeben wird. Das zweite Element ist der Datenfluss. Der Datenfluss wird durch einen Pfeil mit Namen dargestellt. Das dritte Element ist die Funktion, welche durch einen Kreis mit Namen dargestellt wird. Das vierte und letzte Element ist die Schnittstelle. Sie wird durch ein Rechteck mit Namen dargestellt.

Abbildun 4.18 zeigt ein Datenfluss-Diagramm. Dieses enthält drei Datenspeicher, acht verschiedene Datenflüsse, zwei Funktionen und zwei Schnittstellen. Von der Kundenschnittstelle werden Bestellungsdaten an die Bestellung bearbeiten Funktion gesendet. Von hier aus werden die Kundendaten aus der Kunden Datei und Bestandsdaten aus der Lagerbestands Datei geladen. Die Bestellung bearbeiten Funktion schickt dann Entnahmedaten an die Lagerbestands Datei und Lieferungsdaten an die Funktion „Rechnung schreiben"'. Diese greift wieder auf die Kundendatei zu, schickt Rechnungssummen an die Debitorendatei und sendet schließlich die Rechnungsdaten an die Kundenschnittstelle.

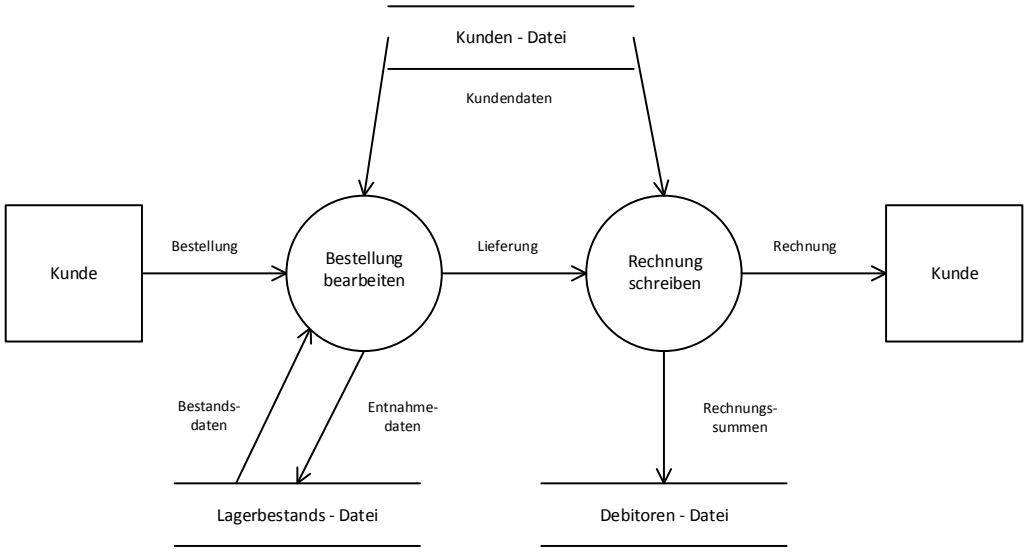

Abbildung 4.18: Beispiel eines Datenflussdiagramms nach [Win13].

Das Datenflussdiagramm ist, da es nur auf vier Elementen aufgebaut ist, sehr schnell und einfach erlernbar. Zusätzlich ist es eben durch diese vier verschiedenen Elemente, welche das visuelle Vokabular ausmachen, sehr gut verständlich und leicht lesbar. Eine weitere Eigenschaft des Datenflussdiagramms ist, dass man den theoretischen mit dem tatsächlichen Ablauf vergleichen kann. Jedoch kommt bei dieser ansonsten schönen Diagrammart, die Unübersichtlichkeit bei größeren Diagrammen zum Vorschein.

4.2.6 UML-Aktivitätsdiagramm

Das UML-Aktivitätsdiagramm ist ein Verhaltensdiagramm. Es beschränkt sich auf die dynamischen Prozesse des Systems. Aktivitätsdiagramme stellen die Verknüpfung von elementaren Aktionen mit Kontroll- und Datenflüssen graphisch dar. Aktivitätsdiagramme eignen sich sehr gut zur Beschreibung von Anwendungsfällen. In Aktivitätsdiagrammen existieren Token, welche den Durchlauf durch das Aktivitätsdiagramm darstellen. Innerhalb des Aktivitätsdiagramms existieren zudem Aktionen, welche auch als Unteraktionen dargestellt werden können. [AMA, RJB99]

Anhand von Abbildung 4.19 wird nun die allgemeine Funktionsweise eines Aktivitätsdiagramms erläutert. Die erste Aktion zu welchem der erste Token gesendet wird, ist die Gast erscheint Aktion. Von dieser Funktion wird nun eine Subaktion anhand von dem Status der Eingabe geöffnet. Falls der Gast neu ist, wird die Aktion Gast aufnehmen aufgerufen. Der Gast wird als aufgenommen markiert und kann nun verwaltet werden. Falls der Gast als bekannt erkannt worden wäre, wäre die Gast verwaltet Aktion direkt nach der Gast erscheint Aktion aufgerufen worden. Die letzte Aktion, die hier im Beispiel aufgerufen wird, ist die Belegung erstellen Aktion. Danach ist das Aktivitätsdiagramm abgearbeitet.

An obigem Beispiel konnte man sehr gut die Vorteile des Aktivitätsdiagramms verdeutlichen. Aktivitätsdiagramme enthalten vereinheitlichte Symbole welche der Prozessdarstellung dienen. Durch diese Standardisierung, muss nur einmal die Syntax verstanden worden sein um sie auf alle anderen Aktivitätsdiagramme anwenden zu können. Dies ermöglicht das Abläufe leicht und effizient nachvollzogen werden können. Zudem kommt hinzu, dass Aufgrund des UML-Standards ein relativ hoher Bekanntheitsgrad existiert, welches die Verwendung von Aktivitätsdiagrammen positiv beeinflusst. Es existieren jedoch auch hier Nachteile die, die Verwendung von Aktivitätsdiagrammen für Topologie-Darstellungen nicht begünstigen. Der erste dieser ist, dass durch Aktivitätsdiagramme hauptsächlich Abläufe dargestellt werden können, welche technisch orientiert sind. Zudem gibt es nur unzureichende Vorgaben zur Struktur des Prozessmodells. Aktivitätsdiagramme lassen sich auch nur für den Einzelprozess anwenden, da keine Möglichkeiten gegeben sind um komplexere Prozesse, oder welche die in sich verschachtelt sind, darstellen zu können. Aussagen über die Prozessgüte lassen sich außerdem nicht aus dem Modell ablesen. Das Aktivitätsdiagramm eignet sich nicht für die direkte Automatisierung eines Prozesses. [AMA, RJB99]

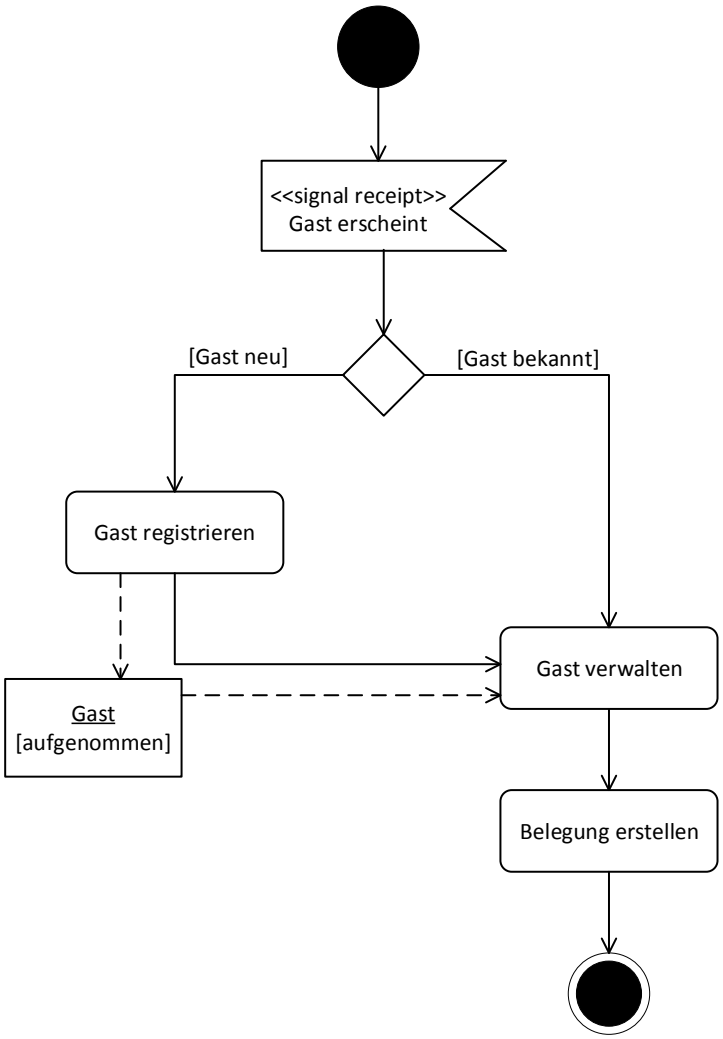

Abbildung 4.19: Beispiel eines UML-Aktivitätsdiagramm nach [Inf13].

4.2.7 Business Process Model and Notation (BPMN)

Ein weiterer Diagrammtyp ist die Business Process Model and Notation (BPMN). BPMN ist eine graphische Spezifikationssprache in der Wirtschaftsinformatik. Mithilfe der bereitgestellten Symbole lassen sich Geschäftsprozesse und Arbeitsabläufe modellieren und dokumentieren. Ab 2001 wurde BPMN von IBM entwickelt und angewandt und seitdem weiterentwickelt. BPMN ist mit EPK verwandt. In der BPMN existieren folgende graphische Elemente: Flow-Objekts sind die Knoten in den Geschäftsprozessdiagrammen. Connecting-Objekts verbinden Knoten in den Geschäftsprozessidagrammen. Pools und Swimlanes sind die Bereiche in welchen Aktoren und Systeme dargestellt werden. Artifakts sind weitere Elemente wie dataobjects groups und annotions zur weiteren Dokumentation. In folgendem Beispiel wird ein Vertriebssystem dargestellt. Begonnen wird mit einer Mitteilung in Schriftform,

welche an das Subsystem Lieferauftrag buchen und einmal an das Datenmodel Lieferung geschickt wird. Lieferung buchen legt eine neue Lieferung an. Dies geschieht in einem Subsystem. Diese liefert wiederrum Daten an den Lieferung warten Prozess. Bedingungen können ebenfalls in die Systemabläufe eingespielt werden. Hier wird geprüft ob für eine Lieferung das Avis System gebraucht wird oder nicht. Im Falle, dass es nicht gebraucht wird, folgt der Endzustand und das Vertriebssystem ist bis zur Eingabe neuer Daten inaktiv.

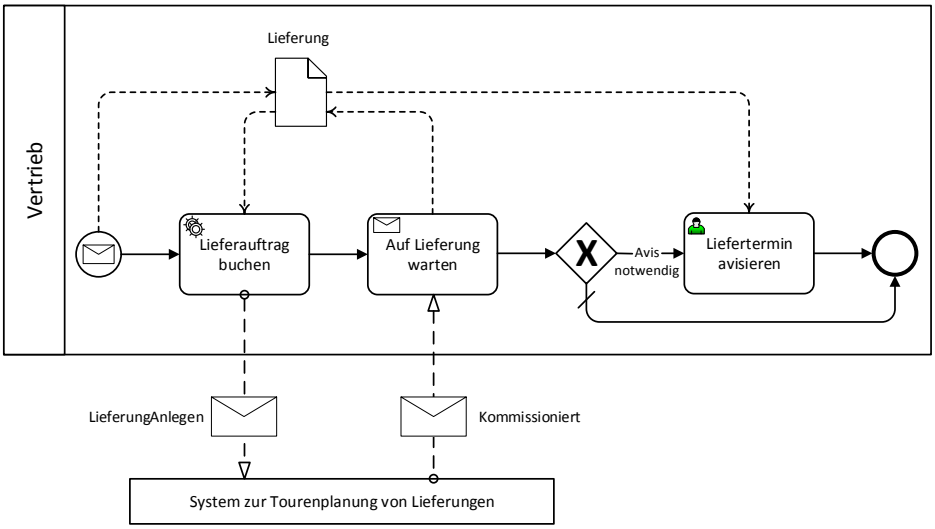

Abbildung 4.20: Beispiel eines BPMN-Diagramms nach [RÖ09].

In Abbildung 4.20 kann man die Grundstruktur der BPMN sehr gut erkennen. Die Prozesse lassen sich sehr einfach in den Lanes darstellen. Bei Bedarf kann man auch weiter ins Detail gehen wie im zweiten Beispiel zu sehen war. BPMN eignet sich sehr gut um IT und Geschäftsprozesse miteinander zu verbinden. Auch ist sie gut für größere Projektteams geeignet, welche aus unterschiedlichen Bereichen kommen können. BPMN ist wie UML durch die OMG standardisiert. Andererseits ist BPMN nicht so leicht zu erlernen, da ein sehr großes Vokabular existiert was die Komplexität der Diagramme immens erhöht, was jedoch anhand der Beispiele nicht auf Anhieb ersichtlich ist. Viele fachliche Aspekte des Prozesses werden durch BPMN nicht dargestellt.

4.2.8 Ereignisgesteuerte Prozesskette (EPK)

Die Ereignisgesteuerte Prozesskette (EPK) ist eine weitere graphische Visualisierung für Pläne. Geschäftsprozesse werden hier als zeitliche logische Abfolge von betriebswirtschaftlichen Aufgaben dargestellt. EPK wurde 1992 an der Universität

des Saarlandes entwickelt. EPK beinhaltete drei grundlegende Notationselemente: Event, Funktion, und Logische Operatoren (And, Or, Xor) [Wik13b, Enz, re-]

In Abbildung 4.21 ist eine einfache EPK dargestellt. Das erste Event Auftrag ist eingegangen, löst die Funktion Kundendaten Prüfen aus. Nachdem die Kundendaten geprüft wurden, wird das Event Kundendaten sind geprüft aufgerufen, was sofort die nächste Funktion aufruft. Dies geschieht bis keine Funktion mehr nach einem Event aufgerufen werden kann.

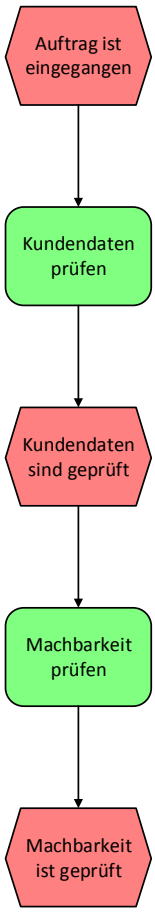

Abbildung 4.21: Beispiel eines EPK-Diagramms nach [Ges12].

Das Beispiel veranschaulicht die Einfachheit von EPK. EPK eignet sich aufgrund seiner Struktur für eine breite Anzahl von Anwendungen. EPK ist außerdem sehr umfassend und bietet Konstrukte für organisatorische als auch IT Aspekte. Ebenfalls lässt sich EPK gut in andere Notationen integrieren. Durch seine Simplizität kann EPK auch von nicht IT-Leuten verstanden werden. Durch seine weite Verbreitung findet EPK auch in der Praxis Anwendung. Zu den genannten Vorteilen liegen bei EPK auch einige Nachteile anbei. Einige Konstrukte sind nicht eindeutig spezifiziert. Dies führt zu mehrfachen Auslegungsmöglichkeiten der Konstrukte, was nicht der Verständlichkeit dient. Um EPK für weitere Anwendungsbereiche nutzen zu können,

ist eine grundsätzliche Überarbeitung für diese Anwendungsbereiche nötig. EPK ist nicht intuitiv, wodurch ein hoher Schulungs- und Einarbeitungsaufwand gefordert wird.[Wik13b, Enz, re-]

4.3 Auswertung und Schlussfolgerungen

Vino4TOSCA (siehe Abschnitt 4.1.1) ist die einzige visuelle Notation für Anwendungstopologien, die bereits auf TOSCA-Topologien ausgerichtet ist und zudem die aufgestellten Anforderungen aus den Abschnitten 3.1, 3.2 und 3.3 erfüllt. Lediglich die Anforderungen A16, A17 und A20 werden nicht erfüllt, da mit Vino4TOSCA, wie bereits erwähnt, ausschließlich Topology Templates visualisiert werden können. Für Node Types und Relationship Types existieren folglich keine grafischen Elemente[10]. Auch sieht die Notation keine Möglichkeit zur Integration von visuellen Prozessnotationen vor. Damit die genannten Anforderungen erfüllt werden, müssen entsprechende Anpassungen bzw. Erweiterungen an der bestehenden Vino4TOSCA-Notation vorgenommen werden.

Aus dem genannten Grund wurde daher die Entscheidung getroffen, die neue Notation auf Basis von Vino4TOSCA zu entwerfen bzw. eine weiterentwickelte Version von Vino4TOSCA bereitzustellen. Bedingt durch diese Entwurfsentscheidung wird als Name für die neue Notation Vino4TOSCA 2 gewählt.

Alle weiteren visuellen Notationen im Bereich der Anwendungstopologien erfüllen die Anforderungen A10, A16, A17, A19, A20 nicht, da keine visuellen Dialekte vorgesehen sind bzw. nicht die Möglichkeit zur Erstellung von Profilen angeboten wird und sie weiterhin nicht TOSCA-spezifisch ausgerichtet sind. Komponentendiagramme, HIPO-Diagramme und Acme erfüllen zusätzlich A23 nicht, da keine visuellen Elemente vorhanden sind, die sich durch abgerundete Ecken auszeichnen, was eine visuell ansprechende Eigenschaft darstellen würde. Die letzten beiden Notationen sind zudem nicht selbsterklärend, sodass A21 nicht erfüllt ist. Für ER-Diagramme ist A5 nicht gegeben, da weder Abstraktion noch Modularisierung unterstützt wird. HIPO-Diagramme und ER-Diagramme sind nicht erweiterbar, d. h. deren Elemente können nicht mit zusätzlichen Informationen oder Eigenschaften versehen werden. Aus diesem Grund ist für die genannten Notationen A18 nicht erfüllt.

Für Vino4TOSCA 2 wurde aus den vorgestellten Notationen für Anwendungstopologien, mit Ausnahme von Vino4TOSCA, lediglich zwei Entwurfsentscheidungen übernommen bzw. als brauchbar eingestuft: Wie bereits angesprochen, wird in einem

[10] In einem Vino4TOSCA Node Template Shape bzw. Relationship Template Shape kann die ID und der Name des zugehörigen Node Types bzw. Relationship Types definiert werden. Separate grafische Elemente für Typ-Elemente existieren nicht.

Komponentendiagramm eine Schnittstelle einer Komponente durch einen nicht ausgefüllten Kreis dargestellt, der mittels einer Linie mit der Komponente verbunden ist. Die Verwendung der Schnittstelle durch eine weitere Komponente wird durch einen offenen Halbkreis visualisiert, welcher den Kreis halb umschließt. In Vino4TOSCA 2 soll diese Darstellungsweise für eine Beziehung zwischen der Schnittstelle eines Node Type bzw. Relationship Type und einer Aktivität eines Plans, welche eine Operation der Schnittstelle aufruft, zum Einsatz kommen. Da in einem Komponentendiagramm lediglich auf Schnittstellen-Ebene eine Beziehung dargestellt werden kann, hier jedoch konkret eine Operationsaufruf einer Schnittstelle dargestellt werden soll, müssen Anpassungen an der Darstellung vorgenommen werden. Ebenso müssen Änderungen bzw. Erweiterungen vorgenommen werden, damit zwischen „Source Interfaces“ und „Target Interfaces“ eines Relationship Type unterschieden werden kann. Weiterhin wird in einem Komponentendiagramm eine Komponente mit dem Schlüsselwort bzw. Stereotyp „«component»“ dargestellt. In Vino4TOSCA 2 soll die ID des zugehörigen Typ-Elements durch spitze Klammern eingeschlossen bzw. durch diese hervorgehoben werden.

Unter den vorgestellten visuellen Notationen für Geschäftsprozesse existiert keine Notation, die sich für das Management von Anwendungen etabliert hat. Wir nehmen an, dass diese Feststellung auf alle visuellen Prozessnotationen übertragbar ist. Am ehesten würde sich BPEL für die genannte Aufgabe eignen. Keller beschreibt die automatisierte Provisionierung (Bereitstellung) von Anwendungen mit BPEL [KB04]. Jedoch existiert für BPEL keine standardisierte visuelle Notation. Stattdessen existieren eine Reihe von Ansätzen, welche die Konstrukte von BPEL mittels Icons oder Shapes visualisieren. Falls Modellierer und Leser unterschiedliche grafische Elemente einsetzen, führt dies u. U. zu Missverständnissen oder Fehlinterpretationen.

Aus den genannten Gründen wurde die Entscheidung getroffen, alle visuelle Notationen für Geschäftsprozesse zur Integration mit TOSCA-Topologien (mit zugehörigen Node Types und Relationship Types) zu akzeptieren. Da die vorgestellten Notationen keine signifikanten Gemeinsamkeiten zeigen, ist es nicht möglich, die grafischen Elemente für TOSCA-Konstrukte so zu entwerfen, dass immer eine eindeutige Trennung zwischen Topologie und Plänen gegeben ist (A25). Zur Erfüllung dieser Anforderung müssen im Rahmen des Entwurfs von Vino4TOSCA 2 daher zusätzliche visuelle Variablen definiert werden.

5 Vino4TOSCA 2

Vino4TOSCA 2 ermöglicht die grafische Modellierung von TOSCA Topology Templates, bestehend aus (1) Node Templates und (2) Relationship Templates, (3) Node Types, (4) Relationship Types und (5) Plänen eines TOSCA-Modells (A16, A17, A20). Für letztere Konstrukte definiert Vino4TOSCA 2 keine visuellen Elemente. Stattdessen sollen (entsprechend der TOSCA Spezifikation) bereits existierende visuelle Notationen für Geschäftsprozesse eingesetzt werden, insbesondere BPMN [TOS13]. Eine Beziehung zwischen einer Aktivität eines Plans und einer Schnittstelle eines Node Types bzw. Relationship Types, die zustande kommt, falls in der Aktivität eine Operation der Schnittstelle aufgerufen wird, kann visualisiert werden (A26).

Die visuelle Notation erlaubt die Definition von Profilen (A7, A10, A23). Wie bereits erwähnt, ist ein Profil eine Erweiterung bzw. Anpassung der Notation für bestimmte Bedürfnisse oder Fähigkeiten von Benutzern in einer bestimmten Domäne [BBK+12a]. Eine Aufgabe kann so effektiv und intuitiv gelöst werden kann [BBK+12a]. Profile beschränken die Variabilität von visuellen Variablen. Beispielsweise wäre es denkbar als Textfarbe lediglich schwarz zuzulassen. Basis-Notation und Profile bilden visuelle Dialekte [BBK+12a]. Im folgenden Abschnitt 5.1 werden die visuellen Variablen genannt, die durch Profile eingeschränkt werden dürfen.

5.1 Visuelle Variablen

Vino4TOSCA 2 verwendet die acht visuellen Variablen von Bertin [Ber83] zur Kodierung von Informationen (A7): (1) Horizontale und (2) vertikale Position, (3) Größe, (4) Helligkeit, (5) Muster, (6) Farbe, (7) Ausrichtung und (8) Form. Abbildung 5.1 veranschaulicht die Variablen bzw. zeigt beispielhaft, welche Änderungen durch diese hervorgerufen werden können.

Die Variablen werden in folgende Kategorien eingeordnet:

1. Variablen, die fest definiert sind.
2. Variablen, die durch Profile eingeschränkt werden dürfen.
3. Freie Variablen.

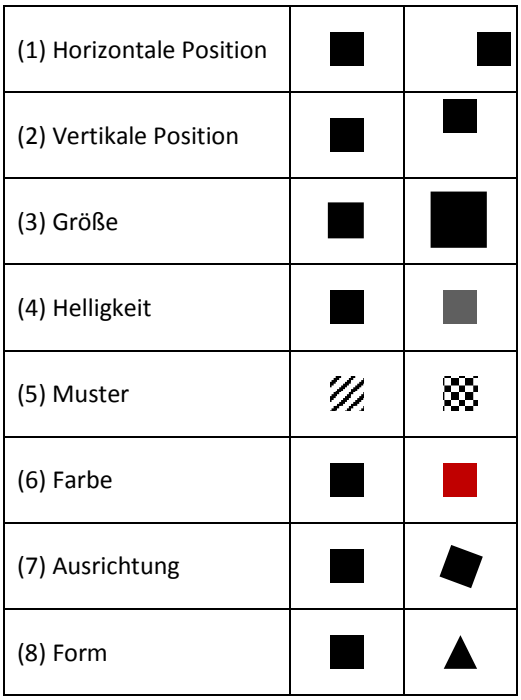

Abbildung 5.1: Die acht visuellen Variablen von Bertin.

In Vino4TOSCA 2 sind die Formen und deren Ausrichtung fest vorgegeben, dürfen also nicht verändert werden (Kategorie 1). Alle weiteren Variablen sind frei (Kategorie 3), sofern sie nicht durch Profile definiert sind. In einem Profil dürfen die Variablen Farbe, Helligkeit, Muster und Größe festgelegt werden (Kategorie 2), da mit diesen eine höhere kognitive Effektivität erzielt werden kann. Falls bspw. in einem Profil, das bei der Modellierung eines Diagramms angewendet werden soll, keine Farbe für die Linien einer Form definiert ist, so darf diese frei gewählt werden (visuelle Variabilität). Generell freie Variablen sind die horizontale und vertikale Position eines visuellen Elements (Kategorie 3). Diese dürfen folglich nicht durch ein Profil vorgegeben werden.

5.2 Visuelle Elemente

Für die meisten visuellen Elemente von Vino4TOSCA 2 wurden abgerundeten Formen gewählt. Diese Entscheidung wurde getroffen, da entsprechend einer Studie von Bar und Neta [BN06] abgerundete Formen von Personen bevorzugt werden (A23). Auch sind jene Formen leichter mit der Hand zu zeichnen (A22) [BBK+12a]. Dies ist wichtig, da Formen fest definierte Variablen sind (siehe Abschnitt 5.1).

Icons können u. a. zur Beschreibung von visuellen Elemente eingesetzt werden (A3, A7, A8, A23). Sie benötigen weniger Platz, weisen eine höhere Informationsdichte auf

als Text und werden bevorzugt gegenüber Formen. Weiterhin können sie schneller verarbeitet und erlernt werden. Icons müssen in einem visuellen Element an die linke obere Position gesetzt werden (Icon Area), da diese von Personen am meisten beachtet wird. [BBK+12a]

Die Schriftart von Text in einem visuellen Element ist nicht vorgegeben [BBK+12a]. Sie kann folglich frei gewählt werden, sofern keine Beschränkung durch ein Profil vorliegt. IDs müssen unterstrichen werden [BBK+12a]. Die ID des zugehörigen Typ-Elements (Referenz) ist nicht unterstrichen. Sie wird stattdessen durch vier spitze Klammern eingeschlossen (vgl. UML-Stereotyp). Die genannten Entwurfsentscheidungen tragen zu einer schnelleren Wiedererkennung bei (A3, A8, A21).

Die meisten Formen von Vino4TOSCA 2 können um eine „Additional Information Area“ erweitert werden, in der beliebige Informationen (Text und Bilder) hinterlegt werden können (A16, A18, A21) [BBK+12a]. Insbesondere dient dieses Feld zum Hinterlegen von Daten, die aus TOSCA-Konstrukten bzw. -Elementen stammen, die mit Vino4TOSCA 2 nicht direkt repräsentiert werden können [BBK+12a]. Hierunter fallen bspw. die Properties (Eigenschaften) eines Node Templates oder Relationship Templates.

In den folgenden Unterabschnitten soll nun auf die visuellen Elemente von Vino4TOSCA im Detail eingegangen werden. Jedes Element wird (insbesondere) durch die Form, den optionalen bzw. erforderlichen Inhalt, die Semantik und die visuelle Variabilität beschrieben. Wie bereits erwähnt, kann letztere ggf. durch ein Profil beschränkt werden.

5.2.1 Node Template Shape

Das Node Template Shape, dargestellt in Abbildung 5.2, repräsentiert ein Node Template eines Topology Templates. Es handelt um ein Rechteck (A3) mit abgerundeten Ecken, dargestellt mittels einer durchgezogenen Linie. Das Node Template kann durch ein (1) Icon in der Icon Area (links ausgerichtet), (2) den Namen oder (3) die ID des Node Template definiert werden (A3), wobei mindestens eine dieser drei Informationen angegeben werden muss. Falls mehrere textuelle Informationen dargestellt werden sollen, so muss die visuelle Reihenfolge aus Abbildung 5.2 eingehalten werden. Mittels einer optionalen Additional Information Area können weitere Informationen (Text oder Bilder) hinterlegt werden, die nicht direkt in der Form definiert werden können. Dabei handelt es sich um ein weiteres abgerundetes Rechteck, dargestellt mittels einer durchgezogenen Linie, das unter die Hauptform gesetzt wird. Es befindet sich hinter der Hauptform, sodass die oberen Ecken verdeckt sind. [BBK+12a]

Die Hauptform darf ein beliebiges Hintergrundbild enthalten. Icon Area und Text dürfen folglich nicht überdeckt werden. Die Linienfarbe der Hauptform und Additional Information Area darf beliebig gewählt werden, wohingegen der Linienstil nicht verändert werden darf (A2). [BBK+12a]

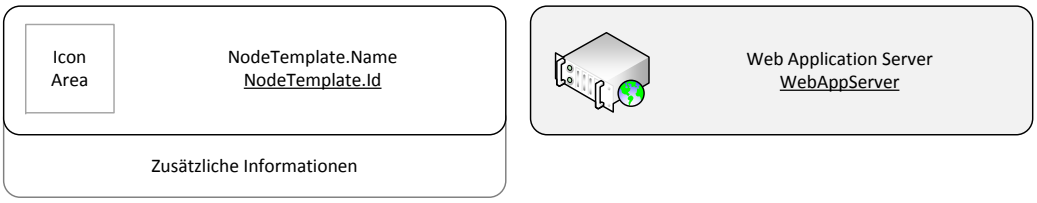

Abbildung 5.2: Vino4TOSCA 2 Node Template Shape mit Beispiel nach [BBK+12a].

5.2.2 Node Type Shape

Das Node Type Shape, dargestellt in Abbildung 5.3, repräsentiert ein Node Type als ein abgerundetes Rechteck mit durchgezogener Linie. Es kann eine beliebige Anzahl von Node Template Shapes (siehe Abschnitt 5.2.1) enthalten (auch keine), die Node Templates repräsentieren, welche das Node Type referenzieren (A4). Das Node Type kann durch ein (1) Icon in der Icon Area (links ausgerichtet), (2) den Namen oder (3) die ID des Node Type definiert werden (A3), wobei mindestens eine dieser drei Informationen angegeben werden muss. Die genannten Informationen sind durch eine durchgezogene Linie von den Node Template Shapes getrennt, auch falls keine Node Templates vorhanden sind. Falls mehrere textuelle Informationen dargestellt werden sollen, so muss die visuelle Reihenfolge aus Abbildung 5.3 eingehalten werden. Eine optionale Additional Information Area ermöglicht das Hinterlegen von weiteren Informationen. Diese ist ebenfalls ein abgerundetes Rechteck, dargestellt mittels einer durchgezogenen Linie. Es wird hinter die Hauptform gesetzt, sodass die oberen Ecken nicht sichtbar sind.

In der Hauptform darf ein beliebiges Hintergrundbild platziert werden. Icon, textuelle Node Type-Informationen und Node Templates Shapes dürfen folglich nicht überdeckt werden. Die Linienfarbe der Hauptform und Additional Information Area darf beliebig gewählt werden, wohingegen der Linienstil nicht verändert werden darf (A1, A2).

Die Darstellung eines Node Types ist optional. Ein Node Template Shape muss nicht im entsprechenden Node Type Shape dargestellt werden.

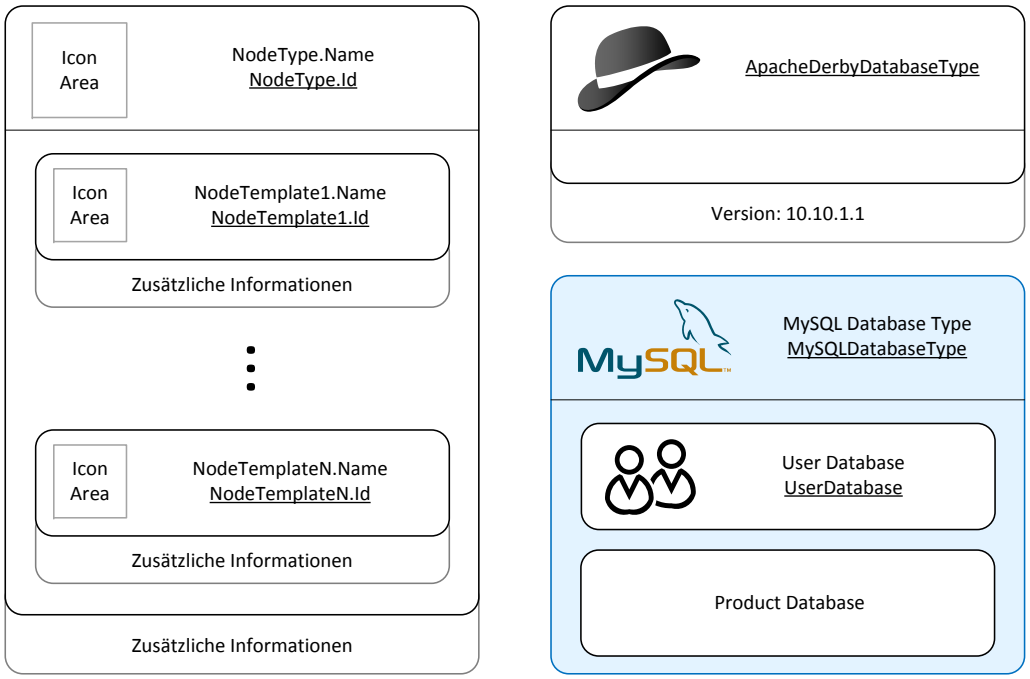

Abbildung 5.3: Vino4TOSCA 2 Node Type Shape mit Beispielen.

5.2.3 Relationship Template Shape

Das Relationship Template Shape, dargestellt in Abbildung 5.4, repräsentiert ein Relationship Template eines Topology Templates als eine Linie (beliebig ausgerichtet), die an jedem ihren Enden eine beliebige (kleine) Form besitzt, z. B. einen Pfeil (A3). Der Linienstil darf nicht gestrichelt sein (wie bei einem Visual Group Shape, siehe Abschnitt 5.2.9), im Übrigen kann er frei gewählt werden (A2). Die Linienfarbe kann beliebig gewählt werden. Ein Relationship Template Shape verbindet zwei Node Template Shapes (siehe Abschnitt 5.2.1) oder Collapsed Visual Group Shapes (siehe Abschnitt 5.2.9). In letzterem Falle zeigen Quelle und Ziel jeweils auf ein (nicht sichtbares) Element in der Gruppe. Das Relationship Template kann durch ein (1) Icon in der Icon Area (links ausgerichtet), (2) den Namen, (3) die ID des Relationship Template oder über (4) die ID des zugehörigen Relationship Type definiert werden (A3). Letztere Information wird durch vier spitze Klammern eingeschlossen (A3). Sie referenziert auf ein Relationship Type Shape (siehe Abschnitt 5.2.4) bzw. Relationship Type mit der angegebenen ID. Zumindest eine der genannten Informationen muss definiert werden. Falls mehrere textuelle Informationen angegeben werden, so muss die visuelle Reihenfolge aus Abbildung 5.4 eingehalten werden. Die Icon Area befindet sich über der Linie, falls diese horizontal ist, andernfalls (vertikal oder diagonal) auf einer beliebigen Seite. In einer optionalen Additional Information Area können weitere, beliebige Informationen hinterlegt werden. Dabei handelt es sich um ein

abgerundetes Rechteck, dargestellt mit einer durchgezogenen Linie, welche unter die Relationship Template-Linie gesetzt wird und diese berührt, falls die Linie horizontal ist. Andernfalls kann sich die Additional Information Area an einer beliebigen Seite befinden. Die Linienfarbe der Additional Information Area ist beliebig, wohingegen der Linienstil nicht verändert werden darf (A1, A2). [BBK+12a]

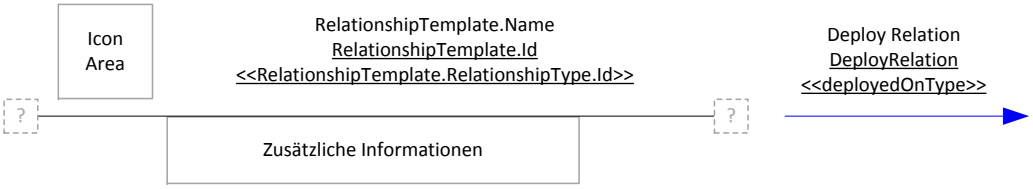

Abbildung 5.4: Vino4TOSCA 2 Relationship Template Shape mit Beispiel nach [BBK+12a].

5.2.4 Relationship Type Shape

Abbildung 5.5 zeigt ein Relationship Type Shape, das ein Relationship Type repräsentiert. Es handelt sich um ein abgerundetes Rechteck, deren rechte Seite um ein abgerundetes Dreieck erweitert wird, sodass ein Richtungspfeil (A3) entsteht (das Dreieck ist die Pfeilspitze). Das aufgesetzte Dreieck sollte möglichst gleichschenklig sein. Die rechte Seite des Rechtecks bzw. die Basis des gleichschenkligen Dreiecks ist unsichtbar. Der Linienstil ist durchgezogen. Das Relationship Type kann durch ein (1) Icon in der Icon Area (links ausgerichtet), (2) den Namen oder (3) die ID des Relationship Type definiert werden (A3), wobei mindestens eine dieser drei Informationen angegeben werden muss. Falls mehrere textuelle Informationen definiert werden, so muss die visuelle Reihenfolge aus Abbildung 5.5 eingehalten werden. Eine optionale Additional Information Area ermöglicht das Hinterlegen von weiteren Informationen. Diese ist ein abgerundetes Rechteck, dargestellt mittels einer durchgezogenen Linie, das unter die Hauptform (ohne Pfeilspitze) gesetzt wird. Es wird hinter die Hauptform gelegt, sodass die oberen Ecken überdeckt sind.

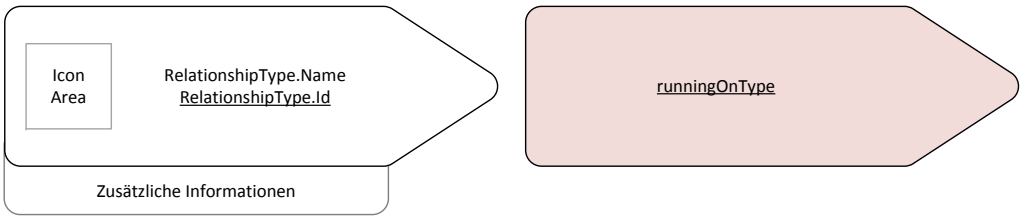

Abbildung 5.5: Vino4TOSCA 2 Relationship Type Shape mit Beispiel.

Die Hauptform kann ein beliebiges Hintergrundbild enthalten, d. h. Icon und Text dürfen nicht überdeckt werden. Die Linienfarbe der Hauptform und Additional Information Area darf beliebig gewählt werden. Eine Änderung des Linienstils hingegen ist nicht erlaubt (A1, A2).

Die Repräsentation eines Relationship Types ist optional. Ein Relationship Template Shape kann dargestellt werden, ohne das das zugehörige Relationship Type Shape existieren muss.

5.2.5 Node Type Interface Shape

Ein Node Type Interface Shape repräsentiert eine Schnittstelle eines Node Types. Die Schnittstelle wird durch einen nicht ausgefüllten Kreis dargestellt (A3), der mittels einer Linie (beliebig ausgerichtet) mit einem Node Type Shape (siehe Abschnitt 5.2.2) verbunden ist. Der Linienstil ist durchgezogen. Der Name der Schnittstelle muss angegeben werden. Ein Node Type Shape kann mit einer beliebigen Anzahl von Node Type Interface Shapes verbunden sein. In Abbildung 5.6 ist ein Node Type Interface Shape dargestellt, das mit einem Node Type Shape verbunden ist.

Die Linienfarbe eines Node Type Interface Shapes kann beliebig gewählt werden. Der Linienstil darf nicht verändert werden darf (A1, A2).

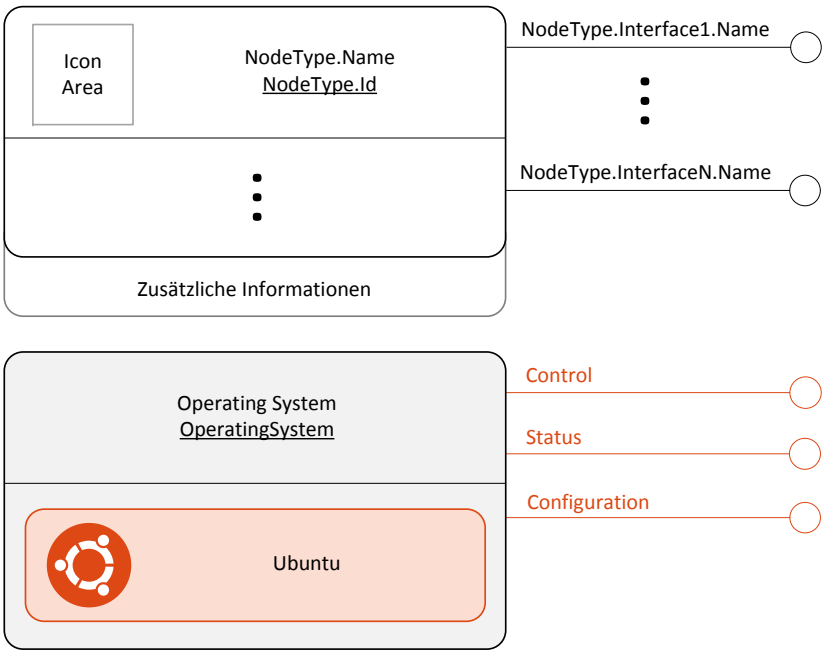

Abbildung 5.6: Vino4TOSCA 2 Node Type Interface Shape mit Beispiel.

5.2.6 Relationship Type Interface Shape

Die Quell- und Zielschnittstellen (Source Interfaces und Target Interfaces) eines Relationship Types können durch Relationship Type Interface Shapes repräsentiert werden. Ein Relationship Type Interface Shape entspricht einem Node Type Interface Shape (siehe Abschnitt 5.2.5) bis auf die folgenden Eigenschaften: Der Kreis, der die Schnittstelle darstellt, muss ein „S“ (Source) enthalten, falls es sich um eine Quellschnittstelle handelt (A3, A8). Eine Zielschnittstelle wird durch ein „T“ (Target) visualisiert (A3, A8). Ein Relationship Type Shape kann mit einer beliebigen Anzahl von Relationship Type Interface Shapes verbunden sein. Abbildung 5.7 zeigt ein Relationship Type Interface Shape, das mit einem Relationship Type Shape verbunden ist.

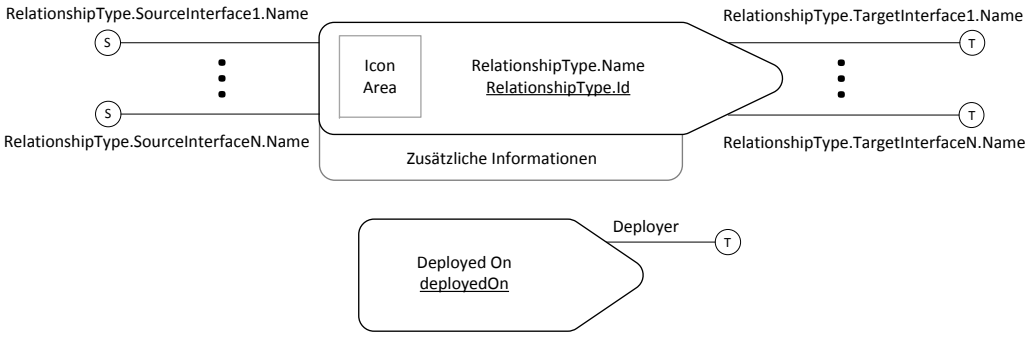

Abbildung 5.7: Vino4TOSCA 2 Relationship Type Interface Shape mit Beispiel.

5.2.7 Plan Shape

Das Plan Shape, dargestellt in Abbildung 5.8, repräsentiert einen Plan. Es handelt sich um ein abgerundetes Rechteck, unterteilt in zwei Bereiche, die durch eine Linie getrennt werden. Der Linienstil ist durchgehend. Im oberen Bereich kann ein (1) Icon in der Icon Area (links ausgerichtet), (2) der Name, (3) die ID, der (4) Typ (URI)[11] oder die (5) Sprache des Plans (URI)[12] definiert werden (A3). Eine der ersten drei Informationen müssen mindestens angegeben werden. Im unteren Bereich wird der Plan selbst dargestellt. Hierzu kann eine beliebige visuelle Notation für die entsprechende Prozesssprache (in unveränderter Form) eingesetzt werden (A24). Eine standardisierte Notation sollte bevorzugt werden. Die Semantik des Prozesses darf nicht verändert werden (A28). Weitere Informationen können in einer optionalen Additional Information Area hinterlegt werden. Diese ist ein abgerundetes Rechteck,

[11] Z. B. Build Plan (instanziiert einen Service):
`http://docs.oasis-open.org/tosca/ns/2011/12/PlanTypes/BuildPlan`
[12] Z. B. BPMN 2.0-Plan: `http://www.omg.org/spec/BPMN/20100524/MODEL`

bestehend aus einer durchgehenden Linie, das unter die Hauptform gesetzt wird, sodass die oberen Ecken nicht sichtbar sind.

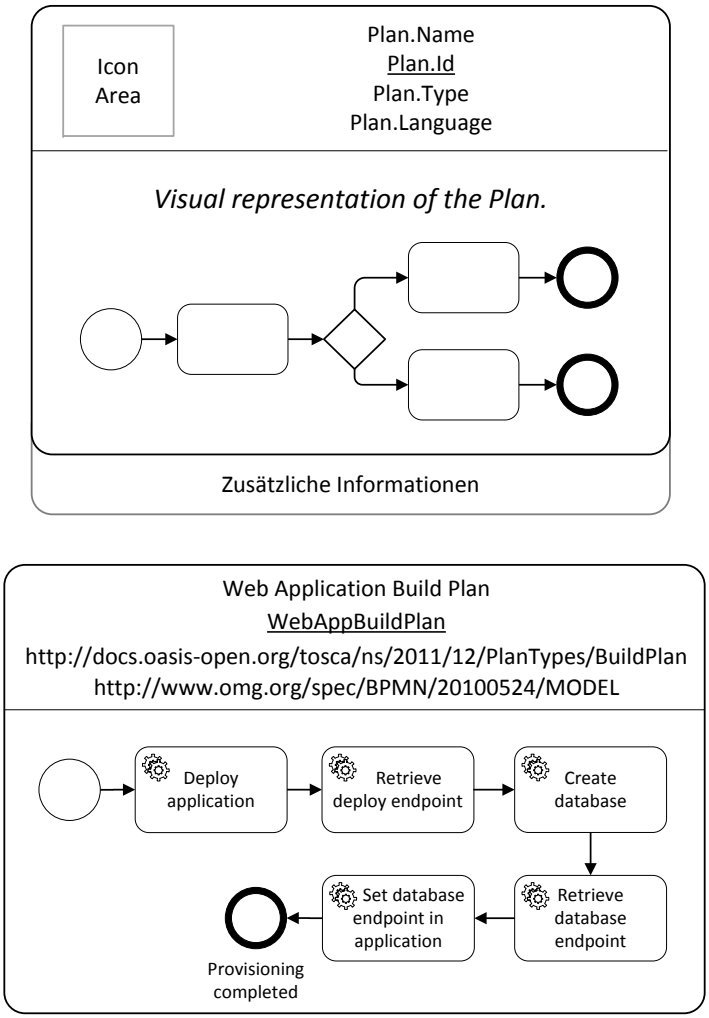

Abbildung 5.8: Vino4TOSCA 2 Plan Shape mit Beispiel.

Der obere Bereich der Hauptform darf ein beliebiges Hintergrundbild enthalten. Icon und Text dürfen folglich nicht überdeckt werden. Die Linienfarbe der Hauptform und Additional Information Area kann beliebig gewählt werden, wohingegen der Linienstil nicht verändert werden darf (A2). Die visuelle Repräsentation des Prozesses unterliegt den Vorgaben (visuelle Syntax) der eingesetzten visuellen Notation und kann nicht durch ein Profil beschränkt werden (A24).

Die Pläne bzw. Plan Shapes werden durch eine doppelte durchgehende Linie (beliebig ausgerichtet) von der Repräsentation des Topology Templates, der Node Types und Relationship Types getrennt (siehe Abbildung 5.9; A14, A25). Die Linienfarbe kann frei gewählt werden, der Linienstil hingegen darf nicht verändert werden (A1, A2).

Wird ein Service Template (mit Node Types und Relationship Types) mit Vino4TOSCA 2 repräsentiert, so obliegt es dem Modellierer, welche Pläne dargestellt werden sollen.

5.2.8 Plan Invoke Operation Shape

Eine Verbindung zwischen einer Aktivität und einer Schnittstelle eines Node Type bzw. Relationship Type, die zustande kommt, falls in der Aktivität eine Operation der Schnittstelle aufgerufen wird, kann durch ein Plan Invoke Operation Shape repräsentiert werden. Das Plan Invoke Operation Shape ist ein offener Halbkreis, der mittels einer durchgezogenen Linie (beliebig ausgerichtet) mit der Aktivität des Plans bzw. Prozesses (in einem Plan Shape, siehe Abschnitt 5.2.7) verbunden ist, in welchem die Operation aufgerufen wird (A3). Der Name der Operation kann optional angegeben werden. Sieht die verwendete visuelle Prozessnotation ein anderes Element zum Aufruf von Operationen bzw. zum Ausführen von Aufgaben vor, so wird das Plan Invoke Shape mit diesem Element verbunden. Der Halbkreis umschließt den Kreis, der die Schnittstelle repräsentiert (siehe Abschnitt 5.2.5 bzw. 5.2.6), welche die aufzurufende Operation definiert (A4). Existiert der Halbkreis bereits, da ein weiteres Plan Invoke Shape einer Aktivität dargestellt ist, das eine Operation aus selbiger Schnittstelle aufruft, so wird lediglich die Linie zwischen Aktivität und Halbkreis gezeichnet. Abbildung 5.9 zeigt Plan Invoke Operation Shapes zwischen Schnittstellen von Node Types und Aktivitäten eines Plans.

Die Linienfarbe eines Plan Invoke Operation Shape kann frei gewählt werden, wohingegen der Linienstil nicht verändert werden darf (A1, A2).

5.2.9 Visual Group Shapes

Die Visual Group Shapes setzen sich aus dem Expanded und Collapsed Visual Group Shape zusammen. Beide sind in Abbildung 5.10 dargestellt.

Mit dem Expanded Visual Group Shape können Elemente visuell gruppiert werden (A5, A19). Die Form kann beliebig gewählt werden und muss die entsprechenden Elemente enthalten (A4). Der Linienstil ist gestrichelt. Im oberen Bereich der Form, der durch eine gestrichelte Linie getrennt ist, kann (1) ein Icon in der Icon Area (links ausgerichtet), (2) ein Name oder (3) eine ID definiert werden (A3). Eine der genannten Informationen muss mindestens angegeben werden. Falls mehrere textuelle Informationen definiert werden, so muss die visuelle Reihenfolge aus Abbildung 5.10 eingehalten werden. Auf die visuellen Elemente eines Plans (in einem Plan Shape, siehe Abschnitt 5.2.7) darf das Expanded Visual Group Shape nicht angewendet werden (A24). [BBK+12a]

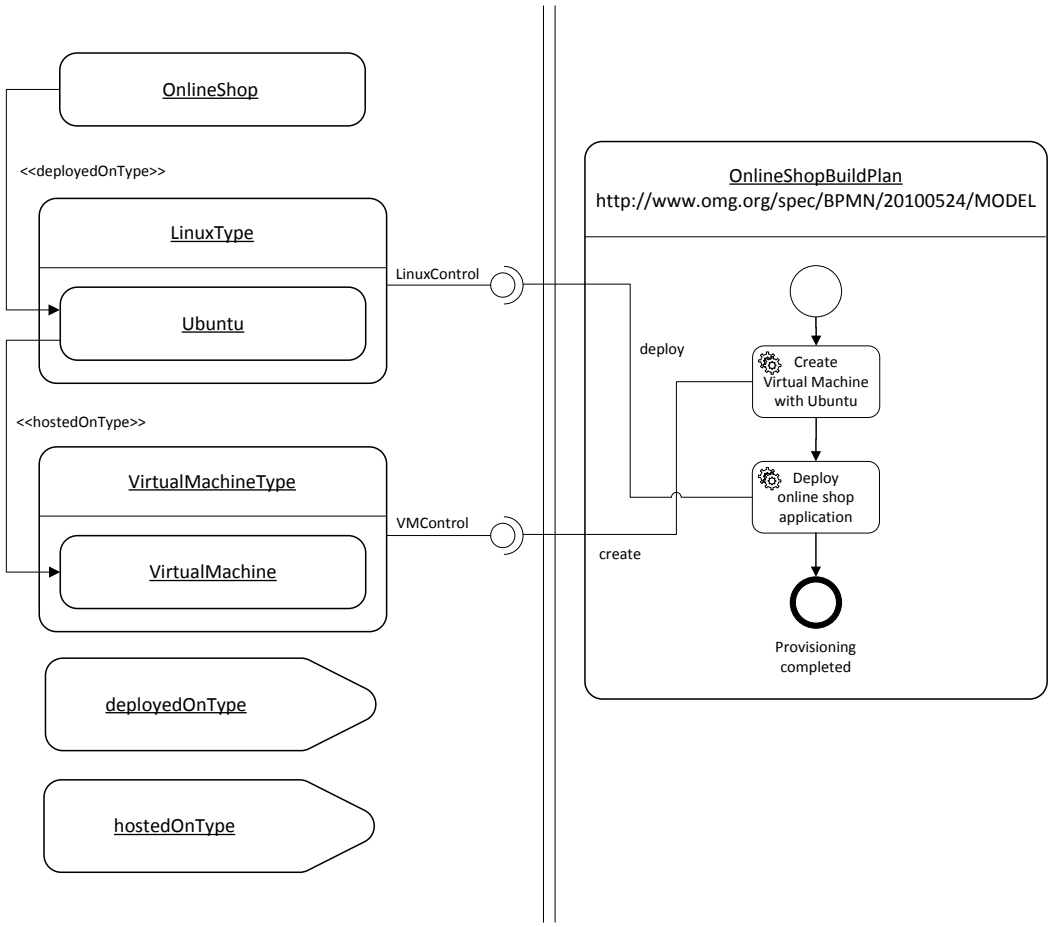

Abbildung 5.9: Vino4TOSCA 2 Plan Invoke Operation Shape Beispiel.

Das Collapsed Visual Group Shape ist ein Oval, das eine beliebige Anzahl von visuellen Elemente repräsentiert bzw. diese abstrahiert (A5, A19). Der Linienstil ist gestrichelt. Ein kleines Quadrat, dass an der Unterseite der Form mittig angeordnet ist und ein Plus-Zeichen enthält, soll die Abstraktion bzw. den geschlossenen Zustand symbolisieren (A3). Die Form kann (1) ein Icon in Icon Area (links ausgerichtet), (2) ein Name oder (3) eine ID enthalten, wobei zumindest eine der genannten Daten angegeben werden muss (A3). Das Collapsed Visual Group Shape darf nicht auf die visuellen Elemente eines Plans (in einem Plan Shape, siehe Abschnitt 5.2.7) angewendet werden außer die visuelle Notation des Plans sieht keine Möglichkeiten zur Abstraktion vor (A27). [BBK+12a]

Die Linienfarbe der Visual Group Shapes kann frei gewählt werden. Eine Änderung des Linienstils hingegen ist nicht erlaubt (A1, A2). Auch darf kein Hintergrundbild gesetzt werden. [BBK+12a]

Die Visual Group Shapes können auch zur Integration von anderen Diagrammen eingesetzt werden (A6). Das integrierte Diagramm wird durch das Icon, die ID oder

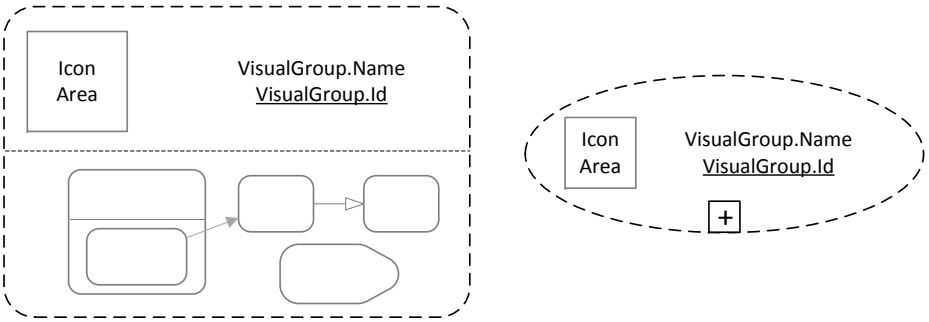

Abbildung 5.10: Vino4TOSCA 2 Expanded und Collapsed Visual Group Shapes nach [BBK+12a].

den Namen identifiziert. Das zugrunde liegende TOSCA-Modell wird durch Visual Group Shapes grundsätzlich nicht verändert. [BBK+12a]

5.2.10 Visual Relationship Group Shapes

Zu den Visual Relationship Group Shapes gehören das Expanded und das Collapsed Relationship Group Shape, die in Abbildung 5.11 veranschaulicht werden. [BBK+12a]

Das Expanded Visual Relationship Group Shape dient der visuellen Gruppierung von Relationship Template Shapes (siehe Abschnitt 5.2.3) die Node Template Shapes bzw. Visual Group Shapes verbinden (A5, A19). Es besteht aus zwei gestrichelten Linien, die eine beliebige Anzahl von Relationship Template Shapes enthalten können, mindestens jedoch zwei (A4). Das Element kann oberhalb der Linien durch ein (1) Icon in der Icon Area (links ausgerichtet), (2) einen Namen oder (3) eine ID beschrieben werden, wobei mindestens eine der genannten Informationen angegeben werden muss (A3). Falls mehrere textuelle Informationen definiert werden, so muss die visuelle Reihenfolge aus Abbildung 5.11 eingehalten werden. [BBK+12a]

Mit dem Collapsed Visual Relationship Group Shape können Relationship Template Shapes abstrahiert werden (A5, A19). Es besteht aus einer gestrichelten Linie, die zwei Relationship Template Shapes bzw. Visual Group Shapes verbinden. In der Mitte der Linie befindet sich ein kleines Quadrat mit einem Plus-Zeichen, das verdeutlichen soll, dass die Linie für eine Menge von (nicht sichtbaren) Relationship Template Shapes steht (A3). Über der Linie müssen zumindest eine der folgenden Informationen definiert werden: Ein (1) Icon in der Icon Area (links ausgerichtet), (2) ein Name oder (3) eine ID (A3). Werden mehrere textuelle Informationen angegeben, so muss die visuelle Reihenfolge aus Abbildung 5.11 beachtet werden. [BBK+12a]

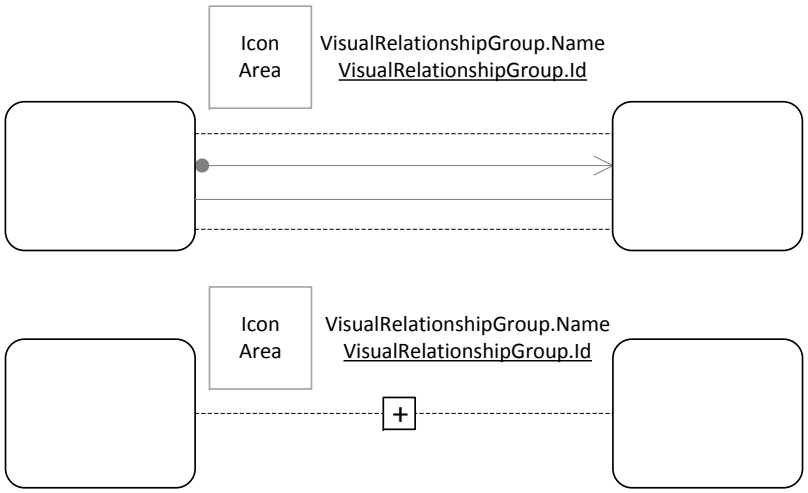

Abbildung 5.11: Vino4TOSCA 2 Expanded und Collapsed Visual Relationship Group Shapes nach [BBK+12a].

Die Linienfarbe der Visual Relationship Group Shapes kann beliebig gewählt werden. Der Linienstil dagegen muss den Vorgaben entsprechen (A1, A2). Ein Hintergrundbild ist nicht erlaubt. [BBK+12a]

5.2.11 Node Template Instanzen

Ein Node Template besitzt zwei optionale Attribute, in denen die minimale und maximale Zahl an zulässigen Instanzen des Node Templates definiert werden kann. Zur Repräsentation dieser Angaben muss eine zweite, durchgehende Linie teilweise um das entsprechende Node Template Shape (siehe Abschnitt 5.2.1) gezeichnet werden[13] (A3) und der min-Wert wird auf die linke Seite, der max-Wert auf die rechte Seite über die Form geschrieben werden. Abbildung 5.12 zeigt ein Node Template Shape eines Node Types, in dem die genannten Attribute definiert sind. [BBK+12a]

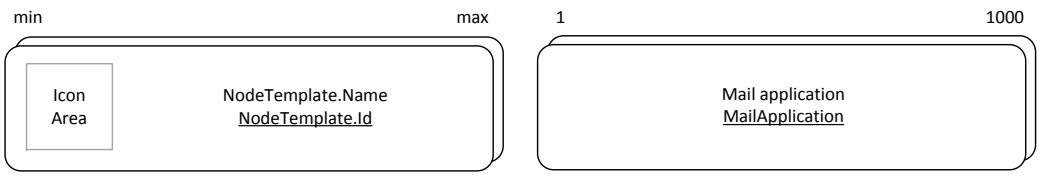

Abbildung 5.12: Vino4TOSCA 2 Node Template Instanzen mit Beispiel nach [BBK+12a].

[13] Dies soll die mögliche Existenz von mehreren Instanzen verdeutlichen.

5.3 Beispiele

Abbildung 5.9 zeigt ein Vino4TOSCA 2-Diagramm eines Online-Shops. Die Anwendung selbst wird durch das Node Template „OnlineShop“ repräsentiert und läuft auf Ubuntu (gleichnamiges Node Template). Das Node Type „LinuxType“ des Betriebssystems definiert eine Schnittstelle „LinuxControl“, die Operationen bereitstellt, mit denen u. a. Anwendungen installiert werden können. Das Betriebssystem läuft auf einer virtuellen Maschine (VM), die durch das Node Template „VirtualMachine“ dargestellt ist. Dieses referenziert den Node Type „VirtualMachineType“ mit der Schnittstelle „VMControl“, die Operationen definiert, mit denen virtuelle Maschinen verwaltet (z. B. erstellt) werden können.
Der BPMN-Plan „OnlineShopBuildPlan“, dargestellt auf der rechten Seite, provisioniert den Online-Shop. Zunächst wird eine Virtuelle Maschine erzeugt, auf der bereits Ubuntu vorinstalliert ist. Abschließend wird die Webanwendung installiert.

Abbildung 5.13 zeigt das Diagramm einer Mailanwendung. Die eigentliche Anwendung wird durch das Node Template „Mail Application“ repräsentiert und ist über mehrere Relationship Templates, die in der Abbildung mittels einem Visual Relationship Group Shape (siehe Abschnitt 5.2.10) abstrahiert dargestellt sind, mit mehreren Datenbanken verbunden. Die Datenbanken und deren Infrastrukturen (Server mit Betriebssystem) sind ebenfalls durch ein Visual Group Shape (siehe Abschnitt 5.2.9) abstrahiert. Die Mailanwendung läuft auf einem Apache Webserver, repräsentiert durch das Node Template „MailAppWebserver“. Der zugehörige Node Type „ApacheWebserverType“ definiert die Schnittstellen „ModuleManagement“ und „AppManagement“, mit deren Operationen Module bzw. Anwendungen auf dem Webserver verwaltet (z. B. installiert) werden können. Windows Server, repräsentiert durch das gleichnamige Node Template, ist das Betriebssystem, auf welchem der Webserver installiert ist. Das zugehörige Typ-Element „WindowsServerType“ stellt eine Schnittstelle „AppManagement“ bereit, deren Operationen zur Verwaltung der Anwendungen auf dem Betriebssystem eingesetzt werden können. Windows Server läuft auf einem Cloud Server von Rackspace, der durch das Node Template „MailAppServer“ repräsentiert ist. Das Anlegen eines entsprechenden Servers bzw. deren Verwaltung erfolgt über Operationen der Schnittstelle „ServerManagement“, die vom zugehörigen Typ-Element „RackspaceCloudServerType“ bereitgestellt ist.
Auf der rechten Seite ist der BPMN-Plan „MailAppBuildPlan“ dargestellt, mit dem die Mailanwendung bereitgestellt bzw. eine Instanz der Topologie erzeugt werden kann. Zunächst wird ein Cloud Server mit bereits vorinstalliertem Windows Server auf Rackspace erzeugt. Anschließend werden nacheinander der Apache Webserver, das PHP Modul und die eigentliche Mailanwendung installiert. Parallel zu den bisher genannten Aktivitäten werden weitere Server erzeugt, auf denen jeweils ein

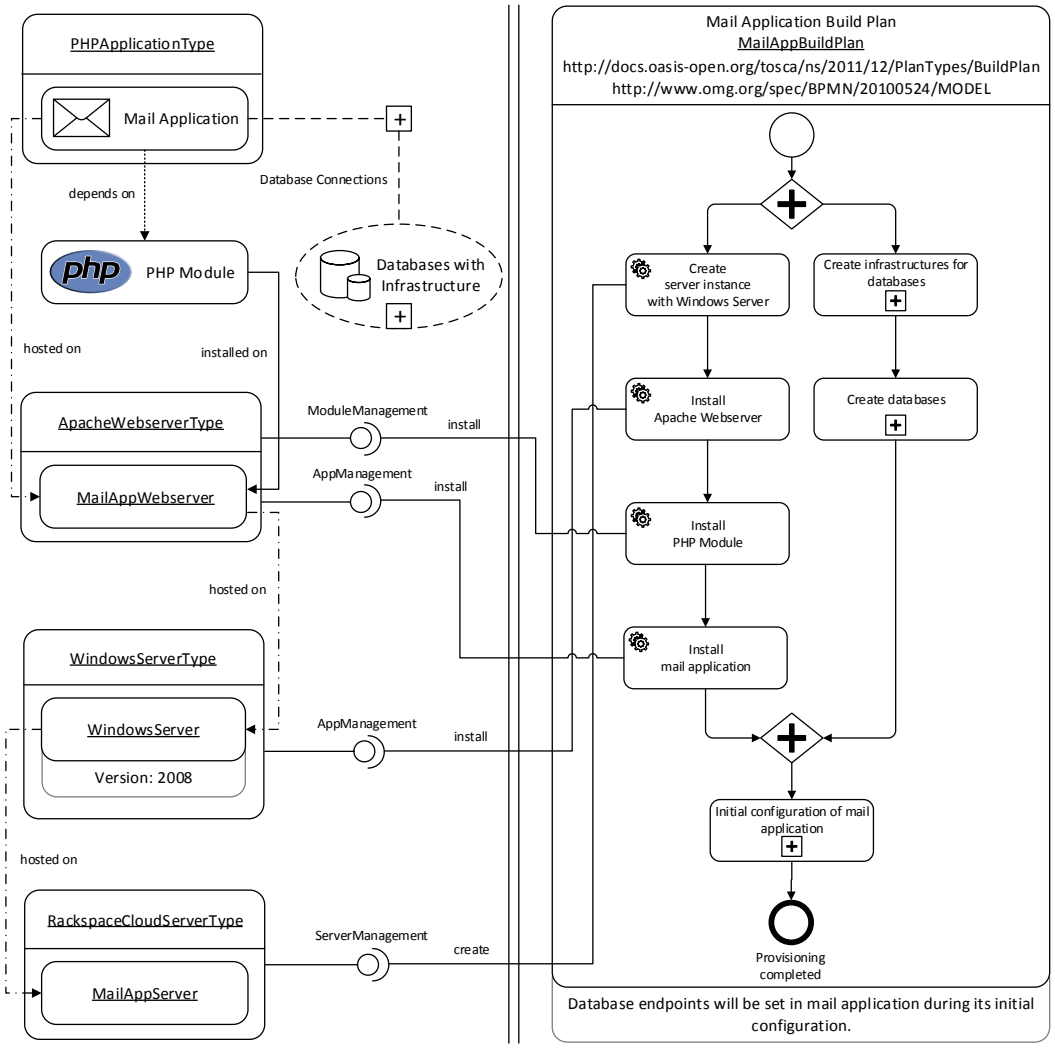

Abbildung 5.13: Vino4TOSCA 2-Diagramm zu einem TOSCA-Modell einer Mailanwendung.

Betriebssystem installiert und schließlich eine Datenbank angelegt wird. Nach der Installation und Erstellung der Datenbanken erfolgt abschließend die Konfiguration der Mailanwendung. Dabei werden u. a. die Endpunkte der Datenbanken in der Mailanwendung gesetzt.

6 Zusammenfassung und Ausblick

Das Ziel dieser Fachstudie war es, eine visuelle Notation zur Darstellung von Topologien und Managementplänen zu erstellen. Hierfür wurden zuerst verschiedene Anforderungen aufgestellt, welche als Grundlagen zur Erstellung eines effektiv einsetzbaren Diagramms dienen. Die Anforderungen wurden dabei in visuelle Anforderungen zur Gestaltung von Diagrammen, TOSCA-spezifische Anforderungen, Usability- und User Experience-Anforderungen sowie Anforderungen bezüglich der Integration von Topologien und Geschäftsprozessen kategorisiert. Weiterhin wurden bestehende Notationen von Anwendungstopologien sowie Geschäftsprozessen betrachtet und anhand der aufgestellten Anforderungen bewertet. Schließlich wurde eine auf Vino4TOSCA aufbauende Notation – Vino4TOSCA 2 genannt – erarbeitet, welche sowohl die grafische Darstellung von Topology Templates als auch von Plänen sowie deren Beziehungen ermöglicht. Die entworfene Notation definiert im Gegensatz zu Vino4TOSCA zusätzlich Formen für Node Types und Relationship Types, deren Schnittstellen, Plänen und den Aufruf von Operationen (in Plänen), die in den genannten Schnittstellen definiert worden sind. An den bereits existierenden Formen für Node Templates, Relationship Templates, Visual Groups, Visual Relationship Groups und Instanzen von Node Templates wurden Anpassungen vorgenommen. Die Expanded und Collapsed Group Template Shapes wurden nicht übernommen, da Group Templates mittlerweile aus der TOSCA Spezifikation entfernt worden sind. Zukünftig wäre es denkbar, dass Formen für weitere TOSCA-Elemente wie z. B. Requirements entworfen werden, um den Informationsgehalt der Notation weiter zu steigern.

Literaturverzeichnis

[Ale64] C. Alexander. *Notes on the Synthesis of Form.* Harvard University Press, 1964.

[AMA] AMADEE GmbH. UML-Aktivitätsdiagramm. URL http://www.amadee.com/docs/de/uml-aktivitaetsdiagramm.html?ContextID=140. Abgerufen am 2013-05-06.

[BBK+12a] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, D. Schumm. Vino4TOSCA: A Visual Notation for Application Topologies Based on TOSCA. In *OTM 2012, Part I*, Band 7565 von *Lecture Notes in Computer Science (LNCS)*, S. 416–424. Springer-Verlag, 2012.

[BBK+12b] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, D. Schumm. Vino4TOSCA Website, 2012. URL http://www.vino4tosca.org.

[Ber83] J. Bertin. *Semiology of graphics.* University of Wisconsin Press, 1983.

[BN06] M. Bar, M. Neta. Humans Prefer Curved Visual Objects. *Psychological Science*, 17(8):645–648, 2006. doi:10.1111/j.1467-9280.2006.01759.x. URL http://pss.sagepub.com/content/17/8/645.abstract.

[Bun] Bundesamt für Sicherheit in der Informationstechnik. Cloud Computing Grundlagen. URL https://www.bsi.bund.de/DE/Themen/CloudComputing/Grundlagen/Grundlagen_node.html. Abgerufen am 2013-04-17.

[Che04] P. P. Chen. Entity Relationship Modellierung, 2004. URL http://ebus.informatik.uni-leipzig.de/www/media/lehre/seminar-pioniere04/sem04swp-hartmann-vortrag.pdf.

[Enz] Enzyklopädie der Wirtschaftsinformatik. EPK. URL http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/Geschaftsprozessmodellierung/EPK. Abgerufen am 2013-05-14.

[FMC] FMC Consortium. Fundamental Modeling Concepts – Quick Introduction. URL http://www.fmc-modeling.org/quick-intro. Abgerufen am 2013-05-05.

[Fra11] Fraunhofer FIRST. The Generic Workflow Description Language Toolbox, 2011. URL http://gridworkflow.org/kwfgrid/gworkflowdl/docs.

[Fri02] G. Friese. Wirtschaftsinformatik I Übung 2 – EPK-Modellierung, Techniken zum Systementwurf, 2002. URL http://www.friese-total.de/uni/bwl/wi/wi-2.shtml. Abgerufen am 2013-05-02.

[Gab] Gabler Wirtschaftslexikon. Struktogramm. URL http://wirtschaftslexikon.gabler.de/Archiv/76260/struktogramm-v7.html.

[Ges12] Geschäftsprozessmanagement Blog. Modellierung von Geschäftsprozessen mit der Ereignisgesteuerten Prozesskette (EPK), 2012. URL http://de.processorientation.com/?p=668. Abgerufen am 2013-05-02.

[GMW00] D. Garlan, R. T. Monroe, D. Wile. Foundations of component-based systems. Kapitel Acme: architectural description of component-based systems, S. 47–67. Cambridge University Press, New York, NY, USA, 2000. URL http://dl.acm.org/citation.cfm?id=336431.336437.

[IBM06] IBM. WebSphere Integration Developer – Service Component Architecture, 2006. URL http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.prodovr.doc/topics/csrvcomparch.html. Abgerufen am 2013-07-11.

[Inf13] Informatik Forum Simon GmbH. UML – Unified Modeling Language, 2013. URL http://www.infforum.de/themen/anwendungsentwicklung/thema_SE-methode_uml.htm. Abgerufen am 2013-05-02.

[KB04] A. Keller, R. Badonnel. Automating the Provisioning of Application Services with the BPEL4WS Workflow Language. In A. Sahai, F. Wu, Herausgeber, *Utility Computing*, Band 3278 von *Lecture Notes in Computer Science*, S. 15–27. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30184-4_2. URL http://dx.doi.org/10.1007/978-3-540-30184-4_2.

[Ley12] F. Leymann. Architectural Diagrams & Styles (Foliensatz), 2012.

[Loh05] N. Lohmann. DBS I – Grundlagen von Datenbanksystemen, 2005. URL http://www2.informatik.hu-berlin.de/~blunk/pdf/dbs1_nlohmann.pdf.

[LR09] B. Lahres, G. Rayman. Objektorientierte Programmierung, 2009. URL http://openbook.galileocomputing.de/oop/oop_kapitel_02_001.htm. Abgerufen am 2013-05-02.

[LS78] J. Ludewig, W. Streng. *Überblick und Vergleich verschiedener Mittel für die Spezifikation und den Entwurf von Software*. Kernforschungszentrum, Karlsruhe, 1978.

[LS87] J. H. Larkin, H. A. Simon. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1):65–100, 1987. doi:10.1111/j.1551-6708.1987.tb00863.x. URL http://dx.doi.org/10.1111/j.1551-6708.1987.tb00863.x.

[MÏ2] M. Müller. *Sichere Nutzung von Cloud-Storage in Datenbanken*. Diplomarbeit, Technische Universität Dresden, 2012. URL http://www.rn.inf.tu-dresden.de/uploads/Studentische_Arbeiten/Diplomarbeit_M%C3%BCller_Mario.pdf. Abgerufen am 2013-04-22.

[MG11] P. M. Mell, T. Grance. SP 800-145. The NIST Definition of Cloud Computing. Technischer Bericht, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011. URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Abgerufen am 2013-04-21.

[Moo09] D. Moody. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Trans. Softw. Eng.*, 35(6):756–779, 2009. doi:10.1109/TSE.2009.67. URL http://dx.doi.org/10.1109/TSE.2009.67.

[NC99] J. C. Nordbotten, M. E. Crosby. The effect of graphic style on data model interpretation. *Information Systems Journal*, 9(2):139–155, 1999. doi:10.1046/j.1365-2575.1999.00052.x. URL http://dx.doi.org/10.1046/j.1365-2575.1999.00052.x.

[Nor02] D. Norman. Emotion & design: attractive things work better. *interactions*, 9(4):36–42, 2002. doi:10.1145/543434.543435. URL http://doi.acm.org/10.1145/543434.543435.

[PQ06] M. Petre, E. de Quincey. A gentle overview of software visualisation. *Psychology of Programming Interest Group (PPIG)*, 2006. URL http://www.ppig.org/newsletters/2006-09/1-overview-swviz.pdf.

[RÖ9] B. Rücker. Bauen wir uns eine BPMN 2.0 Engine, 2009. URL http://www.bpm-guide.de/2009/08/02/bauen-wir-uns-eine-bpmn-20-engine. Abgerufen am 2013-05-02.

[re-] re-wissen.de – Fraunhofer IESE. EPK-Modellierung. URL http://www.re-wissen.de/opencms/Wissen/Techniken/EPK-Modellierung.html. Abgerufen am 2013-05-14.

[RJB99] J. Rumbaugh, I. Jacobson, G. Booch. *The Unified Modeling Language reference manual*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1999.

[sof] software-kompetenz.de – Fraunhofer IESE. Petri-Netze. URL http://www.software-kompetenz.de/?16617. Abgerufen am 2013-05-02.

[TOS13] TOSCA Technical Committee. Topology and Orchestration Specification for Cloud Applications (TOSCA) – Committee Specification 01. Technischer Bericht, OASIS, 2013. URL http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf. Abgerufen am 2013-04-26.

[Wik13a] Wikipedia. Entity-Relationship-Modell – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Entity-Relationship-Modell&oldid=117772832. Abgerufen am 2013-05-05.

[Wik13b] Wikipedia. Ereignisgesteuerte Prozesskette – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Ereignisgesteuerte_Prozesskette&oldid=116791035. Abgerufen am 2013-05-10.

[Wik13c] Wikipedia. Fundamental Modeling Concepts – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Fundamental_Modeling_Concepts&oldid=116170221. Abgerufen am 2013-05-05.

[Wik13d] Wikipedia. Generic Workflow Description Language – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Generic_Workflow_Description_Language&oldid=114950229. Abgerufen am 2013-05-01.

[Wik13e] Wikipedia. HIPO-Diagramm – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=HIPO-Diagramm&oldid=116464651. Abgerufen am 2013-04-06.

[Wik13f] Wikipedia. Komponentendiagramm – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Komponentendiagramm&oldid=116940785. Abgerufen am 2013-07-03.

[Wik13g] Wikipedia. Nassi-Shneiderman-Diagramm – Wikipedia, Die freie Enzyklopädie, 2013. URL http://de.wikipedia.org/w/index.php?title=Nassi-Shneiderman-Diagramm&oldid=118156563. Abgerufen am 2013-05-04.

[Win13] WinfWiki. Dokumentationsanforderungen im IT-Projektmanagement, Abschnitt Datenflussdiagramm – WinfWiki, 2013. URL http://winfwiki.wi-fom.de/index.php/Dokumentationsanforderungen_im_IT-Projektmanagement#Datenflussdiagramm. Abgerufen am 2013-05-02.

[www] www.BWL-Betriebswirtschaft.de. Das Flussdiagramm. URL http://www.bwl-betriebswirtschaft.de/flussdiagramm.html. Abgerufen am 2013-05-02.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift — Lars-Alexander Albrecht

Ort, Datum, Unterschrift — Rene Trefft

Ort, Datum, Unterschrift — Michael Zimmermann