Beschreibung, Verarbeitung und Überprüfung clientseitiger Policies für vertrauenswürdige Cloud-Anwendungen

Dissertation

zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl.- Ing. Jörg Kebbedies
geboren am 26. Dezember 1958 in Stalinstadt, heute Eisenhüttenstadt

Betreuender Hochschullehrer: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Fachreferent: Prof. Dr. Susanne Strahringer

Externer Gutachter: Prof. Dr. Marian Margraf, Freie Universität Berlin

Tag der Verteidigung: 07. Dezember 2017
Tag der Einreichung : 24. Mai 2017

Danksagung

Die Idee zu dieser Arbeit entwickelte sich aus meinen langjährigen Erfahrungen als Berater und Architekt im Bereich der IT-Sicherheit. Das Vorhaben gab mir neben meinen beruflichen Verpflichtungen den Rahmen und Freiraum, dem Thema Vertrauen im Zeitalter zunehmender Digitalisierung einen größeren Schwerpunkt zu widmen.
Dem Engagement und Interesse von Prof. Alexander Schill ist es zu verdanken, dass sich nach ersten gedanklichen Fragmenten ein konstruktives Ringen entwickelte, um für die gesetzte Zielstellung eine angemessene und vertiefende Aufgabenstellung herauszuarbeiten. Ich danke Prof. Schill für das entgegengebrachte Vertrauen. Seine richtungsprägenden Gespräche wirkten auf mich motivierend und seine konsequente moderierende Art half mir, sowohl die inhaltlich vertiefende Ausrichtung als auch die zeitliche Orientierung in Einklang zu bringen.
Den Mitarbeitern an der Professur Rechnernetze der Technischen Universität Dresden möchte ich für ihre freundliche Aufnahme im Rahmen ihrer Klausurtagungen danken, wo ich mich thematisch angezogen und persönlich integriert fühlte. Die lektorische und fachliche Zusammenarbeit mit Frau Dr. Iris Braun, Herrn Dr. Tenshi Hara und Herrn Dr. Josef Spillner entwickelte meine Detailsicht, die in Bezug auf Kommunikation, Präsentation und Ablauforganisation für meine Arbeit notwendig war. Vielen Dank für die richtungsweisende Begleitung in eine anfangs noch nicht vertraute Vorgehensweise.
Ich möchte an dieser Stelle auch meinen Kollegen der Fa. secunet AG einen Dank aussprechen, die trotz anspruchsvoller beruflicher Verpflichtungen noch Zeit fanden, durch einen berufsbezogenen Diskurs meinen Konzeptansätzen eine fachliche Bewertung entgegenzusetzen. Die erfahrungsgeprägten Gespräche mit Dr. Michael Sobirey gaben mir stets neue Impulse, um meine Herangehensweise und Sicht auf die eigene Arbeit zu schärfen.
Ein großer Dank gilt auch den Lektorinnen Frau Katja Völkel und Frau Annushka Sonek-Wienert, die mit unermüdlichem Einsatz meinen gedanklichen Ausdrucksformen folgten und mit großer Sorgfalt darauf achteten, dass Wort und Ausdruck eine angemessene Form behielten.
Vor allem möchte ich mich bei meiner lieben Frau Svenja bedanken. Sie verstand es mit großer Geduld und trotz zeitlicher Entbehrungen, den Fortschritt der Arbeit aufmerksam zu begleiten. Dabei entwickelte sie sehr einfühlsam stets gute Bedingungen für Ausgleich, kreative Rückbesinnung und neue Denkanstöße. Meiner Tochter Sarah verdanke ich es überhaupt, dass ich den Anfang für dieses Vorhaben gefunden habe und bin froh, ihrem kontinuierlichen Rat gefolgt zu sein.

Dresden, im Mai 2017 Jörg Kebbedies

Kurzfassung

Für Geschäftsbereiche mit hohen Anforderungen an Vertraulichkeit und Datenschutz zur Verarbeitung ihrer sensitiven Informationen kann für die Nutzung von Public-Cloud-Technologien keine Benutzerakzeptanz ausgewiesen werden. Die Ursachen dafür erwachsen aus dem inhärenten Strukturkonzept verteilter, begrenzter Verantwortlichkeiten und einem fehlenden Cloud-Anwender-Vertrauen.

Die vorliegende Arbeit verfolgt ein Cloud-Anwender-orientiertes Vorgehen zur Durchsetzung regelnder Policy-Konzepte, kombiniert mit einem holistischen Ansatz zur Herstellung einer durchgehenden Vertrauensbasis.

Der Aspekt Vertrauen erhält eine eigenständige Konzeptualisierung und wird zu einem Cloud-Anwender-Instrument für die Gestaltung vertrauenswürdiger infrastruktureller Eigenschaften entwickelt. Jede weitere Form einer Policy entwickelt ihren verbindlichen regulierenden Wert erst durch eine unlösliche Verbindung mit den hier vorgelegten Konzepten vertrauenswürdiger Entitäten.

Ein ontologisch formalisierter Beschreibungsansatz vollzieht die für eine Regulierung notwendige Konzeptualisierung einer domänenspezifischen IT-Architektur und qualifizierender Sicherheitseigenschaften. Eigenständige Konzeptklassen für die Regulierung liefern den Beschreibungsrahmen zur Ableitung integrierter Trust-Policies.

Darauf aufbauende Domänenmodelle repräsentieren eine vom Cloud-Anwender definierte Erwartung in Bezug auf ein reguliertes Cloud-Architektur-Design und reflektieren die reale Welt auf Grundlage vertrauenswürdiger Fakten. Vertrauen quantifiziert sich im Ergebnis logischer Schlussfolgerungen und ist Ausdruck zugesicherter Cloud-Sicherheitseigenschaften und geregelter Verhaltensformen.

Inhaltsverzeichnis

Abbildungsverzeichnis **xiii**

Tabellenverzeichnis **xv**

1 Einleitung **1**

1.1 Motivation . . . 3
1.2 Forschungsfragen . . . 3
1.3 Zielstellung . . . 4
1.4 Vorgehensweise . . . 5

2 Problembeschreibung **7**

2.1 Public Cloud, Strukturerweiterung einer Organisation . . . 7
2.1.1 Kopplung im sozialen Kontext . . . 7
2.1.2 Strukturelle Kopplung im Cloud-Kontext . . . 8
2.2 Regelungen: strukturbildende Elemente von Organisationen . . . 9
2.2.1 Regelungen im sozialen Kontext . . . 9
2.2.1.1 Rechtliche Regelungen . . . 9
2.2.1.2 Nichtrechtliche Regelungen . . . 10
2.2.1.3 Regelungen in Organisationen . . . 10
2.2.2 Regelungen im Cloud-Kontext . . . 11
2.3 Erwartungen und Unbestimmtheit von Handlungen . . . 11
2.3.1 Erwartungen im sozialen Kontext . . . 11
2.3.2 Erwartungen im Cloud-Kontext . . . 13
2.4 Konformität, Abbildung von Regelungen . . . 14
2.4.1 Konformität im sozialen Kontext . . . 14
2.4.2 Konformität im Cloud-Kontext . . . 14
2.5 Thesen . . . 15

3 Analyse **17**

3.1 Anforderungen . . . 17
3.1.1 Infrastrukturschicht . . . 18
3.1.1.1 Hardwarebasierte Geo-Lokalisierung . . . 19
3.1.1.2 Virtual Machine Monitor . . . 19
3.1.1.3 Netzwerksicherheit . . . 20

3.1.2 Plattform-/Laufzeitschicht . . . 20
3.1.2.1 Virtualisierungstechnologie . . . 20
3.1.2.2 OS-Sicherheitsmodell . . . 20
3.1.2.3 Datensicherheit der Laufzeitschicht . . . 21
3.1.3 Anwendungs-/Serviceschicht . . . 22
3.1.3.1 Anwendungssicherheit . . . 22
3.1.3.2 Prozesssicherheit . . . 22
3.1.3.3 Datensicherheit der Anwendungsschicht . . . 23
3.1.4 Verwaltung/Betrieb . . . 23
3.1.5 Compliance . . . 24
3.1.5.1 Governance . . . 24
3.1.5.2 Klassifizierte Informationen . . . 25
3.1.5.3 Datenschutz . . . 26
3.1.6 Zusammenfassung der Regulierungsziele . . . 27
3.2 Anwendungsfälle einer Multi-User-Cloud-Umgebung . . . 27
3.2.1 TCG-Konzepte und Definitionen . . . 28
3.2.2 UC-Aufbau einer Vertrauensbasis . . . 30
3.2.3 UC-Aufbau einer vertrauenswürdigen Kooperationsbasis . . . 32
3.2.4 UC-kooperative Provisionierung . . . 35
3.2.5 UC-Änderungen von Regeln innerhalb einer kooperativen Domäne . . . 38
3.2.6 Abgeleitete Anwendungsfälle aus TCG-Richtlinien . . . 41
3.3 State-of-the-Art-Betrachtung . . . 43
3.3.1 Thema: Regulierungsziele . . . 45
3.3.1.1 Pattern-based Runtime Management of Composite Cloud Applications 45
3.3.1.2 Unifying Compliance Requirements across Business and IT . . . 48
3.3.2 Thema: Digitale Regelkonzepte . . . 49
3.3.2.1 Policy-Aware Provisioning of Cloud Applications . . . 49
3.3.2.2 Policy-Aware Provisioning and Management of Cloud Applications . . 50
3.3.3 Thema: Vertrauenskonzepte . . . 51
3.3.3.1 Secure Enclaves for REactive Cloud Applications . . . 51
3.3.3.2 Enforcing-Security-and-Assurance-Properties-in-Cloud-Environment . 53
3.3.4 Thema: Technische Standards . . . 54
3.3.4.1 Web Services Policy 1.5 – Framework - Current . . . 54
3.3.4.2 WS-SecurityPolicy 1.3 . . . 55
3.3.4.3 WS-Trust . . . 56
3.3.4.4 Web Services Security: SOAP Message Security 1.1 . . . 57
3.3.5 Thema: Sprachkonzepte . . . 58
3.3.5.1 Using Ontologies to Analyze Compliance Requirements of Cloud-Based Processes . . . 58
3.3.5.2 Policy Language for a Pervasive Computing Environment . . . 59
3.4 Zusammenfassung und Abgrenzungsbeschreibung . . . 60

4 Konzeption **61**
4.1 Ontologie-Konzept . . . 62
4.1.1 Strukturentwurf Ontologie . . . 63
4.1.2 Ziele der ontologischen Konzeptualisierung . . . 64
4.1.3 Ontologie Regulierung . . . 65
4.1.3.1 Haupthierachie *Regulation*-Ontology . . . 66
4.1.3.2 Konzeptklasse *Action* . . . 66
4.1.3.3 Konzeptklasse *Constraint* . . . 66
4.1.3.4 Konzeptklasse *Rule* . . . 67
4.1.3.5 Konzeptklasse *Policy* . . . 67
4.1.3.6 Konzeptklasse *State* . . . 69
4.1.3.7 Konzeptklasse *Transformation* . . . 69
4.1.4 Ontologie Cloud-Domain . . . 70
4.1.4.1 Konzeptklasse *CloudDomain* . . . 70
4.1.4.2 Konzeptklasse *Entity* . . . 71
4.1.4.3 Konzeptklasse *Subject* . . . 72
4.1.4.4 Konzeptklasse *ArchitecturalLayer* . . . 72
4.1.4.5 Konzeptklasse *Object* . . . 73
4.1.4.6 Konzeptklasse *Part* . . . 74
4.1.4.7 Konzeptklasse *Connection* . . . 74
4.1.4.8 Konzeptklasse *CloudService* . . . 75
4.1.5 Ontologie Security . . . 76
4.1.5.1 Konzept einer vertrauensbildenden Sicherheitsstrategie . . . 76
4.1.5.2 Konzeptklasse *Asset* . . . 77
4.1.5.3 Konzeptklasse *PropertySecurity* . . . 77
4.1.5.4 Konzeptklasse *SecurityFunction* . . . 78
4.1.5.5 Konzeptklasse *SecurityRequirement* . . . 79
4.1.5.6 Konzeptklasse *Identity* . . . 79
4.1.5.7 Konzeptklasse *Credential* . . . 79
4.1.5.8 Konzeptklasse SecurityModel (Sicherheitsmodell) . . . 81
4.2 Konzept zur Herausbildung von Vertrauen (Trust) . . . 81
4.2.1 Konzept einer vertrauenswürdigen Entität . . . 82
4.2.2 Konzept einer Authority . . . 83
4.2.2.1 Zusicherung von Entity-Eigenschaften . . . 85
4.2.2.2 Entitäten innerhalb einer *Authority*-Hierarchie . . . 85
4.2.2.3 Entitäten und externe *Authority* . . . 86
4.2.3 Konzept einer Policy zur Entwicklung von Vertrauen . . . 86
4.2.3.1 Spezialisierung der Trust-Policy . . . 87
4.2.3.2 *QualityProperty* – Gegenstand der Vertrauenspolitik . . . 88
4.3 Trust-Establishment-Protokoll . . . 90
4.3.1 Datenmodell . . . 90
4.3.1.1 Verhaltensorientierte Artefakte . . . 90

4.3.1.2 Kryptographische Artefakte . . . 91
4.3.1.3 Protokollspezifische Artefakte . . . 92
4.3.2 Horizontale Etablierung von Vertrauen (Establishment of Trust) . . . 93
4.3.2.1 Phase1: Auswahl einer Cloud-Plattform . . . 94
4.3.2.2 Phase2: Erweiterung der Vertrauensgrundlage auf Cloud-Anbieter-Seite 96
4.3.3 Vertikale Etablierung von Vertrauen (Delegation of Trust) . . . 98
4.3.3.1 Registrierung von Policy-Entitäten . . . 99
4.3.3.2 Registrierung von Domänen-Entitäten . . . 99
4.3.3.3 Ableitung vertrauenswürdiger Entitäten . . . 100
4.3.3.4 Ableitung vertrauenswürdiger Eigenschaften und Aktivitäten . . . 101
4.4 Zusammenfassung . . . 103

5 Validierung 105
5.1 Referenzarchitektur – Trusted Cloud . . . 106
5.1.1 Komponentenbeschreibung – IT-Plattform . . . 107
5.1.2 Komponentenbeschreibung – Laufzeitumgebung . . . 109
5.1.3 Komponentenbeschreibung – Integrierte Systeme . . . 109
5.1.4 Externe Systeme – Key & CA Service . . . 110
5.1.4.1 Bezeichnungen und Namespaces . . . 112
5.1.4.2 TE-Zustandsmodell . . . 113
5.1.4.3 Policy-Zonen und Policy-Anwendungsraum . . . 113
5.2 Trust-Policies und Transformation . . . 114
5.2.1 Szenario (1) – Bereitstellung Virtual Machine Monitor KVM . . . 116
5.2.1.1 Domain-Spezifikation – KVM-Komponente . . . 116
5.2.1.2 Regulation-Spezifikation – KVM-Deployment-Policy . . . 116
5.2.1.3 Prüfung der KVM-Authentizität . . . 117
5.2.1.4 Zusicherung von KVM-Identitätseigenschaften . . . 118
5.2.1.5 Transformation – KVM-Trust-Rule . . . 119
5.2.1.6 Transformation – KVM-Deployment-Rule . . . 120
5.2.2 Szenario (2) – Bereitstellung Virtualisiertes Betriebssystem . . . 120
5.2.2.1 Domain-Spezifikation – Virtual-OS . . . 120
5.2.2.2 Regulation-Spezifikation – Virtual-OS-Deployment-Policy . . . 121
5.2.2.3 Prüfung der TE-Authentizität . . . 121
5.2.2.4 Policy-Zone einrichten – *Z_RUNTIME.DB* . . . 122
5.2.2.5 Vertrauenskette prüfen – *Chain of Trust* . . . 123
5.2.3 Szenario (3) – Bereitstellung Datenbanksystem (DBS) . . . 124
5.2.3.1 Domain-Spezifikation – Datenbanksystem . . . 124
5.2.3.2 Regulation-Spezifikation – DBS-Deployment-Policy . . . 125
5.2.3.3 Prüfung der DBS-Authentizität . . . 126
5.2.3.4 Transformation – DBS-Trust-Rule . . . 126
5.2.3.5 Transformation – DBS-Deployment-Rule . . . 126

5.2.4 Szenario (4) – Externe DBS-Zugangssteuerung . . . 126
5.2.4.1 Domain-Spezifikation – User-to-DB Connection . . . 127
5.2.4.2 Regulation-Spezifikation – DBS-Connection-Policy . . . 127
5.2.4.3 Prüfung der DBS-Endpunkt-Authentizität . . . 128
5.2.4.4 Absicherung der DBS-Verbindung – Verschlüsselung . . . 130
5.2.4.5 Transformation . . . 132
5.3 Attestierung – Vertrauenswürdigkeit . . . 133
5.3.1 Dynamische Methoden der Konzeptklasse *State* . . . 133
5.3.2 Kategorien für Niveaubestimmung von Vertrauenswürdigkeit . . . 134
5.3.3 Semantische Rules für Niveaubestimmung . . . 135
5.3.3.1 Ableitungsregel – Vertrauenswürdigkeit *HOCH* . . . 135
5.3.3.2 Ableitungsregel – Vertrauenswürdigkeit *MITTEL* . . . 136
5.3.3.3 Ableitungsregel – Vertrauenswürdigkeit *GERING* . . . 137
5.3.3.4 Ableitungsregel – Vertrauenswürdigkeit *UNBESTIMMT* . . . 138
5.4 Gegenüberstellung der Szenarien mit den Zielstellungen . . . 138
5.5 Gegenüberstellung der Ergebnisse mit den Kernfragen . . . 140
5.6 Zusammenfassung der Validierung . . . 141

6 Zusammenfassung – Ausblick 143
6.1 Zusammenfassung der Arbeit . . . 143
6.2 Ausblick und abgeleitete Themen . . . 145

Abkürzungsverzeichnis 147

I State-of-the-Art – Kategorien 161

II Hardwareunterstützte Sicherheit für eine IT-Plattform 165
II.1 Trusted Platform Module . . . 165
II.2 Technologie für IT-Plattformsicherheit . . . 165
II.3 Konzept einer hardwarebasierten Vertrauenspolitik . . . 166
II.3.1 Sichere Mikroarchitektur . . . 166
II.3.2 Messung statischer Systemeigenschaften . . . 166
II.4 Kontrollierter Systemstart . . . 167
II.4.1 Identifizierbarer Plattform-Eigentümer . . . 168
II.4.2 Versiegeln von Systemwerten (*Sealing*) . . . 168
II.5 Konzept der Attestierung . . . 168
II.5.1 Attestierungs-Schlüssel . . . 168
II.5.2 Zertifizierung des Attestierungs-Identifikationsschlüssels . . . 169
II.5.3 Attestierungs-Modul . . . 169
II.5.4 Attestierungs-Service . . . 169
II.5.5 Hardwarebasierte Geo-Lokalisierung . . . 170

III Übersicht der Anforderungen 171
III.1 Anforderungen an die Cloud-Infrastruktur-Plattform-Ebene . . . 171

III.2 Anforderungen an die Cloud-Laufzeitebene 172
III.3 Anforderungen an die Cloud-Service-Ebene 173
III.4 Anforderungen an operatives Management 174
III.5 Anforderungen an Cloud-Anwender-Nutzungsebene 176

IV Spezifikation Ontologie **179**

Abbildungsverzeichnis

1.1 Modell der Regulierung zur Sicherstellung von Sicherheits- und Datenschutzstrategien . 6

2.1 Modell einer Organisation 8
2.2 Struktureller Begriff für Konformität 14

3.1 Konzept einer Vertrauensbasis (*Trusted Context*) 32
3.2 Konzept einer kooperierenden Vertrauensbasis (*Cooperated Trusted Context*) 34
3.3 Vertrauenswürdige kooperative Erweiterung von Cloud-Ressourcen 37
3.4 Vertrauenswürdige kooperative Änderung von Cloud-Ressourcen 40
3.5 Abgeleitete Anwendungsfälle als Anforderungsgrundlage 41
3.6 Themenbereiche der State-of-the-Art-Betrachtung 43

4.1 Pattern für ein Verbundkonzept vertrauenswürdiger Entitäten 61
4.2 Ontologie Strukturentwurf 64
4.3 Konzept einer Sicherheitsstrategie 76
4.4 Trustworthy-Entity Pattern 82
4.5 Hierarchien von Authorities 84
4.6 Policy-Konzept zur Entwicklung vertrauensbildender Architekturgrundlagen 89
4.7 Horizontale Etablierung von Vertrauen 93
4.8 Phase1: Horizontale Etablierung von Vertrauen (*Auswahl Cloud-Plattform*) 94
4.9 Phase2: Vertrauensgrundlage auf Cloud-Anbieter-Seite erweitern 96
4.10 Ganzheitliches Anwendungsprinzip vertrauenswürdiger Entitäten 100
4.11 Ebenen der State-Konzeptualisierung 102

5.1 Referenzarchitektur – Validierung 106
5.2 CA-Hierarchy – Validierung 110

Tabellenverzeichnis

3.1 Taxonomie der Sicherheitsaspekte von Cloud Computing 18
3.2 Übersicht der Regulierungsziele 27
3.3 Ablaufbeschreibung – Konzept einer Vertrauensbasis 31
3.4 Ablaufbeschreibung – Konzept einer vertrauenswürdigen Kooperationsbasis 33
3.5 Ablaufbeschreibung – Bereitstellen von vertrauenswürdigen Cloud-Ressourcen 36
3.6 Ablaufbeschreibung – Änderungen in der Policy des Cloud-Anwenders 39

4.1 Ablaufbeschreibung – Auswahl Cloud-Plattform 95
4.2 Ablaufbeschreibung – Vertrauensbasis horizontal erweitern 98

5.1 Auszug aus Ontologie-*Cloud-Domain* für Referenzarchitektur 107
5.2 Übersicht Namespace-Definitionen 112
5.3 Übersicht Präfix-Definitionen 112
5.4 Übersicht Storage-Bereiche 112
5.5 Statusmodell – TE_x-Instanz 113
5.6 Policy-Zonen für Referenzarchitektur 113
5.7 Ablaufbeschreibung – Auswahl Cloud-Plattform 115
5.8 Kategorien als Grundlage zur Attestierung einer bestehenden Vertrauenswürdigkeit . . . 135

1 | Einleitung

Cloud Computing ist zweifelsfrei zurzeit eines der dynamischsten Themen in der IT-Branche. Mit einer extrem hohen Entwicklungsgeschwindigkeit werden potenzielle Anwender fast täglich mit neuen Möglichkeiten, Angeboten und Produkten zur Optimierung ihrer IT-Landschaften konfrontiert. Cloud Computing ist eine Schlüsseltechnologie zur Beschleunigung digitaler Transformationen im Bereich wirtschaftlicher, sozialer und auch politischer Prozesse.
Doch jede neue Entwicklung birgt neben den Potenzialen auch Gefahren, die es zu erkennen und zu minimieren gilt. Dabei besitzt Cloud Computing das Potenzial, die IT-Branche zu transformieren, so dass sie Teil der Dienstleistungsgesellschaft wird. IT-Produkte sollen also nicht mehr selbst betrieben, sondern auf dem freien Markt als Dienstleistung eingekauft werden, um neue Geschäftsmodelle zu eröffnen, die erst mit den Möglichkeiten digitaler Prozesserweiterungen eine völlig neue Flexibilität entfalten.
Die Vorteile dieser Entwicklung erkauft sich der Anwender jedoch mit einem Verlust von Kontrolle seiner genutzten und oftmals zur Erbringung der Kernbereiche auch benötigten IT-Infrastrukturen. Aber es geht nicht allein um Kontrolle der Infrastruktur. Vielmehr benötigt der Anwender Möglichkeiten, seine Strategien in Form moderner digitaler Prozesse nach geregelten Grundsätzen durchzusetzen und zu beeinflussen. Der Kontroll- und Regulierungsverlust muss mit Vertrauen in den Dienstleister ausgeglichen werden [Cur+10]. Doch bleibt die Frage, wie ein Anwender erkennen und bewerten kann, ob das Vertrauen gerechtfertigt ist. Hierzu müssen Kriterien, Verfahren und Konzepte entwickelt werden, die den Anwender bei seiner Auswahl des IT-Dienstleisters unterstützen.
Im Bereich der IT-Sicherheit hat das Bundesamt für Sicherheit in der Informationstechnik (BSI) im Rahmen seines Eckpunktepapiers [BSI12] Mindestanforderungen zur Informationssicherheit bei Cloud-Computing-Diensten für den normalen und hohen Schutzbedarf definiert.
Auch wenn durch das Eckpunktepapier ein wichtiger Schritt zur Etablierung von mehr Vertrauen in Cloud Computing gegangen wurde, besteht speziell bei sehr hohem Schutzbedarf, wie beispielsweise im Geheimschutz oder in der Finanz- und Versicherungsbranche, trotz aller bisheriger Bemühungen kein ausreichendes Vertrauen in diese Form der Datenverarbeitung. Aufgrund des hohen Geschäftspotenzials von Cloud Computing ist jedoch eine hohe Erwartungshaltung vorhanden, auch für diese Branchen technische Voraussetzungen zu schaffen, um das benötigte Vertrauensverhältnis herzustellen.
Einschlägige Normen zum Informationssicherheitsmanagementsystem (ISMS) [ISO13] oder des BSI zum IT-Grundschutz [BSI08] bilden die Grundlagen für eine strukturierte Vorgehensweise sowie ein effektives Management, um ein angemessenes Sicherheitsniveau in einem Unternehmen oder einer Behörde erfolgreich zu erzielen und aufrechtzuerhalten.
Im IT-Bereich drückt Vertrauenswürdigkeit den Grad des Vertrauens aus, den ein Anwender gegenüber einem IT-System aufbringt. Hierunter zählt beispielsweise, dass ein System wie vorgeschrieben und

erwartet funktioniert und nicht ausfällt und dass gesetzte Erwartungen in Bezug auf die Sicherheit und Qualität erfüllt werden.
Durch die Aufgabe der Hoheit über die eigenen Daten muss der Cloud-Anwender dem Cloud-Anbieter ein gewisses Maß an Vertrauen entgegenbringen. Der Anwender ist gezwungen, darauf zu vertrauen, dass die Integrität, Verfügbarkeit und Vertraulichkeit seiner Daten durch den Diensterbringer zugesichert und gewährleistet werden. Eine Studie der Bitkom [Bar+09b] belegt, dass ein fehlendes Vertrauen in die Nutzung von Cloud-Dienstleistungen ein wesentliches Risiko für das Erreichen einer hohen Nutzerakzeptanz darstellt.
Besonders das fehlende Vertrauen in Datenschutz- und Datensicherheitskonzepte sowie die im Einzelfall unklaren rechtlichen Regelungen bilden derzeit das größte Hemmnis für eine schnellere Marktentwicklung im Bereich Cloud Computing. Für die Herausbildung von Vertrauenswürdigkeit hinsichtlich Cloud Computing spielen unterschiedliche Qualitätseigenschaften der IT-Systeme, die für die Service-Erbringung verantwortlich sind, eine Rolle. Für die Entstehung von Vertrauenswürdigkeit lassen sich folgende relevante Qualitätseigenschaften für Cloud-Systeme, die Daten mit sehr hohem Schutzbedarf speichern oder verarbeiten, identifizieren:

- Vertraulichkeit der Daten
- Integrität der Daten
- Verfügbarkeit der Daten
- Authentizität der handelnden Akteure
- Transparenz der Prozesse, in denen die Daten verarbeitet werden
- Nachvollziehbarkeit aller abgelaufenen Prozesse
- Zugesicherte, beschränkte Lokalität der Daten
- Gesichertes Löschen der Daten aus dem System

Im allgemeinen Konsens zur Definition von Cloud-Services stellt ein Cloud-Service-Anbieter auf unterschiedlichen Serviceebenen IT-Dienstleistungen zur Verfügung. Die IT-Serviceleistungen lassen sich zur besseren Abgrenzung verschiedenen Ebenen der IT-Datenverarbeitungskette zuordnen:

- Services auf der Ebene der Verarbeitungsleistung (SaaS)
- Services auf der Ebene der Plattform (PaaS)
- Services auf der Ebene der IT-Infrastrukturen (IaaS)

Bei der Betrachtung der differenzierten Servicedienstleistungen wird deutlich, dass sowohl IT-Verarbeitungssoftware (Nutzungsrechte für IT-Anwendungen) als auch Geschäftsdaten von Kunden oder Unternehmen (Zugriffsrechte auf Daten) einzeln oder in Kombination in die Verantwortung eines externen Cloud-Anbieters übergeben werden können. Beiden Aspekten (Verarbeitungssoftware, Daten) ist gemeinsam, dass die Kontrollfähigkeit schrittweise aus der Sicht des Cloud-Anwenders abgegeben wird.

1.1 Motivation

Cloud Computing als Servicekonzept kann den Handlungsspielraum zukünftiger Cloud-Anwender über die Grenzen ihrer eigenen Organisationsbereiche ausdehnen und vergrößert darüber ihr wirtschaftliches und kommunikatives Potenzial. Unabhängig davon, welche Dienstleistung von einem Cloud-Anwender in Anspruch genommen wird, besteht die Notwendigkeit, basierend auf einem festgelegten Handlungsrahmen eine kooperative Beziehung mit einem Cloud-Anbieter einzugehen.
Gegenwärtige Cloud-Architekturmodelle bieten noch keinen Ansatz, Cloud-Anwender in die Lage zu versetzen, durch Formen technischer Integration auf die Prozess- und die Sicherheitspolitik in Infrastrukturbereichen der Dienstleistungsangebote eines Cloud-Anbieters Einfluss zu nehmen. Bisherige Praktiken reduzieren das Prinzip ihrer Einbeziehung auf das Angebot, fertig abgestimmte und überprüfbare Leistungsangebote auf Grundlage passender Vertragsgrundlagen zu übernehmen und diese entsprechend nachzunutzen.
Cloud-Anwender benötigen effektive Möglichkeiten, Cloud-Serviceangebote selbstbestimmt in die eigene Prozessstrategie zu integrieren und sie als Teil eigenverantwortlichen Handelns auf die Bereiche eines Cloud-Anbieters auszudehnen. Dabei ist die Kontrollfähigkeit über den gesamten Regulierungsrahmen zu gewährleisten. Cloud-Serviceangebote werden als Teil einer Cloud-Anwender-Erwartung verstanden und bilden eingebettete Funktionsbausteine im eigenen Handlungs- und Regulierungsrahmen.
Regulierung wird hier als Prozess zur Begrenzung und gleichzeitiger Ausrichtung von Handlungsweisen bzw. Systemprozessen verstanden, um ein durch Vorgaben festgelegtes, nachvollziehbares und kooperatives Zusammenwirken der beteiligten Cloud-Akteure durchzusetzen. Dabei wird die Einhaltung von System- und Sicherheitseigenschaften eines Cloud-Systems gefordert, durchgesetzt und nachgewiesen. Die Regulierungsansprüche umfassen dabei Funktions-, Sicherheits- und Datenschutzrichtlinien, die in der Regel außerhalb konkreter Cloud-Anwendungen entstehen.
Der Schlüssel für die Umsetzbarkeit einer Cloud-Anwender-bezogenen Regulierungspraxis liegt im Konzept einer *vertrauenswürdigen* Realisierbarkeit. Daher ist diese Betrachtung nur zielführend, wenn dem Cloud-Anwender ein geeignetes Konzept zur Bewertung der Vertrauenswürdigkeit eines Cloud-Anbieters in allen Bereichen seiner Dienstleistung angeboten wird.
Unter Berücksichtigung kontrollierbarer Vertrauensverhältnisse kann eine selbstbestimmte Cloud-Anwender-Strategie auf das Potenzial aktueller Cloud-Dienstleistungen überführt werden.

1.2 Forschungsfragen

Die Nutzung von Public Cloud Computing stellt eine technische Form kooperativer Kopplung dar und erfordert technische Konzepte zur Erweiterung des Regulierungs- und Handlungsbereiches des Cloud-Anwenders. Grundsätzlich führt jede strukturelle Kopplung zu einer erhöhten Komplexität und kann nur durch ein Vertrauenskonzept zwischen Cloud-Anwender und Cloud-Anbieter gelöst werden. Ein Hauptproblem entwickelt sich aus den Prinzipien gegenwärtiger kooperativer Kopplungen im Public-Cloud-Bereich.
Während Cloud-Anbieter festgelegte Serviceangebote zur Verfügung stellen, so besteht für Cloud-Anwender die Notwendigkeit, auf diese Serviceangebote regulierend Einfluss nehmen zu können. Maßnahmen für eine Cloud-Anwender-orientierte Regulierung sind effektiv durchzusetzen und bedürfen ver-

teilter, vertrauenswürdiger IT-Architekturkonzepte, die den Handlungsrahmen des Cloud-Anwenders für einen vertraglich definierten Nutzungszeitraum erweitern. Von diesen Aspekten ausgehend, werden folgende Kernfragen formuliert:

Kernfrage 1: Wie entwickelt sich Vertrauen bei der Nutzung technischer Systeme?
Wie lässt sich der Aspekt von Vertrauen auf eine verteilte Cloud-Architektur übertragen? Welches sind die bestimmenden qualitativen und funktionalen Kriterien, unter denen sich Vertrauen bei einem Cloud-Anwender in Bezug auf die Nutzung von Cloud-Services entwickelt? Können aus diesen Kriterien Werte ermittelt werden, aus denen sich ein Maß für Vertrauen ableitet?

Kernfrage 2: Welche Sprachkonzepte sind für Regulierungsstrategien anzuwenden?
Mit welchen formalen Sprachmitteln kann eine ganzheitliche Repräsentation von unterschiedlichen Regulierungsanforderungen erfolgen, so dass auf technischer Ebene eine semantisch korrekte Interpretation in Bezug auf den Regulierungsgegenstand gewährleistet ist? Dabei ist zu berücksichtigen, dass der Gegenstand der Regulierung, die Systemeigenschaften und Prozesse in einem verteilten Cloud-System, eine entsprechende sprachliche Repräsentation benötigt.

Kernfrage 3: Welche Ausdrucksstärke benötigen Regulierungsanforderungen?
Die Regulierung verfolgt unterschiedliche Zielstellungen, die sich auf z. B. funktionale, sicherheitstechnische und datenschutzrechtliche Aspekte beziehen. Wie können mit einem formalen Sprachkonzept unterschiedliche Regulierungsanforderungen formuliert werden, so dass sich daraus eine erfolgreiche Anschlusshandlung ableitet?

Kernfrage 4: Wie gestaltet sich die Erweiterung der Regulierungshoheit?
Wie vollzieht sich für einen Cloud-Anwender die technische Erweiterung seiner eigenen Regulierungs- und Handlungshoheit auf die Bereiche eines Cloud-Anbieters? Die Erweiterung soll dabei sowohl die Prozesse zur Regulierung als auch Konformitätsbestimmung mit einbeziehen. Wie kann der Nachweis über eine erfolgreiche Umsetzung erbracht werden?

1.3 Zielstellung

Die Promotionsarbeit beschreibt konzeptionell einen Ansatz, um die bisherigen, sich anscheinend ausschließenden Lösungskonzepte in eine technische Konvergenz zu überführen. Die Fokussierung auf einen reinen clientbasierten bzw. Cloud-Anbieter-orientierten Lösungsansatz soll zu einem zwischen Serviceanbieter und Serviceanwender verteilten, wechselseitig verhandelbaren Lösungsansatz weiterentwickelt werden.
Wie in jeder verteilten Kommunikation bestimmen Regelwerke (Policies) den Handlungsrahmen und die Qualitätsmerkmale (z. B. Verfügbarkeit, Sicherheit, Performance) der sich herausbildenden und für einen definierten Zeitraum bestehenden Beziehung zweier Kommunikationspartner.
Die allgemeine Zielstellung besteht darin, Prinzipien und Verfahren für eine verteilte IT-Sicherheitsarchitektur herauszuarbeiten, die es erlauben, auf Grundlage beschreibbarer Interaktions- und Verarbeitungsvorschriften den Aufbau, die Nutzungsbedingungen sowie die Bedingungen für die Auflösung von Trusted Domains zwischen den autarken Systemeinheiten eines Cloud-Anwenders und des Cloud-Service-Anbieters technisch zu realisieren.

Die konkrete Zielstellung orientiert auf eine Beschreibbarkeit von technischen Abläufen (Interaktionsmuster) und Bedingungen zur Nutzung Cloud-gestützter Datenverarbeitungsdienstleistungen in Form einer Cloud Security Policy. Die Beschreibung stützt sich auf Vorgaben des Cloud-Anwenders. Die Durchsetzung und Überwachung von Cloud-Anwender-Policies auf der Seite des Cloud-Anbieters stehen im Vordergrund. Der Entwurf einer interpretierbaren Semantik bildet dafür die Grundlage. Die Semantik ist ausgerichtet auf die Formulierung technischer Interaktionsmuster, Kennzeichnung von Bedingungen und Eigenschaften von Einzelaktivitäten, sowie Vorgabe von Sicherheitszielen und rechtlichen Nutzungsbedingungen bezüglich der Datenverarbeitungsprozesse.
Für die Umsetzung einer Security Policy wird eine architekturelle Umgebung benötigt. Dafür soll in einem ersten Schritt ein Referenzmodell „Trusted Domain“ entwickelt werden. Das Referenzmodell beschreibt eine verteilte und vertrauenswürdige IT-Sicherheitsarchitektur unter Mitwirkung unabhängiger Akteure, die in eine vertraglich fixierte Nutzungsbeziehung (Cloud Security Policy) eintreten.
Die Ausführung einer interpretierbaren Semantik benötigt eine spezifische Sprache, die innerhalb einer verteilten Cloud-Architektur technische Rahmenbedingungen vorfinden muss. Das Referenzmodell identifiziert dabei Bausteine, denen Vertrauen zugeschrieben werden kann, um auf ihrer Grundlage eine Cloud Security Policy durchzusetzen.
In einem zweiten Schritt werden Konzepte zur Entwicklung von Vertrauen bei der Nutzung cloudbasierter Ressourcen und Dienstleistungen herausgearbeitet. Aus den Kernkonzepten heraus konkretisiert sich der Rahmen für die Identifizierung von allgemeinen Handlungs- und Sicherheitspattern. Die konsequente Anwendung einer patternorientierten Verallgemeinerung soll den Entwurf der Cloud-Security-Policy-Semantik begründen.
Die technische Realisierbarkeit des Referenzmodells „Trusted Domain“ soll durch eine prototypische Referenzimplementierung nachgewiesen werden. Basierend auf ausgewählte Anwendungsfälle des Prototyps, werden Cloud Security Policies durch Cloud-Anwender entwickelt und auch modifiziert, um die Durchsetzung geforderter Kernkonzepte und deren gezielte Modifikation zu demonstrieren. Den Schwerpunkt bildet dabei die konkrete Einhaltung von Sicherheitsvorgaben und Ablaufstrukturen, die zur Herausbildung von Vertrauenswürdigkeit beitragen.

1.4 Vorgehensweise

Ausgehend von einem allgemeinen Modell zur Entwicklung von Policy-Pattern [Fer13], lässt sich Regulierung als Ausdruck konkreter strategischer Zielstellungen in Abbildung 1.1 beschreiben. Dabei zeigt sich eine enge Verflechtung zwischen der Policy-Entwicklung und dem Architekturdesign.
Änderungen in der Sicherheits- und Datenschutzstrategie (Modell zur Regulierung der Zuverlässigkeit ist hier ausgeklammert) führen zu strukturellen Veränderungen im Sicherheits- und Funktionsdesign einer Cloud-Architektur. Wechselnde Datenschutz- und Sicherheitsziele benötigen adaptive Architekturkonzepte, um neue Regulierungsformen auf das System zu übertragen.
Die Vorgehensweise beginnt mit einer strukturierten Erhebung von technischen und nicht funktionalen Anforderungen für die Umsetzung regulierter Cloud-Konzeptionen auf der Grundlage verteilter IT-Architekturen. Die Verteilung bezieht sich auf die Differenzierung von IT-Ressourcen der Gruppe von Cloud-Anwendern und der Gruppe von Cloud-Anbietern.

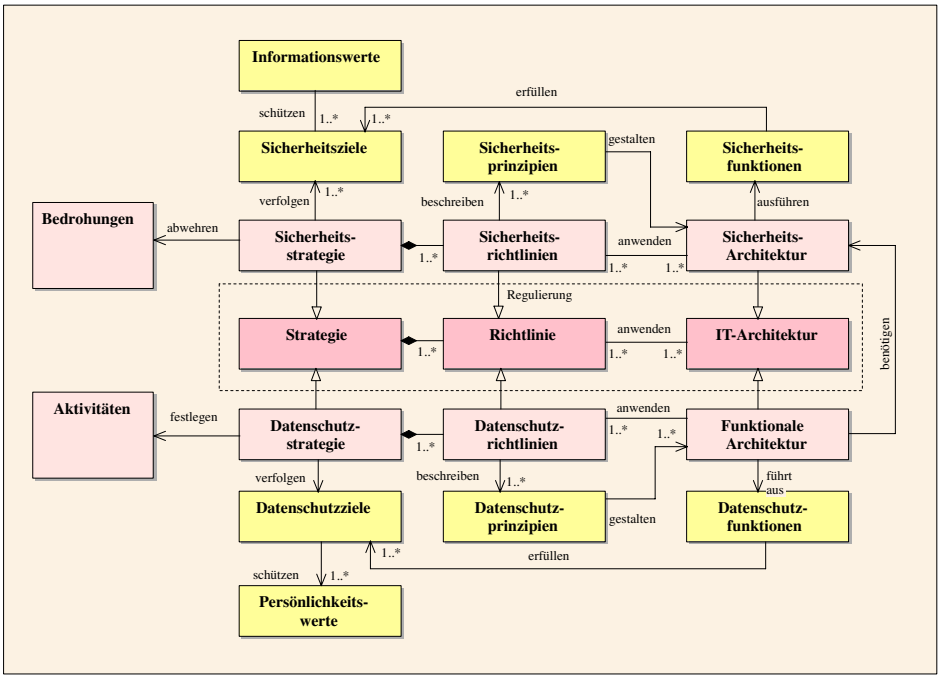

Abbildung 1.1: Modell der Regulierung zur Sicherstellung von Sicherheits- und Datenschutzstrategien

Es folgt eine systematische Entwicklung von Szenarien für die Herausbildung von Vertrauenswürdigkeit in Form von Anwendungsszenarien. Jedes Szenario ist gekennzeichnet durch eine Zielstellung im Kontext der Etablierung, Modifikation, Überwachung und Auflösung einer Sicherheitsdomäne.
In einem nächsten Schritt erfolgt der Entwurf eines Referenzmodells (Trusted Domain), auf dessen Grundlage die Umsetzung der Kernkonzepte für die Herausbildung von Vertrauenswürdigkeit nachgewiesen werden kann. Im Zuge der Herausarbeitung von Kernkonzepten für Vertrauenswürdigkeit wird das Referenzmodell weiter qualifiziert. Die Aspekte von Verlässlichkeit einzelner Komponenten und Compliance sind leitende Prinzipien für den Entwurf des Referenzmodells.
Ausgehend von einem Referenzmodell wird eine Realisierungsspezifikation für die Implementierung der Kernkonzepte für Vertrauenswürdigkeit vorgelegt. Im Ergebnis liegt eine prototypische Implementierung der Referenzarchitektur vor. Die umgesetzten Anwendungsfälle sollen den technischen Nachweis für die Realisierbarkeit von vertrauenswürdigen Ende-zu-Ende-Beziehungen im Kontext einer verteilten IT-Architektur erbringen.
Für die Herausbildung einer allgemeinen Nutzungsakzeptanz und den Nachweis einer produktiven Anwendbarkeit werden am Beispiel cloudbasierter Referenzbeispiele Datenverarbeitungskonzepte vorgestellt, die sich auf Grundlage des Prototypen gemäß bestehender Sicherheitsanforderungen ausführen lassen. Die Cloud-Referenzbeispiele bilden konkrete Geschäftsprozesse ab. Sie umfassen die sichere Übertragung von Geschäftsdaten in eine Cloud-Umgebung, Verarbeitung, Verteilung und sichere Rückübertragung der Geschäftsdaten. Die Rückübergabe ist mit einer sicheren Datenvernichtung auf der Seite des Cloud-Anbieters verbunden.

2 | Problembeschreibung

Wie in der Einleitung bereits dargestellt, bestätigen aktuelle Praxisbewertungen für die Anwendung sensitiver elektronischer Geschäftsprozesse, speziell im Public-Cloud-Bereich, nur eine sehr geringe Akzeptanz, diese Technologie einsetzen zu wollen.
Für die Darstellung der Problemsituation wird ein Ansatz gewählt, der sich nicht primär auf einen technischen Diskurs stützt. Vielmehr wird gezeigt, dass bestehende Schwierigkeiten im Public-Cloud-Bereich sich über allgemeine Grundprinzipien sozialer Kopplungen und damit über Grundlagen der sozialen Systemtheorie erklären lassen.
Erst in einem zweiten Schritt wechselt der Diskurs auf die Übertragung dieser Grundprinzipien in den Bereich der Cloud-Technologiekonzepte. Die Herausarbeitung der Konzepte systemischer Kopplungen führt zu Schlussfolgerungen darüber, welche Teile dieser Konzepte in die Technologiebetrachtung von Cloud-Systemen einfließen müssen, um darüber einen adäquaten Lösungsansatz zu finden.
Der Zusammenhang zwischen den Problemstellungen im Cloud Computing und den Gesetzmäßigkeiten sozialer Systeme wird durch gemeinsame Begriffe wie Regelungen (Regulation), Konformität (Compliance) und Vertrauen (Trust) hervorgehoben (siehe Abbildung 2.1). Die aufgeführten Begriffe konstituieren sich aus den strukturellen Prinzipien sozialer Systeme und lassen sich adäquat auf Cloud-Technologien als Repräsentation sozialer Kopplungen übertragen. Die eingeführten Begriffe werden nachfolgend als deutsche Begriffe weiter verwendet.

2.1 Public Cloud, Strukturerweiterung einer Organisation

Der Übergang zur Nutzung von Cloud-Dienstleistungen stellt sich für Cloud-Anwender, in einer systemischen Betrachtung, als Kopplung mit einer bis dahin eigenständig operierenden Dienstleistungsorganisation (Cloud-Anbieter) dar. Überführt man die Begriffe Cloud-Anwender und Cloud-Anbieter in einen sozialen Kontext, so beschreiben beide Begriffe das System einer Organisation.

2.1.1 Kopplung im sozialen Kontext

Der Begriff Kopplung beschreibt an dieser Stelle eine strukturelle und keine technische Kopplung. Der Ausdruck dient dafür, formal eine Beziehung zwischen zwei Organisationen abzubilden und ist als Erweiterung der bestehenden sozialen Dimension einer Organisation zu verstehen. Auf Grundlage dieser Beziehung entsteht Kommunikation zwischen beiden Seiten. Konkrete Ausprägungen struktureller Kopplungen sind z. B. Schriftverkehr, Kurieraufträge, Lieferungen, Dienstleistungsverpflichtungen, aber auch IT-gestützte Kommunikationskonzepte.

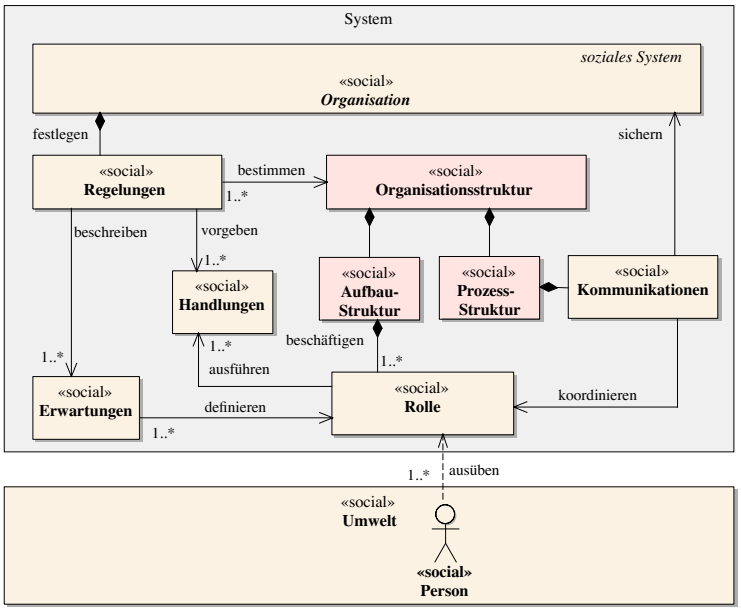

Abbildung 2.1: Modell einer Organisation

Die Kopplung an eine Organisationsstruktur führt zu einer Zunahme an organisatorischer und technischer Komplexität. Die bewusste Erweiterung von Prozessstrukturen ist Ausdruck z. B. wirtschaftlicher Motivationen, das Zulassen neuer Möglichkeiten, um diese zielführend einzusetzen.
Eine zunehmende Komplexität lässt sich nur unter dem Prinzip von Vertrauen bewältigen. *Vertrauen* beschreibt die Strategie mit der größten Reichweite zur Erweiterung von Handlungspotenzialen einer bisher autark agierenden Organisation. Eine Organisation kann sich auf unsichere Prämissen einlassen und hebt dabei deren Sicherheitswert, denn es fällt schwer, erwiesenes Vertrauen zu täuschen [Luh87, S. 180]. Vertrauen schafft größere Handlungsspielräume und Möglichkeiten bei gleichzeitiger Reduzierung der Komplexität [Luh14, S. 30]. Wenn eine Organisation Vertrauen schenkt, so kann sie sich auf neue Kooperationen einlassen. Es entstehen neue Chancen für eine komplexere Rationalität [Luh14, S. 28].
Misstrauen, als zweite Strategie, vergrößert ebenfalls den Handlungsrahmen, wenn auch einschränkender. Im Unterschied zu Vertrauen lässt man sich auf Risiken nur dann ein, wenn entsprechende Sanktionen vorbereitet sind oder ausreichende Versicherungen vorliegen. Vertrauen muss jedoch mit der Möglichkeit des Umschlagens in Misstrauen gekoppelt sein [Luh87, S. 180].

2.1.2 Strukturelle Kopplung im Cloud-Kontext

Die Einführung von Cloud Computing bedeutet auch aus Sicht eines Cloud-Anwenders eine Zunahme an organisatorischer und technischer Komplexität. Die Steigerung der Komplexität entsteht durch die Öffnung bzw. durch die Kopplung an eine bis dahin isoliert agierende Organisationsstruktur des Cloud-Anbieters. Bestehende Risiken und die Absicht, kooperative Verbindungen eingehen zu wollen, erfordern eine Strategie von Vertrauen, oder auch von Misstrauen.
Für die Nutzung von Cloud Computing stellt sich der Begriff Vertrauen darüber in den Mittelpunkt. Gleichzeitig manifestiert sich Vertrauen als Problem, da jeder Nachweis über erwiesenes Vertrauen nur künftig überprüft werden kann. Es ist eine der großen Herausforderungen, geeignete Kontrollfunktionen (siehe Thema Konformität, Abschnitt 2.4) zu schaffen, die den Nachweis über die Richtigkeit von

erwiesenem Vertrauen erbringen. Der geringste Hinweis darüber, dass erwiesenes Vertrauen gebrochen wird, führt zu einer schwer korrigierbaren Veränderung einer bestehenden Beziehung zwischen Cloud-Anwender und Cloud-Anbieter. Diesem Grundprinzip folgend entstehen Chancen, dass sich die Beziehungen gekoppelter Organisationen auf Basis von Cloud-Anwendungen weiter stabilisieren.
Der Unterschied der Reichweite soll auch für den Aspekt Vertrauen im Bereich Cloud Computing Berücksichtigung finden. *Blindes Vertrauen* im Bereich Cloud Computing kann nicht akzeptiert werden.

2.2 Regelungen: strukturbildende Elemente von Organisationen

Die wirtschaftlichen Zielstellungen zurückgestellt, konstituieren sich Organisationen durch geltende *Regelungen* zur Steuerung von Leistungen und dem Verhalten ihrer Mitarbeiter [KK92].
Da der Begriff *Regelung* ein Kernkonzept der vorliegenden Arbeit darstellt, soll an dieser Stelle eine Begriffsbestimmung erfolgen. Die Begriffsbestimmung erklärt zum einen die Zielorientierung von Regeln, zum anderen wird die Vielschichtigkeit von Quellen solcher Regelungen verdeutlicht.
Es wird in dieser Arbeit der Begriff *Regulierung* verwendet. Regulierung beschreibt einen Prozess, der Aufgaben für die Definition, Ausführung und Überwachung von Regelungen beinhaltet. Ein regulierender Prozess umfasst zusätzlich die Koordinierung von gleichzeitig ablaufenden, geregelten Handlungen, um bei verschiedenen und gegensätzlichen Motivationen auf Grundlage geregelter Koordinationsweisen entstehende Probleme zu bewältigen bzw. aufzulösen.

2.2.1 Regelungen im sozialen Kontext

Regeln sind soziale Konstrukte. Gesellschaften ohne Regeln gibt es nicht [Hof16, S. 37].
Es besteht eine Pluralität von Normproduzenten und Normdurchsetzungsregimen [Sch15]. Regeln umfassen nicht nur Rechtsnormen, sondern greifen auch in nicht-rechtliche Bereiche ein.

2.2.1.1 Rechtliche Regelungen

In Gesellschaften besitzen viele Regeln qualifizierende Eigenschaften einer Rechtsnorm. *Rechtliche Eigennormativität funktioniert als Selbstreproduktion normativer Vorgaben aus geltendem Recht.* [Teu14, S. 213]

Definition 2.1 *Rechtliche Regeln*
Rechtliche Regeln enthalten als verbindlich geltende und meist sanktionsbewehrte Vorgaben und Rahmensetzungen für ein gedurftes, ermöglichtes oder gesolltes, aber auch für ein untersagtes Verhalten [Hof16, S. 38].

Recht normiert auch Institutionen sowie Strukturen als Voraussetzung der Möglichkeit normgerechten Verhaltens (des Handelns oder Unterlassens).

2.2.1.2 Nichtrechtliche Regelungen

Definition 2.2 *Nichtrechtliche Regeln*
Nichtrechtliche Regeln umfassen kulturelle oder moralische Normen, aber auch allgemeine Verhaltenserwartungen sowie spezifische technische Regeln [Hof16, S. 37].

Im Zeitalter der Informationstechnik hat sich *Code* als eine spezifische Steuerungsform nichtrechtlicher Regeln herausgebildet und gewinnt zunehmend an Bedeutung. Der Begriff *Code* bildet ein normierendes Element sowohl für die Software zur Gestaltung von Softwarearchitekturen als auch für digitale Kommunikationen [Les06]. Speziell sind es Algorithmen, die als konstituierende Methoden digitale Strukturen und Verhaltensformen bestimmen.
Beispiele sind Funktionen für die Filterung, Verteilung oder die Festlegung von Optionen im Kontext von Auswahlaktivitäten. Algorithmen können gesetzte Voreinstellungen und Rahmenbedingungen für bestimmte Zielsetzungen durchsetzen. Obwohl das technische Vorgänge sind, so werden im Zuge der Voreinstellungen wertende bzw. einschränkende Vorgaben umgesetzt. Aus funktionaler Sicht ist *Code* dem Prinzip einer Steuerung durch nichtrechtliche Regeln [Hof16, S. 41] zuzuordnen.
Diese Bewertung kennzeichnet ein für digitale Systeme grundlegendes und folgenreiches Problem. Die Anwendung von Codes im Bereich digitaler Systeme liegt damit außerhalb der Grundprinzipien rechtlicher Ordnung.

2.2.1.3 Regelungen in Organisationen

Das Vorhandensein und die Ausführung von Regelungen sichert in erster Zielstellung die tagtägliche Existenz einer Organisation. In zweiter Linie verfolgen Organisationen wirtschaftliche oder soziale Ziele. Die repetitive Anwendung eigener Regelungen ist der Garant für das Fortbestehen der Organisation.
Organisatorische Regelungen beschreiben, ordnen und reduzieren zugleich den möglichen Handlungsrahmen. Abgeleitet aus diesen Handlungsgrundsätzen bildet sich die *Organisationsstruktur* heraus.
Die Struktur einer Organisation konstituiert den Sinn von allen Handlungen [Luh87, S. 383]. Sowohl die *Aufbau-Struktur* als auch die *Prozess-Struktur* sind Ergebnisse regulativer Festlegungen. Die Strukturen bilden dabei Muster einer gewissen Dauerhaftigkeit, die dem Wechsel von Personal standhalten, wie einem gewissen Ausmaß des tatsächlichen Verhaltens, das Individuen beisteuern [Wei95].
Die aus den Regelungen abgeleiteten Organisationsstrukturen mit ihren Handlungen bilden die Elemente einer Organisation. Es ist dabei nicht entscheidend, welche konkrete Person eine Handlung ausführt, sondern in welcher *Rolle* die Person ihre Aufgaben in der Organisation erfüllt. Rollen bilden die Bestandteile der Aufbauorganisation, die sich wiederum in Ordnungsprinzipien auf Ebene von Organisationseinheiten gliedern können.
Für die Durchsetzung von Regelungen bedarf es fortdauernder Kommunikation. Kommunikation kann nicht als eigentlicher Handlungsakt verstanden werden. Kommunikation sind Ereignisse, die ein koordiniertes Handeln zwischen Rollen erst ermöglichen. Jede Rolle erfüllt eine Grundmenge endlich komplexer Einzelaufgaben. Die aus den Kommunikationen heraus zerlegten Handlungen verursachen gewollte Anschlusskommunikation [Luh87, S. 193]. Eine fortdauernde Kommunikation sichert den Fortbestand einer Organisation. Die Gesamtheit aller Regelungen einer Organisation bildet den Regulierungsrahmen, der für die Organisation verbindlich ist.

2.2.2 Regelungen im Cloud-Kontext

Konnten sich innerhalb einer nicht gekoppelten Organisationsstruktur, auf Grundlage durchsetzbarer und bestätigter Erwartungen, veränderliche, aber geregelte Strukturen und Handlungsabläufe herausbilden (z. B. etablierte Prozesse in einer Private Cloud), so sieht sich ein Cloud-Anwender im Rahmen einer Public-Cloud-Kopplung mit einem Risiko konfrontiert. Das Risiko besteht im Verlust seiner vollständigen Regulierungshoheit.
Die Kopplung zwischen beiden Seiten benötigt Kommunikation, um der einzigen Absicht eines Cloud-Anwenders eine Grundlage zu geben, seine erwarteten Handlungsabläufe zu koordinieren. Welcher tatsächliche Handlungsrahmen im Ergebnis möglicher Kommunikationen erreicht wird, bleibt jedoch für einen Cloud-Anwender unberechenbar (siehe Abschnitt 2.3).
Die strukturelle Unabhängigkeit beider Seiten erlaubt es einem Cloud-Anwender nicht, seinen selbstbestimmten Regulierungsrahmen einseitig und wirksam auf die Seite des Cloud-Anbieters zu erweitern. Kommunikation wird bisher einseitig als reiner Handlungsakt im Sinne des fachlichen Datenaustausches und der Prozesssteuerung unterstützt.
Bisherige Cloud-Modelle berücksichtigen keine Kommunikation zum Zweck einer geregelten Vermittlung und Durchsetzung von Cloud-Anwender-Erwartungen, die es zu erfüllen gilt. Ein Cloud-Anwender sieht sich mit dem Problem konfrontiert, dass er seine Regulierungsansprüche nicht in einer technisch interpretierbaren Form auf die Seite des Cloud-Anwenders übertragen und steuern kann.
Die Cloud-Anwendung führt gleichzeitig zu einer Verlagerung organisatorischer Regelungen in Bereiche digitaler Systeme. Die Umsetzung normativer Ausdrucksformen stützt sich auf softwarebasierte Methoden und Technologien. Damit besteht das im Abschnitt 2.2.1.2 gekennzeichnete Problem, dass softwarebasierte Regelungskonzepte in Form von Code-Strukturen keine rechtlichen normativen Ausdrucksformen repräsentieren. An dieser Stelle wird dem Cloud-Anwender faktisch keine Grundlage angeboten, seine regulierenden Interessen in Form von Softwarestrukturen rechtsverbindlich zu gestalten und durchzusetzen.
Das bestehende Problem bezieht den Aspekt der Transformation von organisatorischen Regelungen in digitale normative Repräsentationen (Softwarebausteine) mit ein. Der Übergang von geregelten organisatorischen in digitale Prozesse benötigt Schnittstellenfunktionen, deren Fähigkeiten durch rechtliche Regelungen sicherzustellen sind. Abweichungen während der Transformation führen zu einem Bruch normativer Zielstellungen bezogen auf einen Gesamtprozess.

2.3 Erwartungen und Unbestimmtheit von Handlungen

Der Abschnitt führt in eine Problemstellung ein, die sich mit der Durchsetzbarkeit von Erwartungen bezüglich regulativer Definitionen auseinandersetzt. Die Problemstellung ist ein Teil der Spezifik gekoppelter Strukturen und stellt sich grundsätzlich bei der Durchsetzung von Regelungen in kooperierenden Organisationsstrukturen.

2.3.1 Erwartungen im sozialen Kontext

Während die Regelungen die Instrumente zur Steuerung koordinierten Handelns darstellen, manifestieren sich in Erwartungen vorausgedachte Zielstellungen. Erwartungsstrukturen bilden Bedingungen

anschlussfähigen Handelns und erzeugen Handlungssinn [Luh87, S. 392]. Mit der Rollenbeschreibung kommuniziert die Organisation ihre *Erwartungen* an handelnde Akteure.
Es ist nicht allein wesentlich, dass Kooperationen mit ihren Beziehungen durchgesetzt werden, sondern dass sich daraus tatsächlich erwartete Handlungsmuster zusammen mit ihren Ergebnissen ableiten. Im Idealfall erfüllen die Handlungsmuster die gestellten Erwartungen durch konsequente Berücksichtigung geltender Regelungen. Die Entscheidung darüber, welche Anschlusshandlungen einer Kommunikation folgen, sind weder vorher bestimmbar, noch können sie durch Kommunikation erzwungen werden.
In einer kooperativen Ausführung von Handlungen bleibt es jeder Seite überlassen, ob und welche Handlungsoptionen im Zuge eigener Entscheidungen gezogen werden. Kommunikation schafft lediglich Bedingungen eines kooperativen Handelns, jedoch hat sie keinen Einfluss auf ein erwartetes Anschlusshandeln. Kommunikation dient jedoch als Voraussetzung für ein Verstehen des Sinns. Die Annahme oder Ablehnung einer Handlung entsteht im Ergebnis der Selektion des Adressaten als Prämisse eigenen Verhaltens [Luh87, S. 203]. Ob Kommunikation zu einem erwarteten Anschlusshandeln führt, hängt von folgenden Faktoren ab:

Semantik

Mitteilungen, die im Rahmen einer Kommunikation empfangen werden, müssen durch den Akteur verstanden werden. Die Bedeutung von Mitteilungen muss sich erschließen. Das verlangt nach einer für die Zwecke ausgerichteten sprachlichen Grundmenge an Begriffen, um die Komplexität zu reduzieren.

Erreichbarkeit

Die Mitteilungen, die für eine Koordinierung von Handlungen bestimmt sind, müssen ihren Empfänger erreichen. Das setzt die Anwesenheit von Personen voraus. Die Anwesenheit verlangt zusätzlich eine Selektion. Mitteilungen können ihren kommunikativen Zweck nur erfüllen, wenn sie bewusst ausgewählt wurden.

Erfolg

Das Verstehen einer Mitteilung vorausgesetzt, führt diese in jedem Falle zu einem neuen Kenntnisstand der Person. Die Information wurde aufgenommen und führt zu einem Zustandswechsel in Bezug auf den Wissensstand vor dieser Mitteilung. Ein Erfolg wird daran gemessen, ob sich aus diesem Zustand heraus ein Handlungsmuster entwickelt, welches die Kommunikation im Sinne gestellter Zielstellungen fortsetzt.

Innerhalb einer Organisation gilt ein Grundprinzip. Erwartungen, ob sie erfüllt wurden oder nicht, müssen erklärbar sein. Das Problem kann innerhalb einer Organisation positiv oder negativ sanktioniert werden. An dieser Stelle entsteht ein Regulativ und führt mit hoher Wahrscheinlichkeit zu regelkonformen Abläufen. In Bezug auf den Aspekt *Erwartungen* erfüllen die im Abschnitt 2.2.1.1 beschriebenen rechtlichen Regulierungen eine bestimmende Funktion.
Die Aufgabe von Recht in modernen Rechtsstaaten ist es, ein frei gewähltes und selbstbestimmtes Verhalten zu ermöglichen. Dennoch ist dieses Recht auch häufig mit der Erwartung verbunden, dass mit dem Verhalten zugleich mittelbar normativ erwünschte Wirkungen erreicht werden [Hof16, S. 38]. Das entspricht den Zielen einer Organisation, kooperatives Handeln für das Erreichen entsprechender Wirkungen auszurichten.
Die rechtsverbindliche Formulierung solcher Erwartungen als verhaltenslenkende Methode ist der Versuch, dem oben beschriebenen Prinzip der Selektion des Adressaten, die Erfüllung eines normativen Ver-

haltens anzunehmen oder abzulehnen, entgegenzuwirken. Rechtsnormen sind darauf angewiesen, durch Aufklärung, Sanktionen oder durch Förderung von Einsichten die Einlösung der normativen Erwartungen zu unterstützen [Hof16, S. 39].

2.3.2 Erwartungen im Cloud-Kontext

Für die Anwendung der Prinzipien im Cloud-Kontext ist es bedeutend, die konstituierenden Funktionen zur Entwicklung koordinierter Handlungsmuster zu verstehen. Es können Erwartungen ausgesprochen werden, doch sind zukünftige Handlungen weder vorhersehbar, noch direkt beeinflussbar, was sich für einen Cloud-Anwender insbesondere im Public-Cloud-Bereich als weiteres Risiko darstellt.
Theoretisch könnte das zu einem grundsätzlichen Verzicht neuer struktureller Kopplungen führen. Dieser Aspekt wird in der gegenwärtigen großen Zurückhaltung von potenziellen Anwendern in Bezug auf die Einführung und Nutzung von Cloud-Technologien sichtbar.
Ein entscheidender Faktor zur Senkung dieses Risikos liegt in der Sicherstellung von Verständnis, also einer Erfassung der Bedeutung übermittelter Informationen. Die von einem Cloud-Anwender formulierten Regulierungsaspekte als auch Handlungsmuster erfordern formale Sprachmittel, die mit ihrer Expressivität den Gegenstand von Regulierungsaspekten in eine Form eindeutiger, semantischer Interpretierbarkeit überführen.
Auch die Annahme, dass ein Cloud-Anwender in der Lage ist, mit geeigneten Sprachmitteln seine Erwartungen an die auszuführenden Dienstleistungen formal korrekt zu spezifizieren, löst das Problem nur unvollständig. Das Prinzip der Unbestimmtheit von Handlungen macht es einem Cloud-Anwender schwer, eine direkte Beeinflussung auf die Durchsetzung seiner Ansprüche vorzunehmen.
Das Prinzip *Erreichbarkeit* als bestimmendes Element bildet eine weitere Problemstellung und ist unmittelbar verknüpft mit dem Problem der Vertrauenswürdigkeit. An dieser Stelle sind Lösungsansätze gefordert, die im Zusammenhang mit struktureller Kopplung eine Etablierung vertrauenswürdiger und verfügbarer Cloud-Entitäten gewährleisten. Dieser Ansatz bindet verpflichtend den Cloud-Anbieter mit ein, um technische Rahmenbedingungen zu schaffen, so dass derartige Einflussnahmen vonseiten des Cloud-Anwenders möglich werden.
Die Bereitstellung vertrauenswürdiger Cloud-Entitäten beeinflusst den Faktor *Erfolg* im Sinne der Durchsetzung von Handlungsmustern. Der Grad der Vertrauenswürdigkeit präsentiert sich unter anderem durch die zuverlässige Ausführung von Regulierungs- und Handlungsinstruktionen. Nur die Vertrauenswürdigkeit verschafft dem Cloud-Anwender die Gewissheit, dass seine Regulierungsansprüche tatsächlich durchgesetzt werden.
Unter diesen Bedingungen ließen sich Sprachmittel anwenden, die den Grad der Kopplung als Ausdruck einer garantierten Anschlusshandlung festlegen. Zwischen einem Cloud-Anwender und einer vertrauenswürdigen Cloud-Entität ließen sich darüber unterschiedliche Abstufungen kooperativer Kopplungen vereinbaren.

2.4 Konformität, Abbildung von Regelungen

2.4.1 Konformität im sozialen Kontext

Konformität ist ein vergleichender Begriff und lässt sich nicht ohne einen Bezugspunkt bewerten. Die Gesamtheit von Regelungen einer Organisation stellt diese Ordnung her und bildet den Bezugspunkt (siehe Abbildung 2.2).
Maßnahmen zur Überprüfung der Konformität können Abweichungen ausgeführter Handlungsmuster von diesem Bezugspunkt feststellen. Eine Konformität ist erfüllt, wenn keine Abweichungen im Rahmen der Ordnung nachweisbar sind. Der Begriff Konformität entsteht durch die Fähigkeit einer Organisation zur Selbstreferenzierung. Abweichungen können nur im kontinuierlichen Prozess der Selbstbeobachtung von Handlungsmustern festgestellt werden [Luh11, S. 72].

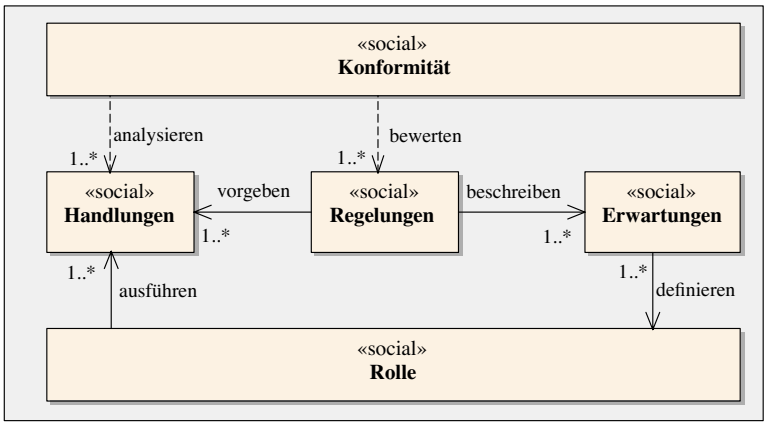

Abbildung 2.2: Struktureller Begriff für Konformität

Die Durchsetzung und Überprüfung von Konformität stellt an eine Organisation existenzsichernde Grundanforderungen, die es zu erfüllen gilt. Der Begriff Konformität erhält seine Bedeutung erst im Zusammenhang mit handelnden Personen. Inwieweit der Sinn und die Ausrichtung von Handlungen von einer einzelnen Person verstanden wurde, kann nur am Handlungsrahmen selbst beobachtet werden. Die Handlungen sollten gemäß den formulierten Erwartungen ausgerichtet sein.
Konformität ist nur nachweisbar, wenn über die Gesamtstruktur einer Organisation hinweg Möglichkeiten bestehen, Handlungsmuster während ihrer Ausführung zu messen bzw. zu bewerten. Darüber kann ein Vergleich entstehen. Die Messbarkeit bedingt strukturelle Einheiten innerhalb der Organisation. Der Umfang an Maßnahmen zur Feststellung von Konformität entwickelt sich mit derselben Dynamik wie die Änderungen in der Organisationsstruktur selbst.

2.4.2 Konformität im Cloud-Kontext

Der Einsatz von Cloud Computing als digitales System vergrößert die Schwierigkeit einer Konformitätsabsicherung. Mit der Auslagerung von Teilen der Geschäftsprozesse in ein Cloud-System sind Funktionen zur Gewährleistung von Konformität auf die Seite des Cloud-Anbieters zu verlagern. Zusätzlich besteht auch das im Abschnitt 2.2.2 formulierte Problem, dass bereits Transformationen von organisato-

rischen Regelungen in digitale normative Methoden sicherzustellen sind. Die Konformitätsbetrachtung muss auf die Schnittstellen zur Transformation erweitert werden.
Der Prozess zur Beobachtung von Abweichungen ist selbst einer Regulierung und damit einer Konformitätsbewertung unterzogen. Die Regulierung beschreibt den Funktionsrahmen, unter dem der Konformitätsnachweis geführt wird. Der Nachweis der Konformität ist damit gleichermaßen mit einer *Erwartung* verknüpft, ebenso wie die Prozesse zur Durchsetzung von Regelungen und zur Ausführung der vom Cloud-Anwender definierten Handlungsmuster. Demzufolge können hier die gleichen Risiken benannt werden, die bereits im Abschnitt 2.3.2 identifiziert wurden.
Eine fehlende Vertrauenswürdigkeit im Prozess zur Konformitätsbewertung entzieht ihr die autoritäre Grundlage. Die Bewertung erscheint ohne Bedeutung. Gleichermaßen wie bei der Durchsetzung von Regelungen sind Lösungen gefordert, die eine Beeinflussung auf die Konformitätsbewertung durch einen Cloud-Anwender möglich machen. Auch an dieser Stelle können vertrauenswürdige Cloud-Entitäten einen technischen Lösungsansatz bieten.

2.5 Thesen

These 1 – Allgemeines Grundprinzip für Regulierung

Im Rahmen dieser Arbeit soll am Beispiel eines Cloud-Systems belegt werden, dass sich aus den Erscheinungsformen für eine differenzierte Regulierung zur Anwendung technischer Systeme ein allgemeines *Grundprinzip der Regulierung* ableiten und beschreiben lässt. Die Verallgemeinerung des Ansatzes schließt die Beschreibung allgemeiner Prinzipien für eine strukturelle, qualitative und verhaltensorientierte Regulierung ein.

These 2 – Formalisierbarkeit von Vertrauen

Eine erfolgreiche Durchsetzung von Regulierungszielen in einem verteilten und föderativen Strukturansatz ist grundsätzlich mit der Anwendung eines allgemeinen Prinzips vertrauenswürdiger Vermittlung von Regulierungsabsichten verbunden. Die vorliegende Arbeit belegt, dass sich das zugrundeliegende *Prinzip von Vertrauen* mit Mitteln einer Konzeptualisierung formalisieren lässt und darüber für reale technische Systeme bestimmbar wird.

3 | Analyse

Die Analysephase untersucht und bewertet bestehende Konzepte, Projekte und Technologien, die in Bezug auf die Realisierung vertrauenswürdiger IT-Konzepte entsprechende Vorarbeiten und Grundlagen liefern. Folgende Zielstellungen bestimmen die weitere Betrachtung:

Kategorien regulativer Aufgaben

Das Thema *Regulierung* durchzieht unterschiedliche Bereiche innerhalb einer Cloud-Domäne. Die Analyse vollzieht eine Klassifikation und führt Kategorien zur Repräsentation von Regulierungszielen ein. Aus den Kategorien leiten sich Policy-Klassen ab. Die Verwendung von Kategorien erlaubt eine Systematisierung in Bezug auf bestehende Einzelanforderungen sowie eine Gesamt- und Vollständigkeitsbetrachtung notwendiger regulativer Zielstellungen.

Anwendungsfälle

Anwendungsfälle dienen als Instrument, um bestehende Anforderungen weiter zu präzisieren und erweitern methodisch das Prinzip der Klassifikation von Regulierungszielen. Die ablauforientierte Beschreibungsform arbeitet prozessbegrenzende Elemente der Regulierung heraus. Im Abschnitt 5 werden ausgewählte Anwendungsfälle aufgegriffen, um Prinzipien vertrauensbildender Regelungen technisch nachzuweisen.

Spezifikationsgrundlagen

Anforderungen an die Regulierung bestimmen Designaspekte im Entwurf einer Referenzarchitektur, so dass regulative Zielstellungen funktional erfüllt werden können. Die Gesamtheit der Einzelanforderungen sind im Anhang III gelistet. Sie bilden die Grundlage zur Ableitung und Spezifikation entsprechender Policies.

State-of-the-Art-Bewertung

Die Vielschichtigkeit im Bereich der Regulierung führt zu einer gleichermaßen geführten State-of-the-Art-Bewertung. Es werden bestehende Arbeiten und Standards im Bereich der Regulierung betrachtet, die sich dem Aspekt Vertrauensbildung nähern oder diesen direkt aufgreifen. Sie umfasst eine vergleichende Analyse mit aktuellen Projekten zu diesem Thema.

3.1 Anforderungen

Bisherige Untersuchungen fokussierten auf Betrachtungen von Sicherheitsaspekten von Public-Cloud-Systemen. Die Studie Streitberger et al. [SR09] legt eine Taxonomie vor, die als Leitfaden für eine

Sicherheitsstrategie im Public-Cloud-Bereich verwendet werden kann. Die Taxonomie ist eine Zusammenfassung bestehender Strukturierungsarbeiten im Bereich der Cloud-Sicherheit und stützt sich auf Vorarbeiten von Bardin et al. [Bar+09a], die später von Arnold et al. [Arn+11] erweitert wurden.
Im Unterschied zur Taxonomie in Streitberge et al. [SR09] wird die Software zur Virtualisierung von Betriebssystemumgebungen der Plattformschicht zugeordnet. Für eine detailliertere Definition von Regulierungsanforderungen wird die Kategorie *Anwendung & Plattform* in die Kategorien *Plattform & Laufzeit* und *Anwendungen & Service* aufgeteilt. Die erweiterte Klassifizierung ist in Tabelle 3.1 dargestellt.

Infrastruktur	**Plattform & Laufzeit**	**Anwendungen & Services**	**Verwaltung & Betrieb**	**Compliance**
Physikalische Sicherheit / GEO-Lokalisierung	OS Sicherheitsmodell	Servicesicherheit	Phasen der Servicenutzung	Datenschutz
Sicherheit VMM (Virtual Machine Monitor)	Middleware Sicherheit	Anwendungssicherheit	Prüfung	Risikomanagement
Sicherheit VM (Virtual Machine)	Laufzeit-Datensicherheit	Prozesssicherheit	Identitäts- und Berechtigungsverwaltung	Rechtlicher Rahmen
Netzwerksicherheit	Sicherheit als Service	Anwendungs-Datensicherheit	Schlüsselverwaltung	Governance
			Interoperabilität & Portabilität	Klassifizierte Informationen

Tabelle 3.1: Taxonomie der Sicherheitsaspekte von Cloud Computing

Die hier eingeführten Ordnungsprinzipien orientieren sich an einem vollständigen Cloud-Architekturmodell. Dadurch ist eine ganzheitliche Betrachtung gewährleistet. Darüber hinaus werden betriebliche Aspekte (*Verwaltung*) und regulatorische Rahmenbedingungen (*Compliance*) berücksichtigt.
Nachfolgend werden erforderliche Regulierungsziele für jede Kategorie herausgearbeitet und beschreiben einen Anforderungsrahmen für die Durchsetzung einer Trust Policy.

3.1.1 Infrastrukturschicht

Die statische Sicherheit von Hardwarebausteinen wird durch die Qualität der *Firmware* (CPU, Chipsatz, BIOS, ROM) bestimmt. Die verhaltensorientierte Sicherheit der infrastrukturellen Hardwareplattform wird durch die Qualität der Software für den Systemstart (*Boot-Software*) und Software zur Virtualisierung (Virtual Machine Monitor, VMM) bestimmt.
Moderne Hardwareplattformen verwenden bereits Technologien, die in Verbindung mit externen Attestierungsdiensten eine vertrauenswürdige Überprüfbarkeit von Hardwareplattformeigenschaften erlauben [FG13; Gre12; YC14]. Gleichzeitig bestehen Konzepte, um auf Ebene der Infrastrukturschicht einen regulierten Start der virtualisierten Systeme auszuführen. Policykonzepte entscheiden darüber, unter welchen infrastrukturellen Voraussetzungen ein Systemstart ausgeführt werden kann und welche Eigenschaften die dafür verwendeten Komponenten nachweisen müssen.

3.1.1.1 Hardwarebasierte Geo-Lokalisierung

Die Möglichkeit zur Bestimmung und zum Nachweis geografischer Koordinaten (Geo-Koordinate) einer installierten IT-Plattform bildet einen Bestandteil regulatorischer Anforderungen. Auf der Grundlage kryptographisch gesicherter geografischer Identifikatoren (*GeoTagging*) kann über die Mechanismen der Attestierung Auskunft darüber gegeben werden, in welchem geografischen Bereich eine IT-Plattform betrieben wird.
Speziell im Bereich der Virtualisierung bilden Geo-Koordinaten ein hinreichendes Identifikationsmerkmal, um Cloud-Dienste bezüglich ihrer Laufzeitumgebung und physischen Präsenz sicher orten zu können. Die Regulierung von Informationsprozessen auf Grundlage gesetzlicher Richtlinien fordert eine eindeutige Identifizierbarkeit von Cloud-Diensten, um Ausführungsbereiche ihrer Verarbeitungsprozesse wirksam steuern zu können.
Der Grad der Vertrauenswürdigkeit wird durch die Verfahren bestimmt, die eine sichere und überprüfbare Bindung geografischer Koordinaten an eine konkrete Cloud-IT-Architektur vornehmen [Ban+12].

3.1.1.2 Virtual Machine Monitor

VMM bildet neben seinen zentralen funktionalen Steuerungsaufgaben den Ausgangspunkt für die architekturelle Systemsicherheit. Die Sicherheit und Verlässlichkeit eines VMM bestimmt das Sicherheitsniveau der darüber gesteuerten virtualisierten Systeme.
Für komplexe virtualisierte Systeme kann eine korrekte Funktionsweise nicht mit einer hinreichenden Wahrscheinlichkeit gewährleistet werden. In der *Common Vulnerabilities and Exposures* Statistik [Cyb16] manifestiert sich die Aussage, dass von der Existenz einer Vielzahl sicherheitsrelevanter Fehler ausgegangen werden muss.
Die Anzahl der *Source lines of code* (SLOC) muss sehr gering gehalten werden (1.000 SLOC, max. 10.000 SLOC), um mit heutigen verfügbaren Methoden einen vertrauenswürdigen Code (*Trusted Computing Base*, TCB) für die sichere Ausführung kritischer Funktionen zu schreiben. Die Vertrauenswürdigkeit einer TCB leitet sich aus den Code-Eigenschaften ab, deren Korrektheit und eine niedrige Fehlerdichte nachgewiesen werden können.
Die TCB ist Bestandteil eines Design-Pattern zur sicherheitsrelevanten Entkopplung komplexer Systeme und zur Kontrolle ihrer Kommunikationsbeziehungen [Rus81]. Die gezielte Abtrennung und gleichzeitige Ausführung vertrauenswürdiger Softwarekomponenten von nicht vertrauenswürdigen Systemen durch eine TCB begrenzt Auswirkungen von Systemfehlern auf das verursachende, nicht vertrauenswürdige System.
Der Einsatz von Separation Kernel [Hoh+04; IAD07] ermöglicht eine konfigurierbare Isolierung und Ausführung von Subjekten (Virtualisierungssoftware) durch eine statische Partitionierung der Ressourcen des Systems. Definierte Sicherheitsbeziehungen kontrollieren die Interaktionen zwischen isoliert laufenden virtualisierten Systemen.
Minimalisierte TCBs (wenige tausend Zeilen) erlauben eine kontrollierte Separierung virtualisierter Maschinen [BR13] unter Anwendung einer INTEL-basierten Hardwarevirtualisierung [Int16c; Int16b; Int16a]. Dieser Lösungsansatz unterscheidet sich wesentlich von Virtualisierungslösungen wie KVM, VirtualBox [Ora16] und VirtualBox auf Nova [Fes15].

3.1.1.3 Netzwerksicherheit

Auf infrastruktureller Ebene erfolgt die Integration der Außenwelt. Unterschiedliche Kommunikations- (Netzwerk) und Speicherungsschnittstellen (Storage) sind in Bezug auf Funktions- und Sicherheitseigenschaften zu integrieren.

Regulierungsziele als Festlegungen über:

RZ1: Autorisierte Code-Eigenschaften der *Firmware*

RZ2: Authentizität der *Firmware*

RZ3: Autorisierte Code-Eigenschaften der *Bootstrapping-Software*

RZ4: Authentizität und Eigenschaften der *Bootstrapping-Software*

RZ5: Minimalisierte und überprüfbare Code-Eigenschaften der *Kernel*-Software bzw. *VMM*-Software

RZ6: Authentizität der *Kernel*-Software bzw. *VMM*-Software

RZ7: Authentizität und Eigenschaften der Virtualisierungssoftware

RZ8: Autorisierte Eigenschaften von Kommunikations- und Speicherungsschnittstellen

RZ9: Authentizität physischer Kommunikations- und Speicherungsschnittstellen

3.1.2 Plattform-/Laufzeitschicht

Die Sicherheit in einer virtualisierten Laufzeitschicht wird durch die eingesetzte Virtualisierungstechnologie und durch das darin virtualisierte Betriebssystem bestimmt. Beide Aspekte müssen ganzheitlich, mit ihren sicherheitsrelevanten Abhängigkeiten in die Analyse einbezogen werden.

3.1.2.1 Virtualisierungstechnologie

Eine *Virtual Machine* (VM) stellt virtualisierte Ressourcen (*network*, *storage*, *memory*, *compute-power*) für das Gastbetriebssystem bereit und gleichzeitig Sicherheitsfunktionen zur Berechtigungssteuerung auf diese Ressourcen. Ein integriertes Betriebssystem (*Operating system*, OS) verwaltet über die OS- spezifischen Mechanismen der VM die bereitgestellten virtualisierten Hardwareressourcen. Die Anforderungen an die Regulierung beziehen sich auf die Auswahl und Sicherstellung von Eigenschaften und Verfahren zur Bereitstellung einer VM.

3.1.2.2 OS-Sicherheitsmodell

Die Betriebssystemsicherheit muss regulativ einem Sicherheitsmodell unterstellt werden [Ott07; FKC03; NSA87]. Das Sicherheitsmodell beschreibt Mechanismen zur Zugriffssteuerung für die kontrollierte Steuerung externer und interner Datenflussbewegungen.
Ein regelbasiertes Modell für den Schutz der Vertraulichkeit ist das Bell-LaPadula-Sicherheitsmodell [BL73]. Das Regelwerk erlaubt die Anwendung überprüfbarer Policies zur Steuerung eines vertraulichen

Informationsflusses und kann durch einen Anwender nicht verändert werden. Im Gegensatz dazu kann mit einem Biba-Modell [Zha09] die Integrität von Informationsflüssen geregelt werden.
Die Vertrauenswürdigkeit eines realen bzw. virtualisierten Betriebssystems lässt sich nicht ohne ein zugrundeliegendes Sicherheitsmodell bestimmen.

3.1.2.3 Datensicherheit der Laufzeitschicht

Für die Datensicherheit sind Funktionen eines OS-Sicherheitsmodells oder kryptographische Methoden anzuwenden. Die kryptographischen Methoden orientieren auf die Vertraulichkeit und Integrität der Daten. Eine weitere Zielstellung für den Einsatz kryptographischer Funktionen besteht in der sicheren Bindung klassifizierender Eigenschaften von Daten (z. B. Geheimhaltungsgrad) an ein ausgewähltes Datenobjekt. Kryptographische Methoden können den Regulierungsbereich auf Bereiche außerhalb einer OS-Umgebung (z. B. Datenimport, Datenexport) erweitern. Damit entstehen Voraussetzungen zur Gestaltung einer systemübergreifenden Vertraulichkeit und Datenhoheit.

File-basierte Daten

Die Datensicherheit bezieht sich auf Bereiche eines Filesystems bzw. auf selektiv auswählbare Fileobjekte (Dateien), die im Bereich eines Filesystems gespeichert werden. Die Ebene eines Filesystems bildet eine erste Autorisierungsstufe. Zugriffsregelungen auf einen Filesystembereich schaffen die notwendige Bedingung, um später selektiv auf Fileobjekte zugreifen zu können.

Datenbanken

Für Daten, die in Form von Datenbanktabellen gespeichert werden, kommen datenbankspezifische Sicherheitskonzepte zur Anwendung. Die Zugriffsverwaltung stützt sich auf ein benutzer- und rollenbasiertes Berechtigungskonzept des entsprechenden Datenbanksystems. Für die Regulierung der Datensicherheit sind Konzepte für eine durchgehende Benutzerabbildung zwischen System- und Datenbenutzer mit in die Policydefinition einzubeziehen.

Datenhoheit

Anforderungen an die Lenkung der Datenhoheit ergeben sich aus datenschutzrechtlichen Forderungen [Rec14; COM12], die Rechtsgrundlagen für die Erhebung, Verarbeitung und Nutzung von Daten beschreiben und das informationelle Selbstbestimmungsrecht regeln.

Der Begriff *Datenhoheit* wird in Bosesky et al. [Bos+13] systematisch abgeleitet und definiert. Es wird jedoch durch die Autoren kein hinreichendes Konzept für eine benutzerzentrierte Ausübung von Datenhoheit in einer digitalen Umgebung herausgearbeitet, um die gemäß der Definition formulierten Anforderungen erfüllen zu können.

Die Auswahl eines Verfahrens zur Datensicherheit leitet sich aus diesen Anforderungen ab.

Regulierungsziele als Festlegungen über:

RZ10: Autorisierte Code-Eigenschaften der *Virtual-Machine*-Software mit integriertem Gastsystem OS

RZ11: Authentizität der *Virtual-Machine*-Software mit integriertem Gastsystem OS

RZ12: Authentizität der Verbindung zwischen *Virtual-Machine*-Software und *VMM*-Software

RZ13: Regeln eines Sicherheitsmodells zur Absicherung der OS-Ressourcen

RZ14: Autorisierte Code-Eigenschaften der *Middleware*-Software

RZ15: Authentizität der *Middleware*-Software

RZ16: Zugriffsbedingungen auf Ressourcen von Datenbanksystemen

RZ17: Bedingungen für den sicheren Datentransfer (*Imp.*, *Exp.*) auf Ebene der *Middleware*-Software

RZ18: Bedingungen für den sicheren Zugriff auf Daten der *Middleware*-Software

3.1.3 Anwendungs-/Serviceschicht

Die Anwendungsebene bestimmt kooperative, zugesicherte Interaktionen zwischen Akteuren. Auf dieser Ebene werden fachspezifische Regulierungen zur Aufgaben- und Datenverteilung, Abstimmung und Entscheidungsfindung (*Compliance*) benötigt. Die Sicherheits- und Funktionseigenschaften von IT-Anwendungen und Services bestimmen die Sicherheit und Zuverlässigkeit bei der Umsetzung von Handlungsschritten und Datenverteilungen.

3.1.3.1 Anwendungssicherheit

Die Funktionen zur Integration von Cloud-Anwendungssystemen, gekoppelt mit Komponenten zur Speicherung und Verteilung von Daten, sind in das Konzept für eine regulierte Cloud-Anwendung einzubeziehen. Qualitätsstandards für die Einführung neuer Service- und Anwendungssysteme sind aufzustellen und einzuhalten. Zusicherungen von Authentizität und Unversehrtheit der Softwarebausteine benötigen eigene Kontrollfunktionen. Es sind nachvollziehbare Prozesse aufzusetzen, um ein qualifiziertes Konfigurations- und Sicherheitsmanagement sicherzustellen.
Die Integration von Service- und Anwendungssystemen darf sich nicht auf die Vertraulichkeits- und Verfügbarkeitsinteressen anderer Cloud-Anwender auswirken, deren Geschäftsprozesse gleichzeitig auf einem Cloud-System ausgeführt werden (*multi-tenant infrastructure domain*).
Das Sicherheitsmanagement umfasst die Regulierung, welche Akteure auf welche Ressourcen der Anwendungsebene (Applikationen und Anwendungsdaten) zugreifen dürfen.

3.1.3.2 Prozesssicherheit

Die Methoden zur Prozesssicherheit steuern das Zusammenwirken von Akteuren in einem Cloud-System und kontrollieren die dafür erforderlichen Steuerungs- und Datenflüsse. Die Koordinierungsaufgaben sind gemäß den gesetzten Zielstellungen festzulegen (*Workflow*) und enthalten Bestimmungen, welche Handlungsschritte einer Nachweisführung unterzogen werden.
Für hohe Anforderungen an die Zurechenbarkeit von ausgeführten Handlungsschritten (*Accountability*) sind während der Handlungsausführung Maßnahmen zur sicheren Authentifizierung eines Akteurs erforderlich. Die Sicherheitsfunktionen transformieren ausgeführte Operationen in nachweisbare Willensbekundungen eines Akteurs. Konzepte für eine sichere und vertrauenswürdige Protokollierung schaffen die Grundlage, die Effektivität von Regulierungsmaßnahmen zusichern zu können.

3.1.3.3 Datensicherheit der Anwendungsschicht

Neben den Aspekten zur Datensicherheit aus dem Abschnitt 3.1.2.3 benötigt die Anwendungsebene erweiterte Verfahren, um eine gesicherte und nachweisbare Kenntnisnahme und Weitergabe von Informationen zu gewährleisten.
Bei hohen Anforderungen an die Vertraulichkeit der Daten ist eine alleinige Anwendung von rollen- oder file-basierten Zugriffsrechten nicht ausreichend. In Bereichen mit klassifizierten Informationen (z. B. Verschlusssachen) werden Informationen mit kryptographischen Methoden (Verschlüsselung) verteilt und Zugriffe auf Informationen mit Methoden der geregelten Weitergabe kryptographischer Schlüssel autorisiert.
Die Datensicherheit auf Anwendungsebene ist eng mit der Prozesssicherheit verknüpft und führt zur Sicherstellung einer hohen Zurechenbarkeit und Vertraulichkeit über alle Bereiche einer Prozessstruktur. Die Weitergabe von Daten ist kontrolliert und fordert die nachweisbare Einwilligung beteiligter Akteure.

Regulierungsziele als Festlegungen über:

RZ19: Autorisierte Code-Eigenschaften der Applikations- bzw. Servicesoftware

RZ20: Authentizität der Applikations- bzw. Servicesoftware

RZ21: Authentizität der Verbindung zwischen Applikations- und Middleware-Software

RZ22: Vertraulicher Daten-Life-Cycle auf Ebene der Applikations- bzw. Servicesoftware

RZ23: Authentizität/Integrität von Anwendungs- und Konfigurationsdaten

RZ24: Authentizität handelnder Akteure (Akteurs-Eigenschaften)

RZ25: Bedingungen für den sicheren Zugriff durch Akteure auf Applikations- bzw. Servicesoftware

RZ26: Bedingungen für den sicheren Zugriff durch Akteure auf Anwendungs- und Konfigurationsdaten

RZ27: Zugelassener Handlungsrahmen handelnder Akteure (Akteurs-Verhalten)

RZ28: Nachweisführung über ausgeführte Aktivitäten (Non-repudiation)

3.1.4 Verwaltung/Betrieb

Die Ebene der Verwaltung erlaubt eine wechselseitige und regulierte Einflussnahme von Cloud-Anbieter und Cloud-Anwender auf die operativen Abläufe. Ein Cloud-Anwender-orientiertes Policy-Management bedeutet die Bereitstellung von Funktionen für das Deployment, das Sicherheitsmanagement und Monitoring einer Cloud-Serviceumgebung (*Phasen der Servicenutzung*).
Einem Cloud-Anwender stehen Überprüfungsfunktionen zur Verfügung, um sich konkrete Cloud-Anbieter-Bedingungen attestieren zu lassen, bevor unter eigener Kontrolle eine Anwendungs- und Serviceumgebung eingerichtet wird. Zwischen Cloud-Anbieter und Cloud-Anwender müssen attestierte Rahmenbedingungen für die Vertragslaufzeit festgeschrieben werden. Die Bedingungen beziehen sich auf die Zusicherung von Cloud-Ressourcen in Bezug auf definierte Qualitäts- und Funktionseigenschaften.

Eine wesentliche Fähigkeit besteht in der Übertragung von Rechten und Bedingungen für die eigenverantwortliche Servicegestaltung und dem Betrieb des Cloud-Systems durch einen Cloud-Anwender. Während dieser Phase der Cloud-Anwendung wird die Mitwirkung des Cloud-Anbieters ausgeschlossen. Der Cloud-Anbieter besitzt die Regulierungs- und Überwachungshoheit (*Prüfung*) über Datenverarbeitungsprozesse und ihre kooperative Anwendung. Dabei wird eine Cloud-Anbieter-definierte Sicherheitspolitik durchgesetzt.
Eine verteilte Cloud-Nutzungshoheit benötigt hierarchische Identity- und Policy-Managementkonzeptionen (*Identitäts- und Berechtigungsverwaltung*) als Grundlage zur Bescheinigung und Delegierung von Vertrauen zwischen Cloud-Anwender und Cloud-Anbieter. Eine einseitige Vertrauenspolitik auf Seite des Cloud-Anbieters entfällt unter diesen Bedingungen. Hierarchische Identity- und Policy-Managementkonzeptionen sind mit einer verteilten Konzeption für das Schlüsselmanagement (*Schlüsselverwaltung*) verknüpft und gewährleisten die eigenverantwortliche Durchsetzung von Vertraulichkeits- und Authentizitätsansprüchen.

Regulierungsziele als Festlegungen über:

RZ29: Verifizierbare Bescheinigung von Cloud-Infrastruktur- bzw. Anwendungsbedingungen

RZ30: Prozesse zur Aushandlung vertrauenswürdiger und funktionaler Cloud-Nutzungsbedingungen

RZ31: Maßnahmen zur Delegierung der Cloud-Nutzungshoheit an einen Cloud-Vertragspartnern

RZ32: Vertrauenswürdige Gestaltung von Cloud-Anwendungsumgebungen

RZ33: Vertrauenswürdige Beschreibung von Funktions- und Sicherheitsrichtlinien

RZ34: Vertrauenswürdige Ausführung von Funktions- und Sicherheitsrichtlinien

RZ35: Vertrauenswürdige Protokollierung operativer Aktivitäten

3.1.5 Compliance

Während sich bereits aus den ersten Kategorien regulative Anforderungen ableiten lassen, so gruppieren sich in der Kategorie *Compliance* explizite regulatorische Anforderungen, die sich aus gesetzlichen Vorgaben oder branchenspezifischen Prozess- und Verfahrensvorgaben mit unterschiedlichen Zielstellungen ableiten.

3.1.5.1 Governance

Dieser Begriff wird in der Literatur nicht einheitlich verwendet und bezieht sich allgemein auf *Formen und Mechanismen der Koordinierung zwischen mehr oder weniger autonomen Akteuren, deren Handlungen interdependent sind, sich also wechselseitig beeinträchtigen und unterstützen können* [Ben+12].
Es werden Abhängigkeiten und Bedingungen deutlich, unter denen eine koordinierte Vorgehensweise auszuführen ist. Ausgehend von einer gesellschaftlichen Koordinations- und Steuerungsbetrachtung, besteht die Herausforderung darin, Methoden für die Transformation in geregelte digitale Handlungsmuster zu entwickeln.

Die Anforderungen orientieren auf die Einhaltung elektronischer, prozessorientierter Kooperations- und Handlungsmuster, um ein sicheres und zielorientiertes Vorgehen innerhalb einer Organisation zu gewährleisten. Sie bestimmen damit maßgeblich den Aspekt der Prozesssicherheit (siehe Abschnitt 3.1.3.2).
Der Begriff *Governance* ist ein Ansatz, um komplexe Anforderungen zur Bearbeitung kollektiver Zielstellungen zu lösen. In Bezug auf einzelne geregelte Abläufe führt *Governance* eine übergreifende regulierende Funktion aus. Die Richtlinien orientieren auf die Vermeidung von Risiken. Die Herausbildung wirksamer digitaler Formen der Koordinierung von Handlungen bzw. Aktivitäten bestimmen zunehmend die Anforderungen an eine moderne Regulierung.
Im Abschnitt 2.2 stellt sich die Durchsetzung einer *Governance* unter den Bedingungen von Cloud-Anwendungen als zentrale Problemstellung dar. Die Konzepte vertrauenswürdiger Kommunikationen bilden die Grundlage, um Regulierungsziele für digitale Prozesse zusichern zu können. Sie bestimmen den Anforderungsrahmen innerhalb dieser Kategorie.

Regulierungsziele als Festlegungen über:

RZ36: Formen zur Definition von verbindlichen, interpretierbaren Richtlinien

RZ37: Regeln für den Aufbau und Begrenzung von Kooperationen

RZ38: Maßnahmen zur Bewältigung von Konflikten zwischen geregelten Handlungsinteressen

RZ39: Regeln für die Neugestaltung bestehender Kooperationsbeziehungen

RZ40: Konzepte für eine gezielte und bewusste Bekundung von Entscheidungen

RZ41: Formen für eine Nachweisführung kooperativer Entscheidungen

RZ42: Regeln über kooperative Verteilungen von Informationen

3.1.5.2 Klassifizierte Informationen

Besondere Anforderungen an die Regulierung entstehen bei der Verwendung klassifizierter Daten.
Unter dem Gesichtspunkt einer Vertraulichkeitsbewertung von Informationen kommt es zu fließenden Übergängen geltender Richtlinien, um die Einhaltung von Prinzipien zur Wahrung einer gesetzten Vertraulichkeit zu gewährleisten [BMI06].
Diesem Aspekt wird gegenwärtig noch eine untergeordnete Rolle beigemessen, der jedoch im Zuge digitaler Transformationsprozesse zu berücksichtigen ist. Ausgehend von den Festlegungen einer allgemeinen *Governance* verstärkt sich der Regulierungsbedarf, um Sicherheitsinteressen aufrechtzuerhalten. Die Erhöhung der Regulierung bezieht sich sowohl auf die Bedingungen, unter denen sich Akteure in kooperativen Prozessen beteiligen können, als auch auf die Formen zur Transparenz und Kontrolle von Informationsverteilungen.
Fließende Übergänge zwischen verschiedenen Regulierungsprinzipien erfordern in ein und denselben kooperativen Umgebungen Festlegungen für die sichere Transition zwischen den Regulierungsformen.

Regulierungsziele als Festlegungen über:

RZ43: Kennzeichnung verschiedener Klassifizierungsniveaus

RZ44: Eigenschaften von prozessbeteiligten Akteuren

RZ45: Richtlinien zur geregelten Verteilung klassifizierter Informationen

RZ46: Maßnahmen für den Nachweis ausgeführter Handlungsschritte

RZ47: Prinzipien zur Änderung von Klassifizierungsmerkmalen

RZ48: Prinzipien zur Kenntnisnahme klassifizierter Informationen

RZ49: Prinzipien für den Nachweis über bestehende klassifizierte Informationen

3.1.5.3 Datenschutz

Neben den funktionalen Anforderungen und Sicherheitsanforderungen [BSI14; ISO13; CC12c; Arn+11] bestehen Anforderungen an den Datenschutz [Rec14; Gru11; Höl+15; COM12]. Während sich Sicherheitsanforderungen auf den Schutz von Informationswerten (*Information Assets*) richten, so formulieren die Bestimmungen an den Datenschutz Maßnahmen für den Schutz der Persönlichkeitsrechte des Einzelnen beim Umgang mit seinen personenbezogenen Daten [Wit10]. Diese beiden Aspekte stehen im Mittelpunkt der hier geführten Analyse (siehe Abbildung 1.1).
Im Bereich der Cloud-Systeme entstehen besondere Herausforderungen an den Datenschutz, da sich gleichzeitig verschiedene Nutzerkreise (*Multi-Tenancy*) [ZB15] mit unterschiedlichen Nutzungsinteressen Ressourcen und Services einer Cloud-Plattform teilen.

Regulierungsziele als Festlegungen über:

RZ50: Verifizierbare Bescheinigung von Cloud-GEO-Informationen [Ban+12]

RZ51: Erhebungs-, Verarbeitungs- und Nutzungsbefugnisse personenbezogener Daten

RZ52: Ablaufstrukturen zur Erhebung, Verarbeitung und Nutzung personenbezogener Daten

RZ53: Nachweisführung über die Kenntnisnahme von personenbezogenen Daten

RZ54: Kontrollfähigkeit von Betroffenen über die Erhebung, Verarbeitung und Nutzung personenbezogener Daten [aca13]

RZ55: Kontrollfähigkeit von Betroffenen über die Verteilung personenbezogener Daten

RZ56: Anwendung von Privacy Design Pattern zur Gestaltung von Cloud Services [Pea09]

3.1.6 Zusammenfassung der Regulierungsziele

Bereich	Regulierungsziele
RZ1 - RZ9	Netzwerksicherheit
RZ10 - RZ18	Datensicherheit der Laufzeitschicht
RZ19 - RZ28	Datensicherheit der Anwendungsschicht
RZ29 - RZ35	Verwaltung/Betrieb
RZ36 - RZ42	Governance
RZ43 - RZ49	Klassifizierte Informationen
RZ50 - RZ56	Datenschutz

Tabelle 3.2: Übersicht der Regulierungsziele

Eine Beschreibung von Anforderungen an ein vertrauenswürdiges Cloud-System befindet sich im Anhang III. Die im Rahmen der Analyse erhobenen Anforderungen bilden die Grundlage für die Ableitung von Regulierungszielen.

3.2 Anwendungsfälle einer Multi-User-Cloud-Umgebung

Eine große Vorarbeit zur Herausarbeitung von Anforderungen zur aktiven Gestaltung vertrauenswürdiger Cloud-Architekturen und Prozesse leistete die *Trusted Computing Group* (TCG) mit ihren Standards für Multi-User-Cloud-Umgebungen [Tru11b; Tru13b].
Während die TCG ihre Anwendungsfälle in *Generic*-, *Provider*- und *Consumer*-Use Cases unterteilt, so wurde im Rahmen der Analyse eine sich am Lebenszyklus zur Entwicklung von Vertrauen (*Trust Establishment*) ausgerichtete Gruppierung vorgenommen:

- Aufbau einer Vertrauensbasis (*Establish Trusted Context*)
- Aufbau einer vertrauenswürdigen Kooperationsbasis (*Establish Cooperated Trustworthy Context*)
- Ablauf vertrauenswürdiger Kooperationen (*Trustworthy Cooperation*)
- Änderung von Regulierungsanforderungen (*Modify Trusted Context Policy*)
- Überprüfung Compliance-Status (*Audit Trusted Context Policy*)

Ausgehend von einer Vertrauensbasis (*Root of Trust*) kann Vertrauen auf verschiedene Bereiche delegiert werden (*Delegation of Trust*) und ein verteiltes Konzept bilden. Diesem Ansatz folgend, fand eine Verallgemeinerung bei der Bewertung der vorliegenden Anwendungsfälle statt, um die Kernansätze für eine policybasierte Entwicklung von Vertrauen herauszuarbeiten und die Grundanforderungen sichtbar zu machen.
In den folgenden Abschnitten werden dafür entsprechende Anwendungsfälle (UC – Use case) vorgestellt. Methodisch werden Anwendungsfalldiagramme [Hit05, S. 174] als Form der Verhaltensmodellierung eingesetzt. Die Anwendungsfälle repräsentieren erste Design-Prinzipien und beschreiben grundlegende Verfahren zur Steuerung und Regulierung einer vertrauenswürdigen Cloud-Architektur (*Trust Management*).

3.2.1 TCG-Konzepte und Definitionen

Der Begriff *Trust* wird als ein Akzeptanzprozess oder als eine Erwartung beschrieben, dass bestehende Sicherheitsrisiken wirksam durch entsprechende Maßnahmen minimiert oder beseitigt werden.

Definition 3.1 *Trust*
Trust can be better thought of as acceptance of risk mitigation as sufficient. The degree of mitigation should exceed the level of risk exposure. If the mitigations are sufficient to address the risks then a solution can be described as trustworthy in that context [Tru13b, S. 7].

Darüber hinaus wird der TCG-Begriff *Policy* eingeführt. Als Policy wird eine Menge von überprüfbaren Beschreibungen definiert, die es erlauben, die Wirksamkeit einer Risikominimierung festzustellen und damit eine vertrauensbildende Maßnahme repräsentieren.

Definition 3.2 *Policy*
Policy in this case is a set of testable statements describing evaluation of the level of mitigation necessary to address the risk and establish trust [Tru13b, S. 7].

Zur Repräsentation zu schützender Ressourcen bzw. Informationen wird in der TCG-Terminologie der Begriff *Asset* verwendet.

Definition 3.3 *Asset*
A functional IT component available for use within a Trusted Systems Domain [Tru13b, S. 8].

Die Definition bezieht sich auf allgemeine Ressourcen einer Domäne.
Darüber hinaus wird der Begriff *Compliant Asset* verwendet. Unter einem *Compliant Asset* werden Ressourcen verstanden, deren Eigenschaften hinsichtlich bestimmter Kriterien bestätigt werden konnten.

Definition 3.4 *Compliant Asset*
An asset that has met the pre-determined criteria for use within the Trusted Systems Domain [Tru13b, S. 8].

Ein zentrales Konzept, das sich aus der Terminologie der TCG ableitet, ist das Konzept einer *System Domain*. Dieser Begriff wird zu einer *Trusted Systems Domain* verfeinert.

Definition 3.5 *Trusted Systems Domain*
A logical grouping containing infrastructure assets, service providers (operators), users, applications and information where a trusted context has been established and governed by a consistent set of operational and security policies [Tru13b, S. 10].

Der Begriff überführt den allgemeinen Cloud-Architekturbegriff in einen regulierten Cloud-Sicherheitsbereich. Die Regulierungsziele diese Bereiches werden durch operationelle Policies und Sicherheitspolicies eines *Consumers* und eines *Providers* beschrieben. Eine vordergründige Zielstellung der Analyse ist die Herausarbeitung der Kernprinzipien zur Transformation einer Cloud-Domäne zu einer vertrauenswürdigen Cloud-Domäne.
Die TCG beschreibt Anforderungen an einen *Trusted Context* als Grundlage für den weiteren kooperativen Aufbau einer *Trusted Systems Domain*. Die nachfolgenden Anforderungen bilden den Rahmen für die spätere Spezifikation einer Trust-Policy zur Steuerung vertrauenswürdiger Bedingungen. In den Anforderungen wird deutlich, unter welchen Bedingungen sich Vertrauen herausbilden kann.

Identifizierbarkeit der Akteure

Eine eindeutige Identifizierung der Akteure muss gewährleistet sein. Die Authentizität von Akteuren ist eine notwendige Bedingung für die Durchsetzung von Regulierungsanforderungen unter hohen Anforderungen nach zurechenbaren Aktionen und Informationen. Sie bildet die Grundlage einer vertrauenswürdigen Kommunikation zwischen Akteuren.

Identifizierbarkeit der Domain

Eine eindeutige Identifizierung technischer Eigenschaften einer *Systems Domain* muss gewährleistet sein. Die Authentizität technischer Eigenschaften beschreiben funktionale und qualitative Rahmenbedingungen, die zur Umsetzung spezifischer Policies als Zusicherung (Invariante) für die vereinbarte Nutzungsdauer vorliegen müssen. Die Überprüfung der Authentizität verlangt die Einführung einer infrastrukturellen Plattformidentität.

Interpretationsbasis von Policies

Sowohl für den Cloud-Anbieter als auch für den Cloud-Anwender besteht die Notwendigkeit, dass Policies des jeweils anderen Kooperationspartners semantisch korrekt interpretiert werden können. Diese Anforderungen beschreiben Rahmenbedingungen für eine wechselseitige und koordinierte Policy-Umsetzung.

Überprüfbare Kooperationsbedingungen

Cloud-Anbieter und Cloud-Anwender benötigen Bedingungen für eine vertrauenswürdige und vertrauliche Kooperation. Beide Kooperationspartner beschreiben technische Artefakte (*Trusted Credential*), die Operationen, Aufgaben oder bereitgestellte Informationen auf Grundlage elektronischer Signaturen attestieren (Attestation Key) und sind durch den jeweilig anderen Kooperationspartner kryptographisch überprüfbar. Die Vertraulichkeit einer Kommunikation erfordert die Bereitstellung von Artefakten für eine kryptographische Verschlüsselung (Encryption Key).

Zusammenfassend lässt sich das Prinzip zur Herstellung einer Vertrauensbasis wie folgt definieren:

Definition 3.6 *Prinzip zur Herstellung einer Vertrauensbasis*
Ausgehend von den beschriebenen Methoden zur Entwicklung eines Trusted Context entwickelt sich eine Vertrauensbasis aus einem Konzept zur gesicherten Attestierung von Asset-Eigenschaften durch identifizierbare kooperierende Akteure. Die Attestierung erfolgt in Form einer überprüfbaren Zertifizierung.

Die folgenden Anwendungsfälle beschreiben schrittweise das Vorgehen für den Aufbau einer *Trusted Systems Domain* und demonstrieren die Herstellung und Erweiterung vertrauenswürdiger Bedingungen unter Anwendung einer vom Cloud-Anwender vorgegebenen Trust-Policy.
Jeder Anwendungsfall beginnt mit einer Kontextbeschreibung für eine festgelegte Domäne, um sowohl die Ausgangssituation als auch die Zielstellungen zu beschreiben. Ein auslösendes Ereignis begründet die Ausführung des Anwendungsfalls, der nur unter den herausgearbeiteten Vorbedingungen in der Lage ist, die nachfolgenden Schritte auszuführen.
Ein Schritt kennzeichnet eine auszuführende Aktivität innerhalb der gesetzten Domäne. Aktivitäten unterliegen einer Policy-Überwachung und verändern den Zustand der Domäne. Ein Farbschema dient der Zustandsabbildung. Komponenten mit einer roten Kennzeichnung repräsentieren technische Akteure mit Sicherheitsfunktionen. Der Wechsel in die grüne Kennzeichnung beschreibt den Übergang in eine vertrauenswürdige Beziehung zwischen den beteiligten technischen Akteuren.

3.2.2 UC-Aufbau einer Vertrauensbasis

Kontext
Die Herstellung eines *Trusted Context* ist verknüpft mit der Beschreibung und dem Aufbau einer *Trusted Systems Domain* und definiert eine Vertrauensbasis. Die Grenzen einer *Trusted Systems Domain* können sich nach Definition 3.5 dynamisch auf weitere Bereiche der verteilten Cloud-IT-Architektur ausdehnen. Ziel des Anwendungsfalls ist es, eine Vertrauensbasis als Grundlage für ein aufbauendes Konzept zur Gestaltung einer hierarchischen, vertrauenswürdigen Cloud-IT-Architektur herzustellen. Im Ergebnis besteht ein *Trusted Context* als Ausgangspunkt (*Root of Trust*) für die weitere Delegierung von Vertrauen innerhalb einer *Trusted Systems Domain*. Der Aufbau einer Vertrauensbasis schafft eine wesentliche Rahmenbedingung, um durch verhaltensregelnde Vorgaben normativ erwünschte Wirkungen zu erreichen [Hof16, S. 38].

Auslösendes Element für den Aufbau einer Vertrauensbasis sind Anforderungen zur Herstellung einer kooperierenden Service-Plattform. Das Vorgehen für den Aufbau einer Vertrauensbasis ist in Abbildung 3.1 dargestellt. Das vertrauensbildende Konzept aus Definition 3.6 wird durch folgenden Ablauf beschrieben:

Status	*Aufbau einer Vertrauensbasis*
Vorbedingung	**Technischer Akteur für Cloud-Anwender** Der Cloud-Anwender verwendet einen technischen Akteur (*Consumer Management Agent* – ConsMA), um seine regulierenden Zielstellungen vertrauenswürdig umzusetzen.
	Technischer Akteur für Cloud-Plattform (PA) Die infrastrukturelle Plattform (*PlatformAss*) verwendet einen technischen Akteur (*PlatformAgent*), um Informationen über ihre Sicherheitseigenschaften überprüfbar zu übermitteln.
Schritt: (1)	**Trusted Systems Domain eröffnen** Zur Bildung einer *Trusted Systems Domain* stellt der Cloud-Anwender eine Policy-Repository (*TSD-PolStore*) bereit. Die Domain wird durch eine Menge beschreibender Attribute definiert (*setDomainProperties*) und ist darüber identifizierbar.
Schritt: (2)	**Policy definieren und bereitstellen** Der Cloud-Anwender definiert und speichert eine Policy (*setPolicy*). Die darin festgelegten infrastrukturellen Sicherheitsbedingungen bilden hinreichende Bedingungen für die Anwendung der infrastrukturellen Plattform.

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Status	*Aufbau einer Vertrauensbasis*
Schritt: (3)	**Eigenschaften der infrastrukturellen Plattform ermitteln** Für die Bestimmung von Systemeigenschaften werden aktuelle Vorgaben (*retrievePolicy*) bezogen. Der CMA stellt eine Anfrage zur Ermittlung der Systemeigenschaften (*measureProperties*) an den technischen Akteur (PA) und bezieht die Ergebnisdaten (*measured platform properties*) nach Abfrage aus einer vertrauenswürdigen Hardwarebasis (*requestProperties*).
Schritt: (4)	**Systemeigenschaften überprüfen** Der CMA überprüft die ermittelten Systemeigenschaften (*validate*) mit den aktuellen Vorgaben aus der Policy.
Schritt: (5)	**Registrierung der Infrastruktur-Compliance** Die Ergebnisse der Validierung im *TSD-PolStore* werden als zugesicherte Werte registriert und bilden die Basis einer nachvollziehbaren Konformitätsbewertung über die ermittelten Systemeigenschaften.
Nachbedingung	***Trusted Systems Domain* ist erzeugt** Im Ergebnis der Registrierung überprüfter infrastruktureller Konformitätsbedingungen wird für den Cloud-Anwender ein *Trusted Context* eingerichtet. Der *Trusted Context* überführt den Cloud-Anwender-Bereich in eine *Trusted Systems Domain* (TSD).

Tabelle 3.3: Ablaufbeschreibung – Konzept einer Vertrauensbasis

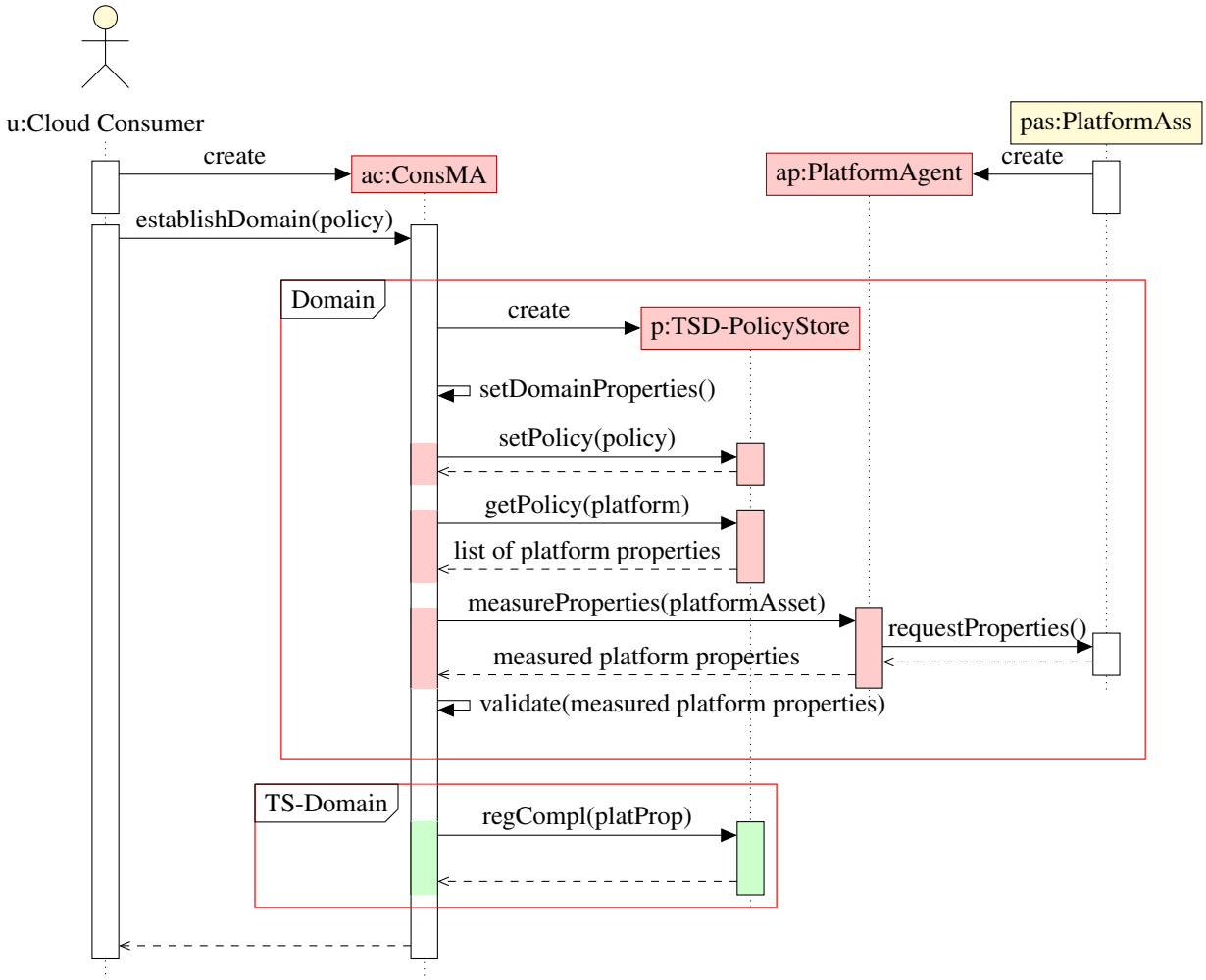

Abbildung 3.1: Konzept einer Vertrauensbasis (*Trusted Context*)

3.2.3 UC-Aufbau einer vertrauenswürdigen Kooperationsbasis

Kontext
Ausgehend von einem bestehenden *Trusted Context* können vertrauenswürdige Beziehungen zu einem oder mehreren Cloud-Anbietern hergestellt werden. Zwischen Cloud-Anwender und Cloud-Anbieter sichert eine überprüfbare Vertrauensbeziehung den wechselseitigen Informationsaustausch. Die Kooperation stützt sich auf Policy-Vorgaben der jeweiligen Kooperationspartner. Ziel des Anwendungsfalls ist es, den *Trusted-Systems-Domain*-Bereich des Cloud-Anwenders auf Bereiche von Cloud-Anbietern zu erweitern. Voraussetzung ist, dass deren Serviceangebote mit den Compliance-Anforderungen des Cloud-Anwenders übereinstimmen. Im Ergebnis entsteht eine erweiterte, vertrauenswürdige Domäne als Grundlage für eine Kooperation zwischen Cloud-Anwender und Cloud-Anbieter.

Auslösendes Ereignis für den Aufbau einer Kooperationsbeziehung ist die Suche nach Cloud-Anbietern, die den Bedarf nach erweiterten Ressourcen (*Assets*) gemäß den Anforderungen des Cloud-Anwenders erfüllen können.

Der Aufbau einer vertrauenswürdigen Kooperationsbasis wird durch folgenden Ablauf beschrieben:

Status	*Aufbau einer vertrauenswürdigen Kooperationsbasis*
Vorbedingung	**Bestehender *Trusted Context*** Sowohl Cloud-Anwender als auch Cloud-Anbieter haben einen eigenen *Trusted Context* hergestellt und folgen dem Geltungsbereich einer eigenen Policy (*TSD-PolStore*). Zusätzlich gelten die Vorbedingungen aus Tabelle 3.3.
	Technischer Akteur für Cloud-Anbieter Der Cloud-Anbieter verwendet einen technischen Akteur (*Provider Management Agent* – ProvMA), um seine regulierenden Zielstellungen vertrauenswürdig umzusetzen.
Schritt: (1)	**Verfügbare Cloud-Ressourcen ermitteln** Der Cloud-Anwender stellt eine Rechercheanfrage (*searchAsset*), ob Cloud-Ressourcen (*Asset*) mit spezifischen Eigenschaften bei einem oder mehreren Cloud-Anbietern zur Verfügung stehen.
Schritt: (2)	**Cloud-Anbieter auswählen** Die Auswahl eines Cloud-Anbieters wird durch die Policy bestimmt (*getPolicy*) und erfolgt über die Weiterleitung einer Rechercheanfrage (*queryAsset()*) an einen ausgewählten Cloud-Anbieter (ProvMA).
Schritt: (3)	**Verfügbare Ressourcen in Bezug auf Compliance-Anforderungen prüfen** Die rechtliche und technische Verfügbarkeit der angeforderten Ressource wird gemäß den vertraglichen Vereinbarungen überprüft (*getPol*). Im Fall einer bestätigten Verfügbarkeit wird die Menge verfügbarer Ressourcen (*available assets*) an den Cloud-Anwender (ConsMA) übermittelt.
Schritt: (4)	**Konformität mit angebotenen Ressourcen ermitteln** Der Cloud-Anwender (ConsMA) überprüft die Systemeigenschaften der bestätigten Ressourcen (*validate*) mit bestehenden Konformitätsanforderungen (*getPol*), abgeleitet aus seinen eigenen Policy-Vorgaben.
Schritt: (5)	**Ressourcen bezüglich der Compliance registrieren** Der Cloud-Anwender (ConsMA) bestätigt die Konformität (*ConfirmComp*) an den Cloud-Anbieter (ProvMA). Cloud-Anbieter und Cloud-Anwender registrieren wechselseitig die bestehende Konformität (*regCompl*) und protokollieren einen überprüfbaren Compliancenachweis.
Nachbedingung	**Kooperative Domäne ist hergestellt** Im Ergebnis der Registrierung vorliegender Konformitätsbedingungen zwischen Cloud-Anwender und Cloud-Anbieter steht für den Cloud-Anwender eine vertrauenswürdige kooperative Domäne (*Cooperated TSD*) zur Verfügung. Der *Trusted Context* wurde auf den Bereich des Cloud-Anbieters erweitert.

Tabelle 3.4: Ablaufbeschreibung – Konzept einer vertrauenswürdigen Kooperationsbasis

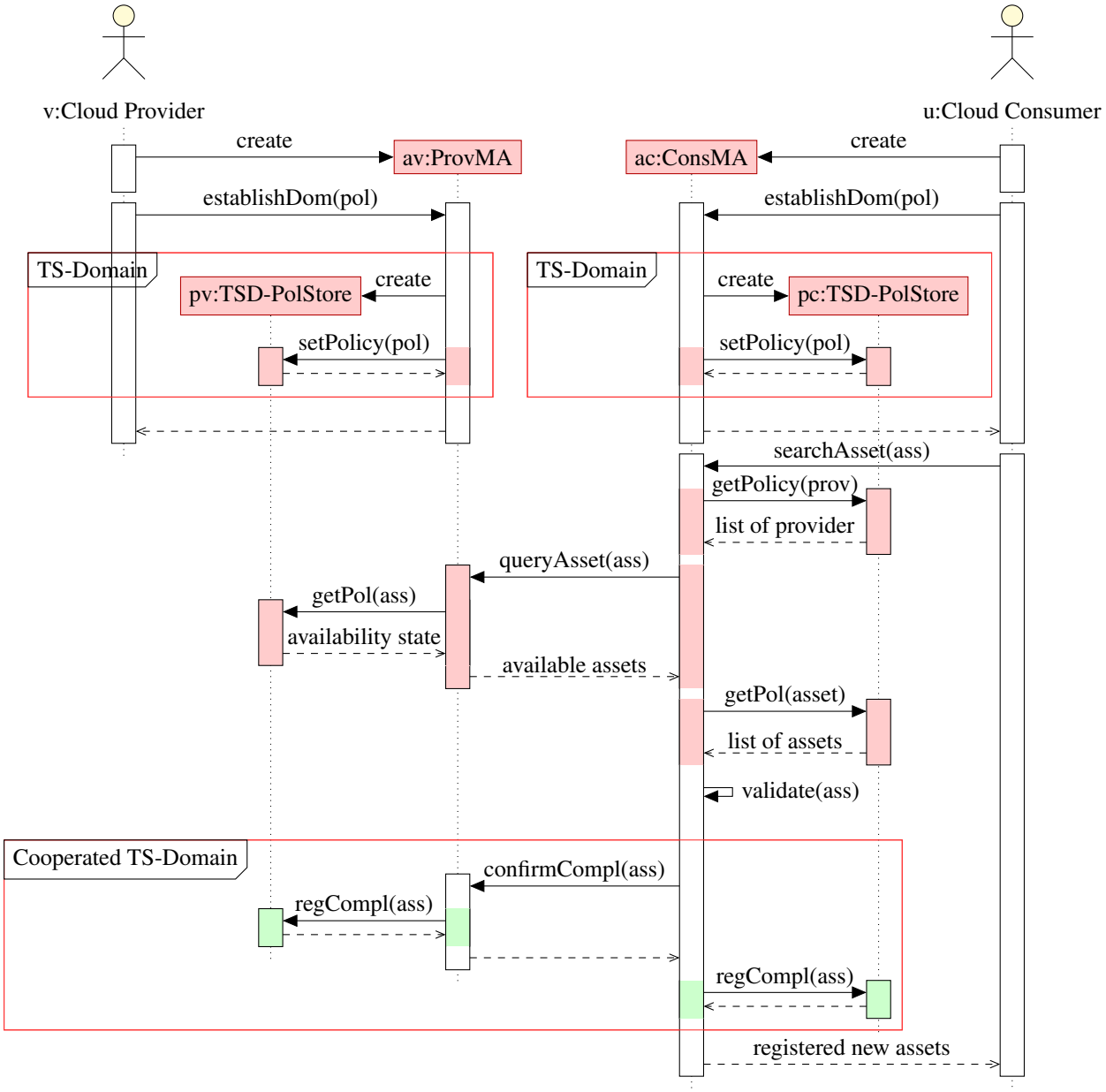

Abbildung 3.2: Konzept einer kooperierenden Vertrauensbasis (*Cooperated Trusted Context*)

Das Vorgehen für die Gestaltung einer vertrauenswürdigen Kooperation in Abbildung 3.2 orientiert auf funktionale Prinzipien und beschreibt die wesentlichen Interaktionsmuster zur Herstellung einer vertrauenswürdigen Kooperationsbasis. Die Kriterien zur Bewertung der Vertrauenswürdigkeit der beteiligten Akteure und der zu integrierenden verteilten Ressourcen (*Assets*) leiten sich ebenfalls aus den gesetzten Policies der Kooperationspartner ab.

3.2.4 UC-kooperative Provisionierung

Kontext
Eine Form der Kooperation im Cloud Computing entsteht durch die flexible und bedarfsgerechte Bereitstellung von Ressourcen (*Assets*). Verfügbare Ressourcen werden bereits im Zuge der Herstellung einer Kooperationsbasis (siehe Abschnitt 3.2.3) ermittelt. Der Anwendungsfall orientiert auf eine Provisionierung vertrauenswürdiger Ressourcen. Ziel des Anwendungsfalls ist es, innerhalb einer kooperierenden Vertrauensbasis einen gemeinsamen Modus für die Koordination des Verhaltens bei der Zusammenstellung von Cloud-Ressourcen zu finden. Der Cloud-Anwender formuliert gegenüber dem Cloud-Anbieter seine Erwartungshaltungen und legt den Modus für die Auswahl der Ressourcen fest. Im Ergebnis stehen dem Cloud-Anwender neue Ressourcen zur Verfügung, um Geschäftsmodelle gemäß seinen Vorgaben auszuführen.

Auslösendes Ereignis für den Anwendungsfall sind Bedingungen, die eine Bereitstellung von virtuellen Ressourcen für den Aufbau einer Softwarearchitektur verlangen.

Status	*Kooperative Provisionierung vertrauenswürdiger Ressourcen*
Vorbedingung	**Bestehende kooperierende Vertrauensbasis** Zwischen Cloud-Anbieter und Cloud-Anwender besteht eine kooperierende Vertrauensbasis (siehe Abschnitt 3.2.3). Der Cloud-Anwender kann auf einen Bestand registrierter, konformer Ressourcen zurückgreifen.
Schritt: (1)	**Konforme Cloud-Ressource anfordern** Der Cloud-Anwender stellt eine Anforderung zur Provisionierung einer virtuellen Ressource (*RequestComplAsset*) mit spezifischen Eigenschaften, die bei einem oder mehreren Cloud-Anbietern zur Verfügung stehen.
Schritt: (2)	**Cloud-Anbieter auswählen** Die Auswahl eines Cloud-Anbieters erfolgt aus dem registrierten Bestand konformer Ressourcen (*getCompl*).
Schritt: (3)	**Ressourcen von ausgewählten Cloud-Anbietern anfordern** Aus dem Registratureintrag wird eine konkrete Anforderung (*requestAsset)* der geforderten Ressource an den Cloud-Anbieter generiert.
Schritt: (4)	**Gültigkeit der Ressourcenanfrage feststellen** Der Cloud-Anbieter (ProvMA) verifiziert die eingehende Ressourcenanforderung (*getPol()*) hinsichtlich ihrer Berechtigung und der Verfügbarkeit der angeforderten Ressource. Kann die Ressource als konforme Ressource bestätigt werden (*validateState()*), so werden dem Cloud-Anwender (ConsMA) die Registratureinträge zur Verfügung gestellt (*asset location*).

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Status	*Kooperative Provisionierung vertrauenswürdiger Ressourcen*
Schritt: (5)	**Ressourcen abrufen und provisionieren** Der Cloud-Anwender (ConsMA) lädt und provisioniert die angeforderte Ressource (*loadComplAsset()*) und überprüft ihre Vertrauenswürdigkeit (*getPol()*). Nach Bestätigung der Vertrauenswürdigkeit wird die bereitgestellte Ressource installiert, konfiguriert (Deployment) und in Betrieb genommen (*startProvAss*).
Schritt: (6)	**Operative Ressource registrieren** Der Cloud-Anwender (ConsMA) registriert die in Betrieb genommene Ressource (*regOpAss*). Die Funktion ist Teil der Nachweisführung, um Rückschlüsse über den aktuellen Stand operativer vertrauenswürdiger Ressourcen treffen zu können.
Nachbedingung	**Konforme Ressource im operativen Betrieb steht bereit** Im Ergebnis der Kooperation wurde einem Cloud-Anwender eine angeforderte Cloud-Ressource zur Verfügung gestellt und in den operativen Betrieb überführt. Die Konformität der Ressource in Bezug auf die Anforderungen des Cloud-Anwenders konnte überprüft werden und ist zugesichert.

Tabelle 3.5: Ablaufbeschreibung – Bereitstellen von vertrauenswürdigen Cloud-Ressourcen

Im Unterschied zur TCG verallgemeinert der in Tabelle 3.6 beschriebene Ablauf das Vorgehen für eine policygestützte Provisionierung und Nutzung virtueller Ressourcen. Der Ressourcenbegriff *ProvAss* repräsentiert dabei spezifische virtuelle Cloud-Ressourcen.

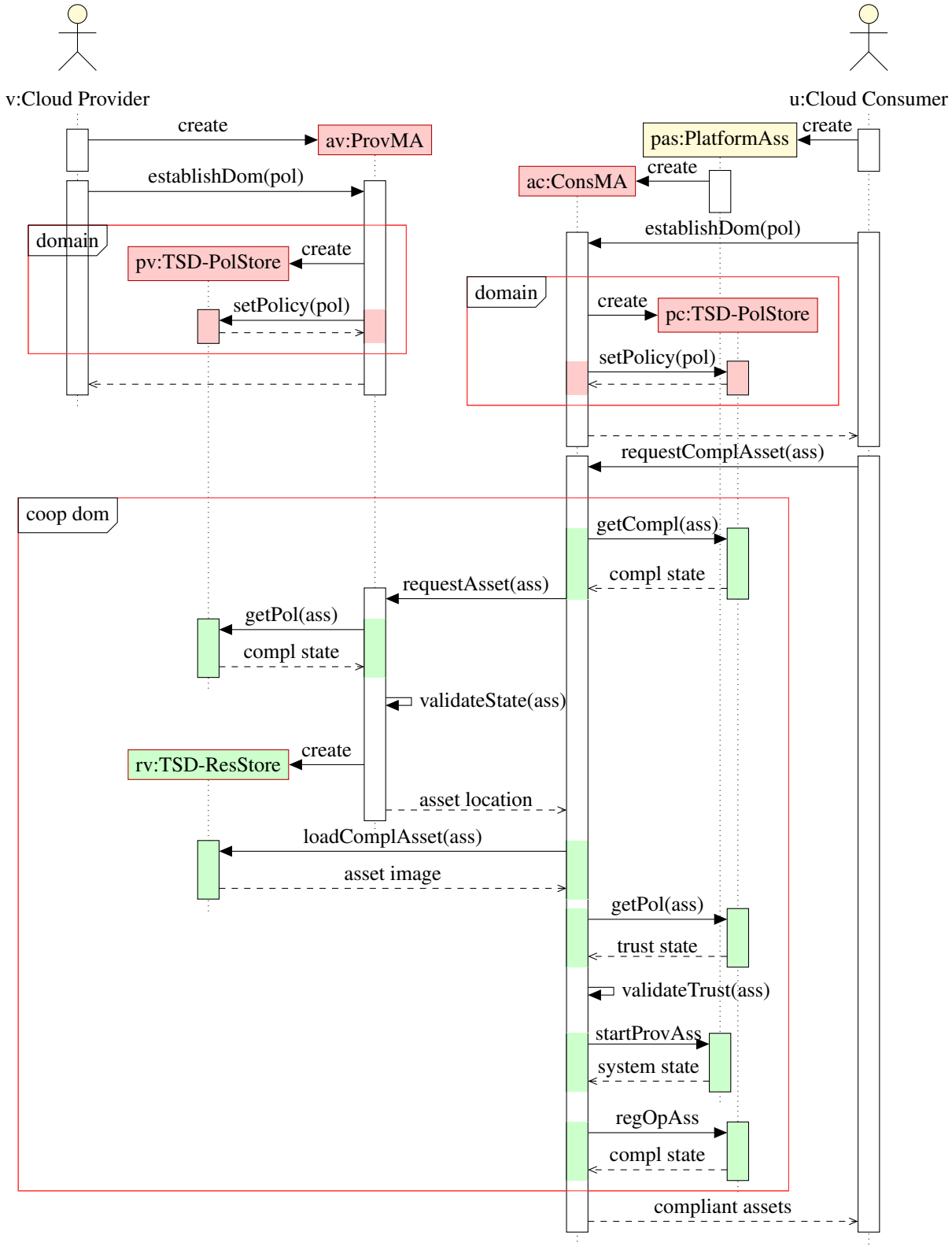

Abbildung 3.3: Vertrauenswürdige kooperative Erweiterung von Cloud-Ressourcen

3.2.5 UC-Änderungen von Regeln innerhalb einer kooperativen Domäne

Kontext
Es liegen Bedingungen vor, um eine etablierte Policy des Cloud-Anwenders zu modifizieren. Die Änderungen können sich auf als konform registrierte Eigenschaften von Ressourcen (*Assets*) beziehen, die bereits betrieblich genutzt werden. Ziel des Anwendungsfalls ist es, innerhalb einer kooperierenden Vertrauensbasis einen gemeinsamen Modus für die Koordination des Verhaltens bei Policy-Änderungen zu finden. Der Cloud-Anwender formuliert gegenüber dem Cloud-Anbieter neue Erwartungshaltungen und legt den Modus für eine Neuauswahl (Re-Provisionierung) der Ressourcen fest. Im Ergebnis stehen dem Cloud-Anwender angepasste Ressourcen zur Verfügung, die den modifizierten Policy-Definitionen entsprechen.

Auslösendes Ereignis für den Anwendungsfall ist die Notwendigkeit, eine bestehende Cloud-Anwender-Policy zu modifizieren.

Status	*Änderung einer Cloud-Anwender-Policy*
Vorbedingung	**Bestehende kooperierende Vertrauensbasis** Zwischen Cloud-Anbieter und Cloud-Anwender besteht eine kooperierende Vertrauensbasis (siehe Abschnitt 3.2.3). Der Cloud-Anwender kann auf einen Bestand registrierter, konformer Ressourcen zurückgreifen.
Schritt: (1)	**Regeln in der Policy-Repository ändern (*TSD-PolStore*)** Der Cloud-Anwender modifiziert Regeln in der Policy-Repository. Im Ergebnis erhält er die Information, dass die neuen Regeln eine Abweichung zu bereits registrierten konformen Ressourcen verursachen.
Schritt: (2)	**Alternative Ressource auswählen** Die Auswahl eines Cloud-Anbieters erfolgt aus dem registrierten Bestand konformer Ressourcen (*getCompl*). Es wird eine Auswahl mit möglichen alternativen Ressourcen (*compliant asset*) vorgeschlagen.
Schritt: (3)	**Alternative Ressource von ausgewähltem Cloud-Anbieter anfordern** Aus dem Registratureintrag wird eine konkrete Anforderung (*requestNewAsset)* der geforderten Ressource an den Cloud-Anbieter generiert.
Schritt: (4)	**Gültigkeit der alternativen Ressourcenanfrage feststellen** Der Cloud-Anbieter (ProvMA) verifiziert die eingehende Ressourcenanforderung (*getPol*) hinsichtlich ihrer Berechtigung und Verfügbarkeit. Kann die Ressource als konforme Ressource bestätigt werden (*validateState*), so werden dem Cloud-Anwender (ConsMA) die Registratureinträge zur Verfügung gestellt (*asset location*).

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Status	*Änderung einer Cloud-Anwender-Policy*
Schritt: (5)	**Ressource abrufen und provisionieren** Der Cloud-Anwender (ConsMA) lädt und stellt die angeforderte alternative Ressource zur Verfügung (*loadNewComplAsset*). Ihre Vertrauenswürdigkeit wird überprüft (*getPol*).
Schritt: (6)	**Nicht-konforme Ressource de-provisionieren** Nach Bestätigung der Vertrauenswürdigkeit wird die nicht-konforme Ressource außer Betrieb genommen (*stopProvAss*) und deinstalliert (*de-ProvAss*).
Schritt: (7)	**Alternative konforme Ressource provisionieren** Nach Bestätigung der Vertrauenswürdigkeit wird die bereitgestellte alternative Ressource installiert, konfiguriert (Deployment) und in Betrieb genommen (*startProvAss*).
Schritt: (8)	**Nicht-konforme Ressource de-registrieren** Der Cloud-Anwender (ConsMA) de-registriert die außer Betrieb genommene Ressource (*de-regOpAss*). Die Funktion ist Teil der Nachweisführung, um Rückschlüsse über den aktuellen Stand operativer vertrauenswürdiger Ressourcen treffen zu können.
Schritt: (9)	**Alternative, operative Ressource registrieren** Der Cloud-Anwender (ConsMA) registriert die in Betrieb genommene alternative Ressource (*regNewOpAss*). Die Funktion ist Teil der Nachweisführung, um Rückschlüsse über den aktuellen Stand operativer, vertrauenswürdiger Ressourcen treffen zu können.
Nachbedingung	**Konforme Ressource im operativen Betrieb steht bereit** Im Ergebnis der Kooperation wurde einem Cloud-Anwender eine alternative Cloud-Ressource zur Verfügung gestellt und in den operativen Betrieb überführt. Die Konformität der Ressource in Bezug auf Anforderungen des Cloud-Anwenders konnte überprüft werden und ist zugesichert.

Tabelle 3.6: Ablaufbeschreibung – Änderungen in der Policy des Cloud-Anwenders

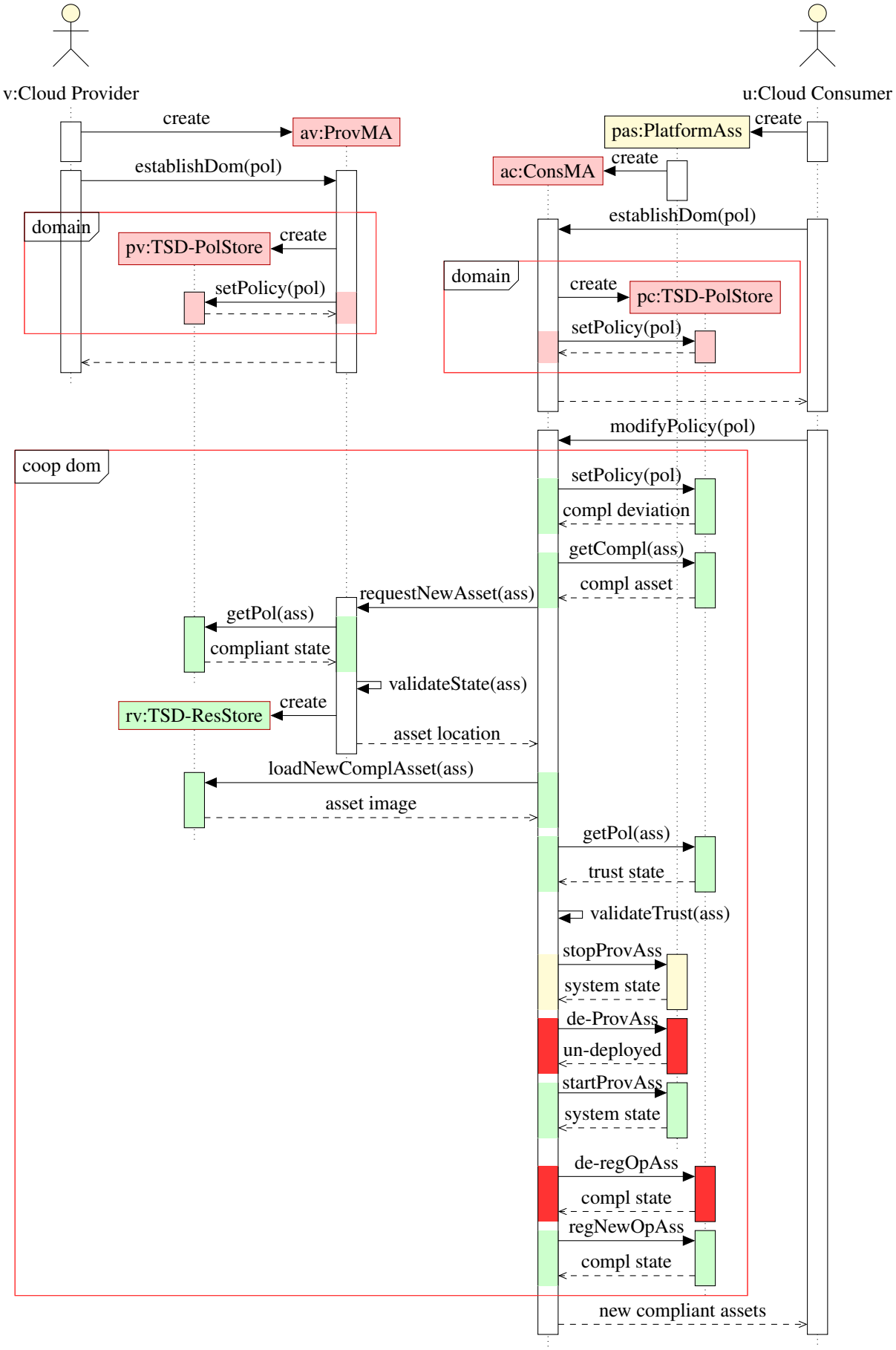

Abbildung 3.4: Vertrauenswürdige kooperative Änderung von Cloud-Ressourcen

3.2.6 Abgeleitete Anwendungsfälle aus TCG-Richtlinien

Im Mittelpunkt der TCG-Spezifikation stehen Handlungskonzepte zur Umsetzung des Prinzips einer *Trusted Domain*, die unter der Federführung eines Cloud-Anwenders (*Cloud Consumer – CU*) und einem Cloud-Anbieter (*Cloud Provider – CP*) mit verteilten Verantwortungsbereichen etabliert wird.

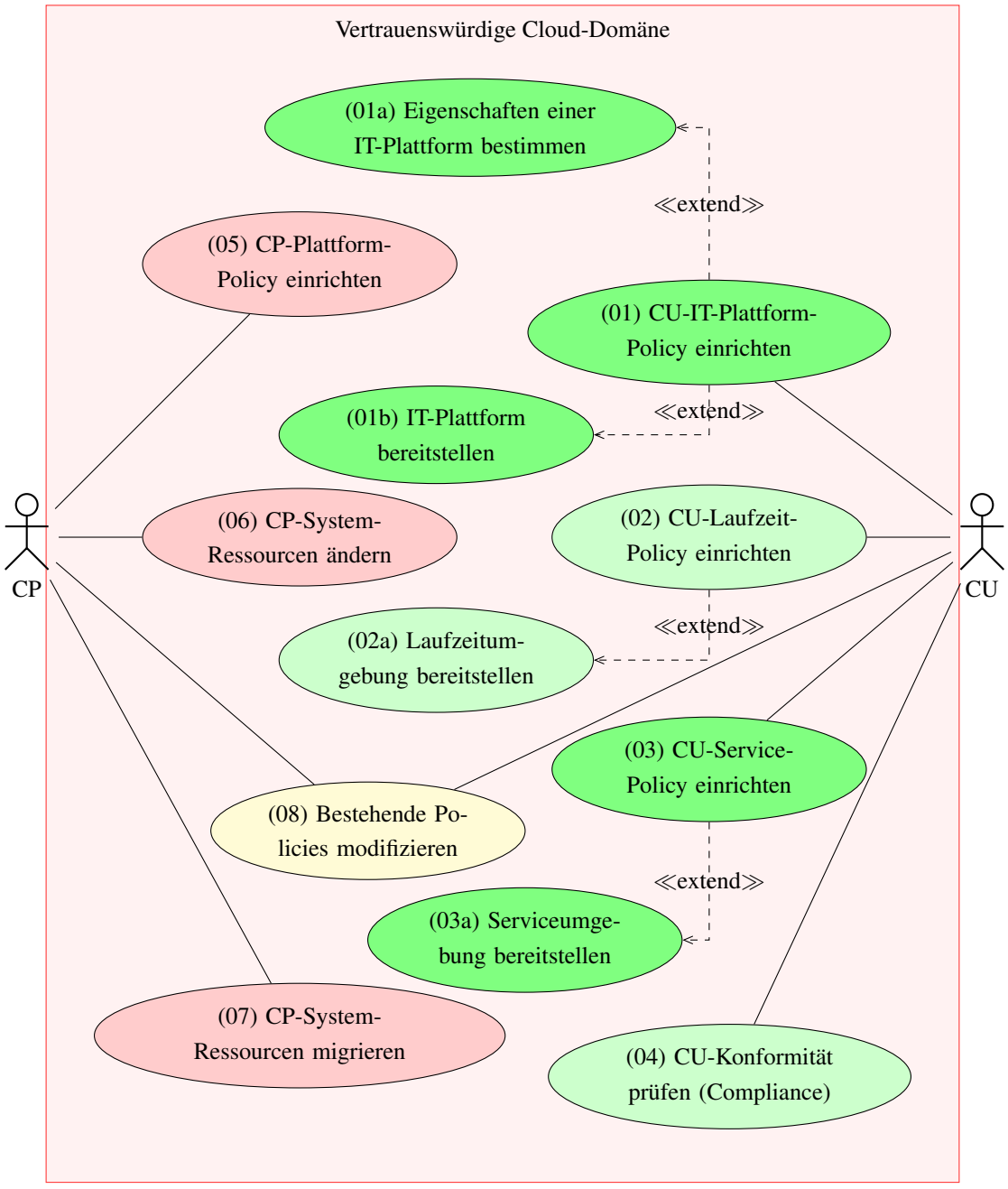

Abbildung 3.5: Abgeleitete Anwendungsfälle als Anforderungsgrundlage

Die in Abbildung 3.5 definierten Anwendungsfälle leiten sich aus den TCG-Vorgaben und Beschreibungen ab und bilden einen Anforderungsrahmen für die weitere Spezifikations- und Evaluationsphase. Die Orientierung an diesen Anwendungsfällen gewährleistet eine strukturierte Anforderungssicht für ein Cloud-Anwender-orientiertes Management von Regulierungs- und Complianceaufgaben.

UC01 : CU-IT-Plattform-Policy einrichten

Der Anwendungsfall stellt eine Policy für den Cloud-Anwender (CU) mit Anforderungen über die Eigenschaften und dem Verhalten einer Cloud-IT-Plattform bereit.

UC01a: Eigenschaften einer IT-Plattform bestimmen

Der Anwendungsfall ermittelt zugesicherte funktionale und sicherheitsrelevante Eigenschaften einer Cloud-IT-Plattform.

UC01b: IT-Plattform bereitstellen

Der Anwendungsfall steuert alle Aktivitäten zur Überprüfung und Bereitstellung von vertrauenswürdigen Komponenten, um eine Cloud-IT-Plattform(IaaS) basierende Vertrauensgrundlage herzustellen.

UC02 : CU-Policy-Laufzeit einrichten

Der Anwendungsfall stellt eine Policy für den Cloud-Anwender (CU) mit Festlegungen über die Eigenschaften und dem Verhalten einer Cloud-Laufzeitumgebung bereit.

UC02a: Laufzeitumgebung bereitstellen

Der Anwendungsfall steuert alle Aktivitäten zur Überprüfung und Bereitstellung von vertrauenswürdigen Komponenten, um eine Cloud-Laufzeitumgebung (PaaS) herzustellen.

UC03 : CU-Policy-Service einrichten

Der Anwendungsfall stellt eine Policy für den Cloud-Anwender (CU) mit Festlegungen über die Eigenschaften und dem Verhalten einer Cloud-Service- und Anwendungsumgebung bereit.

UC03a: Serviceumgebung bereitstellen

Der Anwendungsfall steuert alle Aktivitäten zur Überprüfung und Bereitstellung von vertrauenswürdigen Komponenten, um eine Cloud-Serviceumgebung (SaaS) herzustellen.

UC04 : CU-Konformität prüfen (Compliance)

Der Anwendungsfall überprüft den System- und Sicherheitsstatus eines Cloud-Systems und führt eine vergleichende Analyse in Bezug auf betrieblich wirksame Policies durch.

UC05 : CP-Policy-Plattform einrichten

Der Anwendungsfall stellt eine Policy für einen Cloud-Anbieter (CP) mit Festlegungen über die Eigenschaften und dem Verhalten einer Cloud-IT-Plattform bereit.

UC06 : CP-System-Ressourcen ändern

Für die Behebung von Ressourcenengpässen führt ein Cloud-Anbieter (CP) Maßnahmen durch, um fehlende Ressourcen auszugleichen. Diese Maßnahmen beziehen sich auf alle Ebenen eines Cloud-Systems.

UC07 : CP-System-Ressourcen migrieren

Technische Gründe können die Verlagerung von Ressourcen eines CU-etablierten Cloud-Systems erforderlich machen. Der Wechsel erfordert die Kooperation mit dem Cloud-Anwender.

UC08 : Bestehende Policies modifizieren

Der Anwendungsfall beschreibt Maßnahmen für die Modifikation einer bereits betrieblich wirksamen Policy. Im Ergebnis liegt eine neue Version der entsprechenden Policy vor.

3.3 State-of-the-Art-Betrachtung

Die State-of-the-Art-Betrachtung bezieht sich nicht allein im Kern auf die Entwicklung von Policies zur Definition einer anwenderorientierten Cloud-Strategie. Die in Beziehung stehenden Themenbereiche (siehe Abbildung 3.6) wie z. B. Vertrauen, Regel- und Sprachkonzepte werden mit in die Betrachtung einbezogen. Die gesetzte Zielstellung lässt sich nur in einer ganzheitlichen Betrachtung bewerten und konzeptionell bearbeiten.

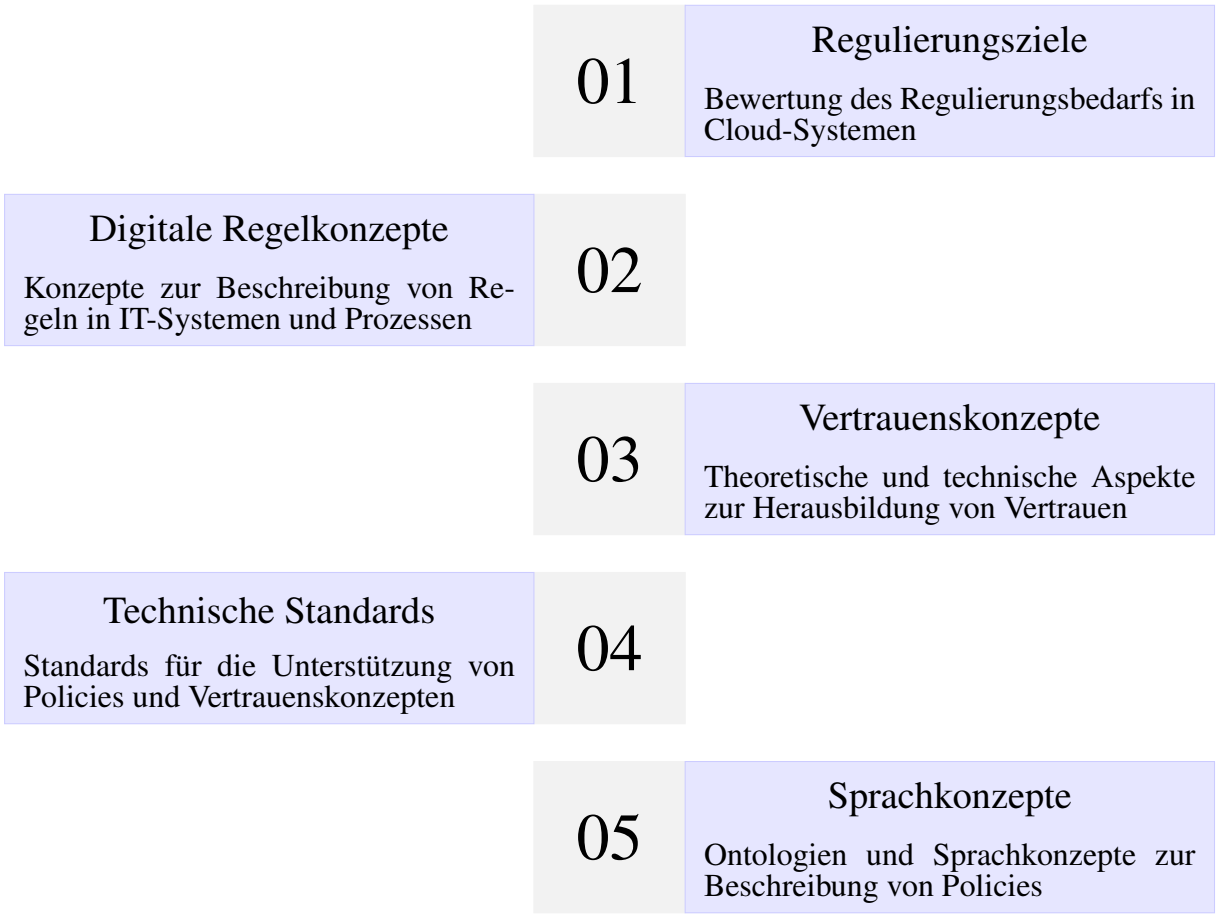

Abbildung 3.6: Themenbereiche der State-of-the-Art-Betrachtung

Ein im Anhang (I) spezifiziertes Template formuliert Kategorien, die Gruppen von spezifischen Bewertungskriterien zusammenfassen.
Die Kategorien werden im Abschnitt 3.3.1.1 vollständig aufgeführt. Für alle weiteren Bewertungen wird der Kontext erläutert und eine Zusammenfassung der Ergebnisse vorgenommen. Das Template umfasst folgende Struktur:

Kontextbeschreibung

Die Kontextbeschreibung erläutert Hintergründe, Rahmenbedingungen (Projekt, Organisation) und Zielstellungen der untersuchten Arbeit. Im Fall von Forschungsarbeiten wird das entsprechende Forschungsthema herausgestellt. Die Einordnung in eines der genannten Themengebiete wird vorgenommen.

Regulierung und Policies

Die Betrachtung orientiert auf die Regulierung von IT-Systemen, im Speziellen in Bezug auf Cloud-Systeme. Welche Regulierungsziele werden verfolgt? Unter welchen Gesichtspunkten werden Policies definiert, klassifiziert und umgesetzt? Welche Bereiche einer IT-Architektur werden durch die betrachteten Policies beeinflusst? Werden Maßnahmen verfolgt, um die Ausführung von Policies zu steuern, zu priorisieren, zu separieren oder abzusichern?

In der Betrachtung muss berücksichtigt werden, dass bereits konkrete geregelte Bereiche innerhalb einer IT-Infrastruktur existieren können.

Methoden zur Formalisierung

Die Betrachtung orientiert auf formale Methoden, die für die Repräsentation von Policies zur Anwendung kommen. Welche Repräsentationsstandards werden verwendet? Wie werden Strukturierungskonzepte formal abgebildet? Besteht die Möglichkeit, formal einen Kontext zu beschreiben?

Etablierung von Vertrauen

Die Betrachtung orientiert auf die Anwendung von Kernprinzipien zur Herausbildung vertrauensbildender Eigenschaften und technischer Verfahren. Wie können vertrauenswürdige Eigenschaften zugesichert werden?

Das Konzept Vertrauen ist kein einzelnes, in sich abgeschlossenes Konzept, sondern wird als ein Gesamtkonzept verketteter Verfahren von vertrauensbildenden Maßnahmen (Vertrauenskette) untersucht. In der Bewertung wird der Aspekt aufgenommen, inwieweit ein Vertrauenskonzept nur einen Teilbereich der Cloud-Architektur berücksichtigt oder ob es einen ganzheitlichen Ansatz verfolgt. Kommen dabei hardware- oder softwarebasierte Funktionen zum Einsatz?

Sicherheit und Systemzustand

Im Hinblick auf die Bewertung von Compliance-Anforderungen werden Untersuchungen in Bezug auf System- und Sicherheitszustände vorgenommen. Mit welchen Methoden erfolgt eine formale Beschreibung und optional eine Absicherung von System- und Sicherheitszuständen? Auf welche Bereiche einer Cloud-Architektur bezieht sich die Zustandsbetrachtung?

Attestierung von Eigenschaften

Die Attestierung von vorliegenden Architektur- und Prozesseigenschaften bildet einen Teil der Untersuchung in Bezug auf vertrauensbildende Funktionen. Welche technischen Konzepte werden in Bezug auf die vertrauenswürdige Attestierung behandelt? Welche Formen von Attestierungen werden untersucht? Welche Sicherheitsfunktionen kommen dabei zur Anwendung, um die Vertrauenswürdigkeit von Attestierungsergebnissen ableiten zu können?

Technische Transformation von Policies

Die Betrachtung orientiert auf Methoden und technische Konzepte zur Transformation interpretierbarer Policies. Welche grundlegenden Konzepte werden für die semantische Bewertung von Policies untersucht? Welche Transformationsmethoden kommen zur Anwendung?

Bewertung und Relevanzabschätzung

Es erfolgt eine Zusammenfassung und Einordnung der Bewertungsergebnisse in Bezug auf ihre weitere Einbeziehung in die Spezifikationsphase.

3.3.1 Thema: Regulierungsziele

3.3.1.1 Pattern-based Runtime Management of Composite Cloud Applications

Projektname / Referenz:	Pattern-based Runtime Management of Composite Cloud Applications	[Bre+13a]
Universität / Organisation:	Institut für Architektur von Anwendungssystemen (IAAS) – Universität Stuttgart	2013
Ziele des Projekts / Standards:		
Der Lösungsansatz stellt eine Formalisierung von Policies vor und beschreibt Prinzipien ihrer Transformation auf Grundlage von Management-Pattern. Die Arbeit nimmt Bezug auf Policy-Transformationsaspekte und betrachtet den Übergang von einem formalen *Application State Model* in ein reales System (*Desired Application State Model*). Die Untersuchungen verfolgen einen methodischen Weg, um von einer konzeptionellen Managementsicht (Management Pattern) eine technische und automatisierte Realisierung der Managementziele zu gewährleisten. Der hier vorgelegte Lösungsansatz ist eng verbunden mit den Spezifikationszielen von TOSCA [OAS13].		
Kategorie	Strukturierte Policyumsetzung	

***Regulierung und Policy*: Ziele der Regulierung**		
PZi1	Regulierung von Sicherheitsfunktionen?	–
PZi2	Policy für Trust vorhanden?	–
PZi3	Regulierung von Sicherheitsmodellen?	–
PZi4	Regulierung von Privacy und Datenschutz?	–
PZi5	Regulierung von Funktion und Deployment?	✓
PZi6	Regulierung von operativem Management?	✓
Klassifikation		
PRu1	Verwendung typisierter Regeln?	–
PRu2	Typen:	–
Policysteuerung		
PSt1	Maßnahmen zur Konflikterkennung	–
PSt2	Unterstützung von Prioritäten	–
PSt3	Werden Workflows für Policyumsetzung unterstützt?	✓
PSt4	Funktionen für die Steuerung von Policies	✓
Separierung		
PSp1	Unterstützung von Policy Domains?	✓
PSp2	Unterstützte Architekturebene für Policy	*Verteilungsebene*, *Runtime (OS-Ebene)*, *Serviceebene*, *Speicher*

Sicherheit		
PSec	Wird eine Policy als Asset behandelt und durch Sicherheitsmaßnahmen geschützt?	–

***Bildung von Vertrauen*: Prinzipien**		
VPr1	Kommen vertrauenswürdige Agenten zum Einsatz?	–
VPr2	Existieren hardwarebasierte Vertrauenskonzepte?	–
VPr3	Werden kryptographische Identitäten genutzt?	–
VPr4	Werden Evaluationsprozesse betrachtet?	–
VPr5	Werden Vertrauensketten verwendet?	–
VPr6	Existieren Protokolle für Vertrauensketten?	–
VPr7	Beziehen sich Vertrauenskonzepte auf Architekturebenen?	–
VPr8	Kann das Niveau an Vertrauen berechnet werden?	–

Aspekte der Formalisierung		
PFo1	Welche syntaktische Repräsentation wird verwendet?	Management Pattern transformieren Application State Models
PFo2	Liegt eine eigenständige Konzeptualisierung vor?	✓ als Pattern Framework
PFo3	Können Policies mehrere Rules enthalten?	–
PFo4	Können Policies gruppiert werden?	✓ *Configuring-*, *Guarding-*, *Extending-Policy*
PFo5	Können Policies verschachtelt werden?	–
PFo6	Wird die Formalisierung eines Kontexts geführt?	–
Systemzustand		
SSy1	Wird der Systemzustand formal behandelt?	✓
SSy2	Unterstützte Architekturebene für Systemzustand?	*Serviceebene*
Sicherheitszustand		
SSi1	Wird der Sicherheitszustand formal behandelt?	–
SSi2	Unterstützte Architekturebene für Sicherheitszustand?	–

Aspekte der Attestierung und Transformation	
ATT1 Wird Vertrauen durch Attestierung zugesichert?	–
ATT2 Wird der Systemzustand durch Attestierung zugesichert?	–
ATT3 Wird der Sicherheitszustand durch Attestierung zugesichert?	–
Transformation von Policies	
TRa1 Besteht Schnittstellenkonzept für Transformation?	✓
TRa2 Welche Technologien zur Transformation werden unterstützt ?	Structural Management Annotation (inklusive Parameter)

Zusammenfassung	
Ziel:	Die vorliegende Arbeit beschreibt eine Policy-Transformation als Kernkonzept für die geregelte Bereitstellung von Ressourcen unter Anwendung einer Amazon-EC2-Referenzarchitektur. Die Topologie wird als *Target Topology Fragment* in Form von Pattern beschrieben. Die Transformationsregeln selbst werden als Annotationen mit den Management-Pattern verknüpft. Die Effektivität in der technischen Umsetzung von Policies wird durch ein Application-State-Modell (ASM) bewertet.
Regulierungsbereich:	Der funktionale Regelungsbereich umfasst die Virtualisierungs-, Laufzeit- und Anwendungsschicht einer IT-Architektur.
Sicherheit:	Sicherheitsaspekte werden im Prozess der Provisionierung nicht diskutiert.
Vertrauen:	Der Aspekt Vertrauen wird in der Betrachtung nicht berücksichtigt und es gibt keine Ansätze zur Definition von Vertrauen in Form einer Policy.
Abgrenzung:	Pattern (inklusive Annotation) sind Low-Level-Aktivitäten für eine Transformation. Die Transformation setzt auf vorhandene Schnittstellen des Cloud-Systems auf. Für die eigene Arbeit werden die Konzepte zur Transformation weiter vertieft. Grundprinzipien für die Anwendung von Policies in Verbindung mit einer System-Zustandsbewertung sind zu berücksichtigen.

3.3.1.2 Unifying Compliance Requirements across Business and IT

Projektname / Referenz:	Unifying Compliance Requirements across Business and IT	[Koe+14]
Universität / Organisation:	Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Institut für Architektur von Anwendungssystemen (IAAS)	2014
Ziele des Projekts / Standards:		
Die Arbeit reflektiert das Problem zur Abbildung unterschiedlicher rechtlicher oder organisatorischer Quellen, aus denen sich Anforderungen für zu regelnde Aspekte innerhalb von IT-Lösungen ableiten. Es wird ein Vorgehen beschrieben, um eine strukturierte und vollständige Erfassung von Regulierungsanforderungen zu gewährleisten. Im Mittelpunkt steht ein Konzeptansatz, wie aus einer bestehenden Anforderungssituation der Nachweis über eine konforme Umsetzung (Nachweis der Compliance) erbracht werden kann. Der hier vorgelegte Lösungsansatz ist eng verbunden mit den Spezifikationszielen von TOSCA [OAS13].		
Kategorie	Strukturierte Policyumsetzung	

Zusammenfassung	
Ziel:	Die Arbeit beschreibt ein Vorgehensmodell für eine ganzheitliche Erfassung von Regulierungsanforderungen mit der Möglichkeit, eine *Traceability* zwischen Anforderungen und gesetzlichen Vorgaben sicherzustellen. Dafür wird ein Referenzkonzept (*Compliance Descriptor*) als Bindeglied zwischen strukturierten Anforderungen und den Vorgaben aus Gesetzen eingeführt.
Regulierungsbereich:	Basierend auf eine Taxonomie wird an Beispielen der Gesetzgebung gezeigt, wie strukturierte Anforderungen an die Regulierung auf Vorgaben der Gesetzgebung als Compliance-Nachweis abgebildet werden.
Sicherheit:	Sicherheitsrelevante Aspekte werden nicht betrachtet.
Vertrauen:	Ein für die Regulierung zugrunde gelegtes Trustkonzept wird nicht betrachtet. Funktionen für das Reporting als Compliancenachweis werden diskutiert, jedoch werden keine Ansätze für ein vertrauenswürdiges Reporting formuliert.
Abgrenzung:	Das Vorgehen dient einer vollständigen Erfassung möglicher Regulierungsziele und dem Abgleich mit technisch umgesetzten Anforderungen. Der Konzeptansatz zeigt keine Regeln für eine vertrauenswürdige Umsetzung der erhobenen Anforderungen zur Regulierung. Für die eigene Arbeit werden Methoden für einen Abgleich regulativer Vorgaben mit konkreten Policies weiter verfolgt. Modellierungstools wie *ProGoalML* und *Composite Application Framework* (CAFE) zeigen konzeptionelle Ansätze, wie aus einer modellierten Fachdomäne die technische Umsetzung, Konfiguration und das Monitoring der erzielten Ergebnisse realisiert wird.

3.3.2 Thema: Digitale Regelkonzepte

3.3.2.1 Policy-Aware Provisioning of Cloud Applications

Projektname / Referenz:	Policy-Aware Provisioning of Cloud Applications	[Bre+13b]
Universität / Organisation:	Institut für Architektur von Anwendungssystemen (IAAS – Universität Stuttgart)	2013
Ziele des Projekts / Standards:		
Der Lösungsansatz stellt eine Formalisierung von Policies vor und beschreibt die Herangehensweise für eine konforme Provisionierung auf unterschiedlichen Cloud-Architekturebenen. Policies werden als Management-Planlet-Konzept beschrieben und berücksichtigen sowohl funktionale als auch nicht-funktionale Aspekte. Sie werden als Workflow miteinander verbunden. Sicherheitsaspekte werden als nicht-funktionale Anforderungen mit in den Provisioning-Workflow integriert. Im Vordergrund steht die Ausführung einer zentralen Managementprozedur unter Anwendung verschiedener technologischer Schnittstellen für die Realisierung nicht-funktionaler Aspekte wie z. B. der Sicherheit.		
Kategorie	Strukturierte Policyumsetzung	

Zusammenfassung	
Ziel:	Im Kern bilden Management Planlets die Grundlage für eine geregelte Bereitstellung von Komponenten. Sie sind die Spezialisierung einer Management Policy. Policy-Aware bedeutet, dass nicht-funktionale Aspekte mit in die Provisionierung einbezogen werden.
Regulierungsbereich:	Management-Plans erlauben die Provisionierung und damit die Regulierung ab der Virtualisierung bis zur Serviceebene.
Sicherheit:	Sicherheit ist Teil der Provisionierung und erhält keine ganzheitliche Betrachtung im Sinne einer Sicherheitsstrategie. Es werden einige Teilaspekte herausgezogen, um die konzeptionellen Erweiterungen vorzustellen.
Vertrauen:	Ein Vertrauenskonzept als Grundlage für Policies wird nicht behandelt.
Abgrenzung:	Eine Trennung von Policies für Deployment zur Bereitstellung von Ressourcen und für Security wird nicht vorgenommen. Ausgangspunkt bildet ein Planlet mit Annotations als Policy. Für die eigene Arbeit wird die Verwendung eines Graphenmodells und die Visualisierung einer Domain [Bre+12] aufgenommen. Das Konzept von Policies als eine separate Formalisierung in Form von Management-Annotations zeigt einen Weg zur Verbindung einer Domain mit nicht-funktionalen Policy-Konzepten. Ein Application State bezieht sich auf funktionale Aspekte. Aussagen über den Security-State werden nicht gemacht. Der Begriff *Constraint* taucht in der Formalisierung nicht auf.

3.3.2.2 Policy-Aware Provisioning and Management of Cloud Applications

Projektname / Referenz:	Policy-Aware Provisioning and Management of Cloud Applications	[Bre+14]
Universität / Organisation:	Institut für Architektur von Anwendungssystemen (IAAS – Universität Stuttgart)	2014
Ziele des Projekts / Standards:		
Der Lösungsansatz stellt eine Formalisierung von Application-Strukturen und Policies vor und beschreibt eine Herangehensweise für eine konforme Realisierung in eine Cloud (Management Plans). Die Policies berücksichtigen sowohl funktionale als auch nicht-funktionale Aspekte (Sicherheitsanforderungen). Der Lösungsansatz beschreibt ein Konzept zur Abbildung einer Application Topology (siehe Domainkonzept).		
Kategorie	Strukturierte Policyumsetzung	

Zusammenfassung	
Ziel:	Die Arbeit ist eine Weiterentwicklung des Konzepts von 2013 (siehe 3.3.2.1). Die Erweiterung umfasst die Automatisierung von Provisionierungsaufgaben im Zusammenspiel mit nicht-funktionalen Sicherheitsaspekten unter Verwendung einer Application-Topologie.
Regulierungsbereich:	Management-Plans erlauben die Provisionierung und damit die Regulierung ab der Virtualisierung bis zur Serviceebene.
Sicherheit:	Sicherheit wird als nicht-funktionaler Aspekt der Provisionierung zugeordnet.
Vertrauen:	Der Vertrauensaspekt wird im Zusammenspiel mit der Provisionierung nicht behandelt.
Abgrenzung:	Es wird eine konkrete Beschreibung einer Application-Topology (Domain-Modell) eingeführt. Regulierung wird als Gegenstand der Application-Topology (Desired Application, State Model) aufgenommen. Das Konzept der Transformation in technische Systeme ist Teil der Management-Plan-Konzeption und beschreibt die Umsetzung in Form von Prozessschritten. Die Attribute einer Management Annotation Policy verweisen auf bestehende Policystandards. In der eigenen Arbeit erfolgt Transformation auf Grundlage einer eigenständigen Konzeptklasse *Transformation* und erlaubt die Adaption an unterschiedliche technische Schnittstellen. Das Konzept der Transformation ist ebenfalls Bestandteil einer Policy (siehe auch 3.3.2.1). Das Ziel der eigenen Arbeit besteht in einer Klassifizierung von Policies und der Bereitstellung einer dafür notwendigen und überprüfbaren Vertrauensbasis. Management Planlets bilden bereits eigene Klassen von Policies, geben jedoch keine geschlossene Übersicht über die Regulierungsstrategie.

3.3.3 Thema: Vertrauenskonzepte

3.3.3.1 Secure Enclaves for REactive Cloud Applications

Projektname / Referenz:	Secure Enclaves for Reactive Cloud Applications	[Pro16]
Universität / Organisation:	European Commission (Objective: ICT-07-2014: Advanced Cloud Infrastructures and Services Grant agreement no: 645011), TU Dresden, Institut für Systemarchitektur, Bereich Systems Engineering	2015
Ziele des Projekts / Standards:		
SERECA ist ein Forschungsprojekt der EU, eröffnet 2014, und wird in Zusammenarbeit mit der TU Dresden koordiniert und ausgeführt. Die Zielstellung des Projekts SERECA ist die Entwicklung eines Architekturdesigns, um innerhalb einer nicht vertrauenswürdigen IT-Stackumgebung eine sichere Ausführungsumgebung für *Reactive Application* bereitzustellen. Die Sicherheit stützt sich dabei auf innovative Hardwaremechanismen. Die Sicherheitsziele verfolgen Vertraulichkeit, Integrität, Verfügbarkeit und Lokalisierbarkeit. Das Konzept einer *secure enclave* soll über ein sicheres Verbundkonzept (*secure channel*) zu einem verteilten Konzept (*distributed secure enclaves*) dieser sicheren Ausführungsumgebungen erweitert werden.		
Kategorie	Vertrauenskonzept	

Zusammenfassung	
Ziel:	Das Projekt greift den Aspekt einer vertrauenswürdigen Hardwareplattform auf und setzt einen Vertrauensanker (*Root of Trust*), auf dessen Grundlage weitere Konzepte aufsetzen können. Das Projekt zielt auf die Anwendung einer minimalen *Trusted Computing Base* (TCB), durch Anwendung der SGX-Technologie (Software Guard Extension) von INTEL. Die Attestierung stützt sich auf Methoden der kryptographischen Identifizierbarkeit einer Hardwareplattform.
Regulierungsbereich:	Die Regulierung bezieht sich auf kryptographische Protokolle, um die Vertrauenswürdigkeit der Hardwareplattform festzustellen und entsprechend die Provisionierung eines Cloud-Anwender-Payloads sicher auszuführen. Die Entwicklung einer spezifischen Policy ist nicht Teil der Spezifikation.
Sicherheit:	Sicherheitsaspekte zur Gewährleistung einer isolierten Vertraulichkeit unter nicht-vertrauenswürdigen Randbedingungen werden betrachtet. Das Sicherheitsziel Vertraulichkeit steht im Mittelpunkt der Betrachtung. Das Thema einer sicheren Identifizierung von vertraulichen Containern kann aus der bestehenden Informationslage nicht abgeleitet werden.

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Zusammenfassung	
Vertrauen:	Im Unterschied zu dem in der Arbeit vorgestellten Vertrauenskonzept beruht das *enclave*-Vertrauenskonzept auf einen hardwaregestützten sicheren Container (*enclave*). Diese Containerform stellt sich aus Sicht einer IT-Infrastruktur als *Single-point-of-Trust* dar. Das Konzept verfolgt zwar eine Kopplung solcher hardwaregesicherten Container, jedoch lässt sich das Prinzip einer *Delegation of Trust* nicht erkennen. Aus den vorliegenden Informationen lässt sich ein holistisches und verteiltes Trustkonzept über verschiedene Schichten einer IT-Infrastruktur unter Anwendung softwarebasierter Sicherheits- und Trustansätze noch nicht erkennen.
Abgrenzung:	In erster Linie wird das Ziel einer vertraulichen und integritätsgesicherten Verarbeitung unter hardwareunterstützten Ausführungsbedingungen verfolgt. Es muss noch ein praktischer Beweis angetreten werden, ob ein *Single-Point-of-Trust*-Ansatz den Anforderungen an Verteilung, Skalierung und Performance gerecht wird. Die Bewertung von Mechanismen zur Gewährleistung der Vertraulichkeit bildet die Grundlage, um bereitgestellte Sicherheitsfunktionen mit einem Schutzbedarf abgleichen zu können. Es ist noch nicht ersichtlich, ob Anforderungen an die Regulierbarkeit im Sinne einer algorithmischen Einrichtung solcher vertraulichen Ausführungsbedingungen (Policy) erfüllt werden können. Der *enclave*-Ansatz kann jedoch die Grundlage legen, um ein hierarchisches Konzept von *enclaves* in Form einer Vertrauenskette aufzubauen. Jeder dieser Bereiche schafft eigene *private* Zonen und schützt darin Schlüsselmaterial und vertrauenswürdige Softwarecodes. Ausgehend von diesen Bereichen lassen sich weitere solcher vertraulichen Bereiche aufbauen. Die Vertrauenswürdigkeit eines jeden Bereiches kann mit Mitteln der Attestierung gemessen werden. Für die eigene Arbeit erhält das Konzept eine Bedeutung, da es ein Kernproblem der Sicherheit zur Laufzeit in Verbindung mit Security-Modellen lösen kann. Die Absicherung von Ausführungsinstanzen kann zu einem Verbundkonzept vertrauenswürdiger Entitäten weiterentwickelt werden. Der Konzeptansatz erfüllt die Anforderung, während der Laufzeit die Integrität einer vertrauenswürdigen Entität sicherzustellen. Damit entstehen in einem System verteilte Vertrauenspunkte, die als Basis für weitere Vertrauensketten dienen.

3.3.3.2 Enforcing-Security-and-Assurance-Properties-in-Cloud-Environment

Projektname / Referenz:	Enforcing Security and Assurance Properties in Cloud Environment	[Bou+15]
Universität / Organisation:	Gemeinschaftsprojekt aus: INSA Centre Val de Loire, Univ. Orleans, Bourges, France, ENS Lyon - Inria - UCB Lyon, Univ. of Lyon - LIP, Lyon, France, Transp. - Technol. & Dev., IKUSI, Spain, Ind. & Adv. Manuf. Dept., Vicomtech-IK4, San Sebastian, Spain, R&D Dept., Nextel S.A., Spain, Inria	2015
Ziele des Projekts / Standards:		
Das Gemeinschaftsprojekt beschäftigt sich mit einem Konzept zur Durchsetzung von Sicherheitspolicies in einer verteilten Cloud-Architektur.		
Kategorie	Vertrauenskonzept	

Zusammenfassung	
Ziel:	Untersuchung und Bewertung verschiedener Policysprachen für die Formulierung von Sicherheitszielen. Es wird ein Verfahren und Modell vorgestellt, um einem Cloud-Anwender erwartete Sicherheitseigenschaften zusichern zu können.
Regulierungsbereich:	Das Policykonzept umfasst die Betriebssystemebene und beschreibt Regeln für die Umsetzung eines vollständigen Sicherheitsmodells. Es bestehen Regelkonzepte, um Verbindungen auf Netzwerkebene zu einem Cloud-System zu autorisieren und Sicherheitsvorgaben in Bezug auf die Vertraulichkeit und Authentizität der Verbindungsendpunkte festzulegen.
Sicherheit:	Die Durchsetzung eines Mandatory Access Control Modells (MAC) orientiert auf die Sicherheitsziele Vertraulichkeit und Authentizität der eingesetzten Komponenten.
Vertrauen:	Mit dem Aspekt *Assurance* wird ein Vertrauensansatz für den Cloud-Anwender entwickelt. Der Begriff *Trust* wird nicht direkt verwendet. Es wird jedoch ein Framework für die Zusicherung von Sicherheitseigenschaften vorgestellt, basierend auf einen *Based Measure Agent*, der Statusinformationen an das Framework überträgt.
Abgrenzung:	Die Arbeit greift eine Reihe von Policy-Konzepten auf, die für die Formulierung von interpretierbaren Sicherheitszielen geeignet sind [Twi+09; HLL10] bzw. die ASPF-Policy in [Bob+14]. Der Vergleich erlaubt eine erste Abschätzung über sprachliche Ausdrucksstärken im Bereich Sicherheit. Obwohl die Zusicherung von Sicherheitseigenschaften als wesentliche Zielstellung erhoben wurde, wird die Betrachtung rein funktional geführt. Eine Bewertung der attestierten Werte wird nicht vorgenommen. Es werden Aussagen zu vertrauensbildenden Konzepten der Common-Criteria [CC12b] gemacht, doch wird der hier vorgestellte Ansatz nicht als Weiterführung dieser Grundprinzipien verstanden und auch bewusst als eigene Entwicklung definiert.

3.3.4 Thema: Technische Standards

3.3.4.1 Web Services Policy 1.5 – Framework - Current

Projektname / Referenz:	Web Services Policy 1.5 – Framework - Current	[Ved+07]
Universität / Organisation:	Organization for the Advancement of Structured Information Standards (OASIS)	2007
Ziele des Projekts / Standards:		
Der Standard repräsentiert ein Rahmenwerk zur Formulierung XML-basierter Policies zur Festlegung von domänenspezifischen Fähigkeiten, Anforderungen, Verhalten und Qualitätseigenschaften von Web-Services. Die Anforderungen werden in Form von regulativen Anforderungen oder Fähigkeiten (*assertions*) definiert und können als Bestandteil einer Web-Service-Spezifikation hinzugefügt werden. Das Policy-Rahmenwerk bildet die Grundlage für weitere Policy-Spezifikationen.		
Kategorie	Technische Standards	

Zusammenfassung	
Ziel:	Der Standard bildet ein offenes Framework zur Definition von Policies und entfaltet seine Wirksamkeit in Verbindung mit anderen Spezifikationen.
Regulierungsbereich:	Der zu regelnde Bereich bezieht sich auf Web-Services und den Übertragungsweg zwischen Akteur und Web-Service-Endpunkt. Bis auf die Ebene der Hardware gibt es keine Begrenzung der Anwendung auf eine IT-Architekturschicht. Er bildet den syntaktischen Rahmen, um entsprechende Policies daraus abzuleiten, die mit Web-Services flexibel umgesetzt werden können.
Sicherheit:	Unter Anwendung weiterer Spezifikationen für Web-Service-Sicherheit können auch Policies für die Durchsetzung von Sicherheitszielen definiert werden.
Vertrauen:	Der Aspekt Vertrauen wird nicht direkt behandelt. Die vorgestellten Konzepte sind jedoch geeignet, um Sicherheitseigenschaften als Anforderung zu definieren und herzustellen. Die Eigenschaften können als vertrauenswürdige Bedingungen überprüft werden.
Abgrenzung:	Für die eigene Arbeit setzt der Standard eine Grundlage, um daraus Sicherheits-Policies entwickeln zu können. Für den Aufbau einer vertrauenswürdigen Beziehung zwischen technischen Entitäten sind Web-Services geeignet, da sie neben funktionalen Aspekten auch nicht-funktionale Anforderungen umsetzen können. Die Einbindung von Sicherheits-Policies bezieht sich auf die Ebene der technischen Transformation. Interpretierbare nicht-technische Policy-Vorgaben werden dann in technische Policy-Konzepte eines konkreten IT-Systems transfomiert.

3.3.4.2 WS-SecurityPolicy 1.3

Projektname / Referenz:	WS-SecurityPolicy 1.3	[Nad+12a]
Universität / Organisation:	Organization for the Advancement of Structured Information Standards (OASIS)	2007
Ziele des Projekts / Standards:		
Der Standard beschreibt Nutzungsbedingungen für die Verwendung von Web-Services in Form von Sicherheits-Policies. Das Rahmenwerk für Sicherheits-Policies schließt Web-Service-Sicherheitsspezifikationen wie WSS11 [Nad+06] und WS-Trust [Nad+12b] mit ein.		
Kategorie	Technische Standards	

Zusammenfassung	
Ziel:	Offenes Framework zur Definition von Policies, das erst in Verbindung mit anderen Spezifikationen wirksam wird.
Regulierungsbereich:	Reguliert den Übertragungsweg (Message-Sicherheit) zwischen Akteur und Web-Service-Endpunkt. Der Endpunkt ist in der Lage, konkrete sicherheitsrelevante Nutzungsbedingungen festzulegen. Bis auf die Ebene der Hardware gibt es keine Begrenzung der Anwendung auf eine IT-Architekturschicht.
Sicherheit:	Für die Message-Sicherheit in der Anwendung von Web-Services kann die Vertraulichkeit der Übertragung auf Anwendungsebene geregelt werden. Darüber können Anforderungen an die Authentizität und Integrität der übertragenen Daten (Messages) gestellt werden. Die Policies lassen sich im Kontext sicherer Verbindungen in das Gesamtkonzept integrieren.
Vertrauen:	Der Aspekt Vertrauen wird in Verbindung mit dem Standard WS-Trust berücksichtigt. Die sichere und vertrauenswürdige Identifikation von Entitäten der Serviceendpunkte wird durch die Anwendung von referenzierbaren Security-Token ausgeführt. Dabei werden je nach Sicherheitsbedarf unterschiedliche Tokenarten verwendet. Die Spezifikation von Policies unterstützt Mechanismen, um eine sichere Identifizierbarkeit von Policies zu gewährleisten. Darüber hinaus kann die Authentizität überprüft und damit die Vertrauenswürdigkeit einzelner Policies bestätigt werden.
Abgrenzung:	Die im Standard vorgestellten Sicherheitsmechanismen erlauben eine protokollbasierte Herstellung vertrauenswürdiger Kommunikationsendpunkte. Für die Sicherstellung vertrauenswürdiger Kommunikationsbedingungen stehen entsprechende Ausdrucksmittel (*Assertion*) zur Verfügung. Ebenso kann auf die Gestaltung des Protokollablaufs regelnd eingewirkt werden. Für die eigene Arbeit erhält der Standard eine Bedeutung, da er in die Entwicklung eines Protokolls zur Herstellung von Vertrauensketten zwischen verteilten Entitäten einfließen kann. Auf Ebene der Transformation können Vorgaben zur Sicherheit mit sprachlichen Mitteln des Standards auf technische Entitäten übertragen und aktiviert werden.

3.3.4.3 WS-Trust

Projektname / Referenz:	WS-Trust	[Nad+12b]
Universität / Organisation:	Organization for the Advancement of Structured Information Standards (OASIS)	2007
Ziele des Projekts / Standards:		
Der Standard bildet die Grundlage für kooperative Prozesse zwischen unterschiedlichen Autoritäten (*Authority*). Autoritäten werden technisch durch Public-Key-Infrastrukturen repräsentiert und sind in der Lage, für einen definierten technischen Hoheitsbereich überprüfbare Zusicherungen über die Echtheit elektronischer Identitäten auszustellen. Die Form ihrer Zusicherungen setzt ein Vertrauensniveau und bildet die Grundlage für nachfolgende technische Entscheidungen. Der Standard stellt Mechanismen bereit, um Security-Token sicher auszutauschen und Protokollgrundlage für ein Security-Token-Management zwischen verteilten Autoritäten. Gleichzeitig schafft er Vorbedingungen für einen vertrauenswürdigen Datenaustausch und stützt sich dabei selbst auf Standards wie WSS11 [Nad+06] für den sicheren Datenaustausch.		
Kategorie	Technische Standards	

Zusammenfassung	
Ziel:	Die Einbeziehung von Konzepten verteilter Autoritäten setzt die Grundlage für den Aspekt von Vertrauen in allen darauf aufbauenden technischen Protokollen.
Regulierungsbereich:	Unter Einbeziehung von Vertrauenskonzepten benötigen verteilte serviceorientierte Prozesse Regelungen für die Herstellung, den Nachweis und über die Auflösung von Vertrauensbeziehungen zwischen verteilten Autoritäten. Der Standard setzt Rahmenbedingungen, um die Vertrauenswürdigkeit von Web-Service-Entitäten protokolltechnisch sicherzustellen. Reguliert werden an dieser Stelle technische Anforderungen an Security-Token und an die Gestaltung von Protokollen, um Security-Token nach Vorgaben anzufordern, zu erzeugen und zu verteilen.
Sicherheit:	Es werden Sicherheitsfunktionen für die Vertraulichkeit und für die Integrität von Messages verwendet.
Vertrauen:	Der Standard beschreibt Mechanismen zur Gestaltung von Vertrauensverhältnissen zwischen Service-Anwender und Web-Services. Die Mechanismen sind für die Entwicklung von Protokollen zur Etablierung von Vertrauensketten zwischen unterschiedlichen Entitäten anwendbar.
Abgrenzung:	Für die eigene Arbeit besitzt der Standard eine Bedeutung, da er eine protokolltechnische Grundlage für die Spezifikation von Protokollen zur Herstellung von Vertrauensbeziehungen zwischen den Autoritäten von Cloud-Anwender und Cloud-Anbieter bildet. Der Standard setzt weiterhin die Grundlage für Konzepte vertrauenswürdiger Entitäten und der Anwendung vertrauenswürdiger Policy-Konzepte. Die Anwendung solcher Methoden kann als Teil in das holistische Vertrauenskonzept einfließen.

3.3.4.4 Web Services Security: SOAP Message Security 1.1

Projektname / Referenz:	Web Services Security: SOAP Message Security 1.1 [Nad+06]
Universität / Organisation:	Organization for the Advancement of Structured Information Standards (OASIS) 2006
Ziele des Projekts / Standards:	
Der Standard stellt Mechanismen bereit, um SOAP Messages bezüglich der Integrität und der Vertraulichkeit zu sichern. Der Standard unterstützt die Übertragung und den Austausch unterschiedlicher Security-Token-Formate und bildet die Voraussetzung für eine sichere Umsetzung der WS-Trust-Spezifikation [Nad+12b]. Die Funktionen für die Integritäts- und Vertraulichkeitsabsicherung von Messages bilden ein Rahmenwerk zur Entwicklung von Sicherheitsprotokollen.	
Kategorie	Technische Standards

Zusammenfassung	
Ziel:	Der Standard stellt Sicherheitsfunktionen für den Austausch von Nachrichten bereit.
Regulierungsbereich:	Reguliert werden die Verfahren zum Integritätsschutz (XML-Signature [Bar+08]). Es handelt sich u. a. um Festlegungen über eingesetzte Hash-Verfahren, Signaturbereiche, Signaturverfahren und Algorithmen. Im Bereich der Verschlüsselung (XML-Encryption [I+02]) werden u. a. Festlegungen über Verschlüsselungsverfahren, Verschlüsselungsmethoden, Message-Authentication (Message Authentication Code) – für den Integritätsschutz von Blöcken und der Key-Verschlüsselung – für den sicheren Schlüsseltransport erforderlich. Die Methoden sichern die Übertragung von Security-Token.
Sicherheit:	Die Mechanismen beschreiben überprüfbare Sicherheitsstandards für die Implementierung von Sicherheitsprotokollen.
Vertrauen:	Der Standard ist Bestandteil der Sicherheitsprotokolle zur Etablierung von Vertrauen [Nad+12b] und benutzt Security-Token für eine vertrauenswürdige Ende-zu-Ende-Übertragung.
Abgrenzung:	Für die eigene Arbeit ist der Standard geeignet, sichere SOAP-orientierte Interaktionen in eigenen Protokollen für die Herstellung von Vertrauensbeziehung zu spezifizieren. Darüber hinaus können Konzepte für den sicheren Datenaustausch auf Applikationsebene realisiert werden, um z. B. die Authentizität bzw. Vertraulichkeit der Daten sicherzustellen.

3.3.5 Thema: Sprachkonzepte

3.3.5.1 Using Ontologies to Analyze Compliance Requirements of Cloud-Based Processes

Projektname / Referenz:	Using Ontologies to Analyze Compliance Requirements of Cloud-Based Processes	[Hum+14]
Universität / Organisation:	Technische Universität Dortmund, Lehrstuhl für Software Engineering und Fraunhofer Institut für Software- und Systemtechnik	2014
Ziele des Projekts / Standards:		
Die vorliegende Arbeit beschreibt ein strukturiertes Vorgehen für die vollständige Erfassung von Regulierungsanforderungen auf Grundlage ontologischer Beschreibungsformen. Als Ontologiesprache wird OWL [B M12] verwendet. Ausgangspunkt bildet die bestehende Problemsituation, dass die Nichteinhaltung von Sicherheitsrichtlinien und Ausführungsbestimmungen (Compliance) als Ursache für eine fehlende Nutzungsakzeptanz im Bereich der Cloud-Technologien darstellt. Die Vorgehensweise verwendet eine Risikobetrachtung als Zielorientierung für die später abzuleitenden Anforderungen zur Erhöhung der Sicherheit. Der Schwerpunkt liegt in einer strukturierten Analyse von textuellen Anforderungen und einer nachgeordneten Formalisierung von Policies.		
Kategorie	Strukturierte Policyumsetzung	

Zusammenfassung	
Ziel:	Formalisierung von Policies durch einen Prozess zur Auswertung textueller Grundlagen und Ableitung von Anforderungen für die Generierung von Policies.
Regulierungsbereich:	Der Prozess orientiert auf die Regulierung von Sicherheitszielen.
Sicherheit:	Ausgehend von Risiken verfolgt der ontologische Ansatz die Ableitung von Sicherheitsanforderungen, die in Form von *Rules* definiert werden. Der Konzeptansatz stellt Beziehungen zwischen dem Risikomodell und Anforderungen an die Sicherheit her. Der Regulierungsbedarf zur Absicherung eines IT-Systems wird logisch abgeleitet.
Vertrauen:	Der Aspekt Vertrauen wird nicht explizit betrachtet.
Abgrenzung:	Die Arbeit stellt nur den Aspekt für das Sicherheitsmanagement in den Vordergrund. Der ontologische Ansatz dient als Instrument, um die Vorschriftensituation mit einem bestehenden Konzept für Sicherheitsanforderungen abzugleichen. Die semantische Interpretation von Vorgaben und Richtlinien kann zukünftig die Definition von Policies vereinfachen und den Bezug auf Anforderungen und Regulierungsgrundlage sicherstellen. Für Richtlinien zur Beschreibung von Bedingungen zur Bildung vertrauenswürdiger Beziehungen kann dieses Vorgehen auch für die Konzeptualisierung vertrauenswürdiger Policies weiterentwickelt werden.

3.3.5.2 Policy Language for a Pervasive Computing Environment

Projektname / Referenz:	Policy Language for a Pervasive Computing Environment	[KFJ03]
Universität / Organisation:	University of Maryland, Baltimore County	2003
Ziele des Projekts / Standards:		
Das Paper beschreibt einen ontologischen Ansatz zur Formalisierung einer Policy im Bereich des *Pervasive computing* (Rechnerdurchdringung). Als Ontologiesprache wird RDF [CWL14] verwendet. Die hohe Dynamik bei der Bereitstellung von Services verlangt eine hohe Adaptivität zur Anpassung von Sicherheitseigenschaften der Endgeräte, ohne dass die Implementierung verändert werden muss. Der Lösungsansatz verwendet neben einem semantischen Sprachkonzept ein deontisches Logikkonzept zur Konzeptualisierung normativer Begriffe wie *Verpflichtung* und *Erlaubnis*.		
Kategorie	Sprachkonzepte	

Zusammenfassung	
Ziel:	Die Definition und Durchsetzung von Sicherheitseigenschaften von mobilen, gekoppelten Endgeräten.
Regulierungsbereich:	Es werden Policies für ein rollen- bzw. gruppenbasiertes Zugriffskonzept unterstützt. Regeln für die Zugriffssteuerung von *Subjekt-Objekt*-Paaren erlauben eine feingranulare Zugriffspolitik.
Sicherheit:	Das Konzept unterstützt Sicherheitsfunktionen für den vertraulichen Datenaustausch und einer sicheren Endgeräte-Authentifizierung unter Verwendung einer PKI-Lösung. Für die Durchsetzung von Zugriffsregeln steht ein eigenes Framework zur Verfügung.
Vertrauen:	Obwohl der Ansatz einer PKI-Lösung gewählt wurde, fokussiert die Arbeit auf Sicherheitsaspekte.
Abgrenzung:	Für die eigene Arbeit wird das Vorgehen bei der Konzeptualisierung von *Rule*, *Policy*, *Action* und *Condition* aufgegriffen und für den Aspekt Vertrauen weiterentwickelt. Normative Begriffe wie *Recht*, *Verpflichtung*, *Verbot* und *Aufhebung, Entbindung* sind für eine Policy-Strategie zur Herausbildung vertrauenswürdiger Strukturen nicht erforderlich, werden jedoch als Optionen mit aufgenommen. Die *Closed-world*-Betrachtung beschreibt nur gültige Aktivitäten bezogen auf konkrete Domänenobjekte. Es werden *TargetObjects* innerhalb der *action*-Konzeptualisierung verwendet. Die formalisierte Beschreibung einer konkreten Domäne in Form einer IT-Architektur erfolgt nicht. Für die eigene Arbeit ist die Domänen-Spezifikation eine Bedingung zur Beschreibung vertrauenswürdiger Prinzipien. Durch die Verwendung einer eigenen Zugriffskontrollkomponente wird die Transformationen von Policies in konkrete Architekturen nicht erforderlich. Die eigene Arbeit fokussiert auf die technische Transformation von Policies unter Nutzung von Sicherheitskomponenten einer konkreten IT-Architektur.

3.4 Zusammenfassung und Abgrenzungsbeschreibung

Die untersuchten Konzeptarbeiten im Bereich der Regulierungsziele [Bre+13a; Koe+14], Digitale Regelkonzepte [Bre+13b; Bre+14], Vertrauenskonzepte [Pro16; Bou+15], Technische Standards [Ved+07; Nad+12a], [Nad+12b], [Nad+06] und Sprachkonzepte [Hum+14; KFJ03] verdeutlichen, dass der Aspekt Vertrauen als Konzept aufgenommen worden ist und durch Standards unterstützt wird, jedoch nur partiell in bestimmten Bereichen von IT-Architekturen zur Anwendung kommt. Keines der untersuchten Verfahren unterliegt dafür einem Gesamtkonzept.
In spezifischen Bereichen elektronischer Services wie in Breitenbücher et al. [Bre+13b; Bre+14] spielen vertrauenswürdige Endpunkte (Akteure-Entitäten) eine besondere Rolle. Die Konzepte verfolgen allein sicherheitsorientierte Zielstellungen und prägen erste Ansätze vertrauenswürdiger Architekturbereiche.
Die Richtlinien der TCG geben mit ihren Spezifikationen einen Leitfaden [Tru13a] für ganzheitliche Konzepte zur Herausbildung von Vertrauen. Ganzheitlich bedeutet im Kontext der TCG, dass alle Ebenen einer IT-Architektur berücksichtigt und in ein Vertrauenskonzept einbezogen sind. Die Beschreibungen in Abschnitt 3.2 der Analysephase basieren auf diesen Prinzipien und spezifizieren daraus anwendbare Szenarien. Die Vorgehensweise setzt Anforderungen an die Gestaltung einer expliziten Trust-Policy. Aussagen über die Policy-Struktur entfallen, jedoch steht die konzeptionelle Vorgehensweise zur Herstellung einer *Trusted Systems Domain* im Vordergrund, die sich unter Anwendung einer Trust-Policy herausbildet.
Das Konzept zur Kennzeichnung von Entitäten-Eigenschaften setzt die Grundlage für ein eigenständiges Vertrauenskonzept und ist nicht allein auf Sicherheitseigenschaften reduziert. Die vorliegende Arbeit nimmt diesen Ansatz auf und verallgemeinert ihn zu einem Konzept von Qualitätseigenschaften.
Der weitergehende und konstituierende Konzeptansatz zur Entwicklung von Vertrauen vollzieht sich über ein Verfahren überprüfbarer Zusicherungen. Das Verfahren führt zur Entwicklung einer deklarativen Trust-Policy-Beschreibung. Zusicherung bedeutet neben einer verifizierbaren Bescheinigung erfasster Eigenschaften eine gleichzeitige Übertragung von Eigenschaften einer vertrauenswürdigen Entität auf eine nicht bewertete Entität. Zusicherungen transformieren Entitäten einer beliebigen IT-Architektur in Instanzen einer vertrauenswürdigen Strukturklasse. Die kontinuierliche und zuverlässige Ausführung dieser Form von Transformation ist Aufgabe und Ziel einer Trust-Policy.
Zusicherungen sind Ergebnisse handelnder, vertrauenswürdiger Akteure. Der Konzeptansatz zur Entwicklung von Vertrauen erweitert sich auf die Spezifikation solcher Akteure als notwendige strukturelle Grundlage. In einigen Konzeptarbeiten kommen für die Ausführung funktionaler Aufgaben technische Agenten zum Einsatz. Die Spezifikation vertrauenswürdiger Entitäten unterscheidet sich von diesen Ansätzen, denn im Mittelpunkt stehen qualitätsorientierte Zusicherungen, die z. B. die Ergebnisse technischer Agenten einer Bewertung unterziehen können.
In der Analyse wurden Arbeiten aufgegriffen, die ontologisch den Aspekt Sicherheit als Schwerpunkt betrachten [FE09; DRI+06; GMT11; CCK14]. Die Grundlagenrecherche bestätigt Einzellösungen für die Formalisierung von Domänenwissen und Sicherheitsprinzipien, jedoch bestehen Ansätze ihrer Integration. Die vorliegende Arbeit verfolgt eine geschlossene Integration zwischen Cloud-Domänenwissen und Sicherheit und erweitert die Integration um eine Konzeptualisierung für Regulierung.

4 | Konzeption

Im Abschnitt 2.3 der Problembeschreibung wurde das Konzept von Kommunikation als Grundlage für eine erfolgreiche Regulierung unter den Bedingungen einer strukturellen Kopplung herausgearbeitet.
Es entstehen Schwierigkeiten, die Ziele von Kommunikation zu erfüllen, solange ihre konstituierenden sozialen Konzepte wie *Semantik*, *Erreichbarkeit* und *Erfolg* nicht zugesichert werden können. Eine unzureichende Umsetzung der genannten Konzepte beschreibt ein Risiko, welches als Verlust der Kontrollfähigkeit unter den Bedingungen einer strukturellen Kopplung definiert wird.
Der technische Einsatz einer vertrauenswürdigen Entität (*Trustworthy entity*), gekennzeichnet durch ein definiertes Verhalten und zugesicherte Eigenschaften, führt zu erwarteten Anschlusshandlungen und sichert den Erfolg von Regulierungsmaßnahmen.

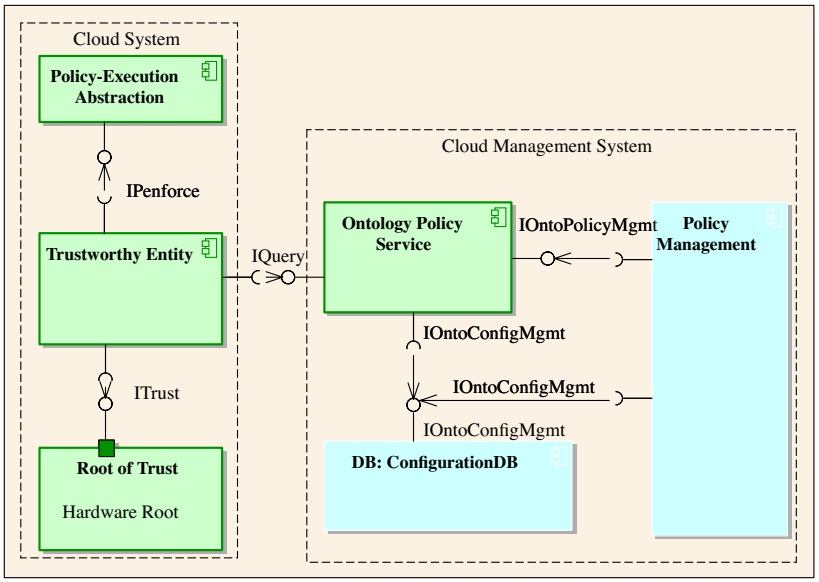

Abbildung 4.1: Pattern für ein Verbundkonzept vertrauenswürdiger Entitäten

Das Pattern Abbildung 4.1 zeigt, dass sich ein bestehender Regulierungsbereich erfolgreich erweitern lässt, wenn die entsprechende Transition (*Policy-Execution Abstraction*) auf der Grundlage einer hardwaregestützten Vertrauensbasis (*Root of Trust*) ausgeführt werden kann. Neben anderen Systemeigenschaften repräsentiert das Pattern einen Vertrauenspunkt (*Point of Trust*) und erweitert technisch den Handlungsbereich eines Cloud-Anwenders zur Durchsetzung seiner Regulierungsanforderungen.
Die konsequente Anwendung von Regulierungsanforderungen auf allen Ebenen einer Cloud-Architektur führt zur Entwicklung eines vollständigen Trusted-Cloud-Architekturmodells, bestehend aus einem Verbund von vertrauenswürdigen Entitäten. Das Pattern in Abbildung 4.1 bildet dafür den Grundbaustein.

4.1 Ontologie-Konzept

Die Entscheidung für einen ontologischen Konzeptansatz entstand aus der Zielstellung, einen formalisierten und von Maschinen interpretierbaren Beschreibungsansatz sowohl für den Aspekt der Regulierung als auch für den zu regulierenden Domänenbereich für Cloud-Systeme anzuwenden.
Ontologien ermöglichen eine Konzeptualisierung und stellen ein Werkzeug für die formale Spezifikation einer spezifischen Domäne bereit. In Gruber et al. werden Ontologien als *formal explicit specifications of a shared conceptualization* [Gru+93] definiert.
Die Abbildung von Regulierungszielen in Bezug auf eine konkrete Domäne muss zwei Seiten berücksichtigen. Die Domäne muss in seiner Realität so beschrieben sein, dass ihr Verhalten beobachtet werden kann (*reality monitoring*). Darüber hinaus greifen Regulierungsmechanismen in die Domäne ein, sie definieren Strukturen und steuern Abläufe in Systemen, die eine Domäne repräsentieren [CRP06, S. 256]. Das erfordert eine konsistente Spezifikation des Systems, bestehend aus Objekten, Beziehungen und Bedingungen, wie sich das System verhalten wird, wenn es als reale Konstruktion überführt wird. Das Modell bietet ein Schema, eine Designbeschreibung über das zukünftige reale System.
Ursprünglich bestand die Absicht, die Ontologiesprache *Description Logic* (DL) [Baa10; Mot+12] einzusetzen, welche dem Prinzip einer *Open-world*-Annahme folgt. Eine *Open-world*-Annahme bedeutet, dass Fakten, die nicht im Modell repräsentiert sind, dennoch auftreten können [HSt11]. Nach der Entwicklung erster konzeptioneller Beispiele zur Abbildung von Konzepten einer Cloud-Domäne [Pan+12; Vel+09] und der Herausarbeitung spezifischer Sicherheitskonzepte [Kne10; HSD07] zeigte sich, dass dieser Ansatz im Kontext der Regulierung und der Durchsetzung von Sicherheitszielen nicht akzeptiert werden kann.
Es gilt der Grundsatz: Nicht regulierte Fakten eines realen Systems können so nicht auftreten. Nur auf dieser Grundlage kann die Effektivität der Regulierung gesichert werden.
Als Alternative zu *Description Logic* wurde die Ontologiesprache *ObjectLogic* ausgewählt, ein Nachfolger von *F-Logic* [KL89]. *ObjectLogic* erweitert die klassische Prädikatenlogik um ein objektorientiertes Konzeptualisierungsparadigma und folgt in Bezug auf die Konzeptualisierung einer *Closed-world*-Annahme.
ObjectLogic führt die logische Beschreibung auf Grundlage von Schemadefinitionen (Schema level statements) und Instanzendefinitionen (Instance level statements) durch.

Schema-Level
Ein Schema repräsentiert in einem objektorientierten Ansatz Klassen als Gegenstand der Konzeptualisierung und deren Beziehungen in Form von Methoden.

In ObjectLogic definieren *signature-F-atoms* spezifische Methoden einer Klasse, die für Instanzen dieser Klasse anwendbar sind [Gmb12]. Jede Methode beschreibt typisierte Parameter und Ergebniswerte, deren Wertebereich über die Kardinalität begrenzt wird. Der objektorientierte Ansatz ermöglicht die Definition von Subklassen (*Subclass-of Statements*) und die Weitergabe von bereits definierten Methoden der Oberklasse. Eine Hierarchie von Klassen wird syntaktisch mithilfe von *subclass-F-atoms* ausgedrückt. Subklassen einer übergeordneten Konzeptklasse sind durch ein '**::**' gekennzeichnet.

Die Menge an *signature-F-atoms* zusammen mit der Hierarchie an Klassen beschreiben das Gesamtschema der Ontologie und bilden das Vokabular.

Instance-Level

Mit einer Instance-Level-Beschreibung erfolgt die Formulierung konkreter Fakten, basierend auf das spezifizierte Schema. Es stehen Syntaxelemente zur Erzeugung von Instanzen (*isa-F-atoms*) von unterschiedlichen Schemadefinitionen zur Verfügung. Die Mitgliedschaft zu einer Konzeptklasse ist durch ein '**:**' ausgedrückt.

In ObjectLogic wird die Anwendung von Methoden mithilfe von *data-F-atoms* formuliert. Darin befindet sich ein Hauptobjekt (host object), eine Methode und ein Bezug auf das Ergebnisobjekt (result object). Methoden können Parameter besitzen, die als einschränkende Eigenschaft einen Bezug auf das Ergebnisobjekt definieren. Die Anwendung von Variablen auf alle Bereiche eines *data-F-atoms* ist möglich und führt zur Formulierung von Anfragen (queries) in Bezug auf eine Methode.

Für die Evaluierung werden Referenz-Cloudarchitekturen als Menge einzelner Architekturbausteine und regulierende Policies mithilfe von Instance-Level-Definitionen beschrieben.

Reasoning-Rules und logische Abfragen

Die Menge an expliziten Fakten, die sich aus der Ontologie ableiten, stellt die Grundmenge des direkt in Beziehung stehenden Faktenwissens dar. Mit *Reasoning-Rules* steht ein Instrument zur Formulierung komplexerer Abhängigkeiten zwischen expliziten Fakten zur Verfügung. Die Anwendung von *Reasoning-Rules* führt zu erweiterten Schlussfolgerungen, die in der direkten Bewertung der Faktenbasis von bestehenden Konzeptklassen nicht sichtbar sind. Für die Konzeptklassen *Vertrauen* oder *Status* können unter Anwendung von *Reasoning-Rules* Werte abgeleitet werden, die auf Grundlage einer konkreten Faktenlage entstehen.

Listing 4.1: Struktur einer *Reasoning-Rule*

```
1  @{MutualFriendship}
2  ?X[friend -> ?Y]
3  :-
4  ?Y:Person[friend -> ?X].
```

Die Struktur einer *Reasoning-Rule* unterscheidet die in Listing 4.1 dargestellten Bereiche: (1) Rule-Bezeichnung, (2) einen Rule-Header und getrennt durch die Zeichenfolge ':-' (3) einen Rule-Body.
Der Rule-Header beschreibt eine Beziehung zwischen Konzeptklassen, die als Schlussfolgerung entsteht. Im Rule-Body werden axiomatisch Abhängigkeiten beschrieben, die zu der im Rule-Header definierten Schlussfolgerung führen können. Durch logische Verknüpfungen kann der Rule-Body zu komplexeren Abhängigkeitsaussagen erweitert werden.

4.1.1 Strukturentwurf Ontologie

Die verteilte Ontologie-Architektur in Abbildung 4.2 ermöglicht die Konzeptualisierung für den Aspekt Regulierung (*Regulation:Ontology*, nachfolgend als Ontologie-*Regulation* bezeichnet) und abstrahiert diesen Teil von einem Cloud-System (*Cloud-Domain:Ontology*, nachfolgend als Ontologie-*Cloud-Domain* bezeichnet), der als Gegenstand der Regulierung zur Anwendung kommt.
In Bezug auf den geführten Diskurs über Regulierung im Abschnitt 2.2 als strukturbildendes Element einer Organisation repräsentiert die Ontologie-*Regulation* eine eigenständige Konzeptualisierung.

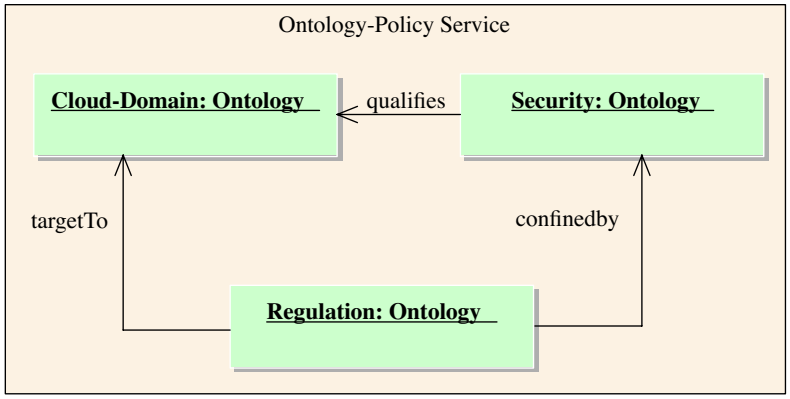

Abbildung 4.2: Ontologie Strukturentwurf

Die Ontologie-*Regulation* erlaubt die Einführung und Beschreibung notwendiger Konzepte für Aktivitäten (*Activity*), ihrer einschränkenden Bedingungen (*Constraint*) und ihrer Definitionsrahmen in Form von Regeln (*Rule*). Die Konzeptualisierung ermöglicht die Entwicklung von Ablaufstrukturen (*Workflow*), deren Bestandteile in Form von Politiken (*Policy*) definiert und mit ihren Abhängigkeiten für unterschiedliche Zielrichtungen gruppiert und auf ein reales Cloud-System in Form technischer Transformationen (*Transformation*) vertrauenswürdig durchgesetzt werden. Die Gruppierung folgt strategischen Aspekten und bildet in ihrer Zusammenfassung eine regulierte Gesamtstrategie ab, die auf ein Cloud-System übertragen werden kann.
Die Ontologie-*Cloud-Domain* dient zur Konzeptualisierung einer Cloud-Architektur und beschreibt den Gegenstand der Regulierung. Architekturbausteine, Anwendungen und Benutzerinteraktionen in einem Cloud-System werden als Konzepte eingeführt, axiomatisch beschrieben und als interpretierbares Domänenvokabular in die Formalisierung für die Regulierung einbezogen (siehe Abschnitt 4.1.4).
Die Verbindung beider Ontologien verfolgt das Prinzip, die Konzeptualisierung struktureller, statischer Konzeptklassen (Ontologie-*Cloud-Domain*) mit einer Konzeptualisierung für eine präzise Regulierung (Ontologie-*Regulation*) zu einem regulierten, formalen Systembegriff (*Cloud-System*) zusammenzuführen.
Die Ontologie-*Security* dient der Formalisierung von Sicherheitsaspekten. Die Ontologie erweitert den funktional basierten Konzeptualisierungsansatz einer Cloud-Domain durch die Einführung qualifizierender Konzepte wie z. B. Vermögenswerte (*Asset*) oder Sicherheitszielen wie Vertraulichkeit (*Confidentiality*), Echtheit (*Authenticity*) und Unversehrtheit (*Integrity*).

4.1.2 Ziele der ontologischen Konzeptualisierung

Die Konzeptualisierung für die Regulierung erfordert eine weitergehende Strukturierung zur Formulierung unterschiedlicher Regulierungsziele, die bereits im Abschnitt 3.1 als Anforderungen erhoben wurden.

Regulierung von Sicherheits- und Datenschutzzielen
Der Regulierungsrahmen umfasst die Prozesse für Deployment und Systemkonfiguration, die mit Regelungen zur Durchsetzung einer Sicherheitspolitik verknüpft werden können, so dass z. B. die Integrität bzw. die Authentizität von Software-Artefakten im Prozess des Deployments sichergestellt wird.

Die Konzeptualisierung von Regulierung erfordert die Formulierung von Einschränkungen zur Sicherstellung von Datenschutzzielen. Dieser Aspekt ist eng verknüpft mit der Regulierung des Benutzerverhaltens, die den Ansprüchen zur Gewährleistung einer *Governance* folgt und später für Maßnahmen zur Konformitätsüberprüfung (*Compliance*) einen Referenzpunkt darstellt.

Regulierung von Vertrauen
Eine eigenständige Konzeptualisierung widmet sich dem Aspekt Vertrauen (*Trust*). Regulierung von Vertrauen wird als eine Methodik für die gezielte Zusicherung von notwendigen Objekt- und Verhaltenseigenschaften einer betrachteten Domäne beschrieben. Die Integration der Ontologie-*Security* dient als formale Grundlage für die Überprüfung und zweifelsfreie Bewertung von Eigenschaften, die durch entsprechende Sicherheitszustände wie z. B. Zustand *identitätsüberprüft* (*authenticated*) oder Zustand *unversehrt* (*integrity-protected*) repräsentiert werden. Die formale Definition und Interpretation von Sicherheitseigenschaften stellt ein qualitätssicherndes Ausdrucksmittel für den Entwurf einer Trust-Policy bereit.

Logische Schlussfolgerungen über implizite Fakten
Ein weiterer Entscheidungsaspekt für einen ontologisch geführten Konzeptansatz liegt im Potenzial zur Anwendung logischer Schlussfolgerungen (*Reasoning*). Durch einen spezifizierten Ansatz logischer Schlussfolgerungen lässt sich aus der Wissensdatenbank implizites Wissen ermitteln, welches sich in der Regel einer expliziten Konzeptualisierung entzieht. Diese Fähigkeit kommt speziell für die Konzepte Systemzustand (*States*), Vertrauenswürdigkeit (*Trustworthiness*) und Risiko (*Risk*) zum Tragen. Das Ontologiekonzept stellt dafür eine Menge von logischen Regelsätzen (Logical Rules) bereit, um implizite Eigenschaften ableiten zu können.

Obwohl in Kagal et al. [KFJ05] der Begriff *Trust Ontology* verwendet wird, so wird mit der vorliegenden Spezifikation keine spezifische Ontologie-*Trust* für die Konzeptualisierung von Vertrauen eingeführt. Der Aspekt Vertrauen soll allein mit Mitteln der logischen Schlussfolgerungen und den Schemadefinitionen der verteilten Ontologiestruktur in Abbildung 4.2 herausgearbeitet werden.

4.1.3 Ontologie Regulierung

Die Konzeptualisierung von *Regulierung* wird durch ein eigenständiges Ontologiekonzept repräsentiert. Unabhängig von einem spezifischen zu regelnden Gegenstand werden allgemeine Konzepte und regulierende Prinzipien herausgearbeitet.
Im Vergleich zu Kagal et al. [KFJ03] vermeidet die Konzeptualisierung komplexe syntaktische Ausdrücke und entwickelt ein formales Vokabular zur Beschreibung unterschiedlicher Regulierungsziele. Das Konfliktmanagement während der Ausführung von Policies ist kein direkter Bestandteil der Policy-Formalisierung, sondern wird über ein implizites Policy-Abhängigkeitskonzept behandelt. Ein Konzept logischer Schlussfolgerungen gewährleistet eine Umsetzung einer Policy gemäß den festgelegten Abhängigkeiten und durch Auswertung gesetzter Prioritäten.

4.1.3.1 Haupthierachie *Regulation*-Ontology

Die Haupthierarchie der Ontologie-*Regulierung* setzt sich aus folgenden Konzeptklassen (siehe Listing 4.2) zusammen:

Listing 4.2: Konzepthierarchie *Regulation*-Ontology – Subclass-F-atoms

```
1  Action[].
2  Constraint[].
3  Policy[].
4  Rule[].
5  State[].
6  Transformation[].
```

4.1.3.2 Konzeptklasse *Action*

Die Konzeptklasse *Action[]* umfasst die formale Beschreibung von Aktivitäten, die in Bezug auf den Regulierungsgegenstand ein Konzept für Handlungsanweisungen beschreiben. Subkonzepte Konzeptklasse *Action* führen Kategorien zur Spezialisierung von Aktivitäten für die Bereiche Deployment, Datenzugriff, Sicherheit und Geschäftsprozesse (*Business*) ein.

Listing 4.3: Subklassen der Konzeptklasse *Action* – Subclass-F-atoms

```
1  ActionBusiness::Action.
2  ActionDataAccess::Action.
3  ActionDeployment::Action.
4  ActionSecurity::Action.
```

Instanzen der Subklassen in Listing 4.3 bilden das Vokabular für einen spezifischen Handlungsrahmen in Bezug auf eine ausgewählte Regulierungsdomäne.

4.1.3.3 Konzeptklasse *Constraint*

Constraints sind spezielle prädikatenlogische Formeln, die Bedingungen oder Einschränkungen beschreiben [HW07]. Die Konzeptklasse *Constraint* formalisiert den Aspekt einer weiteren Spezifizierung und Verfeinerung von Policies und beschreibt zusätzliche Bedingungen in Bezug auf alle Elemente der Konzeptklasse *Rule* (siehe Listing 5.53).

Listing 4.4: Methoden der Konzeptklasse *Constraint* – Signature-F-atoms

```
1  Constraint[is_precondition {0:*} *=> _boolean].
2  Constraint[is_postcondition {0:*} *=> _boolean].
3  Constraint[is_invariant {0:*} *=> _boolean].
```

Die Konzeptklasse *Constraint* definiert für eine Constraint-Typisierung die Methoden *is_precondition*, *is_postcondition* und *is_invariant*.

Listing 4.5: Subklassen der Konzeptklasse *Constraint* – Subclass-F-atoms

```
1  ActionConstraint::Constraint.
2  ObjectConstraint::Constraint.
3  SubjectConstraint::Constraint.
```

Das Subklassenkonzept in Listing 4.5 erlaubt eine spezifische Verfeinerung der Bedingungen bezogen auf die in der Regulierung beteiligten Konzeptklassen *Action*, *Object* und *Subject*.

Listing 4.6: Methoden der Konzeptklasse *ActionConstraint* – Signature-F-atoms

```
1 ActionConstraint[hasDateConstraint {0:1} *=> _dateTime].
2 ActionConstraint[hasTimeConstraint {0:1} *=> _time].
3 ActionConstraint[hasQuantity {0:1} *=> _int].
```

Für jede *Rule*-Instanz kann über eine eigene Instanz der *Constraint*-Konzeptualisierung ein spezifischer Kontext definiert werden. Als Beispiel werden für die Subklasse *ActionConstraint* in Listing 4.6 Methoden für die weitere Begrenzung von Aktivitäten dargestellt. Die Methoden bestimmen z. B. den Zeitpunkt oder die Anzahl möglicher Operationsausführungen.

4.1.3.4 Konzeptklasse *Rule*

Die grundlegenden Prinzipien für eine Regulierung werden durch die Methoden der Konzeptklasse *Rule* in Listing 4.7 definiert. (Die Abkürzung *nsd* in Listing 4.7 repräsentiert einen Namespace zur Verknüpfung mit der Ontologie-*Cloud-Domain*).

Listing 4.7: Methoden der Konzeptklasse *Rule* – Signature-F-atoms

```
1 Rule[transformedBy {0:*} *=> Transformation].
2 Rule[nextRule {0:*} *=> Rule].
3 Rule[hasResult {0:*} *=> RuleResult].
4 Rule[definedContext {1:*} *=> Constraint].
5 Rule[do(Action) {1:1} *=> <nsd#Architecture>].
```

Innerhalb der Konzeptklasse *Rule* erfolgt über die Methode *do(Action)* die Festlegung, welche Aktivität in Bezug auf das zu regulierende Objekt ausgeführt werden soll. Axiomatisch wird über die Konzeptklasse *Action* die Beziehung zwischen der Konzeptklasse *Rule* und der Ontologie-*Cloud-Domain* hergestellt. Die Konzeptklasse *Rule* definiert neben einer Aktivität über die Methode *definedContext* einen Kontext (siehe Abschnitt 4.1.3.3). Für die Attestierung wird über die Methode *hasResult* ein Resultat der Rule-Ausführung gesetzt. Instanzen der Konzeptklasse *RuleResult* repräsentieren einen Status. Zur Steuerung von Abhängigkeiten definiert die Methode *nextRule* eine Instanz der Konzeptklasse *Rule*, die als nachfolgende Regel zur Ausführung kommt. Die Kardinalität erlaubt es, eine oder mehrere Instanzen der Konzeptklasse *Rule* als Nachfolger anzugeben.

4.1.3.5 Konzeptklasse *Policy*

Die Policy-Konzeptualisierung beschreibt eine Menge von Instanzen der Konzeptklasse *Rule* und bietet ein Instrument für die formale Beschreibung spezifischer Regulierungsziele. Die Konzeptklasse *Policy* in Listing 4.8 legt auf oberster Hierarchieebene über die Methoden *targetToZone* den Anwendungsbereich der Policy fest. Mit den Methoden *resultedState* und *expectedState* erfolgt eine Verknüpfung für eine Statusrepräsentation. Abhängigkeiten zwischen einzelnen Instanzen der Konzeptklasse *Policy* werden über die Methode *dependOn* gesteuert. Die Methode *transitionState* beschreibt den Zustandswechsel im Zuge der Policy-Umsetzung.

Listing 4.8: Methoden der Konzeptklasse *Policy* – Signature-F-atoms

```
Policy[targetToZone {1:1} *=> PolicyZone].
Policy[dependOn {0:*} *=> Policy].
Policy[resultedState {0:1} *=> State].
Policy[expectedState {1:*} *=> State].
Policy[transitionState {1:1} *=> _boolean].
Policy[reasonTrustBase {0:N} *=> PolicyTrust].
```

Die Ausführung von Policies wird über die in Listing 4.9 spezifizierte Konzeptklasse *PolicyZone* gesteuert. Instanzen der Konzeptklasse *PolicyZone* beschreiben Zuständigkeitsbereiche einer IT-Architektur und geben den Anwendungsbereich von Policies vor. Der Separierungsansatz vermeidet Konflikte bei der Policy-Ausführung.

Listing 4.9: Methoden der Konzeptklasse *PolicyZone* – Signature-F-atoms

```
PolicyZone[zonetype {0:*} *=> _string].
PolicyZone[assignedEntity {1:*} *=> <nsd#TE>].
PolicyZone[assignedLayer {1:*} *=>  <nsd#ArchitecturalLayer>].
```

Die Methode *assignedLayer* referenziert auf einen spezifizierten Architekturbereich der Ontologie-*Cloud-Domain* (siehe Abschnitt 4.1.4.4). Eine vertrauenswürdige Entität wird mit der Methode *assignedEntity* für die Ausführung von Policies im festgelegten Architekturbereich autorisiert. Das Konzept einer *PolicyZone* folgt der im Abschnitt 3.1 erhobenen Anforderung, dass sich die Ausführung einer Policy auf eine überprüfbare Vertrauensbasis zurückführen lassen muss.
Ausgehend von der Konzeptklasse *Policy*, werden in Listing 4.10 weitere Subkonzeptklassen definiert und erzeugen eine Policy-Taxonomie zur Spezifizierung von Regulierungszielen.

Listing 4.10: Subklassen der Konzeptklasse *Policy* – Subclass-F-atoms

```
PolicyTrust::Policy.
PolicySecurity::Policy.
PolicyPrivacy::Policy.
PolicyDeployment::Policy.
```

Jede Subklasse definiert zusätzliche Methoden *hasRule*, um spezifische Instanzen der Konzeptklasse *Rule* unabhängig von ihrer Ausführungsreihenfolge zu gruppieren. Die Konzeptklasse *PolicyTrust* steht stellvertretend für alle Subklassen der Konzeptklasse *Policy* und erweitert die bestehende Schemadefinition in Listing 4.8 um die folgenden Methoden:

Listing 4.11: Methoden der Konzeptklasse *PolicyTrust* – Signature-F-atoms

```
PolicyTrust[hasTrustRule {1:*} *=> RuleTrust].
PolicyTrust[firstTrustRule {1:1} *=> RuleTrust].
```

Für eine Trust-Policy lassen sich z. B. mit der Methode *hasTrustRule* spezifische Instanzen der Konzeptklasse *RuleTrust* gruppieren. Die Methode *firstTrustRule* definiert die erste Rule-Instanz innerhalb einer Policy, die zur Ausführung kommt. Die Methode *reasonTrustBase*, konzeptionell hinterlegt und ausgeführt als *Reasoning-Rule*, referenziert für alle untergeordneten Policy-Konzeptklassen auf eine bestehende Vertrauensgrundlage, die im Ergebnis einer Trust-Policy hergestellt wurde.

4.1.3.6 Konzeptklasse *State*

Die Konzeptklasse *State* beschreibt in Listing 4.12 unterschiedliche Kategorien von Zuständen und schafft die Voraussetzung für eine nachgelagerte Bewertung der Policyeffektivität in Bezug auf gesetzte Regulierungsziele.

Listing 4.12: Subklassen der Konzeptklasse *State* – Subclass-F-atoms

```
1 StateDeployment :: State .
2 StatePrivacy :: State .
3 StateSecurity :: State .
4 StateTrust :: State .
5 StateService :: State .
```

Auf oberster Hierarchieebene wird eine Beziehung zur Policy gesetzt. Die Instanz einer Policy beschreibt konkrete Regulierungsziele, die durch ein Transformationskonzept (siehe Abschnitt 4.1.3.7) umgesetzt werden.

Listing 4.13: Methoden der Konzeptklasse *State* – Signature-F-atoms

```
1 State[enforcedBy {1:1} *=> Policy].
```

Der im Ergebnis der Transformation erzeugte neue Zustand wird durch eine Instanz der Konzeptklasse *Status* repräsentiert.

4.1.3.7 Konzeptklasse *Transformation*

Die Konzeptualisierung *Transformation* schafft den Übergang von einer wissensbasierten *Policy*-Konzeptualisierung zu einer technischen Abbildung in eine Cloud-Domain. Die regulative Kopplung mit einem Cloud-System benötigt technische Schnittstellen zur Übertragung von Policies in ein Cloud-System. Ausgehend von der Konzeptklasse *Transformation* werden in Listing 4.15 weitere Subkonzeptklassen zur Definition technischer Integrationskonzepte eingeführt. Das Transformationskonzept stellt Methoden bereit, die ein deklarativ beschriebenes Policy-Modell in technologieabhängige Ausführungsschritte überführen.

Listing 4.14: Methoden der Konzeptklasse *Transformation* – Subclass-F-atoms

```
1 ansible :: Transformation .
2 puppet :: Transformation .
3 chef :: Transformation .
4 vagrant :: Transformation .
5 attestation :: Transformation .
```

Im Ergebnis erfolgen darüber Transitionen in Bezug auf einen bestehenden System- oder Sicherheitszustand. Die Expressivität axiomatisch beschriebener Policy-Modelle benötigt auf der Ebene der Konzeptklasse *Transformation* unterschiedliche Schnittstellenkonzepte, um eine adäquate technische Adaption gewährleisten zu können.
Die in Listing 4.15 vorgestellten Subkonzeptklassen stellen einen technischen Rahmen bereit, um das Deployment von Virtualisierungs-, Plattform- und Anwendungskomponenten, ebenso für alle Bereiche

des Konfigurationsmanagements, ausführen zu können. Darüber hinaus müssen Instanzen dieser Subkonzeptklassen auch Funktionen für die Durchsetzung und Einrichtung von Sicherheitsmodellen und den damit verbundenen Sicherheitsfunktionen ausführen können.
Im Rahmen einer Bachelorarbeit [Klu16] wurden erste Transformationstechnologien als Proof-of-Concept untersucht. Diese Untersuchungen werden im Rahmen der Evaluierung fortgesetzt. Die Komponente *Policy-Execution Abstraction* in der Abbildung 4.1 repräsentiert z. B. eine JAVA-basierte Implementierung der *Transformation*-Konzeptualisierung.
Eine Herausforderung der *Transformation*-Konzeptualisierung bildet die Formalisierung komplexer technischer Konfigurationen, die sich aus einer technologieunabhängigen F-Logik-basierten deklarativen Beschreibung von Systemattributen oder Methoden ableiten lassen.
Im Mittelpunkt der Konzeptualisierung steht eine flexible Transformationsbeschreibung, um eine beliebige technische Konfigurationskomplexität abbilden zu können. Die vorgestellte Konzeptualisierung lässt sich um weitere Transformationstechnologien erweitern und stellt einen ersten Entwurf dar, auf dessen Grundlage die technische Umsetzbarkeit der vorgestellten *Policy*-Konzeptualisierung im Abschnitt 4.1.3.5 bewertet werden kann.

Listing 4.15: Ansible Transformation – Signature-F-atoms

```
1  ansible[transformAction(Action) {0:*}    *=> _string].
2  ansible[hasModule {1:1} *=> _string].
3  ansible[setRealParameter {1:1} *=> _string].
4  ansible[hasParameter {1:1} *=> _string].
5  ansible[transformParameter(_string) {0:*}        *=> _string].
```

4.1.4 Ontologie Cloud-Domain

Die Konzeptualisierung für eine Regulierung erfordert die formale Abbildung des Regulierungsgegenstandes. Cloud-Systeme stellen sich als komplexe, verteilte Architekturen mit unterschiedlichen Technologien für die Servicebereitstellung dar. Die Entscheidung, eine eigenständige Ontologie-*Cloud-Domain* einzusetzen, schafft eine Modellierungsgrundlage für die Abbildung unterschiedlicher zu regelnder Cloud-Architekturkonzepte.

4.1.4.1 Konzeptklasse *CloudDomain*

Auf der jeweiligen oberen Ebene werden grundlegende, vererbbare Methoden (Signature-F-atoms) eingeführt, die für alle in der Hierarchie darunter liegenden Konzepte der Ontologie zur Verfügung stehen. Abgeleitete Instanzen dieser Klassen verwenden die Methoden, um konkrete Beziehungen gemäß den Konzeptdefinitionen zu realisieren.
Die Haupthierarchie Ontologie-*CloudDomain* (siehe Listing 4.16 / 4.17) setzt sich aus den Kernkonzepten zusammen:

Listing 4.16: Konzepthierarchie *Cloud-Domain[]* – Subclass-F-atoms

```
1  CloudDomain[].   ArchitecturalLayer::CloudDomain.
2                   Entity::CloudDomain.
3                   Subject::CloudDomain.
```

Die Top-Level-Hierarchiekonzepte *CloudDomain* und *CloudService* bilden die konzeptionellen Einstiegspunkte. Während mit *CloudDomain* die Konzeptualisierung einer IT-Architektur im Vordergrund steht, so werden mit *CloudService* veränderbare Rahmenbedingungen einer Domäne, wie fachliche Zuständigkeiten (*CloudRoles*), geografische Verteilungen (*CloudRegion*) und Geschäftsmodelle (*CloudServiceLayer*), in die Spezifikation einbezogen.

Listing 4.17: Konzepthierarchie *CloudService[]* – Subclass-F-atoms

```
1  CloudService[].  CloudConnection::CloudService.
2                   CloudRegion::CloudService.
3                   CloudRoles::CloudService.
4                   CloudServiceLayer::CloudService.
```

Auf Top-Level-Ebene wird die Beziehung zwischen beiden Hauptkonzepten definiert. Jede Instanz einer konkreten IT-Architektur kann einem oder mehreren konkreten Cloud-Systemen zugeordnet werden.

Listing 4.18: Methoden der Konzeptklasse *CloudDomain* – Signature-F-atom

```
1  Domain[assignedCloud {1:*} *=> CloudService].
```

Die Konzepte unter *CloudDomain* umfassen logikbasierte Beschreibungen von Software-Hardware-Bausteinkonzepten, auf deren Grundlage sich lösungsorientierte Architekturkonzepte ableiten lassen.
Die Architektur eines Softwaresystems besteht aus seinen Strukturen, der Zerlegung in Komponenten und deren Schnittstellen und Beziehungen untereinander [Sta15]. Die Zweckbestimmung führt zu konkreten Entwurfsentscheidungen, so dass die finale Architektur den angestrebten Lösungsansätzen gerecht wird. Beginnend mit den Konzeptklassen aus Listing 4.16, werden weitere Subkonzeptklassen (Subclass-F-atoms) definiert und erzeugen die Taxonomie für eine IT-Architektur.

4.1.4.2 Konzeptklasse *Entity*

Die Konzeptklasse *Entity* bildet die Oberklasse der Subkonzeptklassen *User* und *Object*.

Listing 4.19: Subklassen der Konzeptklasse *Entity* – Subclass-F-atoms

```
1                   User::Entity.
2                   Object::Entity.
```

Die Konzeptualisierung von Entitäten erfolgt in Form von User-Signature Statements (Signature-F-atoms).

Listing 4.20: Methoden der Konzeptklasse *Entity* – Signature-F-atoms

```
1  Entity[assignedIdentity {0:*} *=> <nss#Identity>].
2  Entity[authorizedTo(<nss#Credential>) {0:*} *=> <nss#Identity>].
3  Entity[checkQualityProperty {0:*} *=> <nss#QualityProperty>].
```

Die Methode *assignedIdentity* ordnet Instanzen der Konzeptklasse *Entity* mögliche Identitäten zu. (Die Abkürzung *nss* in Listing 4.20 repräsentiert einen Namespace zur Verknüpfung mit der Ontologie-*Security*.) Die Methode trifft keine Aussage darüber, wie eine sichere Zuordnung zwischen beiden Konzeptklassen gewährleistet wird.

Die Sicherheitsbetrachtung erfolgt mit der Konzeptklasse *Identity* und *Credential* im Abschnitt 4.1.5. Mit der Methode *authorizedTo(<nss#Credential>)* entsteht die konzeptionelle Grundlage, um gestützt auf eine Policy verifizierbare Zuweisungen von User- auf Objekt-Identitäten (Identity-Mapping) zu gewährleisten.
Die Zuordnung von Qualitätseigenschaften auf Teile der Cloud-Domäne schafft den konzeptionellen Ausgangspunkt, um Anforderungen an die Sicherheitseigenschaften syntaktisch auszudrücken und regulativ mit einer Policy zu verbinden. Der konzeptionelle Ansatz wird im Abschnitt 4.2.1 zu einem Konzept vertrauenswürdiger Entitäten weiterentwickelt. Die Konzeptklasse *QualityProperty* beschreibt Sicherheitseigenschaften und wird im Abschnitt 4.1.5 eingeführt.
Die Konzeptklasse *User* bildet die Grundlage zur Anwendung von Cloud-Anwender- und Cloud-Anbieter-Entitäten als handelnde Akteure einer Cloud-Domäne. Die Konzeptklasse *User* repräsentiert natürliche Benutzer.

Listing 4.21: Methoden der Konzeptklasse *User* – Signature-F-atoms

```
1 User[hasDomainRole {0:*} *=> CloudRoles].
```

Die Konzeptklasse *User* übernimmt die Methoden der Konzeptklasse *Entity* und wird in Listing 4.21 um die Methode *hasDomainRole* erweitert. Darauf aufbauend lassen sich repräsentative Instanzen der Konzeptklasse *User* ableiten. Fachliche Rollen (*CloudRoles*) sind der Cloud-Domäne zugeordnet und beschreiben den Berechtigungsumfang einer Instanz der Konzeptklasse *User*.
Die Methode *hasDomainRole* bestimmt in der weitergehenden Sicherheitsbetrachtung den Berechtigungsrahmen und damit den Handlungsrahmen handelnder Akteure (siehe Abschnitt 4.1.5).

4.1.4.3 Konzeptklasse *Subject*

Die Konzeptklasse *Subject* in Listing 4.22 repräsentiert einen Prozess zur Ausführung einer Softwarekomponente als Instanz der Konzeptklasse *Software*. Jede Instanz der Konzeptklasse *Subject* übernimmt dabei die Identität und die damit verbundenen Berechtigungen der zugehörigen Instanz der Konzeptklasse *User*.

Listing 4.22: Methoden der Konzeptklasse *Subject* – Signature-F-atoms

```
1 Subject[hasName {1:1} *=> _string].
2 Subject[runProcessAs(<nss#Identity>) {1:1} *=> Software].
3 Subject[entityID {1:1} *=> ()].
```

Die Einführung der Konzeptklasse *Subject* schafft den konzeptionellen Rahmen zur Bewertung konkreter Cloud-Prozesse und ermöglicht die Ermittlung vertrauenswürdiger Prozesse auf der Grundlage logischer Schlussfolgerungen.

4.1.4.4 Konzeptklasse *ArchitecturalLayer*

Für die Anwendung architekturorientierter Formalisierungen wird die Konzeptklasse *ArchitecturalLayer* als oberste Ebene eingeführt. Die flexible Zuordnung von Komponenten oder Systemkonzepten in ein architektonisches Schichtenmodell wird über die Subklassen *PlatformLayer*, *VirtualisationLayer*, *RuntimeLayer* und *ServiceLayer* repräsentiert.

Die konkrete Abbildung zu Instanzen eines Schichtenmodells erfolgt über die Anwendung logischer Regelsätze, so dass Architekturkonzepte und Designentscheidungen eine eigene interpretierbare Beschreibungsform erhalten.

4.1.4.5 Konzeptklasse *Object*

Die Konzeptklasse *Object* bildet die Grundlage zur Anwendung von Cloud-System-Entitäten innerhalb einer Cloud-Domäne und wird weiter in die Subkonzeptklassen *Part* und *Component* (siehe Listing 4.23) unterteilt. Instanzen der Konzeptklasse *Component* bilden unteilbare Einheiten einer Cloud-Architektur.

Listing 4.23: Subklassen der Konzeptklasse *Object* – Subclass-F-atoms

```
1 Component::Object.
2 Connection::Object.
3 Part::Object.
```

Die Konzeptklasse *Component* definiert die Subkonzeptklassen *Hardware* und *Code*, wobei letztere die Top-Level-Hierarchieebene für die Konzeptualisierung softwarebasierter Komponenten bildet. An dieser Stelle erfolgt die Aufteilung in Hardwarekomponenten und Softwarekomponenten. Softwarebasierte Subkonzeptklassen sind in Listing 4.25 dargestellt und qualifizieren Softwarebausteine unterhalb der Ebene der Konzeptklasse *Code*.
Im Bereich der Architektur wird auf Objektebene das Konzept *State* (siehe Ontologie-*Regulation* im Abschnitt 4.1.3.6) verwendet und erlaubt für alle Instanzen der Architektur die Beschreibung eines spezifischen Status (z. B. Systemstatus, Sicherheitsstatus).
Die Konzeptklasse *Object* übernimmt die Methoden der Konzeptklasse *Entity* und erweitert die Konzeptklasse für die syntaktische Beschreibung von Zustandswerten (Methode *hasSystemState*).

Listing 4.24: Methoden der Konzeptklasse *Object* – Signature-F-atoms

```
1 Object[hasSystemState {1:1} *=> <nsr#State>].
```

(Die Abkürzung *nsr* in Listing 4.24 repräsentiert einen Namespace zur Verknüpfung mit der Ontologie *Regulation*.)
Der Detaillierungsgrad auf Ebene von Subkonzeptklassen folgt dem Prinzip notwendiger Beschreibbarkeit von Systemkomplexität, um gegebene Regulierungsanforderungen im Bereich der Cloud-Domain hinreichend abbilden zu können.

Listing 4.25: Subklassen der Konzeptklasse *Component* – Subclass-F-atoms

```
1 Code::Component.
2 Hardware::Component.
```

Listing 4.26: Subklassen der Konzeptklasse *Code* – Subclass-F-atoms

```
1 DataCode::Code.
2 MachineExecutableCode::Code.
3 ObjectCode::Code.
4 SourceCode::Code.
```

Innerhalb der Konzeptklasse *MachineExecutableCode* wird eine umfangreiche Subklassenspezifikation vorgenommen (siehe Anhang: IV). Es werden nachfolgend wesentliche Subklassen eingeführt, die den konzeptionellen Rahmen schaffen, um auf unterschiedlichen Serviceebenen die Anwendung der Konzeptklassen der Ontologie-*Regulation* anzuwenden.

Listing 4.27: Subklassen der Konzeptklasse *MachineExecutableCode* – Subclass-F-atoms

```
1  OperatingSystemCode::MachineExecutableCode.
2  DatabaseSystemCode::MachineExecutableCode.
3  ServiceCode::MachineExecutableCode.
4  VirtualComponent::MachineExecutableCode.
```

Die Konzeptklasse *OperatingSystemCode* umfasst die Eigenschaften und Beziehungen eines Betriebssystems. Im Rahmen der Konzeptualisierung werden ausgehend von den Grundkonzeptklassen *OSKernelSpace* und *OSUserSpace* alle wesentlichen strukturellen Teilsysteme eines Betriebssystems eingeführt. Die Konzeptklasse *DatabaseSystemCode* repräsentiert den ausführbaren Programmcode einer Datenbank innerhalb einer Laufzeitumgebung und die Konzeptklasse *ServiceCode* den ausführbaren Programmcode einer Serviceanwendung innerhalb der Anwendungsumgebung. Für die Beschreibung virtueller Komponenten und Ressourcen wird eine Konzeptualisierung ausgehend von der Konzeptklasse *VirtualComponent* geführt.

4.1.4.6 Konzeptklasse *Part*

Das *Part*-Konzept in Listing 4.28 enthält Methoden für die Integration von Instanzen der Konzeptklasse *Component* und Instanzen der eigenen Konzeptklasse *Part*. Instanzen des *Part*-Konzepts bilden die konzeptionelle Grundlage für die Beschreibung und den Entwurf modularer Systembausteine.

Listing 4.28: Methoden der Konzeptklasse *Part* – Signature-F-atoms

```
1  Part[isPartOf {0:*} *=> Part].
2  Part[hasSystemName {0:1} *=>_string].
3  Part[hasParts {0:*, inverseOf(isPartOf)} *=> Part].
4  Part[hasComponent {0:*} *=> Component].
5
6  Part[isConnectedTo {0:*} *=> Connection].
```

Die Konzeptklasse *Part* beschreibt neben einem Systemnamen (Methode *hasSystemName*) eine Komponenten-Aggregation (Methode *hasComponent*) bzw. eine Teilsystem-Aggregation (Methode *hasParts*). Architekturelle Ebenen eines IT-Systems (z. B. infrastrukturelle Plattform) lassen sich flexibel als Instanzen der Konzeptklasse *Part* beschreiben. Die vollständige Sicht über Abhängigkeiten und über die Systembaustein-Hierarchie wird mit Regeln der logischen Schlussfolgerung berechnet.

4.1.4.7 Konzeptklasse *Connection*

Die Konzeptualisierung von Verbindungen (*Connection*) wird als Teil der Taxonomie der Cloud-Domain geführt. Aus funktionaler Sicht bestimmen Verbindungen das Systemdesign und entstehen im Ergebnis von Best-Practise-orientierten Entwurfsmustern. Kleine Änderungen im Entwurf strukturbildender Verbindungen führen zu einer neuen Charakteristik des Systems und ändern das Systemverhalten.

Das Sprachkonzept von ObjectLogik beschreibt über die Anwendung von *Signature-F-atoms* notwendige Eigenschaften (*Kardinalität*, *Transititvität* und *Symmetrie*) einer Beziehung (*relation*). Die Eigenschaften einer Verbindung werden in Bezug auf ihre Qualitätseigenschaften durch eine eigene Konzeptklasse weiter präzisiert.

Listing 4.29: Methoden der Konzeptklasse *Connection* – Signature-F-atoms

```
1 Connection[checkQualityProperty {0:*} *=> <nss#QualityProperty>].
2 Connection[connectionType {0:*} *=> _string].
3 Connection[establishedConnection(User) {0:*} *=> VirtualPort].
```

Eine Verbindung (*Connection*) wird konzeptionell als Beziehung zwischen einer Instanz der Konzeptklasse *User* und einer Instanz der Konzeptklasse *VirtualPort* definiert. Die Konzeptklasse *VirtualPort* ist eine Subklasse der Konzeptklasse *VirtualComponent* im Listing 4.27. Über die Methode *checkQualityProperty* können jeder Instanz von *Connection* geforderte Qualitätseigenschaften zugeordnet werden. Eine Verbindung lässt sich über die Methode *connectionType* weiter qualifizieren.
Instanzen der Konzeptklasse *Connection* werden Teil konkreter Systembausteine der Konzeptklasse *Part* im Listing 4.28, falls Verbindungen zu anderen Systembausteinen gefordert sind. Für eine eigenständige Behandlung von Verbindungen wurde in der Part-Konzeptualisierung die Methode *isConnectedTo* eingeführt.

Listing 4.30: Methoden der Konzeptklasse *VirtualComponent* – Signature-F-atoms

```
1 VirtualComponent[isLinkedTo {0:*} *=> Hardware].
```

Die Konzeptklasse *VirtualComponent* stellt zusätzlich eine Methode bereit, um einen virtuellen Port mit einem existierenden Hardwareinterface verbinden zu können.
Aus regulativer Sicht werden über Verbindungen datenschutzrechtliche Verhaltensgrundsätze (*Privacy*) durchgesetzt und gleichzeitig Eigenschaften in Bezug auf die Übertragungssicherheit (*Security*) eingefordert. Die sichere Identifizierbarkeit von Übertragungsendpunkten bildet die notwendige Bedingung für die Gestaltung vertrauenswürdiger Verbindungen (*Trust*).
Der Konzeptansatz *Connection* schafft den Rahmen, um geforderte Qualitätseigenschaften der internen und externen Kommunikation zu formulieren und die Nachvollziehbarkeit bestehender Verbindungen über Reasoning-Konzepte sicherzustellen.

4.1.4.8 Konzeptklasse *CloudService*

Mit der Einführung der Konzeptklasse *CloudService* wird eine logikbasierte Beschreibung auf Fach- bzw. Geschäftsebene geführt (siehe Listing 4.31). Cloud-Servicemodelle bilden Kategorien unterschiedlicher Geschäftsmodelle (*CloudServiceLayer*) und beschreiben charakteristische Merkmale von Serviceangeboten [Bar+09b, S. 22].

Listing 4.31: Cloud-Service Classes – Subclass-of statements

```
1 IaaS::CloudServiceLayer.
2 PaaS::CloudServiceLayer.
3 SaaS::CloudServiceLayer.
```

Das Modell berücksichtigt allgemein akzeptierte Servicemodelle wie *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) und *Software as a Service* (SaaS). Die Form der Konzeptualisierung als orthogonaler Ansatz folgt dem Prinzip der architektonischen Zuordnung im Abschnitt 4.1.4.4 und kann um neue, zukünftige Servicemodelle in Form neuer Subkonzeptklassen der Kategorie *CloudServiceLayer* erweitert werden.

4.1.5 Ontologie Security

Die Formalisierung von Sicherheitszielen und Sicherheitsanforderungen wird eigenständig mit einer Ontologie-*Security* geführt. Wie bereits für den Bereich der Regulierung, wird die Ontologie später als qualifizierende Erweiterung mit der Ontologie-*Cloud-Domain* verknüpft. Ein Ziel der vorliegenden Ontologie-*Security* ist die Integration von Konzepten, auf deren Grundlage eine Wissensrepräsentation zur Bestimmung aktueller Sicherheitszustände erfolgen kann.
Für den Aspekt Security wird auf bestehende Ontologiekonzepte von Herzog et al. [HSD07] und von Fenz et al. [FE09] zurückgegriffen. Beide Ontologieansätze formalisieren den Aspekt Informations- und Datensicherheit als Wissensrepräsentation durch eine Weiterentwicklung bestehender Taxonomien oder durch die Entwicklung spezialisierter Taxonomien für bestimmte Teilbereiche (z. B. Threat-Taxonomie). Die hier vorgelegte Formalisierung stützt sich auf grundsätzliche Prinzipien zur Entwicklung vertrauenswürdiger IT-Sicherheitssysteme und erweitert ein funktional ausgerichtetes IT-Systemdesign (siehe Abschnitt 4.1.4) um ein evaluierbares IT-Sicherheitsdesign. Der Entwurf der Ontologie stützt sich auf das in Abbildung 4.3 vorgelegte Konzeptmodell zur Sicherstellung einer vollständigen Sicherheitsbetrachtung.

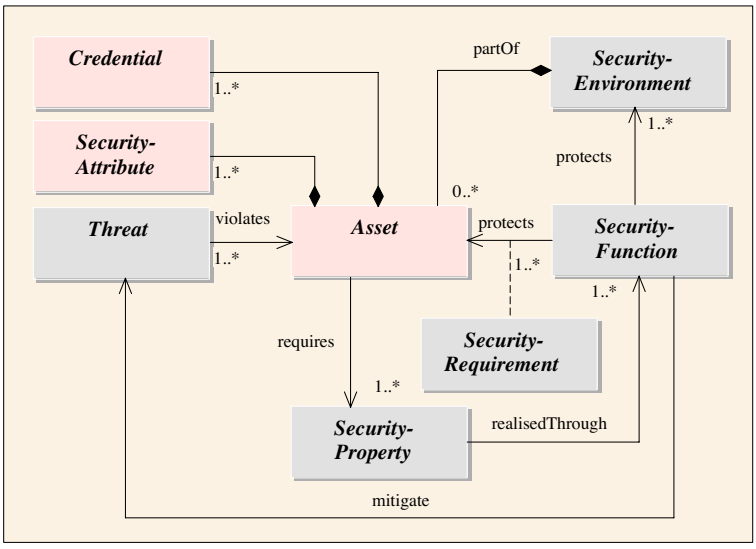

Abbildung 4.3: Konzept einer Sicherheitsstrategie

4.1.5.1 Konzept einer vertrauensbildenden Sicherheitsstrategie

Die Anwendung von Prinzipien für eine Bewertung vertrauenswürdiger IT-Systeme folgt den Richtlinien der Common Criteria (*Common Criteria for Information Technology Security Evaluation*) [CC12a; CC12d] und bildet eine wesentliche Grundlage der Security-Konzeptualisierung.

Das Vertrauen von Cloud-Anwendern in die Nutzung von Cloud-Services entsteht auf Grundlage von Zusicherungen darüber, dass sich der Sicherheitsstatus eines Cloud-Systems nur mithilfe geregelter Mechanismen ändern kann und den Vorgaben einer Sicherheits-Policy unverändert folgt. Die Zusicherungen über einen definierten Sicherheitszustand müssen einem Cloud-Anwender transparent und nachvollziehbar durch vertrauenswürdige Nachweise erbracht werden.
Im Listing 4.32 werden die Top-Level-Kategorien der Ontologie-*Security* eingeführt.

Listing 4.32: ObjectLogic – Regulation Concept Hierarchy

```
1  Asset[].
2  Authority[].
3  Identity[].
4  Credential[].
5  PropertySecurity[].
6  SecurityAttribut[].
7  SecurityFunction[].
8  SecurityModel[].
9  SecurityRequirement[].
```

4.1.5.2 Konzeptklasse *Asset*

Die Kategorie *Asset* bildet den Ausgangspunkt zur Beschreibung von Benutzerwerten. Benutzerwerte besitzen einen Schutzbedarf und erfordern angemessene Sicherheitsfunktionen. Die Zuordnung einer Komponente der Ontologie-*Cloud-Domain* zur Konzeptklasse *Asset* wird axiomatisch über die Methode *assignedObjectBy* mit dem Parameter *SecurityProperty* ausgedrückt. Die Qualifizierung zu einem *Asset* erfolgt über die Forderung zur Gewährleistung spezifischer Sicherheitseigenschaften (*SecurityProperty*). Die Definition von Werten für eine Konzeptklasse wird konzeptionell über die Zuweisung von Qualitätseigenschaften (*SecurityProperty*) ausgedrückt. Die Konzeptklasse wird im Abschnitt 4.1.5.3 eingeführt.
In Anlehnung an die Konzeptarbeiten von Gräuler et al. [GMT11] werden Instanzen der Ontologie-*Cloud-Domain* und der Ontologie-*Regulation* einem *Asset*-Konzept zugeordnet.

Listing 4.33: Methoden der Konzeptklasse *Asset* – Signature-F-atoms

```
1  Asset[assignedObjectBy(SecurityProperty) {0:*} *=> <nsd#Object>].
2  Asset[assignedSubjectBy(SecurityProperty) {0:*} *=> <nsd#Subject>].
3  Asset[assignedPolicyBy(SecurityProperty) {0:*} *=> <nsr#Policy>].
```

(Die Abkürzungen *nsd* und *nsr* in Listing 4.33 repräsentieren einen Namespace zur Verknüpfung mit der Ontologie-*Cloud-Domain* und der Ontologie-*Regulation*.)
Instanzen der Konzeptklasse *Policy* werden aus Cloud-Anwendersicht einer eigenen *Asset*-Kategorie zugeordnet. Diese Zuordnung gewährleistet die Policy-Sicherheit in Bezug auf eine korrekte und verbindliche Umsetzung. Die Konzeptualisierung von Security berücksichtigt damit den Aspekt einer abgesicherten Regulierung.

4.1.5.3 Konzeptklasse *PropertySecurity*

Die Subklasse wird als Teil der Konzeptklasse *QualityProperty* in Listing 4.49 definiert. Das Konzept von Sicherheitszielen orientiert sich an Schutzzielen der IT-Sicherheit [BA10].

Die im Konzept berücksichtigten Sicherheitseigenschaften beziehen sich auf die Schutzziele Vertraulichkeit (*confidential*), Authentizität (*authentic*), Unversehrtheit oder Integrität (*integrity-protected*) und Zurechenbarkeit (*accountable*).

Listing 4.34: Methoden der Konzeptklasse *PropertySecurity* – Subclass-F-atoms

```
1        accountable :: PropertySecurity .
2        authentic :: PropertySecurity .
3        confidential :: PropertySecurity .
4        integrity-protected :: PropertySecurity .
5        accountable :: PropertySecurity .
```

Mit der Eigenschaft *accountable* wird auch das Sicherheitsziel *non-repudiation* verfolgt. Es beschreibt eine Eigenschaft, bei der Entitäten daran gehindert werden, die Mitwirkung an ausgeführten Transaktionen abzustreiten.

4.1.5.4 Konzeptklasse *SecurityFunction*

Die Konzeptualisierung von Sicherheitsfunktionen repräsentiert spezifische Funktionen, die geeignet sind, um definierte Sicherheitsziele zu erfüllen. Zusätzlich besteht eine Beziehung zur Konzeptklasse *SecurityRequirement*, um den Sicherheits-Anforderungsbereich zu definieren.
Das Vorgehen schafft den Rahmen, um über entsprechende ontologische Ableitungskonzepte eine konkrete Sicherheits-Policy mit bestehenden Sicherheitsanforderungen abgleichen zu können.

Listing 4.35: Methoden der Konzeptklasse *SecurityFunction* – Signature-F-atoms

```
1    SecurityFunction[satisfySecRequirements {0:*} *=> SecurityRequirement].
2    SecurityFunction[functionName {1:*} *=> _string].
3    SecurityFunction[basedOnStandard {0:*} *=> SecurityStandard].
4    SecurityFunction[mitigate {1:*} *=> Threat].
```

Die Subkonzepte der Konzeptklasse *SecurityFunction* formalisieren den Aspekt von Anforderungen an Sicherheitsfunktionen.
In Listing 4.37 werden die Subkonzepte der Konzeptklasse *SecurityFunction* eingeführt.

Listing 4.36: Subklassen der Konzeptklasse *SecurityFunction* – Subclass-F-atoms

```
1            Privacy :: SecurityFunction .
2            DataProtectionFunction :: SecurityFunction .
3            CryptographicFunction :: SecurityFunction .
4            Communications :: SecurityFunction .
5            AuthenticationFunction :: SecurityFunction .
6            AuditFunction :: SecurityFunction .
```

Der Methodenentwurf (4) in Listing 4.35 beschreibt eine Beziehung zur Konzeptklasse *Threat* und dient der formalen Beschreibung einer Bedrohungsbetrachtung. Der Konzeptrahmen aus Abbildung 4.3 für die Beschreibung einer Sicherheitsstrategie steht damit vollständig zur Verfügung. Ontologisch geführte Ableitungskonzepte lassen Schlussfolgerungen über die Wirksamkeit einer bestehenden Sicherheitsstrategie zu.

4.1.5.5 Konzeptklasse *SecurityRequirement*

Dieser Teil der Konzeptualisierung formalisiert allgemein akzeptierte Anforderungen an die Sicherheitsfunktionen der *Common Criteria* [CC12c] und überträgt die darin beschriebenen Klassenkonzepte in eine logik-basierte Beschreibungsform. Die Anforderungen formulieren ein erwartetes Sicherheitsverhalten eines Sicherheitsbausteins (*Target of Evaluation*), um entsprechende Sicherheitsziele zu erfüllen.
Die Klasse *FIA: Identification and authentication* wurde bereits in die Konzeptualisierung mit aufgenommen. Das folgende Listing gibt einen Überblick über die Struktur der Klasse.

Listing 4.37: Subklassen der Konzeptklasse *SecurityRequirement* – Subclass-F-atoms

```
1  IdentificationAuthentication(FIA)::SecurityRequirement.
2
3  UserIdentification(UID)::IdentificationAuthentication(FIA).
4  UserAuthentication(UAU)::IdentificationAuthentication(FIA).
5  UserAttributeDefinition (ATD)::IdentificationAuthentication(FIA).
6  UserSubjectBinding(USB)::IdentificationAuthentication(FIA).
7  SpecificationOfSecrets(SOS)::IdentificationAuthentication(FIA).
8  AuthenticationFailures(AFL)::IdentificationAuthentication(FIA).
```

4.1.5.6 Konzeptklasse *Identity*

Die Konzeptklasse *Identity* repräsentiert Entitäten innerhalb eines IT-Systems. Handelnde Akteure, d. h. Instanzen der Konzeptklasse *User* oder lauffähige Software-Artefakte, bilden Instanzen der Konzeptklasse *Identity*. Mit der Methode *propertyInf* in Listing 4.38 werden informelle Eigenschaften einer Identität mit ihren konkreten Werten vorgegeben.

Listing 4.38: Methoden der Konzeptklasse *Identity* – Signature-F-atoms

```
1  Identity[propertyInf(AttributeInformational) {0:*} *=> _string].
2  Identity[propertySec(AttributeSecurity) {0:*} *=> _string].
```

Zusätzlich beschreibt die Methode *propertySec* Sicherheitseigenschaften einer Identität und legt deren Werte fest. Sicherheitseigenschaften werden in Sicherheitsfunktionen berücksichtigt. Sie bestimmen z. B. die Eindeutigkeit einer Identität (Identifizierbarkeit) und Zugriffskontrollentscheidungen, die auf spezifische Identitätsmerkmale (z. B. Rollen-Identifikatoren) ausgerichtet sind.

Listing 4.39: Subklassen der Konzeptklasse *IdentityAttributes* – Subclass-F-atoms

```
1  AttributeSecurity::IdentityAttributes.
2  AttributeInformational::IdentityAttributes.
```

Die Konzeptklasse *IdentityAttributes* in Listing 4.39 legt das Vokabular für mögliche Identity-Eigenschaften fest. Die Subklassen differenzieren nach informellen (*AttributeInformational*) und sicherheitsrelevanten Eigenschaften (*AttributeSecurity*).

4.1.5.7 Konzeptklasse *Credential*

Die Konzeptklasse *Credential* bildet das Kernkonzept für die sichere Zuordnung von Entitäten zu Instanzen der Konzeptklasse *Identity*. Instanzen der Konzeptklasse beschreiben ausgewählte Eigenschaften

von Entitäten, die überprüft, vertrauenswürdig zugesichert und einer Identität fest zugeordnet werden. Das vollständige Konzept für eine vertrauenswürdige Entität wird im Abschnitt 4.2 geführt. Listing 4.40 beschreibt die Spezifikation der Konzeptklasse *Credential*.

Listing 4.40: Methoden der Konzeptklasse *Credential* – Signature-F-atoms

```
1  Credential[verifiedIdentity {1:1} *=> Identity].
2  Credential[verificationMethod {1:*} *=> SecurityFunction].
3  Credential[verifiedPropertyInf(AttributeInformational) {0:*} *=> _string].
4  Credential[verifiedPropertySec(AttributeSecurity) {0:*} *=> _string].
5  Credential[revocationState {1:1} *=> _boolean].
6  Credential[issuedBy {1:1} *=> Authority].
7  Credential[expiredValidity {0:1} *=> _date].
8  Credential[confirmedBy {0:*} *=> <nsd#CloudRoles>].
```

Die Methoden *issuedBy* und *confirmedBy* beschreiben Beziehungen zu Instanzen, die Eigenschaften von Identitäten prüfen und bestätigen. Die damit verbundenen Konzeptklassen *Authority* (z. B. zertifizierte Organisationen) und <*nsd#CloudRoles*> (z. B. Produkthersteller) unterscheiden sich qualitativ in Bezug auf die Prozesse zur Überprüfung und Bestätigung dieser Eigenschaften. Die Qualitätsunterschiede führen zur Ableitung unterschiedlicher Niveaus an Vertrauenswürdigkeit von Entitäten.
Die Methoden *verifiedPropertyInf* und *verifiedPropertySec* beschreiben zugesicherte Eigenschaften einer Identität, *expiredValidity* und *revocationState* legen den Gültigkeitszeitraum fest bzw. setzen kontextbezogen einen Sperrzustand für die zugeordnete Identität. Die Überprüfung von Sicherheitseigenschaften erfordert die Anwendung von Sicherheitsfunktionen und wird mit der Methode *verificationMethod* festgelegt. Die Methode *verifiedIdentity* referenziert auf die bestätigte Identität.
Die Subkonzepte von *Credential* spezialisieren den Aspekt durch qualitativ differenzierte Bindungsarten zwischen Benutzer-Entitäten und Instanzen der Konzeptklasse *Identity*. Listing 4.41 beschreibt mögliche Subkonzepte der Konzeptklasse *Credential*.

Listing 4.41: Credential – Subclass-F-atoms

```
1  Password::Credential.
2  TokenSAML::Credential.
3  TicketPCR::Credential.
4  TicketHash::Credential.
5  Certificate::Credential.
```

Password-Credentials bilden die einfachste Bindungsform und beinhalten als Sicherheitseigenschaft das *Password*. SAML-Token basieren auf ein XML-Format (Security Assertion Markup Language) und enthalten standardisierte SAML-Assertions zur vertrauenswürdigen Bestätigung von Authentifizierungs- und Autorisierungsinformationen [Can+05a]. Der Standard beschreibt Funktionen für die sichere Protokollbindung [Can+05b] und bietet Mechanismen für die sichere Authentifizierbarkeit der im SAML-Protokoll involvierten Parteien.
Hash-Tickets (*TicketHash*) enthalten kryptographische Identifizierungsmerkmale von Softwarekomponenten, die über Hash-Funktionen als digitaler Fingerabdruck berechnet werden. Eine Hash-Funktion ist eine nicht injektive Abbildung, die jedes Objekt des Universums auf eine Hashadresse abbildet [Eck13].
PCR-Tickets (*TicketPCR*) beschreiben, ähnlich wie Hash-Tickets, kryptographische Identifizierungsmerkmale von Hardware-Komponenten einer infrastrukturellen Plattform. Das Prinzip von PCRs (Plat-

form Configuration Register) als kryptographischer Fingerabdruck zur Repräsentation plattformspezifischer Zustandseigenschaften wird in Anhang II beschrieben. Die Zusicherung erfolgt über eine TCB (*Trusted Computing Base*).
Die Anwendung von X.509-Zertifikaten vollzieht eine standardisierte Zusicherung [Coo+08] und kryptographische Bindung eines öffentlichen asymmetrischen Schlüssels an eine Entität. Die Zusicherung erfolgt über das Konzept einer *Authority* (siehe Abschnitt 4.2.2).

4.1.5.8 Konzeptklasse SecurityModel (Sicherheitsmodell)

Ein Sicherheitsschwerpunkt in Cloud-Systemen bildet die Anwendung von Sicherheitspolicies zur Festlegung von Zugriffsbedingungen unter Laufzeitbedingungen. Der Cloud-Anwender benötigt eine Zusicherung, dass die Ausnutzung von Schwachstellen in Anwendungen anderer Cloud-Anwender oder Operationen des Cloud-Anbieters zu keiner Beeinflussung seines vertraglich abgesicherten Nutzungsbereiches führt.
Die Fähigkeiten zur Servicenutzung müssen durch eine Sicherheitspolicy auf definierte Entitäten und Funktionen begrenzt werden. Während der Laufzeit repräsentiert die eingestellte Sicherheitspolicy eine *Invariante* bezogen auf einen zeitlich fixierten Sicherheitsstatus. Der Cloud-Anwender benötigt ein Instrument, um mit eigener Verantwortlichkeit eine Sicherheitspolicy umsetzen zu können.
Eine Lösung für dieses Problem ist die regulative Einbindung von Sicherheitsmodellen. Sicherheitsmodelle stellen eine formale Repräsentation einer Zugriffspolitik dar [Ott07]. Die regulative Anwendung von Sicherheitsmodellen ist Teil der Security-Konzeptualisierung und schafft die Voraussetzung einer vertraglich vereinbarten Praxis verteilter Verantwortlichkeiten zwischen Cloud-Anwender und Cloud-Anbieter.
Die Anwendung eines Mandatory-Access-Control(MAC)-Modells vermeidet die Unzulänglichkeiten eines zur Laufzeitzeit durch Akteure gestaltbaren Zugriffsmodells (Discretionary Access Control) von Standard-UNIX-Betriebssystemen. Die Einbeziehung von MAC-basierten Sicherheitsmodellen sichert während der Systemlaufzeit die Integrität und Vertraulichkeitsziele vertrauenswürdiger Systemkomponenten. Der Cloud-Anwender besitzt die eigene Hoheit zur Anpassung einer Sicherheitspolicy an konkrete Nutzungsbedingungen.
Aktuelle Forschungsinitiativen untersuchen und entwickeln Methoden, um die Integrität und Vertraulichkeit definierter *Assets* mit hardwareunterstützten Technologien zur Laufzeit sicherzustellen [Pro16]. Für die Einbeziehung solcher Konzepte sind perspektivisch MAC-basierte Sicherheitsmodelle zu erweitern und können deren Wirksamkeit erhöhen.

4.2 Konzept zur Herausbildung von Vertrauen (Trust)

Für den Aspekt Vertrauen wurde im Abschnitt 4.1 keine eigenständige Konzeptklasse eingeführt. Vertrauen entwickelt sich als Resultat einer Bewertung vertrauensbildender Fakten, die selbst durch eine spezifische Methodik eingeräumt werden bzw. sich zweifelsfrei bestimmen lassen. Die Methodik benötigt technische Akteure, die nachfolgend als Konzept einer vertrauenswürdigen Entität eine Spezifikationsgrundlage erhalten. Eine vertrauenswürdige Entität ist als Architekturbaustein in Abbildung 4.1 zur Steuerung von Vertrauen aufgenommen. Das Grundpattern integriert sich im Zuge eines verteilten Cloud-Architekturdesigns zu einem Verbund vertrauenswürdiger Entitäten.

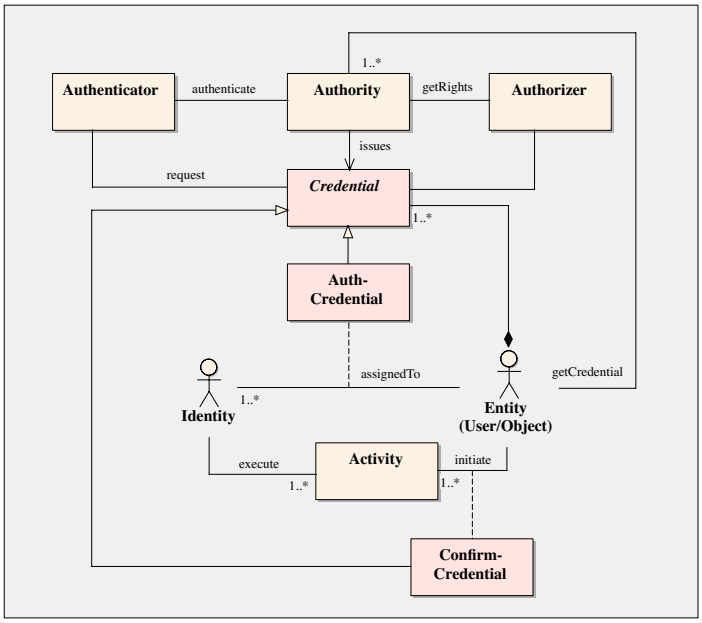

Abbildung 4.4: Trustworthy-Entity Pattern

Abgeleitet aus den Pattern *Circle of Trust*, *Credential* und *Remote Authenticator/Authorizer* [Fer13], stellt das Modell in Abbildung 4.4 die Haupteigenschaften einer vertrauenswürdigen Entität heraus.

4.2.1 Konzept einer vertrauenswürdigen Entität

Ausgehend von einem allgemeinen Begriff, wird nachfolgend eine Konzeptualisierung *Vertrauenswürdige Entität* entwickelt. Es werden folgende Eigenschaften definiert:

Definition 4.1 *Entität*
Eine Entität beschreibt Objekte (z. B. natürliche Personen, programmierbare Agenten (Programm-Code), Komponenten von Computersystemen), die in der Lage sind, elektronische Prozesse auszuführen bzw. mit ihnen zu interagieren. Jede Entität ist durch charakteristische Merkmale identifizierbar.

Die Definition 4.1 wird im Abschnitt 4.1.4 durch die Konzeptklasse *Entity* formal ausgedrückt.

Definition 4.2 *Policy-Entitäten für vertrauensbildende Aufgaben (TE-P)*
Policy-Entitäten bilden eine Subklasse von Entität, die für die Durchsetzung vertrauenswürdiger Bedingungen in Bereichen von IT-Architekturen hergestellt, evaluiert und spezialisiert eingesetzt werden.

Instanzen der Konzeptklasse *Authority* kennzeichnen die Vertrauenswürdigkeit von Policy-Entitäten (siehe Abschnitt 4.2.2).

Definition 4.3 *Domänen-Entitäten für domänenspezifische Serviceaufgaben (TE-D)*
Domänen-Entitäten bilden eine Subklasse von Entität, die für die Ausführung elektronischer, serviceorientierter Dienstleistungen hergestellt, evaluiert und spezialisiert eingesetzt werden.

Instanzen dieser Konzeptklasse stellen dem Cloud-Anwender vertraglich zugesicherte Dienstleistungen in regulierten Architekturbereichen bereit. Instanzen der Konzeptklasse *Authority* können die Vertrauenswürdigkeit von Domänen-Entitäten kennzeichnen.

Das Vertrauen in Entitäten ist mit den Verfahren zur sicheren Bestimmung ihrer elektronischen Identitäten verbunden. Das Maß an Zusicherung darüber, dass ein Verfahren die elektronische Identität einer Entität sicher und korrekt überprüfen kann, bestimmt das Vertrauensniveau der ihr zugeordneten Entität.

Definition 4.4 *Sichere Bestimmung einer Identität*
It is evident that trust in identity is the gate to all factors of trust-management systems. The establishment of an identity as result of a successful authentication process remains a valid fact that is associated with the identity and generally persists throughout a session [Ben06].

Vertrauen entwickelt sich durch den Nachweis darüber, dass Akteure und ausführbare Komponenten eines IT-Systems erwartete Sicherheitseigenschaften ausweisen. Darüber hinaus können die Ergebnisse ihrer ausgeführten Handlungsschritte und Aktionen als ein erwartetes Verhalten im Rahmen einer definierten Sicherheits-Policy nachgewiesen werden.

Definition 4.5 *Erwartetes Verhalten*
Trust is founded on notion of confining expected behavior. The expected behavior is that users and all programming entities remain in line with the security properties and policies adopted by the system [Ben06].

Definition 4.6 *Zurechenbarkeit von Handlungsschritten*
Zurechenbarkeit bezeichnet einen Umstand, dass Aktionen oder Dokumente den urhebenden Personen oder Institutionen (Veranlassern) zugeordnet werden können. Zurechenbarkeit wird als Nachweisbarkeit (detectability), Unleugbarkeit oder Nicht-Abstreitbarkeit (non- repudiation) bezeichnet [BA10].

Die Vertrauenswürdigkeit einer Entität (Konzeptklasse *Entity*), Ausdruck und Repräsentant eines handelnden Akteurs, entwickelt sich über eine überprüfbare und nicht-trennbare Bindung an eine ihr zugeordnete elektronische Identität (Konzeptklasse *Identity*). Als zusicherndes und damit vertrauensbildendes Element bestimmt die Konzeptklasse *Credential* aus Abschnitt 4.1.5.7 den Grad der Bindung zwischen den Konzeptklassen. Vertrauenswürdigkeit entsteht durch zugesicherte Identitätseigenschaften (*Auth-Credential*), während sich der Grad an Vertrauen über Handlungsmuster (*Confirm-Credential*) konstituiert, die vertrauenswürdige Ausführungswerte bereitstellen.

4.2.2 Konzept einer Authority

Ein hohes Maß an Zusicherung gemäß Definition 4.5 und Definition 4.6 wird unter Einbeziehung einer vertrauenswürdigen *Authority* erzielt. Die Konzeptklasse *Authority* steht für eine *Certificate Authority* als Bestandteil einer X.509-konformen Public-Key-Infrastruktur [Coo+08; Yee13].
Das vertrauensbildende Konzept besteht aus einem qualifizierten Prüfprozess zur Bescheinigung geforderter Eigenschaften einer Entität mit Verfahren der elektronischen Signatur [Gal13; PM]. Die Präsentation und Verteilung der Prüfergebnisse erfolgt auf Grundlage von X.509-Zertifikaten, die von einer *Certificate Authority* autorisiert herausgegeben werden und von dieser elektronisch signiert sind. In Bezug auf die Authentizität und Gültigkeit des Zertifikats erlaubt die X.509-Zertifikatsstruktur jederzeit eine kryptographische Überprüfung.
Der Ausgangspunkt einer Vertrauenskette (*Chain of Trust*) gründet sich auf die kryptographische Erzeugung eines asymmetrischen Schlüsselpaares [RSA12; RSA98] und bildet den Vertrauensanker (*Root*

of Trust). Das Vertrauensniveau einer *Certificate Authority* wird durch qualifizierte Konzepte und Maßnahmen bestimmt, um den privaten Schlüssel (*Private Key*) geheim zu halten. Für eine Verteilung der Verantwortlichkeit zur Herausgabe von X.509-Zertifikaten werden Hierarchien als Verbund von *Certificate Authority* gebildet. Die als *Root-Authority* bezeichnete oberste Hierarchiestufe dient allein der sicheren Verwaltung der kryptographischen Artefakte als Vertrauensanker.
Die Konzeption verfolgt das in der Abbildung 4.5 dargestellte Hierarchiekonzept, um Vertrauenskonzepte des Cloud-Anbieters und Cloud-Anwenders bezogen auf ihre Eigenständigkeit zu beschreiben. In einem weiteren Schritt ist zu zeigen, wie ein Vertrauensverhältnis zwischen Cloud-Anwender und Cloud-Anbieter gestaltet wird.

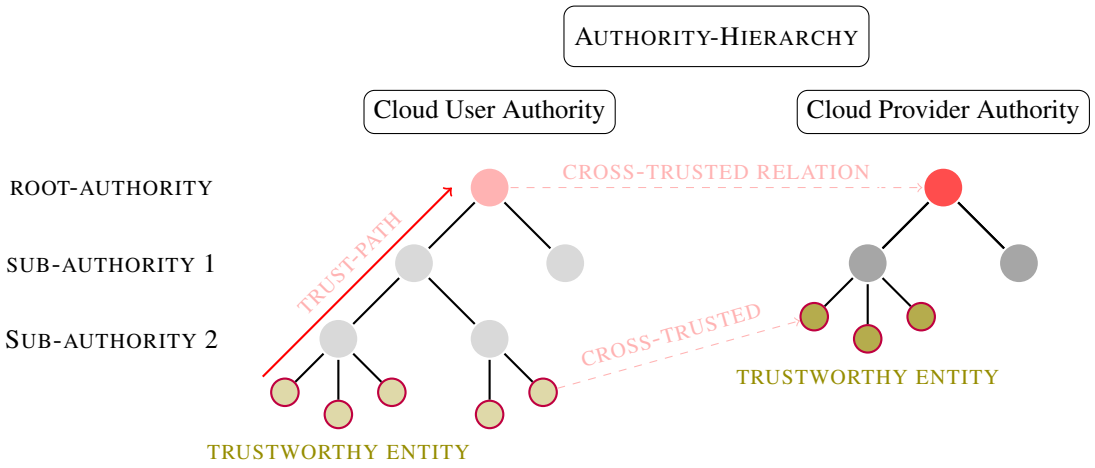

Abbildung 4.5: Hierarchien von Authorities

Die strukturelle Verteilung von Zuständigkeiten für das Trust-Management von Entitäten ist eine Entscheidung der entsprechenden Organisation. Für die weitere Beschreibung wird davon ausgegangen, dass auf Ebene der *Root-Authority* keine Verwaltung von Entitäten vorgenommen wird. Zur Übersichtlichkeit verwalten *Sub-Authorities* nur eine Klasse von Entitäten.
Das Konzept einer *Authority* wird ontologisch mit Listing 4.42 formalisiert beschrieben. Der Modellansatz ermöglicht eine Analyse bestehender Vertrauensbeziehungen zwischen den Konzeptklassen *Entity* und *Authority*.

Listing 4.42: Konzepthierarchie *Authority* – Subclass-F-atoms

```
1 AuthorityConsumer :: Authority .
2 AuthorityProvider :: Authority .
3 AuthorityInspection :: Authority .
```

Die allgemein beschriebenen strukturellen Beziehungen werden über die Signaturbeschreibung in Listing 4.43 repräsentiert.

Listing 4.43: Methoden der Konzeptklasse *Authority* – Signature-F-atoms

```
1 Authority[nameAuthority {0:*} *=> _string].
2 Authority[assignedTrustLevel {1:1} *=> TrustlevelAuthority].
3 Authority[signingCAKey {1:1} *=> PrivateKey].
4 Authority[issuedCACertificate {1:1} *=> CertificateCA].
5 Authority[crossTrustedCA {0:*} *=> CertificateCrossCA].
```

Ein privater Signaturschlüssel (*Methode signingCAKey*) und ein öffentliches Authority-Zertifikat (*Methode issuedCACertificate*) sind im Besitz der Instanzen der Konzeptklasse *Authority*. Prozesse zur Überprüfung des Vertrauensstatus einer Entität verwenden das Authority-Zertifikat. Im Listing 4.43 wurde eine eigene Konzeptklasse *TrustlevelAuthority* als Teil einer *Trustlevel*-Spezifikation aufgenommen. Instanzen der Klasse beschreiben ein Maß an Vertrauenswürdigkeit, deren Wertzuweisung im Konzept logischer Schlussfolgerungen bei der Bewertung von Eigenschaften der Entitäten zur Anwendung kommt. Die Methode erlaubt die flexible Gestaltung verschiedener Vertrauensstufen.

4.2.2.1 Zusicherung von Entity-Eigenschaften

Die Konzeptklasse *Credential* wurde im Abschnitt 4.1.5.7 formal und unabhängig von konkreten Instanzen ihrer Klasse eingeführt. Mit der Methode *issuedBy* im Listing 4.40 vollzieht sich die Übertragung der Vertrauenswürdigkeit einer *Authority* auf eine konkrete Entität. Eine überprüfbare Beziehung zwischen beiden Konzeptklassen ist Ausdruck einer bestehenden Vertrauenskette (*Chain of Trust*) und erfüllt die mit Definition 4.4 gesetzten Anforderungen.
Die Zusicherung einer erwarteten Vertrauenswürdigkeit wird als notwendige Vorbedingung für die Anwendung einer Entität im Cloud-Anwendungskontext definiert. Die Zusicherung bezieht auf die im Abschnitt 4.1.4 eingeführten Subkonzeptklassen *User* und *Object* und wird ganzheitlich im Policyentwurf zur Steuerung der Vertrauenswürdigkeit im Abschnitt 4.2.3 berücksichtigt. Zur Repräsentation einer vertrauenswürdigen Entität wird in der Konzeptarbeit das Acronym *TE* (*Trustworthy Entity*) eingeführt und weiter verwendet.

4.2.2.2 Entitäten innerhalb einer *Authority*-Hierarchie

Das Konzept der Vertrauenshierarchie zwischen Instanzen der Konzeptklasse *Authority* soll weiter formalisiert werden. Entitäten ordnen sich in bestehende Hierarchien ein und bilden darüber hierarchische Vertrauensketten.
Die Sicherstellung einer Hierarchie (siehe Abbildung 4.5) gründet sich auf eine verteilte *Authority*-Architektur. Zu jedem Zeitpunkt muss gewährleistet werden, dass jede Instanz einer *Authority* und die ihr zugeordneten Entitäten auf eine *Root-Authority* zurückgeführt werden kann und in Besitz eines gültigen Signaturzertifikates (Methode *issuedCACertificate*) ist.

Listing 4.44: Methode *hasRootOfTrust* – Reasoning-Rule

```
1   @{RootOfTrustLevel1}
2   ?C[hasRootOfTrust -> ?R]
3   :-
4   ?C:Certificate                  AND
5   ?Y:Authority                    AND
6   ?R:Authority                    AND
7   ?C[issuedBy -> ?Y]              AND
8   ?Y[issuedCACertificate -> ?W]   AND
9   ?W[issuedBy -> ?R]              AND
10  ?R[issuedCACertificate -> ?S]   AND
11  ?S[issuedBy -> ?R].
```

Die Signaturbeschreibung der Konzeptklasse in Listing 4.43 stellt keine expliziten Methoden für die Gestaltung von Vertrauenshierarchien bereit. Über Abhängigkeiten zwischen den Instanzen der Konzeptklassen *CertificateCA* und *Authority* steht implizit das Wissen über eine existierende Hierarchie zur Verfügung. Zur Bestimmung der *Root-Authority* von jeder Hierarchiestufe wird dafür in Listing 4.44 eine *Reasoning-Rule* entwickelt.
Die Beziehung *hasRootOfTrust* der Konzeptklasse *Certificate* wird als Schlussfolgerung im Rule-Header definiert (2). Für die Menge der Instanzen der Konzeptklasse *Certificate* (4) werden die zugehörigen Instanzen der Konzeptklasse *Authority* (5) gesucht, die als Herausgeber (6) eingetragen sind. Das Zertifikat der herausgebenden Instanz (*issuedCACertificate*) wurde selbst von einer *Authority* bestätigt (7). Der Anfang der Vertrauenskette ist erreicht, wenn das Zertifikat der herausgebenden *Authority* (8) von sich selbst bestätigt wurde (9).

4.2.2.3 Entitäten und externe *Authority*

Eine Vertrauensbeziehung zwischen autark organisierten *Authority*-Vertrauenshierarchien wird in Listing 4.43 über die Methode *crossTrustedCA* definiert. In einem vorgelagerten Prozess zur Überprüfung der Vertrauensgrundlagen erfolgt eine wechselseitige oder einseitige Bestätigung der jeweiligen Partner-*Authority*. Die Ausstellung eines Cross-Zertifikates (*CertificateCrossCA*) verknüpft unabhängige Vertrauenshierarchien und delegiert Vertrauen auf die Seite der Partner-*Authority*.

Listing 4.45: Methode *hasRootOfCrossTrust* – Reasoning-Rule

```
1  @{RootOfCrossTrust}
2  ?C[hasRootOfCrossTrust -> ?A]
3  :-
4  ?Z:Certificate                    AND
5  ?K:CertificateCrossCA             AND
6  ?Z[hasRootOfTrust -> ?S]          AND
7  ?K[issuedBy -> ?S]                AND
8  ?K[verifiedIdentity -> ?I]        AND
9  ?I[securityAttribut(DistinguishedName) ->?A].
```

In Abbildung 4.5 wird die Vertrauenshierarchie der Cloud-Anwender-*Authority* mit der Hierarchie der Cloud-Anbieter-*Authority* über eine *Cross-Trusted Relation* verbunden. Zwischen beiden Hierarchien besteht eine Vertrauensbeziehung und für Entitäten beider Hierarchien entwickelt sich ein Vertrauensverhältnis (*Cross-Trusted*). Vertrauensbeziehungen von Entitäten verschiedener Hierarchien können über eine *Reasoning-Rule* in Listing 4.45 bestimmt werden.
Die in Listing 4.44 eingeführte *Reasoning-Rule* fließt als Basis-Rule (6) mit ein und wird um das Konzept einer *CertificateCrossCA* erweitert. Die *Reasoning-Rule* erfüllt die in der Konzeptklasse *Certificate* definierte Reasoning-Methode *reasonRootOfCrossTrust*.

4.2.3 Konzept einer Policy zur Entwicklung von Vertrauen

Unter Anwendung der *Authority*-Konzeption wird ein Prozess zur vertrauenswürdigen Umsetzung von Policies entwickelt. Jede Form einer Policyausführung benötigt als notwendige Bedingung eine qualifizierte Vertrauensbasis. Vertrauen entwickelt sich über den Nachweis darüber, dass gesetzte Policies

gemäß den Anforderungen korrekt umgesetzt und die Regulierungsziele durch eine konforme Systemausführung (*Verhalten*) erfüllt werden.
Die Ausführung erfolgt durch vertrauenswürdige Entitäten mit Eigenschaften, wie sie im Abschnitt 4.2.1 definiert wurden. Die Kernprinzipien für die vertrauenswürdige Umsetzung von Policies werden durch den in Abbildung 4.6 dargestellten Prozess beschrieben:

1. Eine Instanz der Konzeptklasse *Authority* bestätigt eine Policy-Entität (Entität-P) auf Basis von Evaluierungsprozessen (*evaluateEntity*) als Vertrauensbasis (*assureTrustState*) und überführt die Entität in eine vertrauenswürdige Entität (*TE-P*).

2. Nach einer Autorisierung (*activateTE-P*) steuert die vertrauenswürdige Entität den Prozess (*Policy Loop*) zur Ausführung bereitgestellter Domain-Policies (*Policy-D*).

3. Entitäten der Domäne (*Entity-D*), die als Gegenstand einer Policy verwendet werden, müssen in einem Prüfprozess (*Trust Loop*) Sicherheitseigenschaften nachweisen (*checkQualityProperty*), die den Status einer vertrauenswürdigen Entität repräsentieren.

4. Qualitätseigenschaften werden auf Grundlage separater Policies (*Policy-T*) überprüft. Der Prüfprozess bewertet Sicherheits- und Zuverlässigkeitseigenschaften (*apply-RulePT).*

5. Policies vom Typ *Policy-T* sind mit Entitäten verknüpft (*linkedPolicy-T*), deren Qualitätseigenschaften einer Regulierung unterliegen.

6. Die Vertrauensbasis wird bestätigt (*assureTrustState*) und überführt die Entität in eine vertrauenswürdige Entität der Domäne (*TE-D*).

7. Unter den gesetzten Bedingungen wird eine qualifizierte Entität in den Prozess aufgenommen (*activateTE-D*) und der Prozess zur Ausführung der Domain-Policy (*apply-RulePD*) wird fortgesetzt.

4.2.3.1 Spezialisierung der Trust-Policy

Die spezifische Regulierung vertrauenswürdiger Bedingungen führt zu einer Erweiterung der im Abschnitt 4.1.3.5 eingeführten Policy-Konzeptualisierung. Das Subklassen-Konzept in Listing 4.10 berücksichtigt bereits eine *Trust-Policy*, jedoch finden in den vorliegenden Rule-Konzeptklassen die Prinzipien zur Zusicherung von Entity-Eigenschaften über die Konzeptklassen *Credential* und *Authority* noch keine Berücksichtigung.

Listing 4.46: Subklassen der Konzeptklasse *Rule* – Subclass-F-atoms

```
1  RuleTrust::Rule.
2  RuleSecurity::Rule.
3  RulePrivacy::Rule.
4  RuleDeployment::Rule.
```

Die Subklassen-Spezifikation in Listing 4.46 erweitert die Methoden der Konzeptklasse *Rule*, als Instrument für eine sprachliche Spezialisierung von Policy-Klassen. Die Subklasse *RuleTrust* beschreibt die Methode *assuredBy* zur Anwendung der *Credential*-Konzeptklasse.

Listing 4.47: Methoden der Konzeptklasse *RuleTrust* – Signature-F-atoms

```
1   RuleTrust[assuredBy {0:1} *=> <nss#Credential>].
```

Die Methode in Listing 4.47 vollzieht eine vertrauenswürdige Abbildung von Eigenschaften der Konzeptklasse *Entity* auf Eigenschaften der Konzeptklasse *Identity* und repräsentiert eine regulierte Zusicherung von Entity-Eigenschaften der Ontologie-*Cloud-Domain*. Die zugesicherte Verknüpfung zwischen beiden Konzeptklassen manifestiert sich über die Methode *authorizedTo* in Listing 4.20.
Die Vorgehensweise schafft gleichzeitig den Ausgangspunkt für Prozesse einer regulierten Generierung von elektronischen Identitäten. Für hohe Anforderungen an die Vertrauenswürdigkeit muss innerhalb eines definierten Bereiches der Ontologie-*Cloud-Domain* jede Existenz einer Instanz der Konzeptklasse *Identität* durch eine Trust-Policy erklärt werden können.

Listing 4.48: Methode *reasonBasedTrustPolicy* – Reasoning-Rule

```
1   @{RootOfTrustPolicy}
2   ?I[reasonBasedTrustPolicy -> ?P]
3   :-
4   ?P:Policy                                                    AND
5   ?P[hasTrustRule -> ?R]                                       AND
6   ?R[assuredBy -> ?<http://www.trustedcloud.org/security#C>]   AND
7   ?C[verifiedIdentity -> ?<http://www.trustedcloud.org/security#I>] .
```

Für die Überprüfung der Vertrauenswürdigkeit einer Identität ist die in Listing 4.48 spezifizierte *Reasoning-Rule* anzuwenden. Die Methode in Anweisung (6) stellt die logische Verknüpfung zwischen einer Identität (I) und der ihr zugrundeliegenden Trust-Policy (P) her.

4.2.3.2 *QualityProperty* – Gegenstand der Vertrauenspolitik

Bestandteil der Konzeptklasse *Entity* in Listing 4.20 ist die Methode *checkQualityProperty*, um nichtfunktionale Eigenschaften für eine Entität zu kennzeichnen.
Mit der Konzeptklasse *QualityProperty* in Listing 4.51 erfolgt eine Erweiterung der Ontologie-*Security*, um neben Sicherheitszielen zusätzliche qualitätsorientierte (nicht-funktionale) Eigenschaften von Entitäten zu spezifizieren. Beispiele sind Zuverlässigkeit, Verfügbarkeit oder Privatheit.

Listing 4.49: Subklassen der Konzeptklasse *QualityProperty* – Subclass-F-atoms

```
1   PropertyAvailability::QualityProperty.
2   PropertyLiability::QualityProperty.
3   PropertyPrivacy::QualityProperty.
4   PropertySecurity::QualityProperty.
```

Im Entwurf der Subklassen werden Ziele der Verfügbarkeit, Zuverlässigkeit, der Daten- und IT-Sicherheit berücksichtigt. Konkrete Sicherheitsziele werden im Abschnitt 4.1.5.3 definiert.
Aus dem Bereich des Datenschutzes lassen sich Anforderungen in Bezug auf nachweisgeführte Prozessaktivitäten (*provable*) und zugelassene Operationen für den Datenaustausch (*moveable*) beschreiben. Für Anforderungen an die Lokation von Datenspeicherungsprozessen wird die Eigenschaft *locatable* eingeführt.

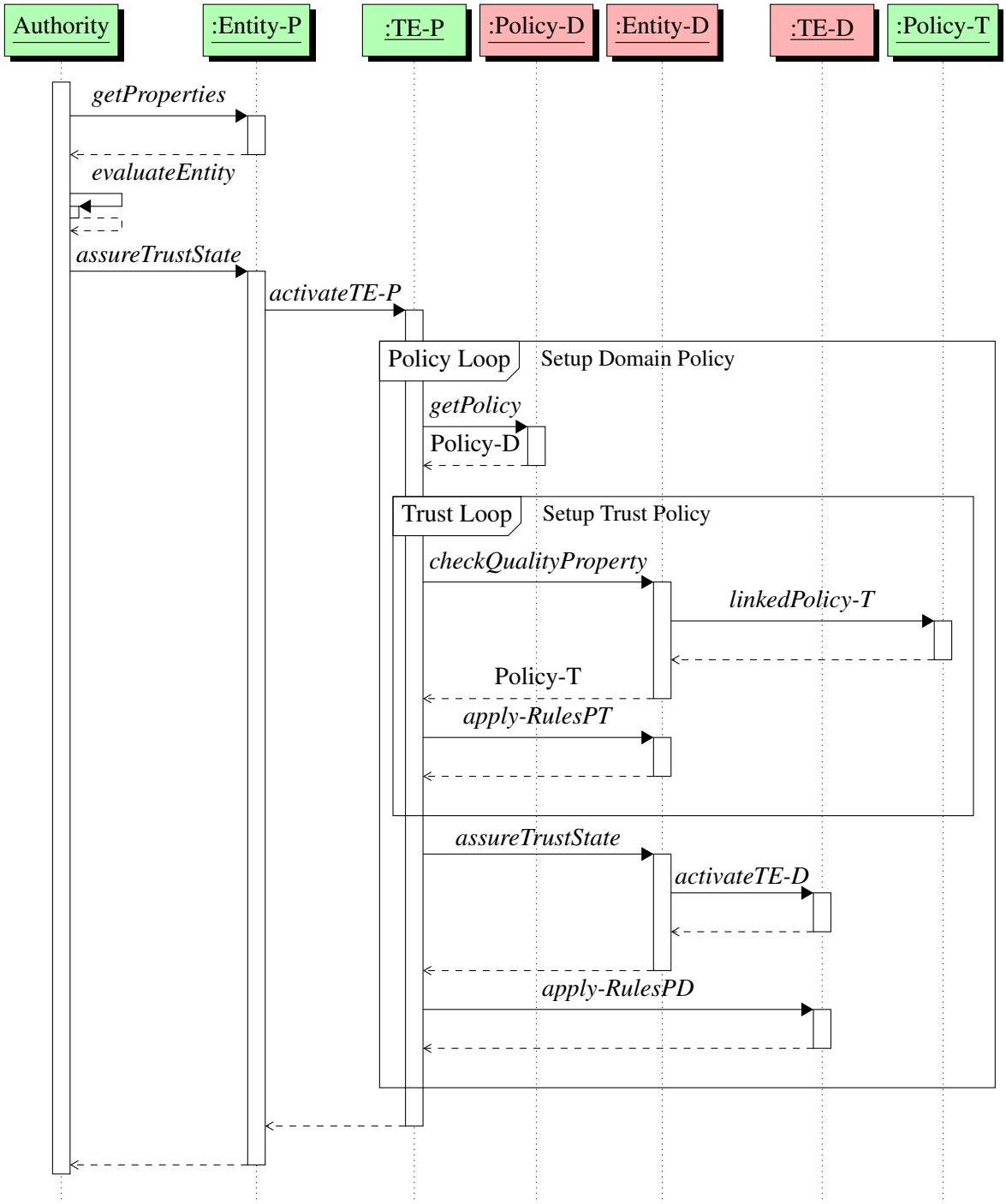

Abbildung 4.6: Policy-Konzept zur Entwicklung vertrauensbildender Architekturgrundlagen

Listing 4.50: Methoden der Konzeptklasse *PropertyPrivacy* – Subclass-F-atoms

```
1 provable::PropertyPrivacy.
2 moveable::PropertyPrivacy.
3 locatable::PropertyPrivacy.
```

Die Umsetzung von Qualitätseigenschaften wird auf die Prinzipien der Policy-Konzeption im Abschnitt 4.1.3 zurückgeführt. Die Methode *enforcedThrough* in Listing 4.51 definiert die Beziehung zu der Konzeptklasse *Policy*.

Listing 4.51: Methoden der Konzeptklasse *QualityProperty* – Signature-F-atoms

```
1  QualityProperty[enforcedThrough {1:*} *=> <nsr#Policy>].
2  QualityProperty[reasonMitigate {1:*} *=> Threat].
```

Die zugeordnete Policy entscheidet über Auswahl und Umfang der benötigten Sicherheitsfunktionen. Die Wirksamkeit eingestellter Sicherheitseigenschaften in Bezug auf eine bestehende Bedrohungssituation kann bereits an dieser Stelle mit der Reasoning-Methode *reasonMitigate* berechnet werden.
Ausgehend von einer Instanz der Konzeptklasse *Rule* (siehe Listing 4.7) wird als Vorbedingung die Überprüfung von Qualitätseigenschaften gesetzt. Die in Listing 4.52 entwickelte *Reasoning-Rule* überprüft bestehende Qualitätsanforderungen. Die Konzeptklasse *Rule* wird dafür um die Reasoning-Methode *reasonQualifiedBy* erweitert und dynamisch ausgeführt.

Listing 4.52: Methode *getQualityProperty* – Reasoning-Rule

```
1  @{getQualityProperty}
2  ?X[reasonQualifiedBy -> ?P]
3  :-
4  ?X:Rule                                    AND
5  ?X[do(?A) -> ?O]                           AND
6  ?O[<http://www.trustedcloud.org/clouddomain#hasQualityProperty>->?P].
```

4.3 Trust-Establishment-Protokoll

Für die Etablierung von Vertrauen wird eine technische Strategie benötigt, um ein Cloud-Anwender-definiertes Vertrauenskonzept auf IT-Bereiche des Cloud-Anbieters zu übertragen. In Anlehnung an [Jür05] erfolgt eine weitergehende Formalisierung des Systembegriffs (siehe Abschnitt 4.1.4), um darauf aufbauend allgemeine Interaktionen mit der Außenwelt abzubilden.

4.3.1 Datenmodell

Die folgende Formalisierung beschreibt Daten, Ereignisse und Nachrichten (Messages). Grundlage bildet die Unified Modeling Language (UML) [Obj15].

4.3.1.1 Verhaltensorientierte Artefakte

Definition 4.7 *Ereignis und Auslöser*
Ein Ereignis (Event) spezifiziert das Auftreten einer Erscheinung, die bzgl. Ort und Zeit messbar ist. Ein Auslöser (trigger) referenziert genau ein Ereignis und stellt die Beziehung zu einem Verhalten her [Wei14, S. 276].

Definition 4.8 *Verhalten*
Bezogen auf einen definierten Zeitbereich wird Verhalten als eine Menge von Aktivitäten beschrieben, die zu einer Änderung des Zustands eines Objekts führen.

Definition 4.9 *Messages*
In einem UML-Modell kommunizieren Komponenten oder Objekte über Nachrichten, nachfolgend als Messages (msg) bezeichnet. Jede eingehende Nachricht wird als Ereignis definiert, nachfolgend als

$Event$ bezeichnet und leitet sich aus einer Menge $\boldsymbol{Events}$ ab. Es gilt $msg \in \boldsymbol{Events}$. $Events$ werden durch einen Message-Namen aus einer Menge $\boldsymbol{MsgNames}$ und optionalen Argumenten repräsentiert. Argumente sind Elemente aus einer Menge von Ausdrücken $\boldsymbol{Exp}$. Messagenamen können über einen Präfix-Namen für Objekt- oder Komponenteninstanzen aus einer Menge $\boldsymbol{OKNames}$ referenzieren.
Objekt- oder Komponenteninstanzen erhalten Messages über eine Eingangswarteschlange $\boldsymbol{inQu_O}$, nachfolgend als Input-Queue bezeichnet, und versenden Messages über eine Ausgangswarteschlange $\boldsymbol{outQu_O}$, nachfolgend als Output-Queue bezeichnet.

Definition 4.10 *Daten*
Wir definieren $\boldsymbol{VAR}$ als Menge von Variablen und $\boldsymbol{DATA}$ als Menge verwendeter Datenwerte. Die Mengen $\boldsymbol{MsgNames}$ und $\boldsymbol{OKNames}$ sind zueinander disjunkt und bilden eine Teilmenge von $\boldsymbol{DATA}$. Es gilt $\boldsymbol{MsgNames} \cup \boldsymbol{OKNames} \subseteq \boldsymbol{DATA}$. Temporäre Daten (nonces) oder andere Datenartefakte, die für die Ausführung kryptographischer Protokolle erforderlich sind, bilden Elemente von $\boldsymbol{DATA}$.

Es entsteht eine mathematische Grundlage für eine Sicherheitsmodellierung in Bezug auf die Datensicherheit in einem System und zwischen verschiedenen Systemen. Das Modell erlaubt eine präzise Beschreibung von Interaktionen mit einem System.

4.3.1.2 Kryptographische Artefakte

Definition 4.11 *Keys*
Wir definieren $\boldsymbol{Keys}$ als Menge kryptographischer Schlüssel $k \in Keys$ und eine injektive Abbildung $()^{-1} : Keys \rightarrow Keys$. Für jeden öffentlichen Schlüssel $k_p \in$ Keys, nachfolgend als Public Key bezeichnet, existiert ein geheimer Schlüssel $k_s = (k_p)^{-1}$ mit $k_s \in K$, nachfolgend als Secret Key bezeichnet.
Public Keys $k_p \in Keys$ werden für die Verschlüsselung von Daten bzw. für die Verifizierung von signierten Daten eingesetzt. Secret Keys $k_s \in Keys$ werden zur Entschlüsselung von Daten oder zur Erstellung digitaler Signaturen verwendet.
Asymmetrische Schlüssel $k \in Keys$ werden entweder zur Verschlüsselung oder zur Entschlüsselung eingesetzt. Symmetrische Schlüssel $k \in Keys$ werden sowohl für die Verschlüsselung als auch für die Entschlüsselung eingesetzt und es gilt $k = k^{-1}$.

Definition 4.12 *Kryptographische Ausdrücke*
Wir definieren eine Algebra kryptographischer Ausdrücke $\boldsymbol{Exp}$ als Quotient einer Termalgebra, gebildet aus den Mengen $\boldsymbol{VAR} \cup \boldsymbol{Keys} \cup \boldsymbol{DATA}$ mit folgenden Operationen:

$\_::\_$	*concatenation*
$\{\_\}\_$	*encryption*
$\mathcal{D}ec\_(\_)$	*decryption*
$\mathcal{S}ign\_(\_)$	*signing*
$\mathcal{E}xt\_(\_)$	*extracting from signature*
$\mathcal{H}ash(\_)$	*hashing*

Ausgehend von den kryptographischen Ausdrücken lassen sich folgende Beziehungen beschreiben:

- $\mathcal{D}ec_{K^{-1}}(\{E\}_K)$ = E (für alle E ∈ $\boldsymbol{Exp}$ und K ∈ $\boldsymbol{Keys}$)
- $\mathcal{E}xt_K(\mathcal{S}ign_{K^{-1}}(\mathrm{E}))$ = E (für alle E ∈ $\boldsymbol{Exp}$ und K ∈ $\boldsymbol{Keys}$)

4.3.1.3 Protokollspezifische Artefakte

Definition 4.13 $Auth_{Prov}$
Als Instanzen der Konzeptklasse Authority zertifizieren sie kryptographische Credentials des Cloud-Anbieters als Basis für eine Vertrauenspolitik.

Definition 4.14 $Auth_{Cons}$
Als Instanzen der Konzeptklasse Authority zertifizieren sie kryptographische Credentials des Cloud-Anwenders als Basis für eine Vertrauenspolitik.

Definition 4.15 *Cloud-Plattform –* P_i
Als Instanzen der Konzeptklasse Part beschreiben sie Bestandteile einer infrastrukturellen Plattform eines verfügbaren Cloud-Systems.

Definition 4.16 *Vertrauenswürdige Entität –* TE_{Con}, TE_{ConCI}
Von einem Cloud-Anwender nach Definition 4.2 bereitgestellte Policy-Entitäten und dienen zur Herstellung einer Vertrauensbasis.

Definition 4.17 *Vertrauenswürdige Entität –* TE_{Prov}
Von einem Cloud-Anbieter nach Definition 4.2 bereitgestellte Policy-Entitäten und dienen zur Herstellung einer Vertrauensbasis.

Definition 4.18 *Trust-Policy für Cloud-Plattform – ITrPolicy*
Policy zur Beschreibung geforderter Eigenschaften der verwendeten Hardwarebausteine und von Anforderungen an die Komponenten zur Ausführung spezifischer Boot- und Virtualisierungstechnologien.

Definition 4.19 *Asymmetrisches Signatur-Schlüsselpaar –* $AK \in Keys$
Kryptographischer Signaturschlüssel für die vertrauenswürdige Attestierung von geforderten Systemeigenschaften.
$AK_{P_i}^{-1}$, AK_{Con}^{-1} *– privater Signaturschlüssel für Plattform* P_i*, Cloud-Anwender*
AK_{P_i}, AK_{Con} *– öffentlicher Signaturschlüssel der Plattform* P_i*, Cloud-Anwender*

Definition 4.20 *Asymmetrisches Authentisierungs-Schlüsselpaar –* $KeySSL \in Keys$
Kryptographischer Authentisierungsschlüssel für die sichere Authentifizierung von beteiligten vertrauenswürdigen Entitäten auf Netzwerkprotokollebene.
$KeySSL_{P_i}^{-1}$, $KeySSL_{Con}^{-1}$ *– privater Authentisierungsschlüssel für Plattform* P_i*, Cloud-Anwender*
$KeySSL_{P_i}$, $KeySSL_{Con}$ *– öffentlicher Authentisierungsschlüssel für Plattform* P_i*, Cloud-Anwender*

Definition 4.21 *Vertrauenswürdiges Zertifikat*
Enthält einen verifizierbaren asymmetrischen öffentlichen Schlüssel, welcher von einer Instanz der Konzeptklasse Authority herausgegeben wurde.
$CertAK_{P_i}$, $CertAK_{Con}$ *– Attestierungs-Zertifikat für Plattform* P_i*, Cloud-Anwender*
$CertSSL_{P_i}$, $CertSSL_{Con}$ *– Authentisierungs-Zertifikat für Plattform* P_i*, Cloud-Anwender*

Definition 4.22 Ont
Instanz einer Ontologie für die Verarbeitung und Speicherung von Policies des Cloud-Anwenders.

Definition 4.23 *Quote (Q)*
Kryptographischer Hashwert zur Repräsentation vertrauenswürdiger Plattformeigenschaften $c_{j \in N}$ einer Cloud-Plattform (P_i) (siehe Definition II.1)
$Q_{P_i} ::= \mathcal{H}ash(c_1::c_j)$

Die im Abschnitt mit den Definitionen 4.7 - 4.23 eingeführten Artefakte und Ausdrücke sind Bestandteil der nachfolgenden Protokollentwicklung für die horizontale und vertikale Etablierung von Vertrauen.

4.3.2 Horizontale Etablierung von Vertrauen (Establishment of Trust)

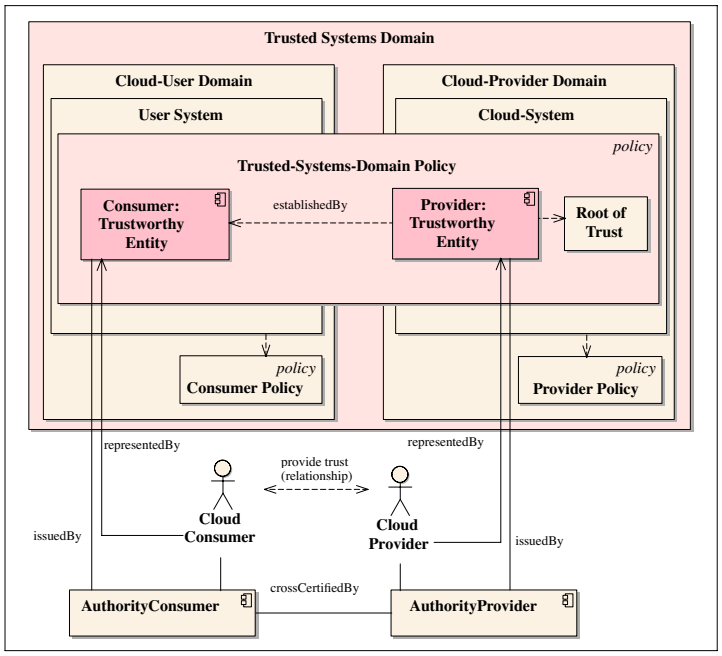

Abbildung 4.7: Horizontale Etablierung von Vertrauen

Als horizontale Etablierung von Vertrauen wird eine Strategie zur Erweiterung einer Cloud-Anwender-definierten Vertrauensgrundlage auf die Seite des Cloud-Anbieters bezeichnet Das Konzept folgt den Prinzipien des Anwendungsfalls zum Aufbau einer Vertrauensbasis im Abschnitt 3.2.2. Aus struktureller Sicht wird in Abbildung 4.7 eine bestehende horizontale Vertrauensbeziehung durch eine verteilte, jedoch für den gemeinsamen Bereich einer *Trusted Systems Domain* wirksame Policy ausgedrückt. Hoheitliche Zuständigkeit manifestiert sich durch getrennte Autoritäten (*AuthorityConsumer*, *AuthorityProvider*) und überträgt auf die jeweilige Policy im Prozess von überprüfbaren Zusicherungen (*issuedBy*) eine domänenspezifische Verbindlichkeit.
Aus einem Konzept separierter Systeme des Cloud-Anwenders (*User System*) und des Cloud-Anbieters (*Cloud-System*) entwickelt sich ein systemübergreifender und hoheitlich durch den Cloud-Anwender bestimmter Regulierungsbereich. Der Zusammenschluss von Autoritäten (*cross-certifiedBy*) vergrößert den Regulierungsbereich durch den Zusammenschluss von Verbindlichkeiten als Maß einer qualifizierbaren Vertrauenswürdigkeit. Die Anwendung vertrauenswürdiger Entitäten gewährleistet die Ausübung hoheitlicher Aufgaben im erweiterten Regulierungsbereich.

4.3.2.1 Phase1: Auswahl einer Cloud-Plattform

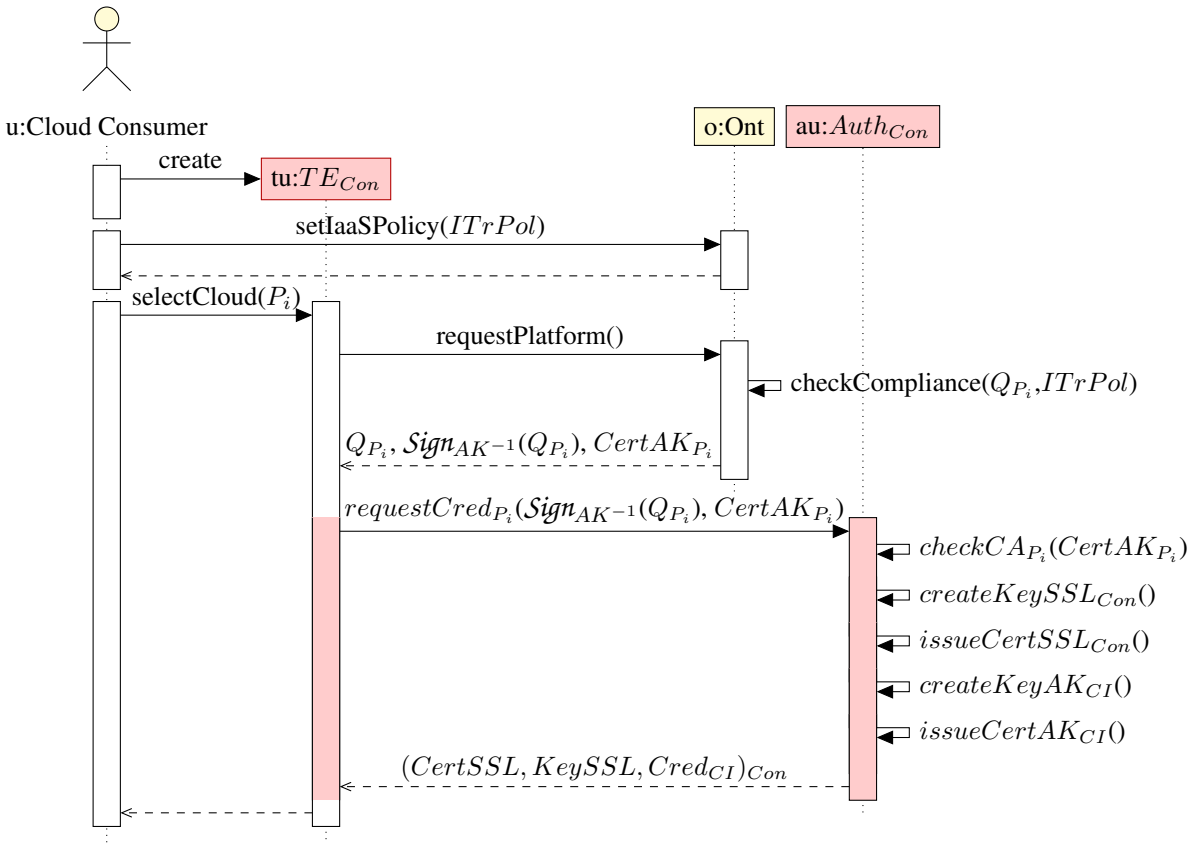

Abbildung 4.8: Phase1: Horizontale Etablierung von Vertrauen (*Auswahl Cloud-Plattform*)

Status	*Auswahl Cloud-Plattform: Horizontale Etablierung von Vertrauen*
Vorbedingung	Der Cloud-Anwender möchte auf Grundlage spezifischer Anforderungen eine Cloud-Plattform auswählen und diese für seine Funktions- und Qualitätsziele einrichten. Die Vertrauenspolitik des Cloud-Anwenders wird durch $Auth_{Cons}$ sichergestellt. Eine vertrauenswürdige Entität (*create* TE_{Con}) auf dem Arbeitsplatz des Cloud-Anwenders gewährleistet die sichere Kommunikation mit einer Cloud-Infrastruktur. Der Cloud-Anbieter kommuniziert über eine vertrauenswürdige Entität (TE_{Prov}) und überträgt zugesicherte Fakten der Cloud-Plattform über eine sichere Verbindung an die Ontologie Ont.
Schritt: (1)	**Policy zur Beschreibung von Cloud-Plattformbedingungen definieren** $setIaaSPolicy$ – Der Cloud-Anwender beschreibt eine Policy und definiert die geforderten infrastrukturellen Eigenschaften einer Cloud-Plattform. Die Beschreibung erfolgt mit einer semantischen Policy-Sprachnotation und wird in der Ontologie gespeichert.

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Status	*Auswahl Cloud-Plattform: Horizontale Etablierung von Vertrauen*
Schritt: (2)	**Cloud-Plattform semantisch auswählen** $selectCloud_{Pi}$ – Die Auswahl einer Cloud-Plattform wird durch die im Schritt (1) hinterlegte infrastrukturelle Policy bestimmt. Die Auswahlanweisung leitet eine semantische Analyse in der Ontologie ein. $checkCompliance$ – Es werden Plattforminstanzen ermittelt, die eine Konformität zur Policy aufweisen.
Schritt: (3)	**Vertrauenswürdige Credentials beantragen** $requestCred_{Pi}$ – Für den Cloud-Anwender werden vertrauenswürdige Credentials als Grundlage für die nachfolgende Etablierungsphase bei der Authority ($Auth_{Con}$) des Cloud-Anwenders beantragt. Im Ergebnis von Schritt (3) liegen am TE_{Con} des Cloud-Anwenders eine signierte Quote ($Sign_{AK^{-1}}(Q_{P_i})$), zusammen mit einem öffentlichen plattformspezifischen Attestierungsschlüssel $CertAK_{P_i}$ der Cloud-Plattform (P_i) vor.
Schritt: (4)	**Plattform-Vertrauenswürdigkeit überprüfen** $checkCA_{P_i}$ – Es ist Aufgabe der Authority, die Vertrauenswürdigkeit der Cloud-Plattform (P_i) auf Grundlage der signierten Quote zu prüfen. Die Gültigkeit des CA-Zertifikats wird überprüft. Als Grundlage für die Prüfung dient der öffentliche plattformspezifische Attestierungsschlüssel $CertAK_{P_i}$.
Schritt: (5)	**Vertrauenswürdige Credentials ausstellen** $createKeySSL_{Con}$ – Die Authority erzeugt ein Schlüsselpaar ($KeySSL_{Con}$) für die sichere Netzwerkkommunikation zwischen den vertrauenswürdigen Entitäten. $issueCertSSL_{Con}$ – Die Authority stellt das zugehörige Zertifikat ($CertSSL_{Con}$) aus. $createKeyAKCI$ – Für den sicheren Identitätsnachweis der vertrauenswürdigen Entität TE_{ConCI} wird ein Schlüsselpaar $KeyAK_{CI}$ erzeugt. $issueCertAKCI$ – Das zugehörige Zertifikat $CertAK_{CI}$ wird ausgestellt (zusammengefasst als $Cred_{CI}$). Nach Übertragung von TE_{ConCI} auf die Seite des Cloud-Anbieters bilden die Credentials ($Cred_{CI}$) die Grundlage für das darauf aufbauende Vertrauenskonzept (siehe Abschnitt 4.3.3).
Nachbedingung	**Vertrauensgrundlage des Cloud-Anwenders hergestellt** Der Cloud-Anwender besitzt vertrauenswürdige Credentials, um die bestehende Vertrauensgrundlage auf die Seite des Cloud-Anbieters erweitern zu können. Der Cloud-Anwender besitzt die Zusicherung, dass vertrauenswürdige infrastrukturelle Plattform-Eigenschaften auf Seite des Cloud-Anbieters vorliegen.

Tabelle 4.1: Ablaufbeschreibung – Auswahl Cloud-Plattform

4.3.2.2 Phase2: Erweiterung der Vertrauensgrundlage auf Cloud-Anbieter-Seite

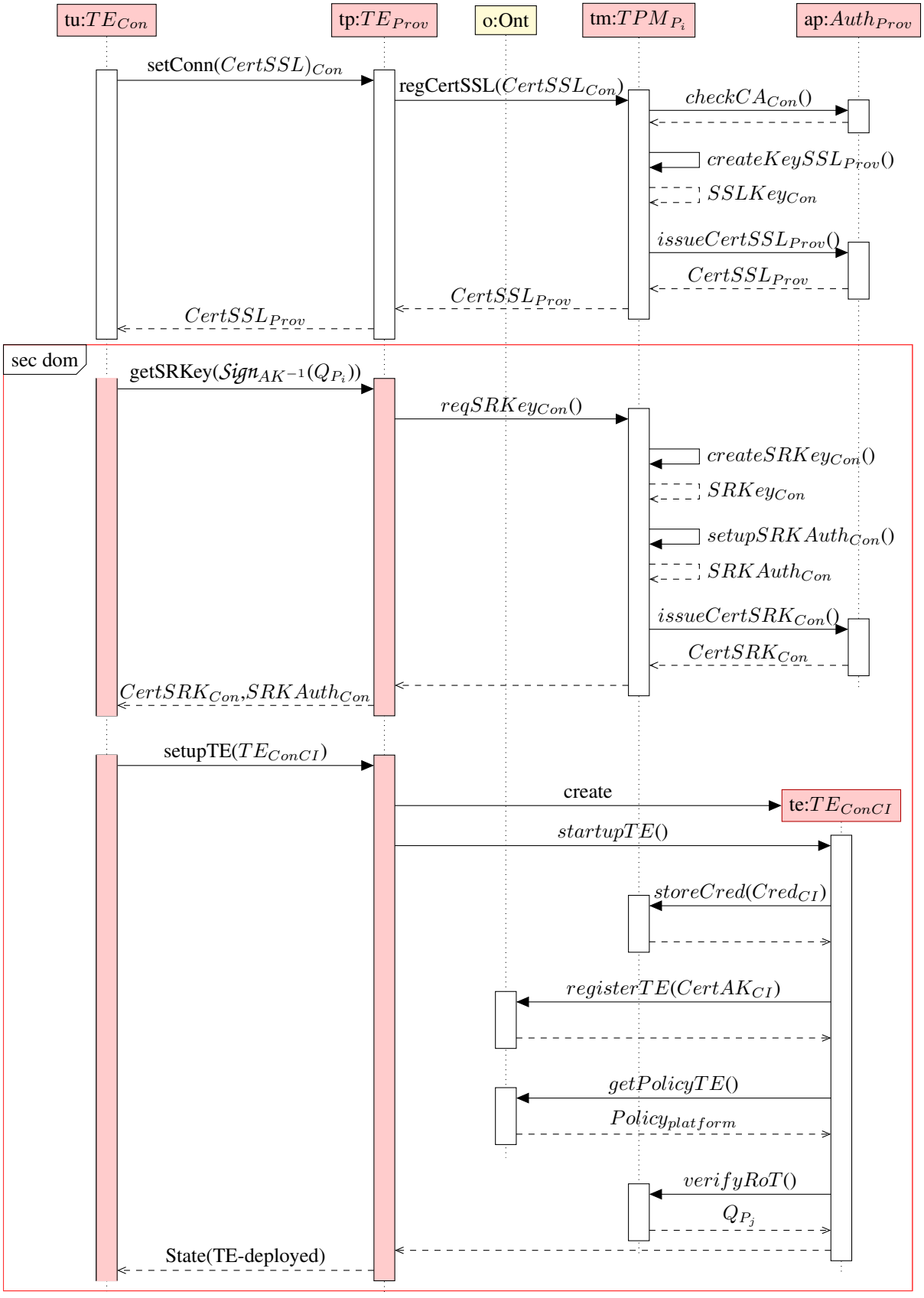

Abbildung 4.9: Phase2: Vertrauensgrundlage auf Cloud-Anbieter-Seite erweitern

Status	*Erweiterung Vertrauensgrundlage: Horizontale Etablierung von Vertrauen*
Vorbedingung	Der Cloud-Anwender hat eine eigene Vertrauensgrundlage eingerichtet (siehe Beschreibung in Tabelle 4.1).
Schritt: (1)	**Transportsicherheit herstellen** $setConn$ – Die vertrauenswürdige Entität des Cloud-Anwenders stellt eine Anforderung, um eine vertrauliche Netzwerkkommunikation (Transportsicherheit) mit der vertrauenswürdigen Entität des Cloud-Anbieters (TE_{Prov}) herzustellen.
Schritt: (2)	**Credentials für Transportsicherheit anfordern** $reqCertSSL$ – Die vertrauenswürdige Entität des Cloud-Providers stellt eine Anforderung, um notwendige kryptographische Credentials für die Transportsicherheit zu erzeugen. Das Cloud-Anwender-Zertifikat ($CertSSL_{Con}$) für die Transportsicherheit wird übergeben.
Schritt: (3)	**Credentials für Transportsicherheit erzeugen** $checkCA_{Con}$ – Die Vertrauenswürdigkeit des Cloud-Anwender-Zertifikats wird überprüft. $createKeySSL_{Prov}$ – Ein *Trusted Platform Module* (TPM_{P_i}) erzeugt ein Schlüsselpaar und beantragt die Ausstellung eines Zertifikats. $issueCertSSL_{Prov}$ – Das von der $Auth_{Prov}$ erzeugte Zertifikat wird anschließend an die vertrauenswürdige Entität des Cloud-Anwenders (TE_{Con}) übermittelt.
Schritt: (4)	**Credentials für vertrauenswürdige Entität anfordern** $getSRKey$ – Unter Nutzung einer vertraulichen Netzwerkverbindung (*sec dom* repräsentiert eine Sicherheitsdomäne) werden vom Cloud-Anbieter vertrauenswürdige Speicher-Credentials ($SRKey_{Con}$) angefordert. Speicher-Credentials (*Storage-Root-Key-Consumer*) dienen zur sicheren Speicherung der vom Cloud-Anwender erzeugten Credentials für die vertrauenswürdige Entität, die auf die Seite des Cloud-Anbieters übertragen wird. (siehe Tabelle 4.1: Schritt **??**)
Schritt: (5)	**Credentials für vertrauenswürdige Entität erzeugen** $createSRKey$ – Für den Cloud-Anwender wird kryptographisch ein Basis-Storage-Key ($SRKey_{Con}$) im *Trusted Platform Module* (TPM) erzeugt und eine Zertifikat angefordert. $setupSRKAuth$ – Für den Cloud-Anwender werden kryptographisch im TPM Autorisierungs-Credentials ($SRKAuth_{Con}$) erzeugt, die für die Zugriffssteuerung im TPM zur Anwendung kommen. $issueCert$ – Das zugehörige Zertifikat ($CertSRK_{Con}$) wird ausgestellt.

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Status	*Erweiterung Vertrauensgrundlage: Horizontale Etablierung von Vertrauen*
Schritt: (6)	**Cloud-Anwender-TE auf Seite des Cloud-Anbieters installieren** $setupTE$ – Der Cloud-Anwender besitzt kryptographische Zugangsdaten, um eine TE_{ConCI}-Instanz auf die Cloud-Anbieter-Infrastruktur netzwerkgesichert zu übertragen. Die Instanz enthält alle notwendigen Credentials ($SRKAuth_{Con}$, $CertSRK_{Con}$, $Cred_{CI}$), um das Deployment und die Speicherung der Schlüssel und Zertifikate vorzunehmen. $create$ – Die Cloud-Anbieter Instanz TE_{Prov} führt das Deployment durch und startet die TE_{ConCI}-Instanz.
Schritt: (7)	**Cloud-Anwender-TE auf Seite des Cloud-Anbieters installieren** $storeCred$ – Die TE_{ConCI}-Instanz speichert TE-spezifische kryptographische Credentials in einem geschützten TPM-Speicherbereich (TPM_{P_i}).
Schritt: (8)	**Registrierung der Cloud-Anwender-Vertrauensbasis** $registerTE$ – Die Registrierung in der Ontologie-*Cloud-Domain* kennzeichnet die TE_{ConCI}-Instanz als vertrauenswürdige Entität für den Infrastruktur- und Plattform-Architekturbereich. Für die Verbindung zum Ontologie-Service erfolgt über eine gesicherte Netzwerkverbindung.
Schritt: (9)	**Policy für Infrastruktur abrufen** $getPolicy$ – Für die Ausführung von Regulierungsaufgaben auf der IT-Plattformebene ruft die TE_{ConCI}-Instanz bereitgestellte Policies ab.
Schritt: (10)	**Infrastrukturelle Vertrauensbasis überprüfen** $verifyRoT$ – Die TE_{ConCI}-Instanz vergleicht den bestehenden infrastrukturellen Zustand (*Root of Trust*) mit dem attestierten Zustand Q_{P_i} aus Schritt (3) in Tabelle 4.1.
Nachbedingung	**Horizontale Erweiterung einer Vertrauensbasis** Der Cloud-Anwender besitzt mit der installierten TE_{ConCI}-Instanz einen erweiterten Hoheitsbereich, der sich auf der Seite des Cloud-Anbieters befindet. Er ist in der Lage, den infrastrukturellen Bereich zu überprüfen und, darauf aufbauend, ein regelndes Policy-Konzept für weitere Architekturebenen auszuführen.

Tabelle 4.2: Ablaufbeschreibung – Vertrauensbasis horizontal erweitern

4.3.3 Vertikale Etablierung von Vertrauen (Delegation of Trust)

Die Vorgehensweise in Abbildung 4.10 beschreibt eine vertikale Etablierung von Vertrauen als eine Strategie zur Bildung hierarchischer, vertrauenswürdiger Anwendungsbereiche innerhalb einer Cloud-Architektur. Das Policy-Konzept zur Entwicklung von Vertrauen in Abschnitt 4.2.3 wird hier aufgenommen und zu einem ganzheitlichen Anwendungsprinzip vertrauenswürdiger Entitäten weiterentwickelt.

Die Klassifikation vertrauenswürdiger Entitäten im Abschnitt 4.2.1 unterscheidet zwischen Policy-Entitäten (*TE-P*-Instanzen) und Domänen-Entitäten (*TE-D*-Instanzen).
Policy-Entitäten übernehmen hoheitliche Aufgaben für den Aufbau einer abgegrenzten und geregelten Cloud-Anwender-Servicedomäne (*Cloud-Consumer domain*) und sind für die sichere Überprüfung, Übernahme und Aktivierung von Domänen-Entitäten verantwortlich (Definition 4.2)..
Bereitgestellte Ressourcen des Cloud-Anwenders oder verschiedener Cloud-Anbieter repräsentieren Entitäten der Servicedomäne (Definition 4.3). Domänen-Entitäten werden als überprüfbare Einheiten bereitgestellt (*Packaged object*).
Die vertikale Etablierung von Vertrauen entwickelt sich in Abbildung 4.10 als ein Prozess verteilter Verantwortlichkeiten zwischen *TE-P*-Instanzen mit Maßnahmen zur sicheren Bestimmung der Vertrauenswürdigkeit und *TE-D*-Instanzen für die geregelte Steuerung und Ausführung von Domänen-Services. Im Zuständigkeitsbereich einer *TE-P*-Instanz (grünes Farbschema) sind eine oder mehrere Domänen-Entitäten (rotes Farbschema) zu berücksichtigen (*Next rule*). Trotz des Nachweises vertrauenswürdiger Eigenschaften wird außerhalb definierter Nutzungsbedingungen (*No valid policy pre-condition*) die Anwendung einer *TE-D*-Instanz ausgeschlossen. Optional können kryptographische Verfahren für die Autorisierung einer TE-D eingebunden werden. Die Bereitstellung eines entsprechenden Schlüssels $K_{(TE-D)}$ erfolgt nur unter Zusicherung überprüfbarer Vorbedingungen.
Nach Aktivierung einer *TE-P*-Instanz wird eine bestehende Vertrauenskette überprüft. Bei Nichterfüllung von bestehenden Vertrauensbedingungen (*No valid trust pre-condition*) wird die vertikale Etablierung von Vertrauen abgebrochen.

4.3.3.1 Registrierung von Policy-Entitäten

Die Registrierung von Policy-Entitäten (*Register entity for authoritative regulation*) schafft die Grundlage für eine nachvollziehbare Policy-Umsetzung. Jede Erfassung einer Policy-Entität repräsentiert eine Erweiterung des Cloud-Anwender-Hoheitsbereiches und vollzieht das Prinzip einer *Delegation of Trust*. Registrierte Policy-Entitäten bilden Fakten in der Ontologie-*Cloud-Domain* und sind Grundlage für die Ableitung bestehender Vertrauensbeziehungen. Sie liefern Vorbedingungen für die Erweiterung einer bestehenden Vertrauensbasis.

4.3.3.2 Registrierung von Domänen-Entitäten

Die Registrierung von Domänen-Entitäten (*Register entity for domain services*) erfasst vertrauenswürdige Eigenschaften einer *TE-D*-Instanz und kennzeichnet diese für die Anwendung in der Cloud-Anwender-Servicedomäne als vertrauenswürdige Entität. Die Registrierung ist eine notwendige Bedingung für die Nutzungsfreigabe (*Activate domain entity*) einer *TE-D*-Instanz.
Während Policy-Entitäten ein einheitliches Vertrauensniveau besitzen, so kann es sich für registrierte Domänen-Entitäten unterscheiden. Bestimmend dafür ist das Verfahren für die Zusicherung von vertrauenswürdigen Eigenschaften einer *TE-D*-Instanz. Die Anwendung verschiedener *Credential*-Konzepte führt zu einer Differenzierung im Vertrauensniveau.

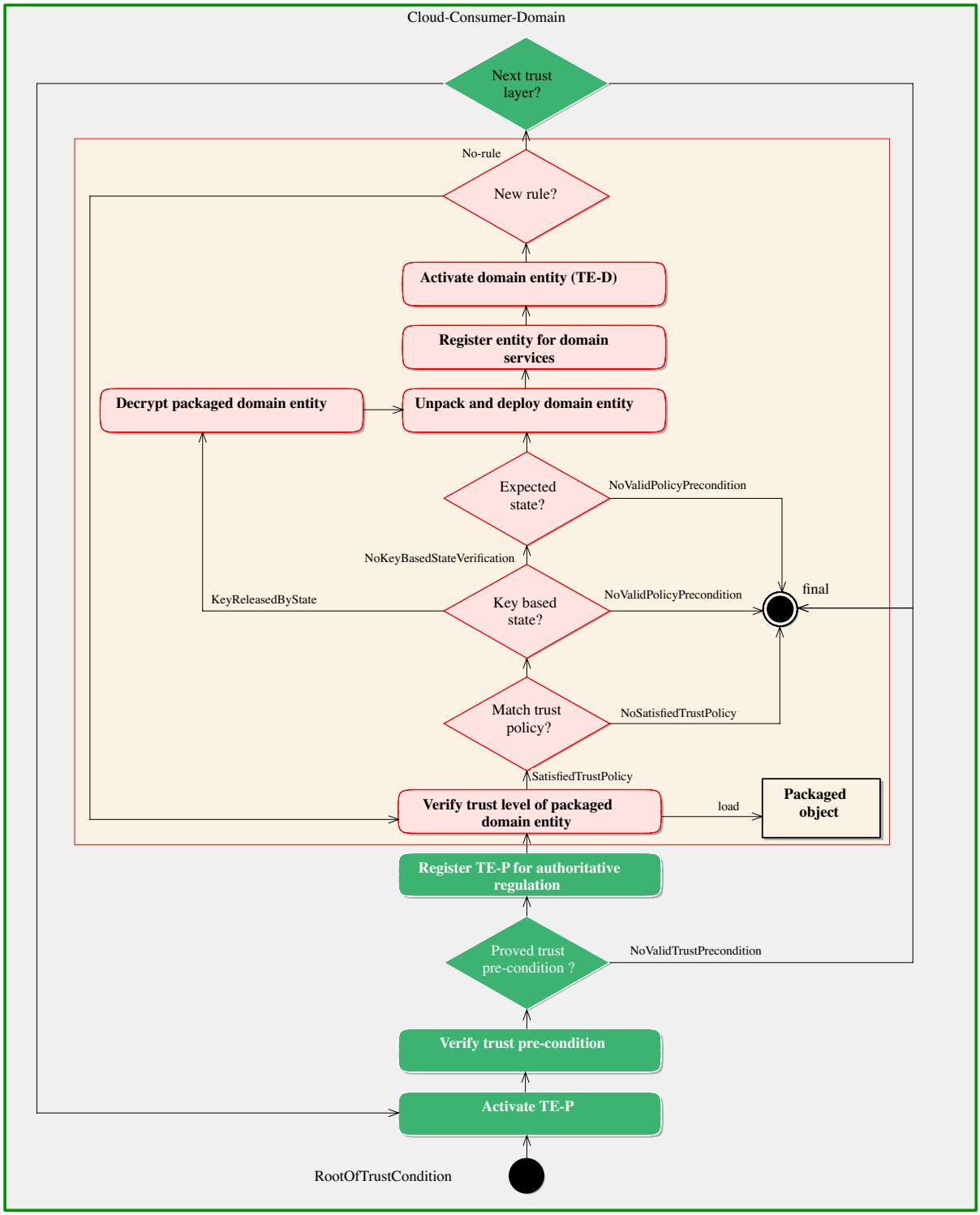

Abbildung 4.10: Ganzheitliches Anwendungsprinzip vertrauenswürdiger Entitäten

4.3.3.3 Ableitung vertrauenswürdiger Entitäten

Das im Abschnitt 4.2.2 eingeführte Konzept einer *Authority* bildet den Ausgangspunkt für eine Strategie der hoheitlichen Regulierung des Cloud-Anwenders in bereitgestellten IT-Architekturbereichen. Die Ausführung von TE-P-basierten Policies schafft eine qualifizierte Faktensituation, da sich die Eigen-

schaften registrierter Domänen-Entitäten auf Prüfprozesse vertrauenswürdiger Policy-Entitäten (TE-P) zurückführen lassen. Die in Definition 4.4 formulierte Anforderung für die sichere Identifizierung einer Entität wird damit erfüllt. Die nachfolgenden Ableitungskonzepte bilden einen Beschreibungsrahmen, um Schlussfolgerungen über bestehende Vertrauensbeziehungen zu ziehen.

Listing 4.53: Methode *reasonBasedTEP* – Reasoning-Rule

```
1  @{RootOfBasedTEP}
2  ?X[reasonBasedTEP->?T]
3  :-
4  ?X:Entity                                                    AND
5  ?X[checkQP->?Y]                                              AND
6  ?Y[<nss#enforcedThrough>->?P]    AND
7  ?P[<nsr#targetToZone>->?Z]       AND
8  ?Z[<nsr#assignedTE>->?T].
```

Über die in Listing 4.53 beschriebene *Reasoning-Rule* werden vertrauenswürdige Bindungen registrierter Domänen-Entitäten als Faktenbasis ermittelt. Die Konzeptklasse *Entity* wird dafür um die Reasoning-Methode *reasonBasedTEP* erweitert und dynamisch ausgeführt. Im Ergebnis repräsentiert die Variable (?T) (2) eine Menge von Policy-Entitäten, die zur Regulierung der erforderlichen Domänen-Entitäten aktuell eingesetzt werden.
Die für eine Policy-Entität zuständige Instanz einer *Authority* kann unter Anwendung der Ableitungskonzepte im Abschnitt 4.2.2.2 (siehe Listing 4.44) ermittelt werden. Die Konzeptklasse *Entity* wird um die Reasoning-Methode *reasonTEPAuthority* erweitert.

Listing 4.54: Methode *reasonTEPAuthority* – Reasoning-Rule

```
1  @{RootTEPAuthority}
2  ?X[reasonBasedTEPAuthority->?A]
3  :-
4  ?X:Entity                                  AND
5  ?X[reasonBasedTEP->?T]                     AND
6  ?T[authorizedTo(?C)->?I]                   AND
7  ?C[<nss#reasonRootOfTrust>->?A].
```

Die *Reasoning-Rule* 4.54 integriert die Methode *reasonBasedTEP* (5) aus Listing 4.53 und die Methode *reasonRootOfTrust* aus Listing 4.44. Im Ergebnis präsentiert die Variable (?A) (7) die Menge der Instanzen, die als *Root Authority* für die entsprechenden Policy-Entitäten eingesetzt werden.

4.3.3.4 Ableitung vertrauenswürdiger Eigenschaften und Aktivitäten

Die Konzeptklassen *Rule*, *Policy* und *State*, eingeführt und beschrieben im Abschnitt 4.1.3, enthalten konzeptionell die Grundlagen für die Ableitung vertrauenswürdiger Eigenschaften von Domänen-Entitäten.

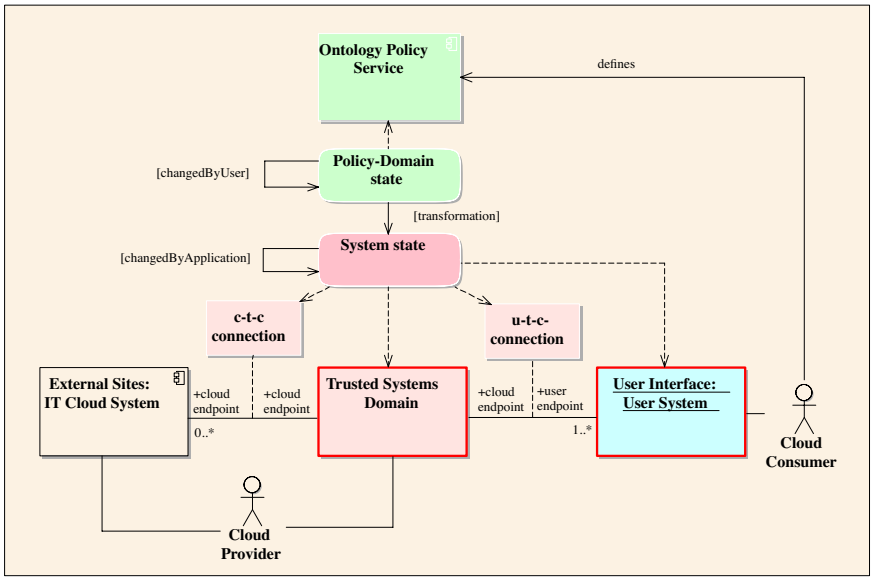

Abbildung 4.11: Ebenen der State-Konzeptualisierung

Sowohl die Konzeptklasse *Rule* als auch *Policy* manifestieren ihre Ergebnisse über Instanzen der Konzeptklasse *State* (siehe Methode *resultedState* in Listing 4.8).
Mit der Einführung der Konzeptklasse *State* werden die Grundprinzipien von Regulierung als Konzept zur Erzeugung definierter System-Eigenschaften axiomatisch abgebildet. Der in Listing 4.12 beschriebene Konzeptrahmen zur Repräsentation von Systemeigenschaften für unterschiedliche Regulierungsziele wird in Abbildung 4.11 allgemein als *System state* wiedergegeben.
Der Konzeptualisierung eines *System state* auf Grundlage einer formal modellierten Policy erweitert den statischen Domain-Konzeptansatz um einen dynamischen Begriff von *Verhalten*. Die Konzeptklassen *Action* und *Transformation*, beide Bestandteile der Rule-Konzeptualisierung, beschreiben ein Systemverhalten als Folge von Transformationen eines *System state*.
Das Modell einer Zustandsmaschine bietet dafür eine mathematische Grundlage, deren Zustände sich über allgemeine mathematische Strukturen beschreiben lassen [BS03]. Daraus abgeleitet, überführt der ontologische Ansatz das Konzept algebraischer Strukturen in eine deklarative formale Beschreibung zugesicherter Eigenschaften einer definierten Domäne.
Die Ergebnisse jeder Instanz der Konzeptklasse *State* können auf die zugehörige Instanz der Konzeptklasse *Policy* zurückgeführt werden (*Policy change*). Die vorgelegte Spezifikation stellt sicher, dass Änderungen im System-Status (*Policy transformation*) sich nur im Ergebnis einer Transformation (siehe Abschnitt 4.1.3.7) als Teil der *Rule*-Konzeptualisierung einstellen lassen.
Die Berechenbarkeit von Zuständen, nur basierend auf Policy-Änderungen, überführt das System in ein vertrauenswürdiges Steuerungskonzept.
Die für einen System-Status prüfende Instanz einer vertrauenswürdigen Entität kann unter dem in Listing 4.55 spezifizierten Ableitungskonzept ermittelt werden. Die Konzeptklasse *State* wird dafür um die Reasoning-Methode *reasonStateAssuredBy* erweitert.

Listing 4.55: Methode *reasonStateVerifiedBy* – Reasoning-Rule

```
1  @{RootStateTEP}
2  ?S[reasonStateAssuredBy->?T]
3  :-
4  ?S:State                    AND
5  ?S[enforcedBy->?P]          AND
6  ?P[targetToZone->?Z]        AND
7  ?Z[assignedTE->?T].
```

Die Repräsentation von Statusänderungen schafft eine qualifizierte Faktensituation, da sich die Kommunikation von Statusänderungen auf Bestätigungsoperationen vertrauenswürdiger Policy-Entitäten (TE-P) zurückführen lässt. Die in Definition 4.6 formulierte Anforderung zur Zurechenbarkeit von Handlungsschritten einer Entität wird damit erfüllt.
In Abbildung 4.11 setzt sich der erwartete Systemzustand aus dem Zustand des Domänen-Modells (*Trusted Systems Domain*) und den Zuständen notwendiger Verbindungen (*Connection*) zusammen. Die Reasoning-Methode in Listing 4.55 kann logisch erweitert werden, um den Zustand bestehender Verbindungen übergreifend abzuleiten (*Compliance-Monitoring*).

4.4 Zusammenfassung

Die Konzeptualisierung von Vertrauen bildet den Kern der Spezifikation im Abschnitt 4.2. Der Abschnitt beschreibt ein Vorgehen, wie sich geregelte vertrauenswürdige Bedingungen unter Anwendung einer eigenständigen Policy herausbilden können.
Als vertrauensbildende Grundelemente werden die Konzeptklassen *Entität* und *Authority* eingeführt. Erst die Bedingungen zur Transformation einer *Entität* in eine *Vertrauenswürdige Entität* überführen ein System in einen vertrauensbildenden Qualitätszustand. Unter Verwendung von Funktionen einer *Authority* erhalten Policies einen zugesicherten und verbindlichen Bezugspunkt.
Innerhalb einer definierten Cloud-Domäne führt die Anwendung regelnder Maßnahmenkonzepte mit *Policy-Entitäten* zu einem Grundprinzip hoheitlicher schrittweiser Erweiterung des Einflussbereiches eines Cloud-Anwenders. Die Aufgabe von *Policy-Entitäten* besteht darin, Bedingungen zu bewerten und in vertrauenswürdige Bedingungen zu überführen. Die konsequente Anwendung dieser Methodik auf alle erforderlichen Ressourcenbereiche der Domäne, bildet sich zu einem ganzheitlichen Konzept vertrauenswürdiger *Domänen-Entitäten* heraus.
Das Trust-Establishment-Protokoll entwickelt einen Ablauf, wie das vorgestellte Prinzip hoheitlicher Erweiterung in horizontaler und vertikaler Richtung zur Anwendung kommt.
Die semantische Beschreibung der vorgestellten Konzeptklassen schafft die Voraussetzung zur Ableitung logischer Schlussfolgerungen über bestehende Vertrauensstrukturen. Die Integration der Ontologie in das Trust-Establishment-Protokoll dient als Instrument zur Ermittlung von vertrauenswürdigen Eigenschaften und Aktivitäten der beteiligten *Policy-Entitäten* und ist Grundlage für ein kooperatives Zusammenwirken.

5 | Validierung

Der folgende Abschnitt überprüft wesentliche Kernaspekte der Trust-Konzeptualisierung am Beispiel einer Cloud-Referenzarchitektur. Die ausgewählten Szenarien präsentieren einen Cloud-Anwender in der Rolle als Architekt einer vertrauenswürdigen Cloud-Architektur. Die Anwendung von Trust-Policies entwickelt systematisch ein Verbundkonzept, bestehend aus vertrauenswürdigen IT-Komponenten. Die technische Implementierung einer vertrauenswürdigen Entität bildet das Bindeglied zwischen der Realisierung einer infrastrukturellen Vertrauenshierarchie und einer sicheren Interpretation semantisch repräsentierter Regulierungsziele und Domänenspezifikationen. Für die Validierung werden folgende Ziele definiert:

(Z1) Referenzarchitektur – Basis zur Überprüfung der Domain-Konzeptualisierung

Am Beispiel einer Referenzarchitektur ist die Erfüllbarkeit der Ontologie-*Cloud-Domain*-Konzeptualisierung nachzuweisen. Die wesentliche Zielstellung liegt in der Überprüfung der Detaillierungsfähigkeit des Modells. Überprüft werden erforderliche Objekteigenschaften, die Verbindungskomplexität (Relation) und die Qualitätsmerkmale der Verbindungen.

(Z2) Trust-Policy – Abdeckung und Ausdrucksstärke

Im Mittelpunkt der Untersuchung steht die Frage, ob das Konzept einer vorgelagerten vertrauensbildenden Trust-Policy mit der Ontologie-*Regulation* vollständig durchgesetzt werden kann. Es ist zu überprüfen, ob sich im Ergebnis der Durchsetzung einer Sicherheits-Policy bestehende Vertrauensgrundlagen ableiten und bewerten lassen. Das Konzeptklasse *Constraint* bestimmt die kontextbezogene Fähigkeit, regulierende Anforderungen weiter einzugrenzen. Welche semantische Ausdrucksstärke steht für eine Policy-Verfeinerung zur Verfügung?

(Z3) Transformation – Technische Integrationsfähigkeiten

Vertrauenswürdige Entitäten benötigen Schnittstellen für die zugesicherte Umsetzung von Policies. Es wird eine Bewertung vorgenommen, welche technischen Fähigkeiten für die Transformation zur Verfügung stehen, um Detailvorgaben von Policy-Definitionen in adäquate technische Operationen überführen zu können. Darüber hinaus ist zu prüfen, ob adaptive Erweiterungen für komplexere Policy-Transformationen möglich sind.

(Z4) Attestierung – Nachweisfähigkeit und Compliance

Das Konzept vertrauenswürdiger Entitäten muss die Messbarkeit von IT-Plattformeigenschaften, Systemzuständen und die Wirksamkeit eingestellter Policystrukturen nachweisen können. Es sind Untersuchungen vorzunehmen, inwieweit der Aspekt Vertrauenswürdigkeit durch ontologische Ableitungsregeln eine unterscheidbare Werte-Repräsentation erhält.

5.1 Referenzarchitektur – Trusted Cloud

Für den Nachweis der Ziele Z1, Z2, Z3, und Z4 wurde das bereits in [Keb+15] eingeführte Cloud-Referenzmodell zu der folgenden *Referenzarchitektur – Validierung* weiterentwickelt.

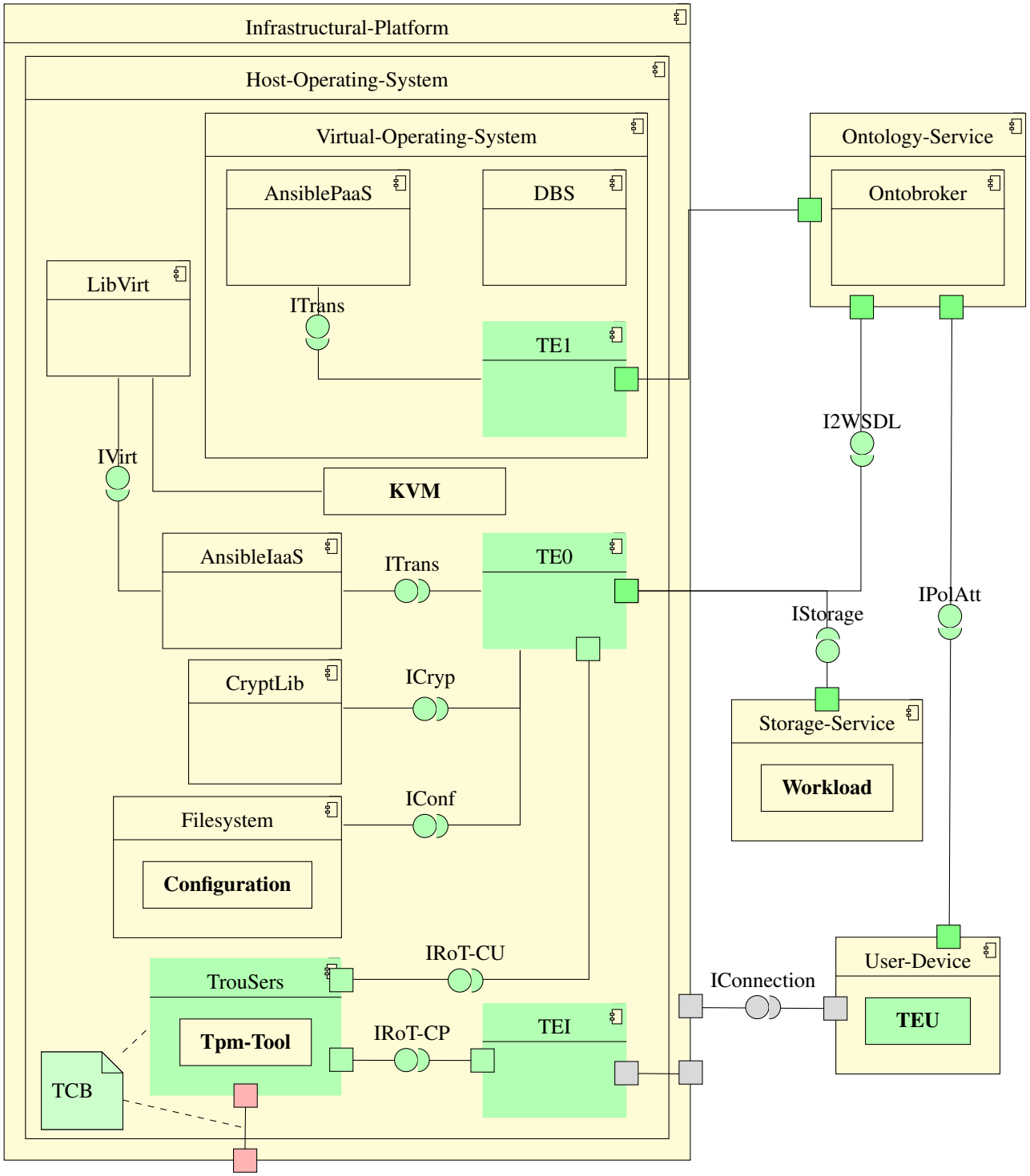

Abbildung 5.1: Referenzarchitektur – Validierung

Die Referenzarchitektur in Abbildung 5.1 beschreibt in einer Bausteinsicht die notwendigen Komponenten und ihre Beziehungen zu externen Komponenten für das Policy-Management, die Attestierung

und die Datenspeicherung. Für die Visualisierung verketteter Vertrauenskonzepte (*Chain of Trust*) werden in allen strukturellen Ebenen der Architektur vertrauenswürdige Komponenten mit einem *grünen* Farbschema hervorgehoben. Die Schnittstellen sind mit dem Präfix (I) gekennzeichnet.
Die Komponenten der Referenzarchitektur und ihre Beziehungen sind vollständig ontologisch modelliert. Die folgende Tabelle gibt eine Übersicht über die Instanzen der Referenzarchitektur als Auszug aus der Ontologie-*Cloud-Domain*.

Instanz der Ontologie-*Cloud-Domain*	**Komponente der Referenzarchitektur**
V_D_Platform-System	Infrastructural-Platform
V_D_Host-OS-Debian	Host-Operating-System
V_D_Virtual-OS	Virtual-Operating-System
V_D_KVMSoftware	KVM
V_D_PostgreDBS	PostgreSQL
TE_0	TE0
TE0-Identity.Virtualisation	TE0-Identität
TE_1	TE1
TE1-Identity.RuntimeOS	TE1-Identität
TE_I	TEI

Tabelle 5.1: Auszug aus Ontologie-*Cloud-Domain* für Referenzarchitektur

5.1.1 Komponentenbeschreibung – IT-Plattform

Die IT-Plattform umfasst die Ebenen *Infrastructural-Platform* und *Host-Operating-System* und enthält Komponenten der Hardware-Infrastruktur, Komponenten für die Virtualisierung und zur Steuerung einer Vertrauenspolitik.

Infrastructural-Platform

Die Serverplattform besteht aus einem DELL Server PowerEdge R520-System mit aktivierten Funktionen VT-enabled (Intel® Virtualization Technology) [Int16d] und TXT-enabled (Trusted Execution Technology) [Gre12]. Die Boot-Prozess-Steuerung erfolgt über *Trusted Boot* (tboot Version 1.8.2) [Tru16].

Vertrauensanker – Root of Trust

Das Validierungskonzept überprüft das Konzept für eine vertikale Delegierung von Vertrauen. Die Schnittstelle *IRoT* (Root of Trust) dient zur Verfiizierung hardwarebasierter Sicherheitseigenschaften der IT-Plattform (siehe Abschnitt II.3). Eine TPM-Lösung Version 1.2 [Tru11a; Tru14] und ein Trusted Computing Software Stack (*TrouSers*) [Tro08] für den Zugriff auf TPM-interne Sicherheitsbereiche bilden dafür eine *Trusted Computing Base* (TCB). Für erweiterte Abfragen von TPM-Registern steht als Teil der *TrouSers*-Schnittstelle *IRoT* eine API zur Nutzung von *Tpm-Tools*-Funktionen [Tro16] zur Verfügung.

Host-Operating-System

Als Grundlage für die Virtualisierung wird als *Host-Operating-System* das Debian-Linux-System Ver-

sion 8 (Jessy) mit Kernel-Version 3.16.0-4 -amd64 eingesetzt. Das *Host-Operating-System* verfügt über 1 GB Hauptspeicher und über eine physische Netzwerkkarte (*eth0*).

KVM

Die Vollvirtualisierung basiert auf *Kernel-based Virtual Machine* (KVM) [KVM16; Rod+12] und Quick-Emulator (QEMU) [Bon+16] für die Hardwareemulierung. KVM arbeitet als Hypervisor für das virtualisierte Betriebssystem und wird als Kernelmodul des Host-Betriebssystems installiert. KVM unterstützt verschiedene Hardwareplattformen.

LibVirt

Die Komponente *LibVirt* [Lib16] enthält eine API und dient als Schnittstelle (*IVirt*) für die technische Steuerung von Virtualisierungsaufgaben unter Anwendung von KVM.

TEI – Trusted Entity (Infrastruktur)

Die Komponente TEI ist eine Service-Implementierung für die sichere und verbindliche Attestierung von infrastrukturellen Eigenschaften. Die Komponente wird von einem Plattform-Anbieter bereitgestellt und repräsentiert den TE_{Prov}-Baustein in Abbildung 4.8. Ein TEI-basierter Service bildet die technische Voraussetzung, um die vertrauenswürdige Entität TE0 des Cloud-Anwenders sicher zu installieren.

Die Installation der Komponente TEI erfolgt während des Systemstarts und ist Teil der Systemkonfiguration des Plattformbetreibers. Die Schnittstelle *IRoT-CP* (Root of Trust – Cloud Provider) repräsentiert eine kryptographische Verbindung zu einem hardwarebasierten Vertrauensanker. In der weiteren Beschreibung wird die Komponente TEI in ihrer Anwendung als TE_I-Instanz bezeichnet.

TE(x) – Trusted Entity (Cloud-Anwender)

Die Komponenten TE0 und TE1 sind Java-basierte Implementierungen einer vertrauenswürdigen Entität und folgen mit ihren Eigenschaften der Spezifikation im Abschnitt 4.2.1. Die Komponenten werden von einem Cloud-Anwender bereitgestellt und repräsentieren den TE_{Con}-Baustein in Abbildung 4.8. In der weiteren Beschreibung werden die Komponenten TE0 und TE1 in ihrer Anwendung als TE_0-Instanz und TE_1-Instanz bezeichnet.

Im Konzept für eine horizontale Erweiterung der Vertrauensgrundlage für den Cloud-Anwender (Phase 2 in Abbildung 4.9) stellt eine TE_I-Instanz notwendige kryptographische Parameter zur Anwendung der Schnittstelle *IRoT-CU* (Root of Trust – Cloud User) bereit.

Ansible – Transformation

Als ein Vertreter der DevOps-Initiative [Kim+16] wird für die technische Transformation von Policies das Automatisierungstool *Ansible* [Hoc14; red] verwendet. Die Konzeptklasse *Transformation* im Abschnitt 4.1.3.7 unterstützt neben *Ansible* weitere DevOps-Technologien. Die Schnittstelle *ITrans* ist eine *Ansible*-Implementierung und stellt Funktionen für die technische Transformation von Deployment- und Sicherheitspolicies, getrennt nach Architekturebenen (*AnsibleIaaS* – Infrastrukturebene, *AnsiblePaaS* – Laufzeitebene), bereit.

CryptLib

Die Schnittstelle *ICrypt* stellt Sicherheitsfunktionen für die Verschlüsselung, für digitale Signaturen, Hash-Funktionen, Netzwerk- und PKI-Sicherheitsprotokolle bereit. Darüber hinaus stehen Funktionen für die Realisierung von CA-Policies und ein X.509-konformes Zertifikatsmanagement zur Verfügung.

Die Security-Software *cryptlib* [Gut15] stellt Implementierungen für verschiedene kryptographische Operationen bereit.

Filesystem

Das Filesystem im Host-Operating-System speichert die TE0-Konfiguration (*Configuration*) mit initialen Parametern für den Start (Identity-Name) und für die Netzwerkkommunikation (IP-Adresse, Adresse Ontobroker). Initiale Parameter werden über die Filesystem-Schnittstelle *IConf* eingelesen. Weitere Konfigurationen erfolgen über den *Ontology-Service*.

5.1.2 Komponentenbeschreibung – Laufzeitumgebung

Die IT-Laufzeitumgebung umfasst die Ebenen *Virtual-Operating-System* und enthält das PostgreSQL-Datenbanksystem.

Virtual-Operating-System (Virtual-OS)

In der Virtualisierung kommt ein Debian-Linux-System Version 8 (Jessie) mit Kernelversion 3.16.0-4 - amd64 zum Einsatz. Das *Virtual-Operating-System* verfügt über 512 MB Hauptspeicher und unterstützt eine virtuelle Netzwerkkarte (*eth0*). Die Bereitstellung erfolgt in Form einer *Virtual Machine* (VM).

PostgreSQL-Datenbanksystem (DBS)

Als freies und objektrelationales Datenbankmanagementsystem (ORDBMS) wird *PostgreSQL* verwendet [Pos17].

5.1.3 Komponentenbeschreibung – Integrierte Systeme

Storage-Service

Der *Storage-Service* dient als externe Speicherkomponente eines Cloud-Systems. Über die Schnittstelle *IStorage* speichert ein Cloud-Anwender eigene *Workloads*, die für das Deployment benötigt werden. Zur Sicherstellung der Vertraulichkeit können Daten und Programme auf dem *Storage-Service* verschlüsselt gespeichert werden.

Ontology-Service

Der *Ontology-Service* steuert den Zugang zum zentralen semantischen Modell und liefert die formale Beschreibung für alle Konzeptbereiche, die für eine vertrauenswürdige Regulierung in einem Cloud-System erforderlich sind. Der *Ontology-Service* stellt über die Schnittstellen *I1WSDL* und *I1WSDL* für alle integrierten TE_x-Instanzen interpretierbare Policies bereit und repräsentiert mit gespeicherten Fakten den aktuellen System- und Sicherheitszustand der Referenzarchitektur.

User-Device

Der Cloud-Anwender-Arbeitsplatz wird über die Komponente *User-Device* abgebildet. Das Interface *IPolAtt* dient als Schnittstelle für das Policy-Management und Spezifikation eines Domain- und Sicherheitsmodells. Die Ermittlung attestierter Resultate eines Policy-Bereiches erfolgt auf Grundlage der *ObjectLogic*-eigenen Abfragesprache (Queries).

Die Komponente *TEU* (Trusted Entity User) steuert den Protokollaustausch mit dem Server-Referenzsystem zur Herstellung einer Vertrauensgrundlage (siehe Abschnitt 4.3.2.1) und wird in der weiteren Beschreibung als TE_U-Instanz bezeichnet.

5.1.4 Externe Systeme – Key & CA Service

Der *Key & CA Service* ist kein direkter Bestandteil der IT-Client-System-Architektur, sondern wird als externe Infrastrukturkomponente bereitgestellt.
Für den Prozess der Validierung wurde das Modell einer hierarchischen Trusted-Domain-Authority entwickelt (siehe Abbildung 5.2). Die Implementierung einer X.509-konformen Public-Key-Infrastruktur stützt sich auf das OpenSSL-Framework [Ope16; Ris13] und repräsentiert die technische Trusted-Domain-Authority.

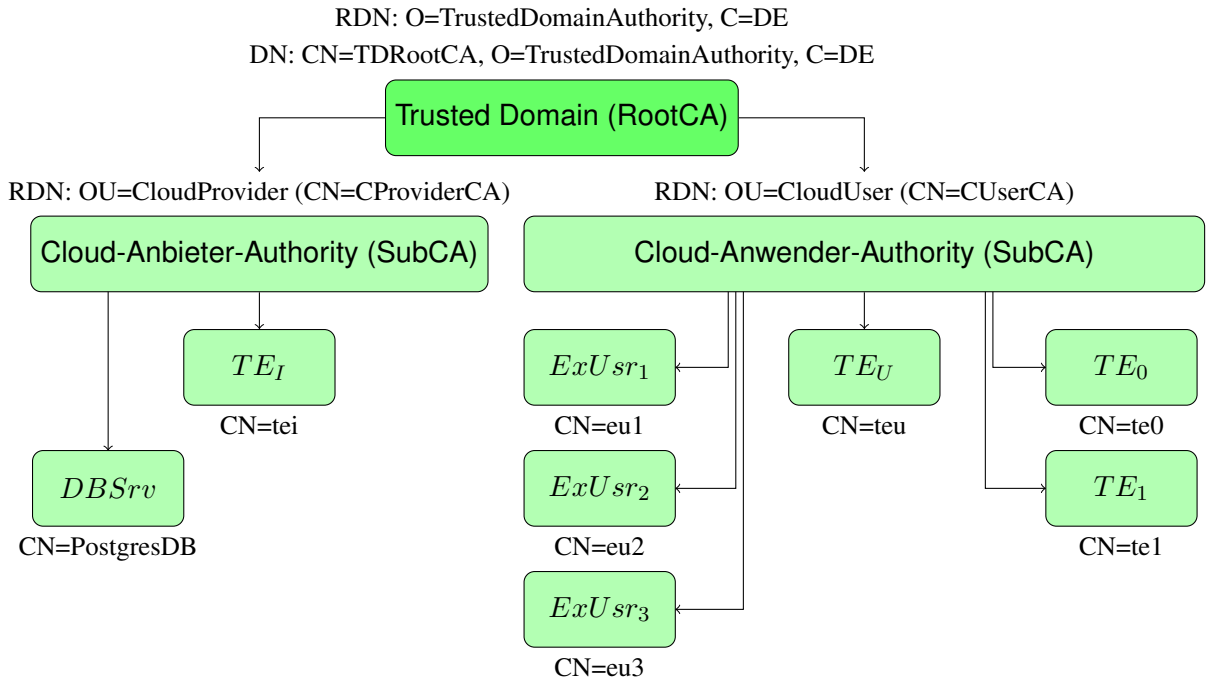

Abbildung 5.2: CA-Hierarchy – Validierung

Ein selbstsigniertes Zertifikat in Listing 5.1 für eine *Certificate-Authority* (CA) bildet den Ausgangspunkt der Hierarchy, nachfolgend als *RootCA* bezeichnet. Zur Repräsentation verteilter Verantwortlichkeiten wird unterhalb der *RootCA*-Ebene, für Cloud-Anwender und Cloud-Anbieter getrennt, eine eigene SubCA-Hierarchiestufe eingeführt. Die Herausgabe und Benennung von X.509-Zertifikaten folgt einem standardisierten Framework für das Management von Public-Key- und Attribut-Zertifikaten [ISO14].
Ein *Distinguished Name* (DN) [SKi95] identifiziert eine Hierarchiestufe eindeutig. Der *Common Name* (CN) beschreibt den Namen der mit dem Zertifikat verbundenen Entität und ist das bestimmende Element im Prozess der sicheren Authentifizierung. Aus Übersichtsgründen wird in Abbildung 5.2 nur für die *RootCA*-Ebene der DN angegeben. Der *Relative Distinguished Name* (RDN) kennzeichnet eine Ebene der Hierarchy. Der vollständige DN bildet sich aus dem CN und den RDN-Anteilen der einzelnen Ebenen.
Als Vorbedingung liegen für alle in Abbildung 5.1 eingeführten Entitäten kryptographische Schlüsselpaare und durch die zuständige *SubCA* herausgegebene X.509-Zertifikate vor.

Listing 5.1: Auszug aus X.509 RootCA-Zertifikat

```
Certificate:
  Data:
      Version: 3 (0x2)
      Serial Number:
          9f:19:3d:01:16:2e:4d:6d
  Signature Algorithm: sha256WithRSAEncryption
      Issuer: C = DE, O = TrustedDomainAuthority, CN = TDRootCA, \\
        emailAddress = support@TDAuthority.de
      Validity
          Not Before: Mar 29 08:23:53 2017 GMT
          Not After : Mar 24 08:23:53 2037 GMT
      Subject: C = DE, O = TrustedDomainAuthority, CN = TDRootCA, \\
        emailAddress = support@TDAuthority.de
      Subject Public Key Info:
          Public Key Algorithm: rsaEncryption
              Public-Key: (4096 bit)
          X509v3 Basic Constraints: critical
              CA:TRUE
          X509v3 Key Usage: critical
              Digital Signature, Certificate Sign, CRL Sign
  Signature Algorithm: sha256WithRSAEncryption
```

Listing 5.2: Auszug aus X.509 Cloud-Benutzer-Zertifikat

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4103 (0x1007)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = DE, O = TrustedDomainAuthority, OU = CloudUser, \\
        CN = CUserCA, emailAddress = support@TDCUser.de
        Validity
            Not Before: Mar 29 13:55:55 2017 GMT
            Not After : Mar 29 13:55:55 2019 GMT
        Subject: C = DE, O = TrustedDomainAuthority, OU = CloudUser, \\
        CN = ExternUser1, emailAddress = externuser1@clouduser.de
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, E-mail Protection
    Signature Algorithm: sha256WithRSAEncryption
```

Die Entitäten TE_0 und TE_1 im Bereich der SubCA-Hierarchie des Cloud-Anwenders repräsentieren in den nachfolgenden Szenarien technische Komponenten, die auf die Seite des Cloud-Anbieters übertragen und ausgeführt werden. Innerhalb der SubCA-Hierarchie stehen X.509-Zertifikate für technische Komponenten zu Verfügung. X.509-User-Zertifikate (siehe Listing 5.2) repräsentieren natürliche Personen. Die Bezeichnung $ExUsr_x$. kennzeichnet einen externen $Cloud\text{-}Anwender_x$.

5.1.4.1 Bezeichnungen und Namespaces

Namespaces für die Referenzierung der Ontologien:

Namespace	Abkürzung	Ontologie
http://www.trustedcloud.org/clouddomain#	nsd#	Ontologie-*Cloud-Domain*
http://www.trustedcloud.org/regulation#	nsr#	Ontologie-*Regulation*
http://www.trustedcloud.org/security#	nss#	Ontologie-*Security*

Tabelle 5.2: Übersicht Namespace-Definitionen

Präfixe zur Klassifizierung von Policies, Rules, Transformationen und Objekt-Instanzen:

Präfix	Bezeichnung
V_PD_	Validierung-Policy-Deployment
V_PS_	Validierung-Policy-Security
V_PT_	Validierung-Policy-Trust
V_RD_	Validierung-Rule-Deployment
V_RS_	Validierung-Rule-Security
V_RT_	Validierung-Rule-Trust
V_T_	Validierung-Transformation
V_QPS_	Validierung-QualityProperty-Security
V_D_	Validierung-Instanz-Ontologie-*Cloud-Domain*

Tabelle 5.3: Übersicht Präfix-Definitionen

Ein Storage stellt Artefakte für die Policy-Ausführung und die technische Transformation zur Verfügung. Transformationsregeln referenzieren auf folgende Speicherbereiche:

Pfad	Speicherbereich
/opt/images	Storage für VM-Images (Virtuelle Maschinen im ISO-Image-Format [ISO88])
/opt/deps	Storage für Packages (Linux Standardpackages im dep-Format)
/opt/template	Storage für Templates (Vorlagen für die automatisierte Konfiguration)
/opt/credentials	Storage für Zertifikate und kryptographische Schlüssel (Keys)

Tabelle 5.4: Übersicht Storage-Bereiche

5.1.4.2 TE-Zustandsmodell

Eine strukturelle Erweiterung der Konzeptklasse *State* in Listing 4.12 durch die Subklasse *StateTE-P* ermöglicht die spezifische Verwaltung von Zuständen für TE_x-Instanzen. Jede TE_x-Instanz besitzt über die Methode *hasSystemState* der Konzeptklasse *Object* in Listing 4.24 eine eigene Status-Instanz.

Listing 5.3: TE-P Subklasse der Konzeptklasse *State* – Subclass-F-atoms

```
1 StateTE-P:: State .
```

Das Management von Zuständen für TE_x-Instanzen erfordert ein Statusmodell. Die Ontologie-*Cloud-Domain* bietet dafür eine Konzeptklasse *StateData* als Subklasse der Konzeptklasse von *DataCode* an (siehe Listing 4.26). Instanzen der Konzeptklasse *StateData* beschreiben Zustandswerte als Teil der TE-Statuspezifikation und kennzeichnen Ablaufbedingungen im Prozess der vertikalen Etablierung von Vertrauen im Abschnitt 4.3.3.

Zustand	Beschreibung
defined:StateData	Instanz spezifiziert und nicht aktiv
authenticated:StateData	Instanz authentifiziert – überprüft Vorbedingungen
registered-compliant:StateData	Instanz registriert – alle Policies erfüllt
registered-non-compliant:StateData	Instanz registriert – einige Policies nicht erfüllt
halted:StateData	Instanz inaktiv
revoked:StateData	Instanz gesperrt

Tabelle 5.5: Statusmodell – TE_x-Instanz

5.1.4.3 Policy-Zonen und Policy-Anwendungsraum

Das Konzept von Policy-Zonen wurde als Teil der Ontologie-*Regulation* in Abschnitt 4.1.3.5 eingeführt und strukturiert eine Cloud-Architektur in Policy-Zonen. Die Verwaltung von Policy-Zonen ist Aufgabe von TE_x-Instanzen. Die Referenzarchitektur unterteilt sich in folgende Policy-Zonen:

Architekturebene	Policy-Zone	Domain-Objekt	Entität
V_Ref.Platform-Layer	Z_IT-Platform	V_D_Platform-System	TE_I
V_Ref.Virtualisation-Layer	Z_VIRT	V_D_Host-OS-Debian	TE_0
V_Ref.Runtime-Layer	Z_RUNTIME.DB	V_D_Virtual-OS-Debian	TE_1

Tabelle 5.6: Policy-Zonen für Referenzarchitektur

Für jede Instanz einer *ArchitecturalLayer* können ein oder mehrere Policy-Zonen existieren. Die lauffähige Instanz einer *PolicyZone* beschreibt den konkreten Policy-Anwendungsraum (*ZoneEnvironment*) einer TE_x-Instanz. Dieser kann sich auf Firmware, Betriebssystem, virtuelles Betriebssystem (VM), Anwendungs-Container [Tur16] oder konkrete Services der *ServiceLayer* (nicht aufgeführt in Tabelle 5.6) beziehen.
Die Zuordnung von Policy-Zonen gestaltet sich dynamisch als Teil des Startprozesses von TE_x-Instanzen.

TE-Komponente benötigen dafür eine eindeutige TE-Identität und Parameter für die Ontologie-Integration (siehe Listing 5.4).

Listing 5.4: TE0-Konfiguration

```
# name of the trustedEntity-instance in the ontology
agentName = V_TE0
# ansible directory
ansibleDir = /etc/ansible
# enable debug mode
debug = true
# set execution interval
executionInterval = 120000
# networkaddress of OntoBroker
ontobrokerAddress = 192.168.100.10:8267
# OntoBroker user
#ontobrokerUser = testuser
# OntoBroker password
#ontobrokerPassword = testpassword
```

Eine TE_x-Instanz ermittelt ihren Policy-Bereich durch Anwendung der in Listing 5.5 aufgeführten ObjectLogic-Query.

Listing 5.5: TE – *ObjectLogic*-Query zur Ermittlung zugeordneter Policies

```
options[outorder(?X)]}
?-
?X:Policy[targetToZone -> ?Y:PolicyZone[assignedTE->" + uri + "]].
```

5.2 Trust-Policies und Transformation

Die nachfolgenden Szenarien verwenden Policies für Trust, Deployment und Sicherheit im Infrastruktur- und Plattform-Architekturbereich der Referenzarchitektur. Die Policyausführung erfolgt unter Anwendung des Prinzips vertrauenswürdiger Entitäten und legt überprüfbare Nachweise vor.

Referenz	**Beschreibung der Policies**
Vorbedingung	Entitäten für Cloud-Anwender und Cloud-Anbieter sind administrativ jeweils einer eigenständigen *Authority* untergeordnet. Als übergreifende Vertrauensbeziehung existiert zwischen beiden Parteien ein gemeinsames Root-Zertifikat . (In einer realen organisatorischen Verteilung entwickeln sich Vertrauensbeziehungen über den Prozess einer Cross-Zertifizierung.) Der Cloud-Anwender wählt nach Überprüfung zugesicherter infrastruktureller Eigenschaften eine Cloud-Plattform aus. Mit der technischen Übertragung der TE_0-Instanz besteht eine horizontale Erweiterung seiner Vertrauensbasis. Das *Host-Operating-System* repräsentiert die Policy-Zone *V_Ref.Virtualisation-Layer*.

Fortsetzung auf der nächsten Seite

... von vorheriger Seite fortgesetzt

Referenz	Beschreibung der Policies
Policy: (1)	**Bereitstellung Ressource – Virtual Machine Monitor (KVM)** Die Policy überprüft die Vertrauenswürdigkeit einer KVM-Entität (KVM-Package). Nach erfolgreicher Überprüfung der KVM-Entität wird diese auf Basis einer Deployment-Policy installiert und konfiguriert.
Policy: (2)	**Bereitstellung Ressource – Virtualisierte Betriebssystemumgebung** Als Vorbedingung wird die Vertrauenswürdigkeit der Entität *Virtual-Operating-System* überprüft. Die Policy regelt die sichere Zuordnung der Entität zu einer *Virtual-Operating-System*-Identität durch Anwendung eines Hash-*Credential*.
Policy: (3)	**Bereitstellung Ressource – Datenbanksystem** Die Policy regelt ein konformes Deployment und die Einrichtung eines *Postgres*-Datenbank-Systems und überprüft dessen Vertrauenswürdigkeit. Als Vorbedingung einer nachgelagerten Sicherheitsstrategie werden für unterschiedliche Benutzerkreise unterschiedliche Datenbanken eingerichtet und repräsentieren separierte Vertraulichkeitsbereiche.
Policy: (4)	**Externe Zugangssteuerung – Datenbanksystem** Die Policy regelt die Absicherung einer externen Netzwerkverbindung über ein TLS-Sicherheitsprotokoll und steuert den Zugriff von externen Benutzern auf Tabellen des Datenbanksystems. Die Absicherung einer Netzwerkverbindung (*IConnection*) in Abbildung 5.1 folgt der Konzeptualisierung von *Connection* in Abschnitt 4.1.4.7. Folgende Sicherheitsfunktionen werden eingerichtet: • Bindung des TLS-Protokolls an Netzwerkkarte • Einrichtung einer TLS-basierten Server-Authentifizierung
Nachbedingung	**Vertrauensgrundlage des Cloud-Anwenders hergestellt** Der Cloud-Anwender besitzt vertrauenswürdige Entitäten, um die bestehende Vertrauensgrundlage auf die Seite des Cloud-Anbieters erweitern zu können. Der Cloud-Anwender besitzt die Zusicherung, dass eine vertrauenswürdige Virtualisierungsschicht und Laufzeitschicht für die kontrollierte Anwendung eines Datenbanksystems auf der Seite des Cloud-Anbieters vorliegen. Die Zugriffe auf das Datenbanksystem sind über eine definierte Datenbankverbindung nur für einen festgelegten Cloud-Anwenderkreis möglich. Die Datenbankverbindung unterliegt Vertraulichkeitsanforderungen zur Anwendung separierter Datenbanken für jeden einzelnen Cloud-Anwender.

Tabelle 5.7: Ablaufbeschreibung – Auswahl Cloud-Plattform

5.2.1 Szenario (1) – Bereitstellung Virtual Machine Monitor KVM

Das Szenario beschreibt eine Sicherheitsstrategie, bei der ausschließlich vertrauenswürdige Software für die Virtualisierung (VMM) in eine bestehende Infrastruktur integriert wird. Die Auswahl und das Deployment der Virtualisierungssoftware KVM ist Teil eines Policy-Konzepts. Der Regulierungsbereich bezieht sich auf Festlegungen zur Anwendung von Virtualisierungs-Technologien (z. B. Auswahl von VMWare vSphere [VMw16], VirtualBox[Ora16], Hyper-V [Mir16]) bzw. auf Festlegungen zu sicherheitsüberprüften Versionen einer bestehenden Virtualisierungstechnologie.

5.2.1.1 Domain-Spezifikation – KVM-Komponente

Trust-Policies überprüfen und bestätigen erwartete Qualitätsanforderungen. Die Definition von sicherheitsrelevanten Qualitätseigenschaften sind Spezifikationen der Ontologie-*Security*.
Die KVM-Spezifikation ist als Instanz der Konzeptklasse *Part* (siehe Abschnitt 4.1.4.6) in Listing 5.6 herausgearbeitet. Die Komponente steht als Software-Package (3) im Storagebereich zur Verfügung.
QualityProperties sind direkt mit dem Domain-Objekt verknüpft und beschreiben qualifizierende Eigenschaften der entsprechenden Instanz. Die Eigenschaft *V_QPS_authentic.Hypervisor* (5) fordert den Nachweis einer ausgewiesenen Authentizität der KVM-Entität.

Listing 5.6: *Module::OperatingSystemCode*: V_D_KVMSoftware

```
1 V_D_KVMSoftwarePackage[hasFileName->quemu-kvm.deb].
2 V_D_KVMSoftwarePackage[hasFileDir->V_D_softwarePath].
3 V_D_KVMSoftware[hasContainer->V_D_KVMSoftwarePackage].
4 V_D_KVMSoftware[authorizedTo(<nss#V_C_TicketHashKVM>)-><nss#V_I_QemuKvm>].
5 V_D_KVMSoftware[checkQP-><nss#V_QPS_authentic.Hypervisor>].
```

5.2.1.2 Regulation-Spezifikation – KVM-Deployment-Policy

Die Policy-Zone für die Virtualisierungsebene (1) der Referenzarchitektur definiert mit Listing 5.7 eine Deployment-Policy, die aus mehreren Deployment-Rules (4) – (5) besteht. Im Fall, dass mehr als eine Rule-Definition vorliegt, bestimmt die Methode *firstPlatformRule* (2) den Anfang des Workflows für die Rule-Ausführung. Die TE_0-Instanz ermittelt über die in Listing 5.5 beschriebene ObjectLogic-Query die Menge der auszuführenden Policies.

Listing 5.7: *PolicyDeployment::Policy*: V_PD_setup.Kvm

```
1       V_PD_setup.Kvm[targetToZone->Z_VIRT].
2       V_PD_setup.Kvm[firstPlatformRule->V_RD_install.Kvm].
3
4       V_PD_setup.Kvm[hasPlatformRule->V_RD_install.Kvm].
5       V_PD_setup.Kvm[hasPlatformRule->V_RD_install.VirtInst].
6       V_PD_setup.Kvm[dependOn->V_PS_identify.platform.components].
7       V_PD_setup.Kvm[reasonPolicyState->StD_Running.kvm].
```

Als notwendige Vorbedingung für die Ausführung der Deployment-Policy ist eine erfolgreich ausgeführte Sicherheits-Policy (6) gesetzt. Die Sicherheitspolicy zur Überprüfung geforderter Plattformeigenschaften liegt im Zuständigkeitsbereich der Policy-Zone *Z_IT-Platform* (siehe Tabelle 5.6) und wurde

bereits durch die TE_I-Instanz ausgeführt. Listing 5.8 enthält einen Auszug von Sicherheits-Rules der Policy *V_PS_identify.platform-components*.

Listing 5.8: *PolicySecurity::Policy*: V_PS_identify.platform-components

```
1  V_PS_verify.Plat-Prop[hasRule->RS_verify.Chipset-State].
2  V_PS_verify.Plat-Prop[hasRule->RS_verify.Bios-State].
3  V_PS_verify.Plat-Prop[hasRule->RS_verify.Cpu-State].
4  V_PS_verify.Plat-Prop[targetToZone->Z_IT-Platform].
```

Die TE_0-Instanz ermittelt gesetzte Vorbedingungen durch Anwendung der in Listing 5.9 entwickelten ObjectLogic-Query.

Listing 5.9: TE – Query zur Ermittlung einer bestehenden Policy-Vorbedingung

```
1  ?- ?X:Policy                    AND
2     ?X[dependOn -> ?P]           AND
3     ?P[resultedState -> ?S].
```

Die Deployment-Rule zur Installation von KVM in Listing 5.10 beschreibt mit Anweisung (5) die Operation und und referenziert auf die Objekt-Instanz der KVM-Entität *V_D_KVMSoftware* in der Ontologie-*Cloud-Domain*.

Listing 5.10: *DeploymentRule::Rule* V_RD_install.Kvm

```
1  V_RD_install.Kvm[expectedState->V_installed.kvm].
2
3  V_RD_install.Kvm[nextRule->V_RD_install.VirtInst].
4  V_RD_install.Kvm[transformedBy->V_T_install.Debs].
5  V_RD_install.Kvm[do(V_ACT_KvmInstall)-><nsd#V_D_KVMSoftware>].
```

Die Rule-Definition setzt den erwarteten Statuswert (1), der nach erfolgreicher Ausführung nachgewiesen werden kann und definiert Vorgaben zur Transformation (4). Statuswerte sind Instanzen der Konzeptklasse *State* und enthalten planbare Größe einer Betriebs- und Sicherheitsstrategie.
Die Ausführung einer Deployment-Rule ist an eine Überprüfung von geforderten *QualityProperties* gekoppelt. Die TE_0-Instanz nutzt dafür die in Listing 4.52 spezifizierte *Reasoning-Rule*.

5.2.1.3 Prüfung der KVM-Authentizität

V_QPS_authentic.Hypervisor fordert in Listing 5.6 einen Authentizitätsnachweis für die KVM-Entität. Die zugehörige Trust-Policy leitet sich für die gesetzte Qualitätseigenschaft direkt über die Methode *enforcedThrough* in Listing 5.11 ab.

Listing 5.11: *PropertyTrust::QualityProperty*: V_QPS_authentic.Hypervisor

```
1  V_QPS_authentic.Hypervisor:QualityProperty.
2  V_QPS_authentic.Hypervisor[enforcedThrough-><nsr#V_PT_verify.Hypervisor>].
```

Die Trust-Policy umfasst in Listing 5.12 nur eine Trust-Rule (4). Grundsätzlich können weitere qualitative Überprüfungen von Entitäten aufgenommen werden, die in direkter Abhängigkeit zum Domain-Object stehen. Das Konzept sieht vor, Qualitätsmerkmale für jedes Domain-Objekt innerhalb einer Rule zu fordern. Daraus lässt sich später ein Maß an Vertrauenswürdigkeit spezifisch für jede Entität ableiten.

Listing 5.12: *TrustPolicy::Policy*: V_PT_verify.Hypervisor

```
1  V_PT_verify.Hypervisor[targetToZone->Z_VIRT].
2  V_PT_verify.Hypervisor[firstTrustRule->V_RT_verifyKvm].
3
4  V_PT_verify.Hypervisor[hasTrustRule->V_RT_verifyKvm].
```

Listing 5.13 beschreibt mit *verifyAuth* (4) die Trust-Operation für das zugehörige Domain-Objekt und legt mit den Anweisungen (2) und (3) Transformationsregeln fest. Als Grundlage für die sichere Zuordnung zu einer KVM-Identität (*V_I_QemuKvm*) dient ein Hash-Ticket (1), eine Instanz der Konzeptklasse *Credential*.

Listing 5.13: *TrustRule::Rule*: V_RT_verify.Kvm

```
1  V_RT_verify.Kvm[assuredBy -> <nss#V_C_TicketHashKVM>].
2  V_RT_verify.Kvm[transformedBy->V_T_verify.Md5Hash].
3  V_RT_verify.Kvm[transformedBy->V_T_get.FileData].
4  V_RT_verify.Kvm[do(verifyAuth)-><nsd#V_D_KVMSoftware>].
```

Die Ausführung von Trust-Policies erfolgt nach diesem Prinzip separiert und vorgelagert zum anstehenden Deployment-Prozess.

5.2.1.4 Zusicherung von KVM-Identitätseigenschaften

Das Kernkonzept für die sichere Zuordnung von Entitäten zu Instanzen der Konzeptklasse *Identity* in Abschnitt 4.1.5.7 erhält in Listing 5.6 über die Anwendung der Konzeptklasse *Credential* eine konkrete Realisierung. Das Hash-Ticket in Listing 5.14 bestätigt überprüfte Eigenschaften der KVM-Entität.

Listing 5.14: *TicketHash::Credential*: V_C_TicketHashKVM

```
1  V_C_TicketHashKVM[verifiedPropertySec(Hash)->"02e2378edffc0a..."].
2  V_C_TicketHashKVM[verifiedPropertyInf(Name)->"KVM-Hypervisor"].
3  V_C_TicketHashKVM[verificationMethod->V_MD5Hash].
4  V_C_TicketHashKVM[verifiedPropertyInf(SerialNumber)->"AZX500-34"].
5  V_C_TicketHashKVM[confirmedBy-><nsd#Sachsen-Provider.Authority>].
6  V_C_TicketHashKVM[verifiedIdentity->V_I_QemuKvm].
```

Die TE_0-Instanz berechnet einen Hash-Wert für das KVM-Software-Package und führt einen Vergleich mit den kryptographischen Vorgaben (1) durch. Das kryptographische Berechnungsverfahren liegt mit Anweisung (3) vor. Im Fall der Übereinstimmung generiert die TE_0-Instanz eine Identität *V_I_QemuKvm* (siehe Listing 5.15) und stellt über die Methode *authorizedTo* in Listing 5.6 eine vertrauenswürdige Bindung zur KVM-Entität sicher.

Listing 5.15: *Identity::SecurityData*: V_I_QemuKvm

```
1  V_I_QemuKvm[securityAttributes(HASH)->"02e2378edffc0a0e1244fb342688dab0"].
2  V_I_QemuKvm[verifiedPropertyInf(Name)->"KVM-Hypervisor"].
```

5.2.1.5 Transformation – KVM-Trust-Rule

Die Trust-Rule-Ausführung ist mit einer technischen Transformation verbunden. Die TE_0-Instanz verwendet dafür die Schnittstelle *ITrans*. Die Anweisungen (3) und (4) in Listing 5.13 verweisen auf Instanzen der Konzeptklasse *Transformation*, die in folgenden Schritten ausgeführt werden:

KVM-Entität als Objekt laden und Hash-Wert bilden

Listing 5.18 beschreibt die Transformationsregel *V_T_getFileData:ansible*, eine Instanz der Konzeptklasse *ansible*. Technische Transformationen werden über ansible-Module gesteuert, die für unterschiedliche IT-Architekturbereiche nachnutzbar vorliegen [Red17a]. Über eine individuelle Modulentwicklung lassen sich komplexere Integrationsanforderungen erfüllen.

Im ersten Transformationsschritt beschreibt das ansible-*stat*-Modul (5) den Ladevorgang für das Softwarepackage der KVM-Entität. Ansible-Transformationsregeln referenzieren über die Angabe von Methoden auf Fakten einer zugewiesenen Ontologie-Instanz. Die Methode *setParameter* (3) referenziert über die Methode *hasContainer* auf das Domain-Object in Listing 5.6. Die Anweisung (1) beschreibt den Softwarepackage-Filename und wird in Listing 5.18 zu einem konkreten technischen Zugriffspfad (*path*) transformiert.

Listing 5.16: *ansible::Transformation*: V_T_get.FileData>

```
1  V_T_get.FileData[additionalParameters(register)->part].
2  V_T_get.FileData[setRealParameter->path].
3  V_T_get.FileData[hasParameter(1)->hasContainer].
4  V_T_get.FileData[hasParameter(2)->authorizedTo].
5  V_T_get.FileData[hasModule->stat].
```

Das ansible-*stat*-Modul [Red17b] berechnet für das Softwarepackage eine Hash-Summe. Über die Verwendung der Modul-Option erfolgt die Auswahl der Hash-Funktion. Die Methode *setOption* (5) referenziert auf das Domain-Object in Listing 5.6. Darüber kann das zugeordnete Hash-Ticket mit Angaben über die geforderte Hash-Funktion gelesen werden. Der berechnete Hash-Wert für die KVM-Entität wird mit Anweisung (1) registriert und für die folgende Transformationsregel bereitgestellt.

Identität der KVM-Entität überprüfen

In Listing 5.17 sind die Regeln für die zweite Transformation beschrieben. Es wird dafür das ansible-*fail*-Modul verwendet (3). Das Modul führt einen kryptographischen Vergleich zwischen den im Register erfassten Hash-Wert (*part.stat.md5*) und einem Wert, der über die Methode *authorizedTo* (2) referenziert wird.

Listing 5.17: *ansible::Transformation*: V_T_verify.Md5Hash>

```
1  V_T_verify.Md5Hash[setRealParameter->msg].
2  V_T_verify.Md5Hash[conditions('part.stat.md5')->authorizedTo].
3  V_T_verify.Md5Hash[hasModule->fail].
```

Die Methode *authorizedTo* referenziert auf die zugeordnete Identität und das sicherheitsrelevante Attribut *HASH* kann für die Vergleichsoperation bestimmt werden. Die Methode *setRealParameter* dient der internen Konfiguration von Rückgabewerten.

5.2.1.6 Transformation – KVM-Deployment-Rule

Nach erfolgreicher Ausführung der KVM-Trust-Policy wird der *expectedState* in Listing 5.13 mit *verifiedKVM* gekennzeichnet. Die notwendige Vertrauensgrundlage ist definiert.
Die Transformationsregel (4) in Listing 5.10 setzt das Deployment mit der Installation des KVM-Software-Packages um.

Listing 5.18: *ansible::Transformation*: V_T_install.Debs>

```
1  V_T_install.Debs:ansible.
2  V_T_install.Debs[hasModule->apt].
3  V_T_install.Debs[hasParameter->hasContainer].
4  V_T_install.Debs[setRealParameter->deb].
```

Die Transformationsregel stellt mit *hasContainer* (3) Informationen bereit, um das KVM-Software-Package über den *Storage-Service* als Instanz der Konzeptklasse *FILE* zu lokalisieren. Unter Anwendung des *apt*-Moduls (2) wird eine automatische Package-Installation ausgeführt.
Der in Listing 5.10 gekennzeichnete *expectedState* (1) für die Deployment-Policy ist erfüllt, wenn die Installation und Konfiguration für das vertrauenswürdige KVM-Software-Package erfolgreich abgeschlossen wurde. Es ist Aufgabe der TE_0-Instanz, den technischen Status zu überprüfen und als Statusfakt auf Ebene der Policy in der Ontologie-*Regulation* zu bestätigen.

5.2.2 Szenario (2) – Bereitstellung Virtualisiertes Betriebssystem

Das Szenario beschreibt das Deployment einer *Virtual-OS*-Entität in Form einer virtuellen Maschine (VM) und erweitert nach Tabelle 5.6 mit dem Start einer VM die bestehende Vertrauenshierarchie. Ausgehend von der Policy-Zone *Z_VIRT*, die sich im Zuständigkeitsbereich der TE_0-Instanz befindet, steuert eine Policy die Erweiterung der bestehenden Vertrauensgrundlage und eröffnet eine darauf aufbauende neue Policy-Zone *Z_RUNTIME.DB*. Eine installierte und lauffähige TE_1-Instanz repräsentiert den erweiterten hoheitlichen Einflussbereich des Cloud-Anwenders.

5.2.2.1 Domain-Spezifikation – Virtual-OS

Listing 5.19 spezifiziert eine *Virtual-OS*-Entität als Instanz der Konzeptklasse *Part*.

Listing 5.19: *VirtualOperatingSystem::OSPart::Software::Part*: V_D_Virtual-OS

```
1  V_D_Virtual-OS[hasSystemName->VM-Debian].
2  V_D_Virtual-OS[hasSoftComponent->V-LAN01].
3  V_D_Virtual-OS[hasSoftComponent->V-Storage].
4  V_D_Virtual-OS[hasSoftComponent
5                 ->Debian-Linux8(Jessie) Server 64Bit].
6  V_D_Virtual-OS[authorizedTo(<nss#TicketHashVirtualOS>)
7                 -><nss#V-I_Vbox.debian>].
8  V_D_Virtual-OS[hasContainer->V_I_VirtualOSImage].
9
10 V_D_Virtual-OS[checkQP-><nss#V_QPS_authentic.VM1>].
11 V_D_Virtual-OS[hasSoftComponent->TE1-Component].
12 V_D_Virtual-OS[isConnectedTo->User-to-VM-Connection].
```

Das Debian-Linux-Image (8) konfiguriert einen virtuellen Speicher (3) und eine virtuelle LAN-Verbindung (2). Die Eigenschaft *V_QPS_authentic.VM1* (10) fordert den Nachweis einer ausgewiesenen Authentizität der *Virtual-OS*-Software.
Für die Cloud-Anwender-System-Integration beschreibt eine Instanz (12) der Konzeptklasse *Connection* geforderte Verbindungseigenschaften bezüglich der Interaktion mit dem virtuellen Betriebssystem. Die Szenariobeschreibung in Abschnitt 5.2.4 stellt ein Policykonzept zur Regulierung interaktiver Verbindungen vor.

5.2.2.2 Regulation-Spezifikation – Virtual-OS-Deployment-Policy

Die Deployment-Policy in Listing 5.20 ist für die Virtualisierungsebene (1) der Referenzarchitektur festgelegt. Die TE_0-Instanz ermittelt über die in Listing 5.5 beschriebene ObjectLogic-Query die Menge der auszuführenden Policies.

Listing 5.20: *PolicyDeployment::Policy*: V_PD_setup.Virtual-OS

```
1  V_PD_setup.Virtual-OS[targetToZone->Z_VIRT].
2  V_PD_setup.Virtual-OS[firstPlatformRule->RD_install.VM].
3  V_PD_setup.Virtual-OS[reasonPolicyState->StD_Running.vm].
4
5  V_PD_setup.Virtual-OS[hasPlatformRule->RD_install.VM].
6  V_PD_setup.Virtual-OS[targetToZone->Z_VIRT].
7
8  V_PD_setup.Virtual-OS[dependOn->PD_setup.KVM].
```

Die Überprüfung der Virtual-OS-Authentizität entspricht der beschriebenen Vorgehensweise im Abschnitt 5.2.1.3. Für die sichere Zuordnung zu einer Virtual-OS-Identität (*V_I_Vbox.debian*) dient in Listing 5.19 ein Hash-Ticket (6), eine Instanz der Konzeptklasse *Credential*.
Die technische Transformation erfolgt gemäß des in Abschnitt 5.2.1.5 dargestellten Ablaufs und steuert nach Überprüfung der Vertrauenswürdigkeit der Package-Software *V_I_VirtualOSImage* das technische Deployment.

5.2.2.3 Prüfung der TE-Authentizität

Die Überprüfung der Vertrauenswürdigkeit der TE_1-Entität ist Teil des Deployments und der vorgelagerten Trust-Politik im Abschnitt 5.2.2.1. Der Nachweis der Vertrauenswürdigkeit der *Virtual-OS*-Entität ist gleichzeitig mit der Zusicherung der Vertrauenswürdigkeit der TE_1-Entität verbunden.
Eine explizite und separate Prüfung der Vertrauenswürdigkeit der TE_1-Entität ist möglich. Für diesen Fall stehen sowohl die *Virtual-OS*-Entität als auch die TE_1-Entität als eigenständige überprüfbare Entitäten zur Verfügung. Es ist Aufgabe der TE_0-Instanz, das Deployment beider Entitäten auszuführen und steuert die Aktivierung der TE_1-Instanz auf Grundlage einer TE-Transformation.
Im Szenario aktiviert (siehe Abbildung 4.10 – *Activate TE-P*) und initialisiert der Startprozess der Instanz *V_D_Virtual-OS* automatisch die TE_1-Instanz. Eine Konfigurationsdatei nach dem Beispiel in Listing 5.4 stellt notwendige Identitäts-Parameter für die Initialisierung bereit.
Die TE_1-Instanz authentisiert sich gegenüber dem *Ontologie-Service* und der Status wird nach erfolgreicher Authentifizierung auf *authenticated* gesetzt (siehe Tabelle 5.5). Für die Überprüfung der Identität

von TE_x-Instanzen liegen *Certificate*-Credential für die Authentizität (*Auth-Credential*) und für die Ausführung von Handlungen (*Confirm-Credential*) vor (siehe Abbildung 4.4).
Das *Certificate*-Credential in Listing 5.21 ist ein Auszug und repräsentiert ein von der SubCA (*CProviderCA*) herausgegebenes X.509-Zertifikat zur Bestätigung der TE_1-Identität.

Listing 5.21: *Certificate::Credential*: V_C_Cert.Comp.X509.TE1-Auth

```
V_C_Cert.Comp.X509.TE1-Auth[verifiedPropertyInf(Name)->TE-1-Auth].
V_C_Cert.Comp.X509.TE1-Auth[verifiedPropertySec(DistinguishedName)\\
        ->C=DE, O=TrustedDomainAuthority,OU=CloudUser, CN=te1].
V_C_Cert.Comp.X509.TE1-Auth[<nsd#assignedIdentity>\\
        ->TE1-Identity.RuntimeOS].
```

Das *Certificate*-Credential in Listing 5.22 ist ein Auszug und repräsentiert ein von der SubCA (*CProviderCA*) herausgegebenes X.509-Zertifikat zur Bestätigung von TE_1-Aktivitäten.

Listing 5.22: *Certificate::Credential*: V_C_Cert.Comp.X509.TE1-Confirm

```
V_C_Cert.Comp.X509.TE1-Confirm[verifiedPropertyInf(Name)->TE-1-Confirm].
V_C_Cert.Comp.X509.TE1-Confirm[verifiedPropertySec(DistinguishedName)\\
        ->C=DE, O=TrustedDomainAuthority,OU=CloudUser, CN=te1conf].
V_C_Cert.Comp.X509.TE1-Confirm[<nsd#assignedIdentity>\\
        ->TE1-Identity.RuntimeOS].
```

Der *Ontologie-Service*, als Teil der Authority-Hierarchie in Abbildung 5.2, authentifiziert eine TE_1-Instanz auf Grundlage des im Listing 5.21 aufgeführten *Auth-Credential*. Nach erfolgreicher Überprüfung der Vertrauenswürdigkeit liegt eine Autorisierung für den Zugriff auf den Ontologiebereich vor. Die TE_1-Instanz ist in der Lage, alle Vorbereitungen für die Herstellung einer neuen Vertrauenshierarchie auszuführen.
Das Zusammenwirken der TE_1-Instanz mit dem *Ontologie-Service* führt zu Veränderungen von Statusinformationen und anderen Fakten der Ontologie. Die Verbindlichkeit von Status- und Faktenänderungen und damit die spätere Verifizierbarkeit ist unter Anwendung des im Listing 5.22 aufgeführten *Confirm-Credential* gewährleistet. Das Prinzip getrennter Identitäten für den Systemzugang und für die Ausführung verbindlicher Handlungsschritte sichert die Attestierung vertrauenswürdiger Fakten innerhalb der Ontologie.

5.2.2.4 Policy-Zone einrichten – *Z_RUNTIME.DB*

Die TE_1-Instanz bestimmt die ihr zugeordnete Policy-Zone durch Anwendung der in Listing 5.23 spezifizierten *Reasoning-Rule*. Sie stellt dafür die Methode *reasonZone* (2) bereit.

Listing 5.23: TE-Methode *reasonZone* – Reasoning-Rule

```
        @{Zone}
        ?T[reasonZone -> ?Z]
        :-
        ?T:<nsd#TE>                         AND
        ?Z:<nsr#PolicyZone>                 AND
        ?Z[<nsr#assignedEntity> -> ?T]      AND
```

Die *Reasoning-Rule* ermittelt alle Instanzen der Konzeptklasse *TE* (4) und Instanzen der Konzeptklasse *PolicyZone* (5). Die Methode *assignedEntity* (6) stellt die Zuordnung zu einer TE-Instanz her. Bei Übereinstimmung wird die Aussage im Header (2) erfüllt und die Variable $?Z$ enthält die Instanz der Konzeptklasse *PolicyZone* (Im Szenario: *Z_RUNTIME.DB*).
Für die Einordnung in die bestehende Vertrauenshierarchie ist die Kenntnis des gültigen Policy-Anwendungsraumes (*ZoneEnvironment*) erforderlich. Die *Reasoning-Rule* in Listing 5.24 ermittelt über die Methode *reasonZoneEnvironment* (2) den Policy-Anwendungsraum.

Listing 5.24: TE-Methode *reasonZoneEnvironment* – Reasoning-Rule

```
1  @{ZoneEnvironment}
2  ?T[reasonZoneEnvironment -> ?E]
3  :-
4  ?T:TE                                  AND
5  ?X:<nsr#PolicyZone>                    AND
6  ?X[<nsr#assignedEntity>->?T]           AND
7  ?X[<nsr#assignedLayer>->?A]            AND
8  ?A[representedThrough->?E].
```

Grundlage ist die Methode *assignedLayer* in Listing 4.9. Sie referenziert auf eine Architekturebene, deren Instanzen einen Policy-Anwendungsraum repräsentieren (im Szenario: *V_D_Virtual-OS-Debian*). Die Schlussfolgerung erfolgt über die Analyse aller Policy-Zonen (5) und einen Vergleich mit der suchenden TE-Instanz (6). Bei Überstimmung beschreibt $?A$ die Architekturebene (7) und die Instanz für den Policy-Anwendungsraum (8) steht als Wert in der Variablen $?E$ zur Verfügung.

5.2.2.5 Vertrauenskette prüfen – *Chain of Trust*

Nach Aktivierung der TE_1-Instanz wird die zugrundeliegende TE_x-Instanz als Vertrauensbasis bestimmt und überprüft (siehe Abbildung 4.10 – *Verify trust pre-condition*). Aus der Kenntnis der Policy-Zone, die für das Deployment der TE_1-Instanz zuständig war, leitet sich durch Anwendung der *Reasoning-Rule* in Listing 5.25 die zugehörige Vertrauensbasis ab. Sie stellt dafür die Methode *reasonTEP* bereit.

Listing 5.25: TE-Methode *reasonTEP* – Reasoning-Rule

```
1  @{HierarchyTE}
2  ?T[reasonTEP->?K]
3  :-
4  ?T:TE                                  AND
5  ?T[reasonZoneEnvironment->?E]          AND
6  ?P:<nsr#PolicyDeployment>              AND
7  ?P[<nsr#hasRule>->?R]                  AND
8  ?R[<nsr#do>(?A)->?E]                   AND
9  ?P[<nsr#targetToZone>->?Z]             AND
10 ?Z[<nsr#assignedEntity>->?K].
```

Die integrierte *Reasoning-Rule* in Listing 5.24 bestimmt den Policy-Anwendungsraum (5). Die Schlussfolgerung erfolgt über die Analyse der Deployment-Policies (6) und einen Vergleich der Ausführungs-

operation aller Deployment-Rules (7) mit dem aktuellen Policy-Anwendungsraum (8). Bei Übereinstimmung leitet sich aus der ermittelten Policy die dafür gültige Policy-Zone ab (9) und die TE_x-Instanz steht als Wert in der Variablen $?K$ zur Verfügung (Im Szenario: TE_0).
Es folgt die Statusüberprüfung der TE_0-Instanz durch Anwendung einer ObjectLogic-Query in Bezug auf ihre Vertrauenswürdigkeit. Nach Prüfung einer gültigen Vertrauensbasis aktiviert die TE_1-Instanz die neue Stufe der Vertrauenshierarchie und der Status wird auf *registered-compliant:StateData* gesetzt (siehe Tabelle 5.5).

Listing 5.26: Abfrage Status TE_0-Instanz – ObjectLogic-Query

```
1   ?-
2   ?T:TE                                   AND
3   ?T[hasSystemState->?S]                  AND
4   ?S[<nsr#hasStateValue>->?V]             AND
```

Nach Ermittlung der zugeordneten TE_0-Status-Instanz (3) liefert die Methode *hasStateValue* (4) den aktuellen Statuswert für die laufende TE_0-Instanz (Im Szenario: *registered-compliant*). Im Szenario wird auf die explizite Überprüfung der Verbindlichkeit vorliegender Statusinformationen verzichtet. Die Vertrauenswürdigkeit des *Ontologie-Service* als Teil der Authority-Hierarchy kann die Vertrauenswürdigkeit der Statusinformationen zusichern. Die Statusänderungen können jederzeit durch ein Konzept logischer Schlussfolgerungen auf die entsprechende vertrauenswürdige Entität zurückgeführt werden.

5.2.3 Szenario (3) – Bereitstellung Datenbanksystem (DBS)

Das Szenario beschreibt den Aufbau und die Absicherung eines Datenbanksystems (DBS) und wird dafür in ein Policy-Konzept eingebunden. Der Regulierungsbereich bezieht sich auf Festlegungen zur Nutzung und Absicherung von PostgreSQL. Die Vertrauenswürdigkeit des Datenbanksystems wird überprüft und nach erfolgreicher Bestätigung in die Referenz-Laufzeitumgebung übernommen. Eine Deployment-Policy steuert die Installation und Konfiguration.

5.2.3.1 Domain-Spezifikation – Datenbanksystem

Listing 5.27 spezifiziert ein Datenbanksystem als Instanz der Konzeptklasse *Part* (siehe Abschnitt 4.1.4.6).

Listing 5.27: *DatabaseSystem::Software::Part*: V_D_PostgreDBS

```
1   V_D_PostgreDBS[hasSystemName->DBS-PostgreSQL].
2   V_D_PostgreDBS[hasSoftComponent->Database.accounting].
3   V_D_PostgreDBS[hasSoftComponent->Database.article].
4   V_D_PostgreDBS[hasSoftComponent->Database.employee].
5
6   V_D_PostgreDBS[checkQP-><nss#V_QPS_authentic.dbspackage>].
7   V_D_PostgreDBS[authorizedTo(<nss#V_C_TicketHashDBS>)\\
8                                   -><nss#V_I_PostgresPackage>].
9   V_D_PostgreDBS[hasContainer->V_I_PostgresPackage].
10  V_D_PostgreDBS[runAsIdentity-><nss#V_I_PostgresServer>].
11
12  V_D_PostgreDBS[isConnectedTo->UsertoDB-Connect].
```

Die Anweisungen (2) – (4) beschreiben die Bestandteile des Datenbanksystems *DBS-Postgres*. Es besteht aus mehreren Datenbanken.
Als Beispiel werden in Listing 5.28 die fachlichen Tabellen der Datenbank *Database.accounting* aufgeführt. *ObjectLogic* besitzt die Fähigkeit, das einer Datenbank zugrundeliegende Konzept von *Relationen* [Ehr13, S. 29] abzubilden. Im Bereich der Ontologie-*Cloud-Domain* endet jedoch die Konzeptualisierung auf Ebene einer *Tabelle* und berücksichtigt keine Relationen-Spezifikation.

Listing 5.28: *DBTable::DatabaseObject*: Tabellen – Datenbank *Accounting*

```
1  Database.accounting[hasDatabaseObject->TDB_account.register].
2  Database.accounting[hasDatabaseObject->TDB_account.report].
3  Database.accounting[hasDatabaseObject->TDB_account.transfer].
```

Die Anweisungen (7) – (10) in Listing 5.27 dienen der Identitätsbestimmung und dem Nachweis der Vertrauenswürdigkeit. Wie bereits im Abschnitt 5.2.1.3 wird auch zur Bestimmung der Vertrauenswürdigkeit des Datenbanksystems ein Hash-Ticket verwendet.
Das Konzept einer *Connection* (12) regelt, welche Verbindungsart ausgewählt wird, wie viele Verbindungen zu einem Datenbanksystem funktional bestehen und welche sicherheitsrelevanten Eigenschaften diese Verbindungen besitzen (siehe Abschnitt 4.1.4.7).

5.2.3.2 Regulation-Spezifikation – DBS-Deployment-Policy

Die Deployment-Policy in Listing 5.29 ist für die Laufzeit-Architekturebene (1) festgelegt. Die ihr zugeordnete TE_1-Instanz wird über die in Listing 5.5 beschriebene ObjectLogic-Query ermittelt.
Die Deployment-Policy besteht aus mehreren Deployment-Rules (4) – (7). Im Fall, dass mehr als eine Rule-Definition vorliegt, bestimmt die Methode *firstPlatformRule* (2) den Anfang des Workflows für die Rule-Ausführung.

Listing 5.29: *PolicyDeployment::Policy*: V_PD_setup.DBS

```
1  V_PD_setup.DBS[targetToZone->Z_RUNTIME.DB].
2  V_PD_setup.DBS[firstRuntimeRule->V_RD_install.dbs].
3
4  V_PD_setup.DBS[hasRuntimeRule->V_RD_setup.dbname].
5  V_PD_setup.DBS[hasRuntimeRule->V_RD_install.dbs].
6  V_PD_setup.DBS[hasRuntimeRule->V_RD_create.dbschema].
7  V_PD_setup.DBS[hasRuntimeRule->V_RD_start.dbsystem.postgres].
8
9  V_PD_setup.DBS[dependOn->V_PD_install.vm-guest].
```

Die Deployment-Policy beschreibt eine notwendige Vorbedingung (9) und überprüft den Status einer zuvor ausgeführten Deployment-Policy für die Bereitstellung einer virtualisierten Betriebssystemumgebung. Die TE_1-Instanz ermittelt bestehende Vorbedingungen durch Anwendung der in Listing 5.9 beschriebenen ObjectLogic-Query.
Die Deployment-Rule zur Installation des Datenbanksystems in Listing 5.30 beschreibt mit Anweisung (5) die Operation und referenziert auf die Objekt-Instanz des Datenbanksystems *V_D_PostgreDBS* in der Ontologie-*Cloud-Domain*.

Listing 5.30: *RuleDeployment::Rule*: V_RD_install.dbs

```
1  V_RD_install.dbs[expectedState->V_installed.DBsystem].
2
3  V_RD_install.dbs[nextRule->V_RD_create.dbschema].
4  V_RD_install.dbs[transformedBy->V_T_install.DBsystem].
5  V_RD_install.dbs[do(V_ACT_DBsystemInstall)-><nsd#V_D_PostgreDBS>].
```

Die Rule-Definition enthält einen erwarteten Statuswert (1), der nach erfolgreicher Ausführung nachgewiesen werden kann. Transformationsregeln werden in Anweisung (4) vorgegeben.
Vor Ausführung einer Deployment-Rule überprüft die TE_1-Instanz über die in Listing 4.52 spezifizierte *Reasoning-Rule*, ob für das Datenbanksystem *QualityProperties* gefordert werden.

5.2.3.3 Prüfung der DBS-Authentizität

In Listing 5.27 fordert *V_QPS_authentic.dbspackage* (6) für das DBS-Software-Package den Nachweis der DBS-Software-Package-Authentizität. Die abgeleitete Trust-Policy für die Sicherstellung der geforderten *QualityProperty* ist mit der Trust-Policy in Listing 5.12 vergleichbar, unterscheidet sich mit *V_D_PostgreDBS* in der zu überprüfenden Entität. Das Hash-Ticket in Listing 5.31 listet informelle Identity-Properties (3) und (4), die für das Datenbanksystem zugesichert werden. Als sicherheitsrelevantes Attribut wird ein MD5-Hashwert (1) verwendet.

Listing 5.31: *TicketHash::Credential*: V_C_TicketHashDBS

```
1  V_C_TicketHashDBS[verifiedPropertySec(Hash)->"03f8a8edeec..."].
2  V_C_TicketHashDBS[verificationMethod->V_MD5].
3  V_C_TicketHashDBS[verifiedPropertyInf(Name)->"DBMS-Postgres"].
4  V_C_TicketHashDBS[verifiedPropertyInf(Version)->"9.2"].
5  V_C_TicketHashDBS[confirmedBy-><nsd#Opensource-Provider.Authority>].
6  V_C_TicketHashDBS[verifiedIdentity->V\_I\_PostgresDBsystem].
```

5.2.3.4 Transformation – DBS-Trust-Rule

Die TE_1-Instanz verwendet für die technische Transformation die Schnittstelle *ITrans*. Der konzeptionelle Ablauf ist bereits in Abschnitt 5.2.1.5 beschrieben. Es gelten die Transformationsregeln in Listing 5.17. Die TE_1-Instanz berechnet während der Transformation den MD5-Hashwert für das DBS-Software-Package und vergleicht das Sicherheitsmerkmal mit dem Eintrag im Hash-Ticket.

5.2.3.5 Transformation – DBS-Deployment-Rule

Für die Installation des DBS-Software-Packages gelten die Transformationsregeln in Listing 5.18. Der konzeptionelle Ablauf ist in Abschnitt 5.2.1.6 beschrieben. Die TE_1-Instanz steuert unter Anwendung des *apt*-Moduls (2) eine automatische DBS-Software-Package-Installation.

5.2.4 Szenario (4) – Externe DBS-Zugangssteuerung

Das Konzeptualisierung von *Connection* verfolgt das Ziel, das Design von Verbindungen unabhängig vom Deployment einzelner Systembausteine festzulegen. Nach dem Deployment des Datenbanksystems

sollen Sicherheitsanforderungen an die Verbindung zwischen Cloud-Anwender und dem Datenbanksystem regulativ beschrieben und durchgesetzt werden. Ausgehend von einer bestätigten vertrauenswürdigen DBS-Software-Package-Entität, repräsentiert eine lauffähige Instanz dieser Entität den vertrauenswürdigen Endpunkt einer DBS-Verbindung.

5.2.4.1 Domain-Spezifikation – User-to-DB Connection

Listing 5.27 spezifiziert eine Verbindung (*Connection*) zwischen Cloud-Anwender und dem Datenbanksystem (*V_UsertoDBConnect*) (siehe Datenbanksystem-Spezifikation im Abschnitt 5.2.3.1).

Listing 5.32: *user-to-system::Connection*: V_UsertoDBConnect

```
1 V_UsertoDBConnect[checkQP->nss#V_QPS_authentic.dbsendpoint>]
2 V_UsertoDBConnect[checkQP->nss#V_QPS_confidential.dbsconnection>]
3 V_UsertoDBConnect[establishedConnection(Everyone)->Port.1.db.postgres].
```

Für die Verbindung sind folgende Sicherheitseigenschaften festgelegt:

V_QPS_authentic.dbsendpoint

Die Verbindung erfüllt Anforderungen an die Authentizität der Kommunikationspartner.
Der Cloud-Anwender muss in der Lage sein, die Identität und damit die Vertrauenswürdigkeit des Datenbanksystems überprüfen zu können.
Die Anwender-Entität *Everyone* (3) in Listing 5.32 beschreibt die Menge zugelassener Datenbank-Benutzer des Datenbanksystems.

V_QPS_confidential.dbsconnection

Die Verbindung erfüllt Anforderungen an die Vertraulichkeit.
Der Datenaustausch zwischen Cloud-Anwender und Datenbanksystem muss verschlüsselt werden. Das erfordert die Auswahl eines ausgewählten Verschlüsselungsverfahrens und die Generierung und Bereitstellung der kryptographischen Artefakte. Die Sicherheitskonfiguration zur Anwendung der kryptographischen Artefakte ist anzupassen.

5.2.4.2 Regulation-Spezifikation – DBS-Connection-Policy

Die Deployment-Policy in Listing 5.33 ist für die Laufzeit-Architekturebene (1) festgelegt. Die ihr zugeordnete TE_1-Instanz wird über die in Listing 5.5 beschriebene ObjectLogic-Query ermittelt. Für die Ausführung der Deployment-Rule (4) ist als Vorbedingung ein installiertes Datenbanksystem (5) sicherzustellen.

Listing 5.33: *PolicyDeployment::Policy*: V_PD_establish.connectiondbs

```
1 V_PD_establish.connectiondbs[targetToZone->Z_RUNTIME.DB].
2 V_PD_establish.connectiondbs[firstRuntimeRule->RD_setup.dbsconnection].
3
4 V_PD_establish.connectiondbs[hasRuntimeRule->RD_setup.dbsconnection].
5 V_PD_establish.connectiondbs[dependOn->PD_install.db].
```

Die Deployment-Rule beschreibt mit *A_setup.Connection* die Operation (1) zur Herstellung der DBS-Verbindung in Listing 5.34 und referenziert auf die *Connection*-Instanz *V_User-to-DB-Connection* in der Ontologie-*Cloud-Domain*. Die Anweisungen (3) und (4) geben Transformationsregeln vor.

Listing 5.34: *RuleDeployment::Rule*: V_RD_setup.dbsconnection

```
1 V_RD_setup.dbsconnection[do(A_setup.Connection)\\
2                                   -><nsd#V_User-to-DB-Connection>].
3 V_RD_setup.dbsconnection[transformedBy->V_T_setup_connection].
4 V_RD_setup.dbsconnection[transformedBy->V_T_setup_connection_users].
5 V_RD_setup.dbsconnection[reasonQualifiedBy {0:*} *=> <nss#QualityProperty>].
```

Die TE_1-Instanz überprüft über die spezifizierte *Reasoning-Rule* (5), ob *QualityProperties* für die *Connection*-Instanz gefordert werden. Als Ergebnis werden die in Listing 5.32 vorgegebenen Qualitätsmerkmale an die TE_1-Instanz zurückgegeben.

5.2.4.3 Prüfung der DBS-Endpunkt-Authentizität

Der zu regulierenden Endpunkt des DBS-Servers ist Bestandteil der *V_UsertoDBConnect*-Spezifikation. Die Trust-Policy für den Nachweis der Datenbank-Server-Authentiziät leitet sich aus den gesetzten *QualityProperties* in Listing 5.35 ab.

Listing 5.35: *PropertyTrust::QualityProperty*: V_QPS_authentic.dbsendpoint

```
1   V_QPS_authentic.dbsendpoint[enforcedThrough\\
2                               -><nsr#PT_authentic.dbsendpoint>].
```

Die Verwendung von Certificate-Credentials erfordert mehrere Prüfschritte, um die Vertrauenswürdigkeit bewerten zu können. Die Trust-Rule (4) in Listing 5.36 überprüft die Authentizität des Endpunktes der DBS-Verbindung (*authenticate.connection.port*). Die sichere Zuordnung des überprüften Certificate-Credentials als Teil der CA-Hierarchy in Abbildung 5.2 ist Aufgabe der Trust-Rule (5).

Listing 5.36: *PolicyTrust::Policy*: V_PT_authentic.dbsendpoint

```
1 V_PT_authentic.dbsendpoint[targetToZone->Z_RUNTIME.DB].
2 V_PT_authentic.dbsendpoint[firstTrustRule \\
3                                   ->V_RT_authenticate.connection.port].
4 V_PT_authentic.dbsendpoint[hasTrustRule->V_RT_authenticate.connection.port].
5 V_PT_authentic.dbsendpoint[hasTrustRule->V_RT_verify.certchain].
```

Trust-Rule (RT_authenticate.connection.port)

Die Anweisung (4) der Trust-Rule in Listing 5.37 beschreibt die Trust-Operation (*A_authenticate.port*) und referenziert auf die Instanz der Konzeptklasse *Connection*. Für die technische Überprüfung ist eine Transformationsregel (3) vorgegeben.

Listing 5.37: *RuleTrust::Rule*: V_RT_authenticate.connection.port

```
1 V_RT_authenticate.connection.port[assuredBy -><nss#V_C_CerticateComponent>].
2 V_RT_authenticate.connection.port[expectedState->St_authenticated.port].
3 V_RT_authenticate.connection.port[transformedBy->V_T_assureCertificate].
4 V_RT_authenticate.connection.port[do(A_authenticate.port) \\
5                                   -><nsd#User-to-DB-Connection>].
6 V_RT_authenticate.connection.port[nextRule->V_RT_verify.certchain].
```

Das *Certificate*-Credential in Listing 5.38 repräsentiert ein von der SubCA (*CProviderCA*) herausgegebenes X.509-Zertifikat zur Bestätigung der DBS-Server-Identität (*DBSrv*).

Die TE_1-Instanz vergleicht die sicherheitsrelevanten Identitätsmerkmale im *Certificate*-Credential mit den in Listing 5.39 ausgewiesenen zertifizierten Eigenschaften (11). Als sicherheitsrelevantes Merkmal enthält das *Certificate*-Credential den *Distinguished Name* (2) und stellt darüber die eindeutige Identifizierung der DBS-Server-Entität sicher.

Listing 5.38: *Certificate::Credential*: V_C_Cert.Comp.X509.DBServer

```
1 V_C_Cert.Comp.X509.DBServer[verifiedPropertyInf(Name)->DBS-Postgres].
2 V_C_Cert.Comp.X509.DBServer[verifiedPropertySec(DistinguishedName)\\
3         ->C=DE, O=TrustedDomainAuthority,OU=CloudProvider, CN=DBSrv].
4 V_C_Cert.Comp.X509.DBServer[<nsd#assignedIdentity>\\
5         ->DB-Identity.PostgresSystem].
6 V_C_Cert.Comp.X509.DBServer[expiredValidity ->2018-04-12^^_date].
7 V_C_Cert.Comp.X509.DBServer[verifiedPropertySec(SerialNumber)->4106].
8 V_C_Cert.Comp.X509.DBServer[issuedBy->Authority.CloudProvider.SubCA].
```

Für die Bestätigung der Vertrauenswürdigkeit ist eine Überprüfung der SubCA-Signatur erforderlich. Kryptographische Artefakte der beteiligten Komponenten können über den *Storage-Service* abgerufen werden (siehe Tabelle 5.4). Die TE_1-Instanz verwendet den *Distinguished Name* als eindeutigen Referenzpunkt, um auf dem *Storage-Service* das konkrete X.509-DBS-Server-Zertifikat (siehe Listing 5.39) zu lokalisieren und zu laden.

Listing 5.39: Auszug aus X.509-DBS-Server-Zertifikat

```
1  Certificate:
2   Data:
3     Version: 3 (0x2)
4     Serial Number: 4106 (0x100a)
5   Signature Algorithm: sha256WithRSAEncryption
6     Issuer: C = DE, O = TrustedDomainAuthority, OU = CloudProvider, \\
7      CN = CProviderCA, emailAddress = support@TDCProvider.de
8      Validity
9        Not Before: Apr  2 17:45:04 2017 GMT
10       Not After : Apr 12 17:45:04 2018 GMT
11    Subject: C=DE, O=TrustedDomainAuthority, OU=CloudProvider, CN=DBSrv
12    Subject Public Key Info:
13       Public Key Algorithm: rsaEncryption
14       Public Key: (2048 bit)
15     X509v3 extensions:
16       Netscape Cert Type:
17          SSL Server
18     X509v3 Key Usage: critical
19        Digital Signature, Key Encipherment
20  Signature Algorithm: sha256WithRSAEncryption
```

Für die Überprüfung der SubCA-Signatur ist das zugehörige SubCA-Zertifikat erforderlich. Eine Referenz auf die Instanz der herausgebenden SubCA-*Authority* befindet sich im *Certificate*-Credential (8) in

Listing 5.38. Unter Anwendung der Konzeptklasse *Authority* in Listing 5.40 ermittelt die TE_1-Instanz mit der Methode *issuedCACertificate* (6) das zugehörige SubCA-Zertifikat-*Credential*.

Listing 5.40: ObjectLogic-Query – Bestimmung SubCA-Zertifikat

```
1  ?-
2  ?C:Certificate                    AND
3  ?Y:Authority                      AND
4  ?R:Authority                      AND
5  ?C[issuedBy -> ?Y]                AND
6  ?Y[issuedCACertificate -> ?W] .
```

Die TE_1-Instanz verwendet den *Distinguished Name* im SubCA-Zertifikat-*Credential* als eindeutigen Referenzpunkt, um auf dem *Storage-Service* das konkrete X.509-SubCA-Zertifikat zu lokalisieren und zu laden.

Trust-Rule (RT_verify.certchain)

Die Trust-Rule in Listing 5.41 überprüft die Vertrauenswürdigkeit der Zertifikat-*Credential* aller Instanzen einer bestehenden *Authority*-Hierarchie (siehe Abbildung 5.2) und verwendet dafür die Transformationsregel in Anweisung (2). Die Anweisung (3) der Trust-Rule beschreibt die Trust-Operation (*A_verify.certchain*) und referenziert auf die Instanz des Zertifikat-*Credential* (4) als Ausgangspunkt der Überprüfung.

Listing 5.41: *RuleTrust::Rule*: V_RT_verify.certchain

```
1  V_RT_verify.certchain[expectedState->St_valid.certchain].
2  V_RT_verify.certchain[transformedBy->V_T_assure.cachain].
3  V_RT_verify.certchain[do(A_verify.certchain) \\
4                          -><nss#V\_C\_Cert.Comp.X509.DBServer>].
```

Als Teil der *Cloud-Anwender-Authority*-Hierarchie ist die TE_1-Instanz in der Lage, die Zertifikatskette vollständig zu prüfen. Auf Ebene der Ontologie-*Security* kann die in Listing 4.44 entwickelte *Reasoning-Rule*-Spezifikation verwendet werden, um für eine bestehende Zertifikatskette die zugehörige *RootCA*-Instanz abzufragen (siehe Abschnitt 4.2.2.2). Anschließend sind alle Instanzen der Konzeptklasse *Authority* bis zur *RootCA*-Instanz zu bestimmen.

Eine rekursive Anwendung der ObjectLogic-Query aus Listing 5.40 führt zu einer Liste von $?W$-Zertifikaten (6) der beteiligten *Authority*-Instanzen ($?W$ repräsentiert die Ergebnismenge der Instanzen der Konzeptklasse *Authority*). Die TE_1-Instanz verwendet aus jedem $?W$-Wert den *Distinguished Name* als eindeutigen Referenzpunkt, um auf dem *Storage-Service* die konkreten SubCA- und RootCA-Zertifikate zu lokalisieren, zu laden und zu überprüfen.

5.2.4.4 Absicherung der DBS-Verbindung – Verschlüsselung

Sicherheitsziele werden als Qualitätseigenschaften einer *Connection* festgelegt und durch Security-Policies erfüllt. In Listing 5.42 leitet sich die zugehörige Security-Policy aus der dafür gesetzten Sicherheitseigenschaft ab (Methode *enforcedThrough*).

Listing 5.42: Sicherheitseigenschaften – Ableitung der Security-Policy

```
1  V_QPS_confidential.dbsconnection[enforcedThrough \\
2                                   –><nsr#PS_protect.dbsconnection>].
```

Die Security-Policy In Listing 5.43 definiert mit den Anweisungen (4) – (6) mehrere Security-Rules und referenziert auf die IT-Laufzeit-Zone-Datenbanksystem (1). Als Vorbedingung ist die Ausführung der Trust-Policy (8) aus Abschnitt 5.2.4.3 sicherzustellen.

Listing 5.43: *PolicySecurity::Policy*: V_PS_protect.db-connection

```
1 V_PS_protect.dbsconnection[targetToZone –>Z_RUNTIME.DB].
2 V_PS_protect.dbsconnection[firstSecurityRule –>V_RS_encrypt.connection].
3
4 V_PS_protect.dbsconnection[hasSecurityRule –>V_RS_encrypt.connection].
5 V_PS_protect.dbsconnection[hasSecurityRule –>V_RS_install.connectioncert].
6 V_PS_protect.dbsconnection[hasSecurityRule –>V_RS_install.connectionkey].
7
8 V_PS_protect.dbsconnection[dependOn–>V_PT_authentic.dbsendpoint].
```

Zur Einrichtung einer vertraulichen DBS-Verbindung sind das kryptographische X.509-DBS-Server-Zertifikat (5) und der dazugehörige private DBS-Server-Schlüssel (6) zu installierten.

Security-Rule (V_RS_encrypt.connection)

Die Konzeptklasse *Constraint*, als Teil der Ontologie-*Regulation* im Abschnitt 4.1.3.3 eingeführt, erlaubt die regulierende Verfeinerung von Policies. Die Security-Rule-Beschreibung in Listing 5.44 legt einen Security-Context (2) als Instanz der Konzeptklasse *Constraint* für die Konkretisierung des Verschlüsselungsverfahrens fest. Die Einschränkung in Listing 5.45 wird in Bezug auf die *Action* definiert, um den Handlungsrahmen auf eine festgelegte Sicherheitsfunktion auszurichten.

Listing 5.44: *RuleSecurity::Rule*: V_RS_encrypt.connection

```
1 V_RS_encrypt.connection[expectedState –>St_ssl–encrypted–connection].
2 V_RS_encrypt.connection[definedContext –>V_CO_securityTLS].
3 V_RS_encrypt.connection[do(A_encrypt.conn)–><nsd#User–to–DB–Connection>].
4 V_RS_encrypt.connection[nextRule –>V_RS_install.connectioncert].
```

Die Security-Rule enthält einen erwarteten Statuswert (*St_ssl-encrypted-connection*), der nach erfolgreicher Ausführung nachgewiesen werden kann. Für den internen Policy-Steuerfluss referenziert die Methode *nextRule* auf die nachfolgende Security-Rule.

Listing 5.45: *ActionConstraint::Constraint*: V_CO_securityTLS

```
1 V_CO_securityTLS : ActionConstraint.
2 V_CO_securityTLS[hasTransformation –>V_T_ssl_encryption].
3 V_CO_securityTLS[applySecurity –><nss#TLSenryptionFunction>].
```

Der Security-Context referenziert auf die Transformationsregeln (2) für die technische Einstellung der kryptographischen Funktion. Die Methode *applySecurity* (3) referenziert auf das geforderte Verschlüsselungsverfahren der Sicherheitsfunktion der Ontologie-*Security*. Nach erfolgreicher Ausführung der

Security-Rule kann der erwartete Statuswert (*St_ssl-encrypted-connectiont*) (1) in Listing 5.44 nachgewiesen werden.

Security-Rule (RS_install.connectioncert)

Die Anweisung (4) der Security-Rule in Listing 5.46 beschreibt die Trust-Operation (*A_install.port-cert*) und referenziert auf die Instanz der Konzeptklasse *Connection*. Die Trust-Rule leitet aus dem DN des Zertifikat-*Credential* (1) das konkrete X.509-DBS-Server-Zertifikat ab und installiert es auf Grundlage der Transformationsregel in Anweisung (3).

Listing 5.46: *RuleSecurity::Rule*: V_RS_install.connectioncert

```
1 V_RS_install.connectioncert[assuredBy –><nss#V_C_CerticateComponent>].
2 V_RS_install.connectioncert[expectedState –>St_activated.cert].
3 V_RS_install.connectioncert[transformedBy –>V_T_activateCertificate].
4 V_RS_install.connectioncert[do(A_install.port–cert) \\
5                                          –><nsd#User–to–DB–Connection>].
6 V_RS_install.connectioncert[nextRule –>V_RS_install.connectionkey].
```

Trust-Rule (RS_install.connectionkey)

Die Anweisung (4) der Security-Rule in Listing 5.47 beschreibt die Trust-Operation (*A_install.port-key*) und referenziert auf die Instanz der Konzeptklasse *Connection*. Die Trust-Rule leitet aus dem DN des Zertifikat-*Credential* (1) den konkreten X.509-DBS-Server-Private-Key ab und installiert den Schlüssel auf Grundlage der Transformationsregel in Anweisung (3).

Listing 5.47: *RuleSecurity::Rule*: V_RS_install.connectionkey

```
1 V_RS_install.connectionkey[assuredBy –><nss#V_C_CerticateComponent>].
2 V_RS_install.connectionkey[expectedState –>St_activated.key].
3 V_RS_install.connectionkey[transformedBy –>V_T_activateKey].
4 V_RS_install.connectionkey[do(A_install.port–key) \\
5                                          –><nsd#User–to–DB–Connection>].
```

5.2.4.5 Transformation

Für die technische Realisierung werden Transformationsregeln für die Policy-Bereiche Deployment, Trust und Security spezifiziert und zur Verfügung gestellt.

DBS-Verbindung-Deployment – V_T_install.Debs

Die Regel referenziert auf Parameter für die automatische Installation eines Software-Packages. Die Lokalisierung des Speicherortes im *Storage-Service* erfolgt über die Methode *hasContainer*.

Listing 5.48: *ansible::Transformation*: V_T_install.Debs>

```
1         V_T_install.Debs:ansible.
2         V_T_install.Debs[hasModule–>apt].
3         V_T_install.Debs[hasParameter–>hasContainer].
4         V_T_install.Debs[setRealParameter–>deb].
```

DBS-Verbindung-Trust – V_T_assure.cachain

Die Regel referenziert über die Methode *reasonIssuedCACertificate* auf eine *Reasoning-Rule* zur Ermittlung eines CA-Zertifikates. Die TE_1-Instanz lädt das CA-Zertifikat und führt eine kryptographische Zertifikatsprüfung durch.

Listing 5.49: *TE::Transformation*: V_T_assure.cachain

```
1 V_T_assure.cachain:TE.
2 V_T_assure.cachain[hasParameter->reasonIssuedCACertificate].
```

DBS-Verbindung-Trust – V_T_assureCertificate

Die Regel referenziert über die Angabe von Methoden auf Werte, die zur Überprüfung von X.509-Zertifikaten benötigt werden. Die TE_1-Instanz prüft alle unter "verifyPropertyInfünd "verifyPropertySec"vorgegebenen Werte und führt den Vergleich durch.

Listing 5.50: *TE::Transformation*: V_T_assureCertificate

```
1 V_T_assureCertificate:ansible.
2 V_T_assureCertificate[hasParameter->verifiedPropertyInf(#)].
3 V_T_assureCertificate[hasParameter->verifiedPropertySec(#)].
```

DBS-Verbindung-Security – V_T_ssl_encryption

Die Regel überführt die SSL-Konfiguration in eine Postgres-Konfigurationsdatei auf Grundlage einer Template-Vorgabe. Das SSL-Protokoll wird für die DBS-Verbindung aktiviert.

Listing 5.51: *ansible::Transformation*: V_T_ssl_encryption

```
1 V_T_ssl_encryption:ansible.
2 V_T_ssl_encryption[hasModule->'template'].
3 V_T_ssl_encryption[hasTemplate->'postgresql.conf'].
4 V_T_ssl_encryption[variableValue->'true'].
5 V_T_ssl_encryption[variableAnsiblePath('1')->'postgresql_config'].
6 V_T_ssl_encryption[variableAnsiblePath('2')->'ssl'].
```

5.3 Attestierung – Vertrauenswürdigkeit

In den vorhergehenden Abschnitten dienten bereits *Reasoning-Rules* zur Analyse bestehender Fakten-Relationen, um implizite Zusammenhänge abzuleiten und darzustellen. Der folgende Abschnitt orientiert auf die Analyse und Quantifizierung einer erreichten Vertrauenswürdigkeit im Bereich der betrachteten Referenzarchitektur. Ausgangspunkt bildet ein semantisch formuliertes Policy-Konzept mit dem Anspruch, auf Grundlage einer gesetzten Vertrauensgrundlage eine zugesicherte Transformation regulierender Konzepte auszuführen.

Die Attestierung von Vertrauen reduziert sich nicht allein auf die Sicherstellung einer vertrauenswürdigen Policy-Transformation als qualifizierende Grundlage. Nachfolgend wird ein erster Entwurf für ein mögliches Quantifizierungsverfahren entwickelt und mit Mitteln der semantischen Schlussfolgerung überprüft. Es stellen sich hier Fragen der Berechenbarkeit und der Vollständigkeit der entwickelten Relationen für die Ableitung einer quantifizierten Vertrauenswürdigkeit.

5.3.1 Dynamische Methoden der Konzeptklasse *State*

Die Konzeptklasse *State* in Listing 5.52 enthält *reason*-Methoden, welche den dynamischen Charakter der Konzeptklasse *State* ausdrücken. Die Methoden repräsentieren Schlussfolgerungskonzepte, für die entsprechende *Reasoning-Rules* existieren.

Listing 5.52: Methoden der Konzeptklasse *StateTrust::State* – Signature-F-atoms

```
1 StateTrust[reasonStateChangingRule {1:*} *=> Rule].
2 StateTrust[reasonStateValueTrust {0:*} *=> <nsd#StateDataTrustlevel>].
3 StateTrust[enforcedBy {1:1} *=> Policy].
4 StateTrust[reasonStateAssuredBy {1:1} *=> <nsd#TE>].
```

Das in Abschnitt 4.3.3.4 entwickelte Schlussfolgerungskonzept für die Methode *reasonStateAssuredBy* bestimmt eine vertrauenswürdige Policy-Entität (4), die als TEP-Instanz den Status zusichert. Der Status repräsentiert das Ergebnis einer ausgeführten Trust-Policy (3) und kennzeichnet mit dem Nachweis einer Policy-Entität die Vertrauenswürdigkeit der entsprechenden Entität.
Die Subklasse *StateTrust* spezialisiert sich über die Methode *reasonStateValueTrust* und berechnet den Wert für das Niveau einer bestehenden Vertrauenswürdigkeit. Im Vordergrund steht nachfolgend die Ableitung definierter Vertrauenswerte für eingesetzte Entitäten innerhalb der Referenzarchitektur.

5.3.2 Kategorien für Niveaubestimmung von Vertrauenswürdigkeit

Die Bestimmung des Niveaus an Vertrauenswürdigkeit basiert auf Kategorien zugesicherter Qualitätseigenschaften von Konzeptklassen der Ontologie-*Cloud-Domain*. Das Niveau drückt sich über einen Wertebereich *HOCH*, *MITTEL*, *GERING* und *UNBESTIMMT* aus, der sich in Abhängigkeit von erfüllten Kategoriebereichen ableitet. Die Konzeptklasse *Entity* besitzt die Methode *reasonETlevel* (Entity-Trustlevel) für die Wertberechnung.
Das Konzept *QualityProperty* aus Abschnitt 4.2.3.2 mit seinen Subklassen setzt in Listing 4.49 den Rahmen, welche Eigenschaften einer Entität zugesichert werden. Entscheidend für das Vertrauensniveau sind die Methoden, wie die Zusicherung von Eigenschaften erfolgt.

Kategorie: Organisation (Hersteller/Anbieter)

Das Vertrauensniveau eines Herstellers oder Anbieters ermittelt sich aus organisatorischen Nachweisen über die Prozesse zur Herstellung, Qualitätssicherung und Verteilung von Entitäten, die als Instanzen der Ontologie-*Cloud-Domain* zur Auswahl stehen.

Die Konzeptklasse *CloudRoles* in Listing 4.21 beschreibt domänenspezifische Rollen und kennzeichnet über die Methode *roleTrustLevel* ein aus den Prozessen der Organisation heraus abgeleitetes Vertrauensniveau. Die Subklasse *Manufacturer* besitzt die Methode *reasonMTlevel* (Manufacturer-Trustlevel) und kennzeichnet den Wert des Vertrauensniveaus als Ergebnis einer logischen Schlussfolgerung.

Kategorie: Methode zur Zusicherung einer Identität

Die Methoden bestimmen den Grad der Überprüfbarkeit von Identitäts-Eigenschaften. Die Subklassen der Konzeptklasse *Credential* repräsentieren unterschiedliche Formen des Identitätsnachweises (z. B. Hash-Ticket und *Certificate*-Credential). Die Zusicherung von Identitätsmerkmalen kombiniert Aussagen über Funktions- und Verhaltensmerkmale einer Entität. Das Vertrauensniveau der bestätigenden Instanz (z. B. ein Hersteller) überträgt sich implizit auf die Entität.

Kategorie: Externe Evaluierung

Die externe Evaluierung ist eine Zusicherung von Funktions- und Verhaltensmerkmalen einer Entität. Es beschreibt ein formales Verfahren, bei dem eine Organisation die Eigenschaften einer Entität über eine zertifizierte Prüfstelle verfizieren und bestätigen lässt. Der Prozess endet mit einer Zertifizierung durch eine Autorität. Zertifizierungen sind der Entität überprüfbar zugeordnet.

Die nachfolgende Konzeptklasse *Certification* der Ontologie-*Regulation* repräsentiert diesen formalen Prozess einer Evaluierung unter Führung einer Autorität (*AuthorityInspection*).

Listing 5.53: Methoden der Konzeptklasse *Certification* – Signature-F-atoms

```
1 Certification[attestedProperties {1:*} *=> <nss#QualityProperty>].
2 Certification[assuredTlevel {1:1} *=> <nss#Trustlevel>].
3 Certification[assuringAuthority {1:1} *=> <nss#AuthorityInspection>].
```

Konzeptklasse	Instanzen von Konzeptklasse
CloudRole	Behördliche Autorität Hersteller mit Herstellernachweisen Unbekannt
Credential	Kryptographisches Schlüsselpaar und Zertifikat – Verbindlichkeit Kryptographisches Schlüsselpaar und Zertifikat – Authentisierung Hash-Ticket Passwort
Certification	Produkt-Zertifizierung – Vertrauensstufe HOCH Produkt-Zertifizierung – Vertrauensstufe MITTEL Produkt-Zertifizierung – Vertrauensstufe GERING

Tabelle 5.8: Kategorien als Grundlage zur Attestierung einer bestehenden Vertrauenswürdigkeit

5.3.3 Semantische Rules für Niveaubestimmung

Die nachfolgenden Rule-Definitionen beschreiben allgemein ein Zuordnungskonzept zu einem Wertebereich auf Grundlage logischer Bedingungen. Tabelle 5.8 präsentiert dafür einen ersten Bewertungsansatz und identifiziert vertrauensbildende Elemente, die in einen Zusammenhang überführt werden müssen und einer kontinuierlichen Verfeinerung unterliegen.
Für jeden Wert an Vertrauen im gesetzten Wertebereich muss der Nachweis einer Policy-Entität als vertrauensbildende Grundlage und damit als wertbildende notwendige Vorbedingung sichergestellt sein.

5.3.3.1 Ableitungsregel – Vertrauenswürdigkeit *HOCH*

Die *Reasoning-Rule* in Listing 5.54 attestiert ein Vertrauensniveau mit dem Wert *HOCH*.
Der Nachweis einer Policy-Entität (11) ist Grundlage der weiteren Berechnungen.

Listing 5.54: Methode *reasonETlevel* für Vertrauensniveau *HOCH* – Reasoning-Rule

```
1  @{LevelofTrustH}
2  ?X[reasonETlevel->"HIGH"]
3  :-
4  ?X:Entity                                    AND
5  ?A:Authority                                 AND
6  ?C:Certificate                               AND
7  ?M:Manufacturer                              AND
8  ?Z:Certification                             AND
9  ?I:Identity                                  AND
10
11 ?X[reasonBasedTEP->?T]                       AND
12
13 ?X[authorizedTo(?C) -> ?I]                   AND
14 ?C[issuedBy->?A]                             AND
15
16 ?X[evaluatedBy -> ?Z]                        AND
17 ?Z[assuringAuthority->?A                     AND
18 ?Z[assuredTlevel->"HIGH"                     AND
19
20 ?X[manufacturedBy -> ?M]                     AND
21 ?M[reasonMTlevel->"HIGH"] .
```

Die Konzeptualisierung von Rollen einer Domäne bildet eine erste Schnittstelle zur Ontologie-*Cloud-Domain*, so dass noch keine vollständige Sicht auf Eigenschaften und Relationen der Konzeptklasse *Manufacturer* als Berechnungsgrundlage vorliegt. Aus diesem Grund unterliegt im Szenario das Vertrauensniveau eines Herstellers (21) einer expliziten Definition.
Der Wert an Vertrauen leitet sich unter Einbeziehung der höchsten Evaluierungsstufe ab (18) und bezieht eine zertifizierende Autorität (17) mit ein. Die sichere Zuordnung von Identitätsmerkmalen stützt sich ebenfalls auf eine Instanz der Konzeptklasse *Authority* (14), die als Herausgeber eines *Certificate*-Credentials (13) nachzuweisen ist.

Anwendung in der Referenzarchitektur:

Die Validierung erweitert die Policy für eine KVM-Entität im Abschnitt 5.2.1 um eine Instanz der Konzeptklasse *Certification*. Als Subklasse der Konzeptklasse *Authority* bestätigt die zertifizierende Instanz von *AuthorityInspection* ein hohes Vertrauensniveau.

Zusätzlich besitzt die KVM-Entität ein *Certificate*-Credential für die sichere Überprüfung der KVM-Authentizität. Die Cloud-Provider-Autorität in Abbildung 5.2 erteilt dafür eine kryptographische Zusicherung. Das KVM-OpenSource-Projekt [KVM16] bildet die Instanz der Konzeptklasse *Manufacturer*.

5.3.3.2 Ableitungsregel – Vertrauenswürdigkeit *MITTEL*

Die *Reasoning-Rule* in Listing 5.55 attestiert ein Vertrauensniveau mit dem Wert *MITTEL*.
Das Vertrauensniveau entsteht entweder durch einen Hersteller mit einem hohen Vertrauensniveau (21) oder über den Nachweis einer mittleren Evaluierungsstufe (18) in Bezug auf die Entität. Für die Zusi-

cherung von Identitätsmerkmalen ist eine Instanz der Konzeptklasse *Authority* (14) nachzuweisen, die als Herausgeber für ein *Certificate*-Credential (13) ausgewiesen ist.

Listing 5.55: Methode *reasonETlevel* für Vertrauensniveau *MITTEL* – Reasoning-Rule

```
1   @{LevelofTrustM}
2   ?X[reasonETlevel->"MIDDLE"]
3   :-
4   ?X:Entity                                   AND
5   ?A:Authority                                AND
6   ?C:Certificate                              AND
7   ?M:Manufacturer                             AND
8   ?Z:Certification                            AND
9   ?I:Identity                                 AND
10
11  ?X[reasonBasedTEP->?T]                      AND
12
13  ?X[authorizedTo(?C) -> ?I]                  AND
14  ?C[issuedBy->?A]                            AND
15
16  (?X[evaluatedBy -> ?Z]                      AND
17  ?Z[assuringAuthority->?A                    AND
18  ?Z[assuredTlevel->"MIDDLE")                 OR
19
20  ?X[manufacturedBy -> ?M]                    AND
21  ?M[reasonMTlevel->"HIGH"]) .
```

Anwendung in der Referenzarchitektur:

Die Validierung erweitert die Policy für ein virtualisiertes Betriebssystem im Abschnitt 5.2.2 um eine Instanz der Konzeptklasse *Certification*. Als Subklasse der Konzeptklasse *Authority* bestätigt die zertifizierende Instanz von *AuthorityInspection* ein mittleres Vertrauensniveau.

Zusätzlich besitzt die VM-Entität ein *Certificate*-Credential für die sichere Überprüfung der VM-Authentizität. Die Cloud-Anwender-Autorität in Abbildung 5.2 erteilt dafür eine kryptographische Zusicherung. Das VirtualBox-OpenSource-Projekt [Ora16] bildet die Instanz der Konzeptklasse *Manufacturer*.

5.3.3.3 Ableitungsregel – Vertrauenswürdigkeit *GERING*

Die *Reasoning-Rule* in Listing 5.56 attestiert ein Vertrauensniveau mit dem Wert *GERING*.
Das Vertrauensniveau entsteht alternativ durch einen Hersteller mit einem mittleren Vertrauensniveau (21) oder über den Nachweis einer geringen Evaluierungsstufe (18) in Bezug auf die Entität.
Für dieses Vertrauensniveau steht die Organisation im Vordergrund, die eine Entität bereitstellt. Der Herstellers bestimmt mit seinen Prozessen und Maßnahmen das Vertrauensniveau.
Für die Zusicherung von Identitätsmerkmalen sind *TicketHash*-Credentials (13) ausreichend. Eine Instanz der Konzeptklasse *Manufacturer* (14) steht in der Verantwortung, korrekte *TicketHash*-Credentials auszustellen und zu verteilen.

Listing 5.56: Methode *reasonETlevel* für Vertrauensniveau *GERING* – Reasoning-Rule

```
1   @{LevelofTrustG}
2   ?X[reasonETlevel->"LOW"]
3   :-
4   ?X:Entity                                   AND
5   ?A:Authority                                AND
6   ?H:TicketHash                               AND
7   ?M:Manufacturer                             AND
8   ?Z:Certification                            AND
9   ?I:Identity                                 AND
10
11  ?X[reasonBasedTEP->?T]                      AND
12
13  ?X[authorizedTo(?H) -> ?I]                  AND
14  ?H[issuedBy->?M]                            AND
15
16  (?X[evaluatedBy -> ?Z]                      AND
17  ?Z[assuringAuthority->?A                    AND
18  ?Z[assuredTlevel->"LOW")                    OR
19
20  ?X[manufacturedBy -> ?M]                    AND
21  ?M[reasonMTlevel->"MIDDLE"]) .
```

Anwendung in der Referenzarchitektur:
Die Validierung unterstützt eine Policy für ein Datenbanksystem (DBS) im Abschnitt 5.2.3 ohne die Anwendung einer Zertifizierungsinstanz. Die DBS-Entität besitzt nur ein *TicketHash*-Credential für die sichere Überprüfung der DBS-Authentizität. Das PostgreSQL-OpenSource-Projekt [Pos17] bildet die Instanz der Konzeptklasse *Manufacturer*.

5.3.3.4 Ableitungsregel – Vertrauenswürdigkeit *UNBESTIMMT*

Die Nichterfüllung aller in den Abschnitten 5.3.3.1, 5.3.3.2 und 5.3.3.3 entwickelten *Reasoning-Rules* führt zu einer Zuweisung der Vertrauenswürdigkeit mit dem Wert *UNBESTIMMT*.

5.4 Gegenüberstellung der Szenarien mit den Zielstellungen

Der Abschnitt unterzieht die Ergebnisse der einzelnen Validierungsaufgaben einer Bewertung in Bezug auf die gesetzten Zielstellungen Z1, Z2, Z3, und Z4 im Abschnitt 5.
Der Entwurf der Referenzarchitektur in Abbildung 5.1 repräsentiert gleichermaßen das semantische Modell der Referenzarchitektur als eine Komposition einzelner Instanzen von Konzeptklassen der Ontologie-*Cloud-Domain* zu einer logisch verbundenen Architekturinstanz. Als Vorbedingung zur Validierung der Policy-Fähigkeiten ist die semantische Definition der Referenzarchitektur als Teil von Z1 notwendig. Die Methodik überprüfte Details der Konzeptualisierung der zu regulierenden Domäne und identifizierte in manchen Bereichen Konzeptualisierungslücken, die rückwirkend zu einer Konzeptanpassung führten.

Die vorliegende Menge an Konzeptklassen der Ontologie-*Cloud-Domain* ermöglichte für die Erfüllung von Z1 einen flexiblen und kompositorischen Designansatz.
Dabei bestätigte sich eine gute Konzeptabdeckung, angefangen von hardwarebasierten Anforderungsdetails bis zu Aspekten der Virtualisierung von Betriebssystemen und der Steuerung von Datenbanksystemen auf Portebene. Die bestehende Konzeptualisierung besitzt bereits einen Konzeptumfang, der bestehende Architektur-Cloud-Konzepte relativ vollständig semantisch abbilden kann.
Auf der Ebene der Policies kam das gleiche Bewertungsprinzip zum Tragen. Die Integrationsfähigkeit der Konzeptklasse Ontologie-*Regulation* zur sprachlichen Beschreibung von Regulierungsanforderungen an einen Teil der Domäne unterzog sich einer kontinuierlichen praktischen Überprüfung.
Für die Erfüllung von Z2 (Trust-Policy – Abdeckung und Ausdrucksstärke) bildete der Teilaspekt zur technischen Transformation eine Einheit und bezog Z3 (Transformation – Technische Integrationsfähigkeiten) in die Betrachtung mit ein. Jede regulierend einwirkende Rule-Definition benötigte eine angepasste Transformationsbeschreibung zur vollständigen Realisierung der gesetzten technischen Anforderungen. Auch die Integration der Ontologie-*Security* verlangte entsprechende Transformationsverfahren, die konkrete Sicherheitseinstellungen konfigurativ auf den entsprechenden Teil der Domäne übertrugen. Die mit der Konzeptklasse *Transformation* in Abschnitt 4.1.3.7 vorgestellten Schnittstellenkonzepte konnten bestehende Transformationsanforderungen erfüllen.
Einen besonderen Schwerpunkt in der Validierung bildete die Überprüfung des Konzeptansatzes aus Abschnitt 4.2.3 zur Sicherstellung einer vorgelagerten Trust-Policy. Im Unterschied zu regulären Policies für das Deployment oder die Absicherung von Domänenbereichen vollzieht sich in der Policy-Anwendung ein zusätzlicher semantischer Interpretationsaufwand zur Prüfung auf vorliegende Qualitätsparameter für jede zu regulierende Domänen-Instanz. Zur Unterstützung der Policy-Entitäten erfolgte eine Integration der dafür spezifizierten *Reasoning-Rule* (siehe Listing 4.48) in die Ontologie und eine Bewertung in Bezug auf die Schlussfolgerungsfähigkeit während der Policy-Ausführung. Für die bestehenden Qualitätsanforderungen ließ sich die Wirksamkeit einer vorgelagerten Trust-Policy als Vorbedingung für die nachfolgenden Policy-Schritte bestätigen.
In Bezug auf die Realisierung einer Trust-Policy stellten sich die Fragen nach ihrer Transformationsfähigkeit. Die in den Abschnitten 5.2.1 bis 5.2.4 vorgelegten Transformationsregeln kamen nach Anpassungen in einigen Bereichen der Ontologie-*Cloud-Domain* erfolgreich zur Anwendung.
Das Szenario in Abschnitt 5.2.2 stellte sich als das umfangreichste Regulierungskonzept heraus. Neben der Ausführung von Policies für die vertrauenswürdige Implementierung eines virtuellen Betriebssystems in Form einer VM vollzieht sich das in Abschnitt 4.3.3 entwickelte Konzept einer vertikalen Erweiterung der Vertrauensbasis. Der Wechsel einer Policy-Zone und die nachfolgende Synchronisation zwischen den jeweiligen TE_x-Instanzen rückte in den Mittelpunkt der Betrachtung.
Die Anwendung, Implementierung und Überprüfung von vertrauenswürdigen Policy-Entitäten für die Verwaltung getrennter Policy-Zonen ergänzte die Untersuchung bezüglich der Policy-Fähigkeiten. Die erfolgreiche Umsetzung eines der Kernkonzepte der Arbeit lässt sich über die in Abschnitt 5.3 vorgestellten Regeln zur Attestierung überprüfen.
Obwohl der Bereich der Attestierung sich auf mehrere Schwerpunkte richten kann, so orientiert sich die Untersuchung auf den Aspekt der Quantifizierung von Vertrauenswürdigkeit. Die Ergebnisse sind schwer zu bewerten, da die Vorgehensweise zur Bewertung einer Vertrauenswürdigkeit sich keiner Verallgemeinerung unterziehen lässt. Es spielen eine Reihe von Faktoren eine Rolle, die kontextbezogen

eine unterschiedliche Bewertung erhalten. Wenn z. B. die Vertrauenswürdigkeit einer Organisation in die Bewertung einfließt, so kann die Menge an Kriterien, die ein hohes oder mittleres Vertrauensniveau bescheinigen, je nach Betrachter abweichen. Aus diesem Grund entstand im Abschnitt 5.3.2 ein eigenständiger Bewertungsansatz.
Ziel des Abschnittes ist der Nachweis über ein semantisch formuliertes Konzept zur Bewertung von Vertrauenswürdigkeit. Für den aufgestellten Wertebereich stellen logikbasierte Regeln (siehe Listing 5.54) die entsprechende Zuordnung zu einem der Bereiche sicher. Der erfolgreiche Nachweis, dass unter Anwendung eines ersten Bewertungsansatzes die Vertrauenswürdigkeit eine quantifizierten Ausdruck findet, bildet das Ergebnis von diesem Abschnitt.
Mit der Erweiterung der bestehenden Policies besteht die Referenzarchitektur aus integrierten Entitäten mit einem unterschiedlichen Niveau an Vertrauenswürdigkeit. Es ist Aufgabe des vorgelegten semantischen Bewertungsverfahrens, das entsprechende Vertrauensniveau auszuweisen.

5.5 Gegenüberstellung der Ergebnisse mit den Kernfragen

Im Anschluss an die Validierung lässt sich ein Bezug zur Beantwortung der in Abschnitt 1.2 aufgestellten Kernfragen herstellen. Die Frage nach konstituierenden Entwicklungsbedingungen für Vertrauen im Bereich technischer Systeme steht im Mittelpunkt von **Kernfrage 1**. Die geregelte Überprüfung erwarteter System- und Verhaltenseigenschaften zieht sich als qualifizierende Methode durch alle vorgestellten Szenarien der Validierung.
Eine vertrauensbildende Entwicklung entsteht durch zwei wesentliche Kernkonzepte. Dabei bildet eine vertrauenswürdige Entität (Policy-Entität) den Ausgangspunkt für die Verlagerung oder Erweiterung einer bestehenden Vertrauensbasis. Die Konzepte zur horizontalen und vertikalen Etablierung von Vertrauen in den Abschnitten 4.3.2 und 4.3.3 beschreiben die Grundlagen für ein solches Vorgehen.
Das Konzept einer vertrauenswürdigen Entität erweitert sich in einem zweiten Schritt zu einem ganzheitlichen Anwendungsprinzip vertrauenswürdiger Entitäten. Der erste Ansatz zur Überprüfung erwarteter System- und Verhaltenseigenschaften und der vertrauenswürdigen Repräsentation dieser Eigenschaften, was sich ursprünglich nur auf Policy-Entitäten bezog, lässt ich als strukturiertes Vorgehen (siehe Abbildung 4.10) auf alle Entitäten einer Domäne übertragen.
Das konstituierende Prinzip zur Herausbildung von Vertrauen gestaltet sich durch eine verifizierbare Zusicherung erwarteter Eigenschaften, deren Grundlage eine Hierarchie von Policy-Entitäten bildet. Die Methoden zur Feststellung einer vertrauenswürdigen Zusicherung sind Gegenstand der in der Validierung vorgestellten Szenarien.
Mit **Kernfrage 2** stehen erforderliche Sprachkonzepte für die Beschreibung von Regulierungsstrategien im Mittelpunkt. Hier geben die Szenarien der Validierung durch die Verwendung semantischer Sprachmittel eine hinreichende Antwort. Sowohl der Aspekt einer präzisen Domänenbeschreibung als auch die Konzeptualisierung von Regulierungszielen sind bereits als interpretierbare Ausdrucksmittel für die Referenzarchitektur im Einsatz. Die Integrationsfähigkeit der verschiedenen Ontologien stellt das sprachliche Potential unter Beweis, um Definitionen für Regulierungsanforderungen auf semantische Domänenkonzepte wirksam anzuwenden und zusätzlich qualitätsfordernde Sicherheitseigenschaften mit den Konzepten für eine IT-Architektur-Dömane zu verknüpfen.
Die Frage nach der Ausdrucksstärke in **Kernfrage 3** beantwortete sich bisher durch die bestehende Fä-

higkeit, verschiedene Policy-Zielstellungen für Deployment, Trust, Security oder auch der Attestierung mit den bestehenden semantischen Konzepten zu definieren und anzuwenden. Das Potential der bestehenden sprachlichen Ausdrucksstärke muss sich an zukünftigen Anforderungen an die Regulierung bewähren. Im Bereich der bestehenden Anforderungen zeigten sich bisher keine Begrenzungen in ihrer Anwendung.
Die Hoheitsfrage, verbunden mit technischen Möglichkeiten, eine bestehende Regulierungshoheit auf Bereiche des Cloud-Anwenders auszuweiten, stellt sich mit **Kernfrage 4**. Die technische Beantwortung dieser Frage beschreibt ein Vorgehen, um Hoheitsbereiche regulativ festzulegen und für eine Vertragslaufzeit sicherzustellen. Die Konzepte zur horizontalen und vertikalen Etablierung von Vertrauen stellen dafür ein Sicherheitsprotokoll bereit, denn die Delegierung von Vertrauen muss sich auf zugesicherten Sicherheitseigenschaften der beteiligten Seiten stützen. Das Sicherheitsprotokoll für die horizontale Etablierung von Vertrauen ließ sich aus Aufwandsgründen nicht als ein eigenständiges Szenario überprüfen. Mit dem Szenario (2) in Abschnitt 5.2.2 erfolgte eine Überprüfung der Prinzipien zur Erweiterung der Policy-Zonen unter Anwendung einer Hierarchie von Policy-Entitäten. Dieses Vorgehen stellt das Grundprinzip vertikaler Hoheitserweiterung vor und identifizierte während der Ausführung ein Potential für eine weitergehende semantische Protokollunterstützung.

5.6 Zusammenfassung der Validierung

Die Validierung stellte deutlich heraus, dass ein Cloud-Anwender im Bereich des Policy-Managements technisch in der Lage ist, das Design und die Funktionseigenschaften einer Cloud-Architektur auf seine konkrete Problemsituation anzupassen und die Verarbeitung im System einer eigenen Regulierung zu unterziehen.
Obwohl der Handlungsrahmen sich auf einen abgegrenzten IT-Architekturbereich bezieht, demonstriert er jedoch wesentliche Konzeptbeiträge aus dem Abschnitt 4 in Bezug auf eine vertrauensbildende Regulierung. Die vorliegenden Szenarien veranschaulichen die Wirksamkeit einer verallgemeinerten Konzeptualisierung für Regulierung in Form eines semantischen Modells und ihre Anwendbarkeit auf eine spezifische IT-Cloud-Domäne.
Dabei reicht es nicht aus, definierte Regelsätze zu interpretieren und auszuführen. Mit jedem Schritt kombinieren regulierende Abläufe eine Funktion zur Analyse mit einer technischen Operation als Ergebnis einer logischen Schlussfolgerung. Der Nachweis einer vorgelagerten Analysefähigkeit unterstreicht den methodologischen Ansatz einer ontologisch geführten Konzeptualisierung. Jedes Szenario erhält darüber eine adaptive Fähigkeit, sich während der stufenweisen und qualifizierten Integration weiterer Entitäten auf neue Regulierungsziele einzustellen.
Im Kern stützen sich alle Aktivitäten auf eine komplementäre und dynamisch integrierte Architektur verteilter vertrauenswürdiger Entitäten. Als Instanz einer vertrauenswürdigen Policy-Entität stellt das implementierte Pattern in Abbildung 4.1 erweiterte Cloud-Anwender-Schnittstellen für das Policy-Management und die technische Transformation dieser Vorgaben bereit. Der Nachweis über die infrastrukturelle adaptive Erweiterungsfähigkeit auf Grundlage neuer vertrauenswürdiger Policy-Entitäten beschreibt ein wesentliches Ergebnis der Validierung. Diese Grundmethode lässt sich auf alle weiteren Architekturebenen erweitern, konnte aber bereits repräsentativ als vertrauensbildendes Kernkonzept in einem eingegrenzten Bereich nachvollziehbar zur Anwendung kommen.

6 | Zusammenfassung – Ausblick

6.1 Zusammenfassung der Arbeit

Das Thema *Beschreibung, Verarbeitung und Überprüfung clientseitiger Policies für vertrauenswürdige Cloud-Anwendungen* setzt den Schwerpunkt der vorliegenden Arbeit allgemein auf die Regulierung von IT-Systemen hinsichtlich ihrer Nutzung und Architekturausprägung. Die Fokussierung auf Cloud-Systeme stellt ein spezifisches Anwendungsparadigma heraus. Regulierungsansprüche des Cloud-Anwenders stehen im Vordergrund, dem sich grundsätzlich die Bereiche eines Cloud-Anbieters in Bezug auf eine selbstbestimmte Verarbeitung und Speicherung seiner eigenen Daten entziehen. Dennoch sieht er sich im gegenwärtigen Cloud-Nutzungskonzept verpflichtet, den digitalen Cloud-Anbieter-Bereich zur Erfüllung eigener Zielstellungen zu verwenden.
Ein wesentlicher Kern der Problembeschreibung stellt heraus, dass Regeln soziale Konstrukte darstellen (siehe Abschnitt 2.2.1) und gleichermaßen mit ihren Zielstellungen auf technische Systeme zu übertragen sind. Dafür wird eine Gegenüberstellung zwischen einem sozialen und digitalen Kontext (Organisation vs. Cloud-System) geführt. In der Gegenüberstellung zeigt sich, dass in beiden Bereichen Regelungen die Instrumente zur Steuerung koordinierten Handelns darstellen und sich darin vorausgedachte Zielstellungen als Erwartungen manifestieren,
Kooperative Kopplungen zwischen Organisationen im sozialen Kontext repräsentieren sich im digitalen Bereich als Serviceparadigma verteilter Cloud-Systeme. Über die Fragestellung, wie sich Kooperationen in Cloud-Systemen durchsetzen und gestalten lassen, dass sich daraus tatsächlich erwartete Handlungsmuster im Sinne des Cloud-Anwenders ableiten, führt zum Thema Vertrauen. Die Aufwertung des Themas um den Aspekt *vertrauenswürdige Cloud-Anwendungen* erweitert die Regulierungsbetrachtung. Vertrauen beschreibt die Strategie mit der größten Reichweite zur Erweiterung von Handlungspotenzialen des Cloud-Anwenders. Für Vertrauen gab es bisher keine eigenständige Form der Erwartungsbeschreibung, so dass sich Vertrauen als zukünftiger Gegenstand der Regulierung zu einem konzeptionellen Schwerpunkt herausbildet. Aus den konstituierenden Elementen vertrauenswürdiger kooperativer Kopplung wie *Erwartung* und *Kommunikation* leiten sich die Konzeptschwerpunkte im Abschnitt 4 ab.
Der Cloud-Anwender benötigt ein semantisches Modell zur Formulierung seiner Erwartungen, so dass eine technische Interpretierbarkeit als Vorbedingung vorliegt.
Bisherige Cloud-Modelle berücksichtigen keine Kommunikation zum Zweck einer geregelten Vermittlung von Cloud-Anwender-Erwartungen in Form dedizierter Trust-Policies. Die Einführung einer vertrauenswürdigen Cloud-Entität begründet sich mit bestehenden Anforderungen an die vertrauenswürdige Durchsetzung von Regulierungs- und Handlungsinstruktionen. Das Design-Pattern in Abbildung 4.1 beschreibt ein Komponentenmuster zur Entwicklung einer vertrauenswürdigen Cloud-Entität. Die An-

wendung einer Cloud-Entität dient der Definition hoheitlicher Vertrauensbereiche und stellt eine Schnittstelle zur Verfügung, über die der Cloud-Anwender seine Regulierungsansprüche durchsetzen kann und darüber Gewissheit erhält, dass diese tatsächlich durchgesetzt werden.
Die Entwicklung eines semantischen Modells und der Entwurf einer vertrauenswürdigen Cloud-Entität steht in Bezug auf die im Abschnitt 2.5 aufgestellten Thesen. Das verteilte Ontologie-Design vollzieht eine Entkopplung zwischen Regulierung (Ontologie-*Regulation*) und einer Cloud-System-Architektur, dem eigentlichen Gegenstand der Regulierung. Regulierung als semantisches Modell stellt logik-basierte Ausdrucksmittel zur Verfügung, die erst in einem zweiten Schritt in Form konkreter Policies eine Zuordnung auf das Modell einer Cloud-Domäne (Ontologie-*Cloud-Domain*) erhalten (**These 1 – Allgemeines Grundprinzip für Regulierung**).
Die Ontologie-*Cloud-Domain* orientiert auf die Beschreibung von Konzeptklassen, die eine IT-Architektur mit Verbindungsrelationen zu weiteren Systemklassen beschreiben. Orthogonal dazu definieren spezifische Konzeptklassen ein Cloud-Servicemodell. Das Domänenmodell besitzt die Fähigkeit, beliebige Verteilungskonzepte von IT-Architekturen abzubilden und zusätzlich in einen Cloud-Kontext zu überführen bzw. auf zukünftige Servicemodelle anzuwenden.
Die Analyse zu vertrauensbildenden Konzepten stellte heraus, dass die Zusicherung von Eigenschaften und Statuswerten zu vertrauenswürdigen Eigenschaften einer Entität führt. Vertrauen entwickelt sich als Resultat einer Bewertung vertrauensbildender Fakten, die wiederum durch Sicherheitsfunktionen einen verifizierbaren Wert erhalten. Die Entwicklung einer eigenen Ontologie-*Security* beschreibt notwendige Konzeptklassen, die den Grundstein für nachfolgende Konzepte zur Zusicherung von Eigenschaften beliebiger Entitäten legten.
Der Aspekt Zusicherung erfordert eine eigenständige Konzeptualisierung von Vertrauen. Die semantische Spezifikation der Konzeptklasse *Authority* in Abschnitt 4.2.2 als Teil der Ontologie-*Security* setzt ein Maß an Zusicherung gemäß Definition 4.5 und Definition 4.6 und unterstützt die Gestaltung verteilter Vertrauenshierarchien. Aussagen über bestehende Vertrauenshierarchien, gültige Cross-Beziehungen oder Root-Zertifikate entstehen im Ergebnis logischer Schlussfolgerungen integrierter *Reasoning-Rule*-Spezifikationen der Ontologie-*Security* (siehe Listing 4.44). Als Form einer verifizierbaren Zusicherung von Identitätsmerkmalen beliebiger Entitäten entstand die Konzeptklasse *Credential*.
Die vertrauenswürdige Zusicherung findet in zwei Spezifikationsformen ihre Anwendung. Die Zusicherung von Identitätsmerkmalen und Sicherheitseigenschaften einer Entität entsteht unter Anwendung differenzierter *Credential*-Klassen für den sicheren Nachweis ihrer Identität und der Zurechenbarkeit ausgeführter Handlungsschritte. Diese Form der Zusicherung ist Gegenstand von Abschnitt 4.2.1 und führt den Begriff einer Policy-Entität (*TE-P*-Instanz) ein.
Die zweite Spezifikationsform wendet die Vorgehensweise auf beliebige Entitäten der Domäne an und führt zur Unterscheidung den Begriff einer Domänen-Entität (*TE-D*-Instanz) ein. Die Zusicherung sicherheitsrelevanter Eigenschaften einer Entität als Instanz der Konzeptklasse *QualityProperty* entwickelt sich in Abschnitt 4.2.3.2 zu einer vorgelagerten Trust-Policy-Spezifikation (**These 2 – Formalisierbarkeit von Vertrauen**).
Die Konzeptphase endet mit zwei Strategien zur Erweiterung einer Cloud-Anwender-definierten Vertrauensgrundlage. Die Unterscheidung in eine horizontale und vertikale Etablierung von Vertrauen drückt im ersten Fall die Erweiterung hoheitlicher Bereiche des Cloud-Anwenders auf IT-Architekturbereiche des Cloud-Anbieters aus. Die Überprüfung vertrauenswürdiger Bedingungen für eine horizontale Stra-

tegie stützt sich auf zugesicherte kryptographische Eigenschaften der Infrastruktur-Plattform des Cloud-Anbieters und ist Teil einer Sicherheitsprotokoll-Spezifikation (siehe Abschnitt 4.3.2).
Ausgehend von einer *Root of Trust* kennzeichnet die vertikale Etablierung von Vertrauen die Erweiterung hoheitlicher Bereiche auf unterschiedliche Architekturbereiche des Cloud-Anbieters. Die vertikale Strategie setzt im Bereich einer IT-Architektur-Ebene eine Vertrauensbasis in Form einer *TE-P*-Instanz und definiert eine zugehörige Policy-Zone. Für jede Ebene vollzieht sich das Deployment von Entitäten in Form von *TE-D*-Instanzen unter Anwendung einer Trust-Policy und gewährleistet die Quantifizierung der Vertrauenswürdigkeit unter Anwendung integrierter *Reasoning-Rule*-Spezifikationen der Ontologie-*Security*.

6.2 Ausblick und abgeleitete Themen

Die Schwerpunkte der Spezifikation im Abschnitt 4 in Verbindung mit den Ergebnissen der Validierung in Abschnitt 5 stellen erste Ansätze für hoheitliche Regulierung vor, Grundstrukturen für vertrauensbildende Konzepte im Bereich technischer IT-Architekturen. Setzt man die Ergebnisse in einen Anwendungskontext, so leiten sich daraus nachfolgende Themenbereiche für die Weiterentwicklung ab.

Vertrauenswürdiges Compliance-Monitoring

Als Teil von Regulierungsaufgaben stellte bereits die Validierung das Prinzip logischer Schlussfolgerung heraus. Die Durchsetzung von Regulierungszielen ist gleichzeitig verbunden mit einer Generierung von Zustandsfakten der realen Cloud-Architektur und ihrer Rückführung auf die semantische Modellbasis der Ontologie. Die Ziele der Regulierung repräsentieren sich als überprüfbare und vertrauenswürdige Fakten und entwickeln für den Cloud-Anwender ein vertrauensbildendes Informationsmodell.

Der Aspekt von *Compliance* repräsentiert sich als implizites Wissen über eine Menge authentischer Fakten im Vergleich mit bestehenden Regulierungsanforderungen in Form konkreter Policies. Das verteilte Ontologiekonzept bildet eine Struktur für die Verbindung von Regulierung im Sinne einer *Governance* [Rüt+10] unter Berücksichtigung von Sicherheitsanforderungen. Darin enthalten ist bereits die Konzeptklasse *Threat* und stellt einen Zusammenhang zu bestehenden Risiken her.

Die Einhaltung von Richtlinien entwickelt sich zu einer eigenen Forschungsdisziplin, da ihr Gegenstand GRC (Governance, Risk and Compliance) eine wachsende gesellschaftliche Bedeutung erhält. Die Untersuchung von Racz et al. [Rac+10] arbeitet die Forschungsziele, Methoden und den GRC-Stellenwert in Bezug auf die Prozesse von Organisationen anschaulich heraus.

Obwohl der Diskurs einen holistischen GRC-Ansatz für Unternehmen und Organisationen verfolgt, um die Effektivität ökonomischer Ziele zu verbessern, steht GRC noch nicht im Kontext einer Sicherheits- und Vertrauensstrategie. Die Konzeptansätze der vorliegenden Arbeit, insbesondere die semantische Domänen-Modellierung in Verbindung mit vertrauensbildenden Regulierungsansätzen, können bestehende GRC-Forschungsaufgaben methodologisch wie auch qualitativ erweitern.

Dynamisierung vertrauensbildender hoheitlicher Erweiterungen

Das Konzept zur Verlagerung vertrauenswürdiger Bereiche auf die Seite des Cloud-Anbieters reduziert sich auf eine technisch abgegrenzte Cloud-Architektur. Die Anwendung kryptographischer Protokolle für eine horizontale Erweiterung setzt notwendige Vertrauensanker zur Ausweitung regulierender Hoheits-

bereiche für einen Cloud-Anwender. Es hat den Anschein, dass sich diese vertrauensbildende Vorgehensweise in Widerspruch zu den vom NIST (National Institute of Standards and Technology) entwickelten Cloud-Anwendungsmodellen befindet.

In der NIST-Studie [MG09] bzw. im Standard [Hog+11] zählen die Elastizität (*Rapid elasticity*) und die dynamische Ressourcen-Bereitstellung (*On-demand self-service*) zu wesentlichen Nutzungsparadigmen. Dieser Widerspruch ist Ursache für eine nur geringe Unterstützung zur Verwendung kryptographischer Prozesse im Cloud-Architekturbereich.

Die ressourcenorientierte Flexibilität benötigt eine weitergehende Konzeptualisierung für die Dynamisierung vertrauenswürdiger Regulierungsbereiche. Aktuelle TCG-Standards für TPM 2.0 [Gro16] erweitern das Konzept migrierbarer kryptographischer Schlüssel (*Migratable Keys*) im Bereich der TPM-Lösungen. Dem vorliegenden ontologischen Modell fehlen dafür Konzeptklassen wie *Migration* in Verbindung mit spezifischen Konzeptklassen für Cloud Computing wie *Elasticity* und *Pooling*. Diese Erweiterungen beziehen sich im Wesentlichen auf die Ontologie-*Regulation* und bilden den zukünftigen regulierenden Ausgangspunkt, um dynamische Cloud-Konzepte gleichzeitig auf die Basis eines holistischen Vertrauensansatzes zu stellen.

Digitale Hoheit

Der gegenwärtig häufig verwendete Begriff *Digitale Souveränität* drückt eine Zielstellung der eigenen Arbeit aus. Die Fähigkeit eines Cloud-Anwenders, seinen regulierenden Einfluss auf die Seite eines Cloud-Anbieters zu erweitern, bei gleichzeitiger Zusicherung vertrauenswürdiger technischer Bedingungen, realisiert ein souveränes Handeln im digitalen Raum. In den Untersuchungen von Friedrichsen et al. [FB16] wird festgestellt: *Diese Souveränität ist derzeit wenig gegeben, da der Einzelne keine Hoheit über seine Daten im Internet, bei Unternehmen und Behörden hat, er die gespeicherten Daten nicht kennt und nicht weiss, welche Auswertungen daraus gezogen werden*.

Der Begriff *Hoheit* führt zu der eigentlichen Problemstellung und ist Gegenstand der Problembeschreibung in Abschnitt 2.2.2. Für den digitalen Raum existieren derzeit keine Instanzen zur Regulierung hoheitlicher Zielstellungen. Die Sicherstellung hoheitlicher Bereiche benötigt dafür Konzepte für die vertrauenswürdige Erweiterung bereits bestehender digitaler Hoheitsbereiche. In der eigenen Konzeptarbeit repräsentieren die Konzeptklassen *Authority* und vertrauenswürdige Entität (*Trustworthy entity*) einen Vertrauensansatz.

Die Entwicklung vertrauenswürdiger Entitäten ist ein technischer Anfang, der Beginn für die Anwendung hoheitlicher digitaler Repräsentanten, für die Transformation gesellschaftlicher Normen in digitale Umgebungen. Es genügt nicht, Regeln aufzustellen und Rechtsansprüche zu formulieren [Alb+16], die keine Instanzen für ihre Durchsetzung und Sanktionierung bei Verstoß besitzen. Hoheitliche Bereiche verlangen nach regulierenden digitalen Strukturen mit wirksamer Einflussnahme.

Die Fortsetzung des Konzeptrahmens bedarf einer strategischen Ausrichtung, weiterer Grundlagenarbeiten in Bereichen digitales Recht und Steuerung digitaler hoheitlicher Aufgaben und ist eng verbunden mit dem Konzept digitaler Autoritäten. Regulierende digitale Strukturen sind die infrastrukturelle Basis vertrauensbildender Konzepte und repräsentieren hoheitliche Zuständigkeiten in digitalen Umgebungen.

Abkürzungsverzeichnis

ACM Authentication Modul

API Application Programming Interface

ASM Abstract State Machine

ASPF Policy-based Security Framework

BIOS Basic Input/Output System

Bitkom Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.

BSI Bundesamt für Sicherheit in der Informationstechnik

CA Certificate Authority

CAFE Composite Application Framework

CC Common Criteria

CMA Consumer Management Agent

CPU Central Processing Unit

CRL Certificate Revocation List

DBSRV Database-Server

DBS Datenbanksystem

DL Description Logic

DN Distinguished Name

EK Endorsement Key

GRC Governance, Risk and Compliance

IaaS Infrastructure as a Service

IAAS Institut für Architektur von Anwendungssystemen

IAO Fraunhofer-Institut für Arbeitswirtschaft und Organisation

ISMS Information Security Management System

KVM Kernel-based Virtual Machine

MAC Mandatory Access Control

MBR Master Boot Record

MD5 Message-Digest Algorithm 5

MLE Measured Launch Environment

NIST National Institute of Standards and Technology

OASIS Organization for the Advancement of Structured Information Standards

OCL Object Constraint Language

OMG Object Management Group

ORDBMS Objektrelationales Datenbankmanagementsystem

OS Operating System

OWL Web Ontology Language

PaaS Platform as a Service

PCR Platform Configuration Register

PKI Public Key Infrastructure

PMA Provider Management Agent

QEMU Quick Emulator – Virtualisierungssoftware, die die gesamte Hardware eines Computers emuliert

RDF Resource Description Framework

RDN Relative Distinguished Name

ROM Read Only Memory

RTM Root of Trust for Measurement

RTR Root of Trust for Reporting

RTS Root of Trust for Storage

SaaS Software as a Service

SAML Security Assertion Markup Language

SGX INTEL – Software Guard Extension Technologie

SINIT-ACM Secure Initialization ACM

SLOC Source Lines of Code

SOAP Simple Object Access Protocol

SSL Secure Socket Layer

Tboot Kernel Module – Trusted Boot

TCB Trusted Computing Base

TCG Trusted Computing Group

TEI Trusted Entity Infrastructure

TLS Transport Layer Security

TMI Trusted Multi-Tenant Infrastructure

TOSCA OASIS Topology and Orchestration Specification for Cloud Applications

TPM Trusted Platform Module

TSD Trusted System Domain

TXT Intel – Trusted Execution Technology

UML Unified Modeling Language

VM Virtual Machine

VMM Virtual Machine Monitor

VT Intel – Virtualization Technology

WSDL Web Services Description Language

X.509 ITU-T-Standard für eine Public-Key-Infrastruktur zum Erstellen digitaler Zertifikate

XML Extensible Markup Language

Literatur

[aca13] acatech-(Hrsg.) *Privatheit im Internet. Chancen wahrnehmen, Risiken einschätzen, Vertrauen gestalten*. Springer Verlag, 2013.

[Alb+16] Jan Philipp Albrecht u. a. *Charta der Digitalen Grundrechte der Europäischen Union*. 2016. URL: https://digitalcharta.eu.

[Arn+11] John Arnold u. a. "Security Guidance for Critical Areas of Focus in Cloud Computing". In: CSA cloud security alliance, 2011.

[B M12] B. Parsia B. Motik P. F. Patel-Schneider. "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax W3C Recommendation". In: World Wide Web Consortium, December 2012. URL: http://www.w3.org/TR/owl2-syntax/.

[BA10] Mark Bedner und Tobias Ackermann. "Schutzziele der IT-sicherheit". In: *Datenschutz und Datensicherheit-DuD* 34.5 (2010), S. 323–328.

[Baa10] Franz Baader. *The Description Logic Handbook: Theory, Implementation and Applications"*. 2 edition. Cambridge University Press, 2010.

[Ban+12] Erin K. Banks u. a. *Trusted Geolocation in the Cloud: Proof of Concept Implementation (Draft)*. NIST Interagency Report 7904 (Draft). National Institute of Standards und Technology (NIST), 2012.

[Bar+08] Mark Bartel u. a. *XML Signature Syntax and Processing (Second Edition)*. Hrsg. von Donald Eastlake u. a. World Wide Web Consortium, Recommendation REC-xmldsig-core-20080610. Juni 2008.

[Bar+09a] Jeff Bardin u. a. "Security Guidance for Critical Areas of Focus in Cloud Computing". In: *Technischer Bericht*. CSA cloud security alliance, 2009.

[Bar+09b] Prashant Barot u. a. *Cloud Computing - Evolution in der Technik, Revolution im Business*. Hrsg. von Dr. Mathias Weber. BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V., 2009.

[Ben+12] Arthur Benz u. a., Hrsg. *Handbuch Governance: Theoretische Grundlagen und empirische Anwendungsfelder (German Edition)*. 2007. Aufl. VS Verlag für Sozialwissenschaften, Feb. 2012. ISBN: 9783531147482.

[Ben06] Messaoud Benantar. *Access control systems: security, identity management and trust models*. Springer Science & Business Media, 2006.

[BL73] D Elliott Bell und Leonard J LaPadula. *Secure computer systems: Mathematical foundations*. Techn. Ber. DTIC Document, 1973.

[BMI06] BMI. *Allgemeine Verwaltungsvorschrift des Bundesministeriums des Innern zum materiellen und organisatorischen Schutz von Verschlusssachen (VS-Anweisung - VSA) vom 31. März 2006*. Bundesministeriums des Innern, 2006.

[Bob+14] Laurent Bobelin u. a. "An advanced security-aware cloud architecture". In: *High Performance Computing & Simulation (HPCS), 2014 International Conference on*. IEEE. 2014, S. 572–579.

[Bon+16] Paolo Bonzini u. a. *QEMU (generic and open source machine emulator and virtualizer)*. 2016. URL: `http://www.qemu-project.org`.

[Bos+13] Pino Bosesky u. a. *Datenhoheit in der Cloud - Studie*. Lorenz-von-Stein-Inst. für Verwaltungswiss. an der Christian-Albrechts-Univ. Interdisziplinäre Studien zu Politik, Recht, Administration und Technologie e.V. (ISPRAT), 2013.

[Bou+15] Aline Bousquet u. a. "Enforcing security and assurance properties in cloud environment". In: *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE. 2015, S. 271–280.

[BR13] Reto Buerki und Adrian-Ken Rueegsegger. "Muen-an x86/64 separation kernel for high assurance". In: *University of Applied Sciences Rapperswil (HSR), Tech. Rep* (2013).

[Bre+12] Uwe Breitenbücher u. a. "Vino4TOSCA: A visual notation for application topologies based on TOSCA". In: *OTM Confederated International ConferencesÖn the Move to Meaningful Internet Systems"*. Springer. 2012, S. 416–424.

[Bre+13a] Uwe Breitenbücher u. a. "Pattern-based Runtime Management of Composite Cloud Applications". In: *Proceedings of the 3rd International Conference on Cloud Computing and Service Science, CLOSER 2013*. SciTePress, 2013.

[Bre+13b] Uwe Breitenbücher u. a. "Policy-Aware Provisioning of Cloud Applications". In: *Proceedings of the Seventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*. Xpert Publishing Services (XPS), 2013, S. 86–95.

[Bre+14] Uwe Breitenbücher u. a. "Policy-Aware Provisioning and Management of Cloud Applications". In: *International Journal On Advances in Security* 7.1&2 (2014), S. 15–36.

[BS03] Egon Börger und Robert Stärk. *Abstract State Machines - A Method for High-Level System Design and Analysis*. Springer-Verlag Berlin Heidelberg, 2003.

[BSI08] BSI. *Standard 100-2*. IT-Grundschutz-Vorgehensweise. 2008.

[BSI12] BSI. *Sicherheitsempfehlungen für Cloud Computing Anbieter*. 2012.

[BSI14] BSI. *IT-Grundschutz-Kataloge*. Techn. Ber. Bundesamt für Sicherheit in der Informationstechnik, 2014.

[Can+05a] Scott Cantor u. a. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. Organization for the Advancement of Structured Information Standards. 2005. URL: `https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[Can+05b] Scott Cantor u. a. *Bindings for the OASIS Security Assertion Markup Language V2.0 (SAML)*. Organization for the Advancement of Structured Information Standards. 2005. URL: https://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf.

[CC12a] CC. *Common Criteria for Information Technology Security Evaluation CEM: Common Methodology for Information Technology Security Evaluation*. Common Criteria, 2012.

[CC12b] CC. *Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model*. Common Criteria, 2012.

[CC12c] CC. *Common Criteria for Information Technology Security Evaluation Part 2: Security functional components*. Version 3.1, Revision 4. Common Criteria. 2012.

[CC12d] CC. *Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components*. 2012.

[CCK14] Chang Choi, Junho Choi und Pankoo Kim. "Ontology-based access control model for security policy reasoning in cloud computing". In: *The Journal of Supercomputing* 67.3 (2014), S. 711–722.

[COM12] EUROPEAN COMMISSION. *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)*. Techn. Ber. EUROPEAN PARLIAMENT und OF THE COUNCIL, Jan. 2012. URL: http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf.

[Coo+08] D. Cooper u. a. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". In: *Request for Comments: 5280, DOI 10.17487/RFC5280*. 2008.

[CRP06] Coral Calero, Francisco Ruiz und Mario Piattini. *Ontologies for software engineering and software technology*. Springer Science & Business Media, 2006.

[Cur+10] Sam Curry u. a. *Infrastructure Security: Getting to the Bottom of Compliance in the Cloud*. RSA Security Brief. 2010.

[CWL14] Richard Cyganiak, David Wood und Markus Lanthaler. "RDF 1.1 Concepts and Abstract Syntax". In: World Wide Web Consortium, 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[Cyb16] International Community Cybersecurity. *CVE - Common Vulnerabilities and Exposures (The Standard for Information Security Vulnerability Names)*. 2016. URL: https://cve.mitre.org.

[DRI+06] S DRITSAS u. a. "The Secure e-Poll Paradigm". In: *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government: 5th IFIP Conference on e-Commerce, e-Business, and e-Government (I3E'2005), October 28-30 2005, Poznan, Poland*. Bd. 189. Springer. 2006, S. 187.

[Eck13] Claudia Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Walter de Gruyter, 2013.

[Ehr13] Hartmut Ehrig. *Mathematisch-strukturelle Grundlagen der Informatik (Springer-Lehrbuch)*. 2. Aufl. 2001. Springer, Okt. 2013. ISBN: 9783540419235.

[FB16] Mike Friedrichsen und Peter Bisa. "Einführung–Analyse der digitalen Souveränität auf fünf Ebenen". In: *Digitale Souveränität*. Springer, 2016, S. 1–6.

[FE09] Stefan Fenz und Andreas Ekelhart. "Formalizing information security knowledge". In: *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*. ACM. 2009, S. 183–194.

[Fer13] Eduardo Fernandez-Buglioni. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns*. Wiley Software Patterns Series. John Wiley & Sons, 2013, 2013.

[Fes15] Norman Feske. *Transplantation of VirtualBox to the NOVA microhypervisor*. genode-labs. 2015.

[FG13] William Futral und James Greene. *Intel Trusted Execution Technology for Server Platforms: A Guide to More Secure Datacenters (Expert's Voice in Security)*. 1. Aufl. Apress, Sep. 2013. ISBN: 9781430261483.

[FKC03] David Ferraiolo, D Richard Kuhn und Ramaswamy Chandramouli. *Role-based access control*. Artech House, 2003.

[Gal13] Cameron F. Kerry; Patrick D. Gallagher. "FIPS PUB 186-4 Digital Signature Standard (DSS)". In: *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*. National Institute of Standards und Technology. 2013.

[Gmb12] semafora systems GmbH. "ObjectLogic Tutorial". In: (2012).

[GMT11] Matthias Gräuler, Benedikt Martens und Frank Teuteberg. "Entwicklung und Implementierung einer Ontologie für das IT-Sicherheitsmanagement im Cloud Computing". In: (2011).

[Gre12] James Greene. "Intel® Trusted Execution Technology - Hardware-based Technology for Enhancing Server Platform Security". In: Intel Corporation, 2012.

[Gro16] Trusted Computing Group. *Trusted Platform Module Library Specification, Family "2.0*. 2016.

[Gru+93] Thomas R Gruber u. a. "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2 (1993), S. 199–220.

[Gru11] Dr. Waldemar Grudzien. *Rechtliche Anforderungen an Cloud Computing – Sichere Cloud-Dienste*. 2011.

[Gut15] Peter Gutmann. *cryptlib security software*. 2015. URL: `https://www.cs.auckland.ac.nz/~pgut001/cryptlib/`.

[Hit05] M. Hitz. *UML@work: objektorientierte Modellierung mit UML 2*. dpunkt.Verlag, 2005. ISBN: 9783898642613.

[HLL10] Ruan He, Marc Lacoste und Jean Leneutre. "A policy management framework for self-protection of pervasive systems". In: *Autonomic and Autonomous Systems (ICAS), 2010 Sixth International Conference on*. IEEE. 2010, S. 104–109.

[Hoc14] Lorin Hochstein. *Ansible: Up and Running*. Ö'Reilly Media, Inc.", 2014.

[Hof16] Wolfgang Hoffmann-Riem. *Innovation Und Recht - Recht Und Innovation: Recht Im Ensemble Seiner Kontexte (German Edition)*. Mohr Siebeck, Mai 2016. ISBN: 9783161544415.

[Hog+11] Michael Hogan u. a. "Nist cloud computing standards roadmap". In: *NIST Special Publication* 35 (2011).

[Hoh+04] Michael Hohmuth u. a. "Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors". In: *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM. 2004, S. 22.

[Höl+15] Tobias Höllwarth u. a. *Cloud Privacy Check(CPC) (Datenschutzrechtliche Anforderungen, die ein Kunde vor der Nutzung von Cloud Services einhalten muss)*. EuroCloud Deutschland_eco e.V., 2015.

[HSD07] Almut Herzog, Nahid Shahmehri und Claudiu Duma. "An ontology of information security". In: *International Journal of Information Security and Privacy (IJISP)* 1.4 (2007), S. 1–23.

[HSt11] H.Stuckenschmidt. *Ontologien:* Informatik im Fokus. Springer, 2011. ISBN: 9783642054044. URL: https://books.google.de/books?id=ndwfBAAAQBAJ.

[Hum+14] Thorsten Humberg u. a. "Using Ontologies to Analyze Compliance Requirements of Cloud-Based Processes". In: *Cloud Computing and Services Science*. Springer, 2014, S. 36–51.

[HW07] Petra Hofstedt und Armin Wolf. *Einführung in die Constraint-Programmierung: Grundlagen, Methoden, Sprachen, Anwendungen*. Springer-Verlag, 2007.

[I+02] Takeshi Imamura, Blair Dillaway, Edi Simon u. a. "XML encryption syntax and processing". In: *W3C recommendation* 10 (2002).

[IAD07] IAD. *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03*. 2007.

[Int16a] Intel. "Intel® 64 and IA-32 Architectures Software Developer's Manual". In: Bd. Volume 3A: System Programming Guide, Part 1. Intel, 2016.

[Int16b] Intel. "Intel® 64 and IA-32 Architectures Software Developer's Manual". In: Bd. Volume 3C: System Programming Guide, Part 3. Intel, 2016.

[Int16c] Intel. *Intel® Virtualization Technology for Directed I/O (Architecture Specification)*. 2016.

[Int16d] Intel. *Intel® Virtualization Technology for Directed I/O – Architecture Specification*. 2016. URL: http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/vt-directed-io-spec.pdf.

[ISO13] ISO. *27001 – Information security management systems – Requirements*. Informationstechnik - IT-Sicherheitsverfahren - Informationssicherheits-Managementsysteme - Anforderungen. 2013.

[ISO14] ISO. *ISO/IEC 9594-8:2014 Information technology – Open Systems Interconnection – The Directory – Part 8: Public-key and attribute – certificate frameworks*. Techn. Ber. International Organization for Standardization, 2014.

[ISO88] ISO. *ISO 9660:1988 Information processing – Volume and file structure of CD-ROM for information interchange*. Techn. Ber. International Organization for Standardization, 1988.

[Jür05] Jan Jürjens. *Secure systems development with UML*. Springer Science & Business Media, 2005.

[Keb+15] Jörg Kebbedies u. a. "Conceptualized Policy Design for User-Regulated Trusted Clouds". In: *8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, December 7-10, 2015*. 2015, S. 601–606. DOI: 10.1109/UCC.2015.105. URL: http://doi.ieeecomputersociety.org/10.1109/UCC.2015.105.

[KFJ03] Lalana Kagal, Tim Finin und Anupam Joshi. "A policy language for a pervasive computing environment". In: *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*. IEEE. 2003, S. 63–74.

[KFJ05] Lalana Kagal, Tim Finin und Anupam Joshi. *Developing secure agent systems using delegation based trust management*. Techn. Ber. DTIC Document, 2005.

[Kim+16] Gene Kim u. a. "The DevOps handbook". In: (2016).

[KK92] Alfred Kieser und Herbert Kubicek. *Organisation*. 3. völlig neu bearbeitete Auflage. ISBN 3-11-013499-3. de Gruyter, 1992.

[KL89] Michael Kifer und Georg Lausen. "F-logic: A Higher-order Language for Reasoning About Objects, Inheritance, and Scheme". In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. SIGMOD '89. Portland, Oregon, USA: ACM, 1989, S. 134–146. ISBN: 0-89791-317-5. DOI: 10.1145/67544.66939. URL: http://doi.acm.org/10.1145/67544.66939.

[Klu16] Felix Kluge. "Entwicklung und Konzeption zur Umsetzung einer Transformation von einer ontologischen beschriebenen Policysemantik in eine sichere agentenbasierte Ablaufsteuerung". Magisterarb. Technische Universität Dresden, 2016.

[Kne10] Martin Knechtel. *Access Restrictions to and with Description Logic Web Ontologies*. Dissertation. Technische Universität Dresden, 2010.

[Koe+14] Falko Koetter u. a. "Unifying Compliance Requirements across Business and IT". English. In: *Proceedings of the IEEE EDOC Conference*. IEEE, Sep. 2014, S. 1–10. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2014-79&engl=1.

[KVM16] Project Community KVM. *Kernel Virtual Machine (full virtualization solution for Linux on x86 hardware)*. 2016. URL: https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792.

[Les06] Lawrence Lessig. *Code: And Other Laws of Cyberspace, Version 2.0*. 2nd Revised ed. Basic Books, Dez. 2006. ISBN: 9780465039142.

[Lib16] Project Community LibVirt. *libvirt – Virtualization API*. 2016. URL: https://libvirt.org.

[Luh11] N. Luhmann. *Organisation und Entscheidung*. Rheinisch-westflische Akademie Der Wissenschaften. VS Verlag für Sozialwissenschaften, 2011. ISBN: 9783531178172. URL: http://books.google.de/books?id=gBI0GxehKCQC.

[Luh14] N. Luhmann. *Vertrauen: Ein Mechanismus der Reduktion sozialer Komplexität*. Uni-Taschenbücher S. UTB GmbH, 2014. ISBN: 9783825240042. URL: http://books.google.de/books?id=KJP0AgAAQBAJ.

[Luh87] Niklas Luhmann. *Soziales System*. suhrkamp taschenbuch. Suhrkamp Verlag Frankfurt am Main, 1987.

[MG09] P Mell und T Grance. *Draft NIST working definition of cloud computing, Referenced on June 3rd (2009)*. 2009.

[Mir16] Mircrosoft. *Hyper-V (Hypervisor-basierte Virtualisierungstechnik)*. 2016. URL: https://www.microsoft.com/de-de/cloud-platform/server-virtualization.

[Mot+12] B. Motik u. a. *OWL 2 Web Ontology Language Profiles (Second Edition)*. Techn. Ber. Wide Web Consortium, December 2012. URL: http://www.w3.org/TR/owl2-syntax/.

[Nad+06] Anthony Nadalin u. a. *Web Services Security: SOAP Message Security 1.1 (ws-security 2004)*. 2006. URL: https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[Nad+12a] Anthony Nadalin u. a. *WS-SecurityPolicy 1.3*. Norm. 2012. URL: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.html#_Toc325573553.

[Nad+12b] Anthony Nadalin u. a. *WS-Trust 1.4. 25 April 2012. OASIS Standard incorporating Approved Errata*. 2012. URL: http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/errata01/os/ws-trust-1.4-errata01-os-complete.html.

[NSA87] NSA. "A Guide to Understanding Discretionary Access Control In Trusted Systems". In: Bd. The Rainbow Books. NCSC-TG-003-87. NATIONAL COMPUTER SECURITY CENTER, 1987.

[OAS13] OASIS. "Topology and Orchestration Specification for Cloud Applications Version 1.0". In: *Organization for the Advancement of Structured Information Standards* (18 March 2013).

[Obj15] Object Management Group. "Unified Modeling Language (UML) Version 2.5". In: *Normative Documents* formal/15-03-01 (2015).

[Ope16] Project Community OpenSSL. *OpenSSL OpenSSL – Cryptography and SSL/TLS Tool*. 2016. URL: https://www.openssl.org.

[Ora16] Oracle-Corporation. *Oracle VM VirtualBox – Technical documentation*. Oracle, 2016. URL: http://download.virtualbox.org/virtualbox/UserManual.pdf.

[Ott07] Amon Ott. *Mandatory Rule Set Based Access Control in Linux: A Multi-policy Security Framework and Role Model Solution for Access Control in Networked Linux Systems*. Aachen, Germany, Germany: Shaker Verlag GmbH, Germany, 2007. ISBN: 383226423X, 9783832264239.

[Pan+12] J.Z. Pan u. a. *Ontology-Driven Software Development*. SpringerLink : Bücher. Springer Berlin Heidelberg, 2012. ISBN: 9783642312267.

[Pea09] Siani Pearson. "Taking account of privacy when designing cloud computing services". In: *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*. IEEE Computer Society. 2009, S. 44–52.

[PM] Penny Pritzker und Willie E. May. "Secure Hash Standard (SHS)". In: *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*. National Institute of Standards und Technology.

[Pos17] Project Community PostgreSQL. *PostgreSQL – objektrelationales Datenbankmanagementsystem*. 2017. URL: `https://www.postgresql.org`.

[Pro16] EU Projekt. *Secure Enclaves for REactive Cloud Applications (SERECA)*. 2016. URL: `http://www.serecaproject.eu`.

[Rac+10] Nicolas Racz u. a. "Governance, risk & compliance (GRC) status quo and software use: results from a survey among large enterprises". In: *ACIS 2010 Proceedings, Paper* 21 (2010).

[Rec14] G. Recht. *Bundesdatenschutzgesetz (BDSG) (German Edition)*. CreateSpace Independent Publishing Platform, Juni 2014. ISBN: 9781500100025. URL: `http://amazon.com/o/ASIN/1500100021/`.

[red] redHat. *Continuous Integration & Delivery with Ansible*. URL: `https://www.ansible.com/it-automation`.

[Red17a] Inc Red Hat. *ansible module library*. 2017. URL: `http://docs.ansible.com/ansible/modules.html`.

[Red17b] Inc Red Hat. *ansible stat-module*. 2017. URL: `http://docs.ansible.com/ansible/stat_module.html`.

[Ris13] I. Ristic. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Computers / Security. Feisty Duck, 2013. ISBN: 9781907117046.

[Rod+12] M. Rode u. a. *KVM Best Practices: Virtualisierungslösungen für den Enterprise-Bereich*. dpunkt.verlag, 2012. ISBN: 9783864911170.

[RSA12] RSA-Laboratories. "PKCS #1: RSA Cryptography Specifications Version 2.2". In: *Request for Comments: 3447*. Network Working Group. 2012.

[RSA98] RSA-Laboratories. "PKCS #13: Elliptic Curve Cryptography Standard". In: RSA Laboratories. 1998.

[Rus81] John Rushby. "Design and Verification of Secure Systems". In: *Reprint of a paper presented at the 8th ACM Symposium on Operating System Principles*. Bd. 15. 5. Pacific Grove, California, Dez. 1981, S. 12–21. URL: `http://www.csl.sri.com/papers/sosp81/`.

[Rüt+10] Andreas Rüter u. a. *IT-Governance in der Praxis: Erfolgreiche Positionierung der IT im Unternehmen. Anleitung zur erfolgreichen Umsetzung regulatorischer und wettbewerbsbedingter Anforderungen (Xpert.press)*. Hrsg. von Andreas Rüter u. a. Springer, 2010.

[Sch15] Gunnar Folke Schuppert. "The World of Rules: Eine etwas andere Vermessung der Welt". In: (2015).

[SKi95] S.Kille. *A String Representation of Distinguished Names*. ISODE Consortium, 1995. URL: http://www.rfc-editor.org/info/rfc1779.

[SR09] W. Streitberger und A. Ruppel. *Cloud Computing Sicherheit: Schutzziele, Taxonomie, Marktübersicht*. Fraunhofer-Institut für Sichere Informationstechnologie SIT, 2009. URL: http://books.google.de/books?id=JgNUcgAACAAJ.

[Sta15] G. Starke. *Effektive Softwarearchitekturen: Ein praktischer Leitfaden*. Carl Hanser Verlag GmbH & Company KG, 2015. ISBN: 9783446444065.

[Teu14] Gunther Teubner. *Recht und Sozialtheorie*. 2014.

[Tro08] Project Community TrouSerS. *TrouSerS TSS 1.2 (The open-source TCG Software Stack)*. 2008. URL: http://trousers.sourceforge.net.

[Tro16] Project Community TrouSerS. "TPM-Tools Version 3.9". In: (2016). URL: https://sourceforge.net/projects/trousers/files/tpm-tools/.

[Tru11a] Trusted Computing Group. *TPM Main Part1 Design Principles*. 2011.

[Tru11b] Trusted Computing Group. *Trusted Multi-Tenant Infrastructure Use Cases Specification*. 2011.

[Tru13a] Trusted Computing Group. *TCG TMI Reference Framework*. 2013.

[Tru13b] Trusted Computing Group. *Trusted Multi-Tenant Infrastructure Work Group Use Cases*. 2013.

[Tru14] Trusted Computing Group. *TPM Keys for Platform Identity for TPM 1.2*. 2014.

[Tru16] Project Community TrustedBoot. *Trusted Boot (open source, pre- kernel/VMM module)*. 2016. URL: https://sourceforge.net/projects/tboot/?source=navbar.

[Tur16] James Turnbull. *The Docker Book*. 2016. URL: https://www.dockerbook.com.

[Twi+09] Kevin Twidle u. a. "Ponder2: A policy system for autonomous pervasive environments". In: *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*. IEEE. 2009, S. 330–335.

[Ved+07] Asir S Vedamuthu u. a. *Web Services Policy 1.5 - Framework*. Norm. 2007. URL: http://www.w3.org/TR/2007/REC-ws-policy-20070904/#WS-SecurityPolicy.

[Vel+09] Joaquin Lasheras Velasco u. a. "Modelling reusable security requirements based on an ontology framework". In: *Journal of Research and Practice in Information Technology* 41.2 (2009), S. 119.

[VMw16] Inc. VMware. *VMWare vSphere*. 2016. URL: http://www.vmware.com/products/vsphere.html.

[Wei14] Tim Weilkiens. *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur. Mit einem Geleitwort von Richard Mark Soley (German Edition)*. Sep. 2014.

[Wei95] K.E. Weick. *Der Prozess des Organisierens*. Suhrkamp-Taschenbuch Wissenschaft. Suhrkamp, 1995. ISBN: 9783518287941. URL: `http://books.google.de/books?id=-JM1SgAACAAJ`.

[Wit10] Bernhard C. Witt. *Datenschutz kompakt und verständlich: Eine praxisorientierte Einführung (Edition) (German Edition)*. 2., akt. und erg. Aufl. 2010. Vieweg+Teubner Verlag, März 2010.

[YC14] Raghuram Yeluri und Enrique Castro-Leon. *Building the Infrastructure for Cloud Security: A Solutions View (Expert's Voice in Internet Security)*. 1. Aufl. Apress, März 2014. ISBN: 9781430261452.

[Yee13] P. Yee. "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". In: *Request for Comments: 6818, DOI 10.17487/RFC6818*. 2013.

[ZB15] Sherali Zeadally und Mohamad Badra, Hrsg. *Privacy in a Digital, Networked World: Technologies, Implications and Solutions (Computer Communications and Networks)*. Okt. 2015.

[Zha09] Mingxi Zhang. "Strict integrity policy of Biba model with dynamic characteristics and its correctness". In: *Computational Intelligence and Security, 2009. CIS'09. International Conference on*. Bd. 1. IEEE. 2009, S. 521–525.

I | State-of-the-Art – Kategorien

Für die Erhebung des State-of-the-Art dient ein Kriterienkatalog als Grundlage für eine differenzierte Erfassung von Entwicklungsarbeiten im Bereich Policies und digitaler Vertrauenskonzepte.

Projektname / Referenz:	
Universität / Organisation: Ziele des Projekts / Standards:	
Kategorie	Strukturierte Policyumsetzung

***Regulierung und Policy*: Ziele der Regulierung**		
PZi1	Regulierung von Sicherheitsfunktionen?	–
PZi2	Policy für Trust vorhanden?	–
PZi3	Regulierung von Sicherheitsmodellen?	–
PZi4	Regulierung von Privacy und Datenschutz?	–
PZi5	Regulierung von Funktion und Deployment?	–
PZi6	Regulierung von operativem Management?	–
Klassifikation		
PRu1	Verwendung typisierter Regeln?	–
PRu2	Typen:	–
Policysteuerung		
PSt1	Maßnahmen zur Konflikterkennung	–
PSt2	Unterstützung von Prioritäten	–
PSt3	Werden Workflows für Policyumsetzung unterstützt?	✓
PSt4	Funktionen für die Steuerung von Policies	✓
Separierung		
PSp1	Unterstützung von Policy Domains?	✓
PSp2	Unterstützte Architekturebene für Policy	Angabe von IT-Bereichen, auf die sich die Policies beziehen
Sicherheit		
PSec	Wird eine Policy als Asset behandelt und durch Sicherheitsmaßnahmen geschützt?	–

***Bildung von Vertrauen*: Prinzipien**		
VPr1	Kommen vertrauenswürdige Agenten zum Einsatz?	–
VPr2	Existieren hardwarebasierte Vertrauenskonzepte?	–
VPr3	Werden kryptographische Identitäten genutzt?	–
VPr4	Werden Evaluationsprozesse betrachtet?	–
VPr5	Werden Vertrauensketten verwendet?	–
VPr6	Existieren Protokolle für Vertrauensketten?	–
VPr7	Beziehen sich Vertrauenskonzepte auf Architekturebenen?	–
VPr8	Kann das Niveau an Vertrauen berechnet werden?	–

Aspekte der Formalisierung		
PFo1	Welche syntaktische Repräsentation wird verwendet?	Angabe der formalen Beschreibungsmethode
PFo2	Liegt eine eigenständige Konzeptualisierung vor?	✓Angabe, welche Konzeptualisierung für *Vertrauen* verwendet wird
PFo3	Können Policies mehrere Rules enthalten?	–
PFo4	Können Policies gruppiert werden?	✓Angabe der Aspekte für eine Policy-Gruppierung
PFo5	Können Policies verschachtelt werden?	–
PFo6	Wird die Formalisierung eines Kontexts geführt?	–
Systemzustand		
SSy1	Wird der Systemzustand formal behandelt?	✓
SSy2	Unterstützte Architekturebene für Systemzustand?	Angabe von IT-Bereichen, auf die sich die Betrachtung zum Systemstatus beziehen
Sicherheitszustand		
SSi1	Wird der Sicherheitszustand formal behandelt?	✓
SSi2	Unterstützte Architekturebene für Sicherheitszustand?	Angabe von IT-Bereichen, auf die sich die Betrachtung zum Sicherheitsstatus beziehen

Aspekte der Attestierung und Transformation		
ATT1	Wird Vertrauen durch Attestierung zugesichert?	–
ATT2	Wird der Systemzustand durch Attestierung zugesichert?	–

ATT3	Wird der Sicherheitszustand durch Attestierung zugesichert?	–
Transformation von Policies		
TRa1	Besteht Schnittstellenkonzept für Transformation?	✓
TRa2	Welche Technologien zur Transformation werden unterstützt ?	Angabe von Technologien für eine Policy-Transformation

Zusammenfassung	
Ziel:	
Regulierungsbereich:	
Sicherheit:	
Vertrauen:	
Abgrenzung:	

II | Hardwareunterstützte Sicherheit für eine IT-Plattform

II.1 Trusted Platform Module

Die Trusted Computing Group (TCG) stellt im Rahmen ihrer Trusted Platform Module (TPM) Work-Group-Spezifikationen [Tru11a] Eigenschaften und Konzepte hardwarebasierter Kryptokomponenten für die Entwicklung und Anwendung von Sicherheitsprotokollen bereit.
TPMs bieten einen Zugriffsschutz auf kryptographische Assets eines IT-Systems, die als Grundlage für die Etablierung von Vertrauenskonzepten eingesetzt werden. Hardwarebasierte Kryptographie garantiert eine von außen nicht manipulierbare und sichere Speicherung sowohl von kryptographischen Schlüsseln als auch von Systemeigenschaften, die im Rahmen einer vertrauenswürdigen Sicherheitsbewertung eine hohe Relevanz besitzen.

II.2 Technologie für IT-Plattformsicherheit

Das Thema einer vertrauenswürdigen IT-Plattform wurde durch die Firma Intel im Jahr 2012 unter dem Technologienamen *Trusted Execution Environment* beschrieben [FG13] und technisch verfügbar gemacht.
Die Firma Intel folgt den Standards und Spezifikationen der Trusted Computing Group (TCG) durch die Integration von TPMs als fest installierten Bestandteil der Intel-Hardware. Die TXT-Technologie erlaubt die Anwendung folgender TCG-Konzepte:

Root of Trust

Es beschreibt einen Ausgangspunkt zur Sicherstellung eines erwarteten Verhaltens aller in einer IT-Plattform integrierten Komponenten. Den Ausgangspunkt bildet dabei eine separierte, nicht manipulierbare und bezüglich ihres Funktionsverhaltens als vertrauenswürdig eingestufte Plattformkomponente.

Root of Trust for Storage – RTS

Es beschreibt die Fähigkeit einer IT-Plattform, ermittelte kryptographische Werte als Ergebnis einer Integritätsüberprüfung und andere sensitive Daten sicher zu speichern.

Root of Trust for Measurements – RTM

Es beschreibt die technische Fähigkeit einer IT-Plattform, die Integrität und Authentizität ihrer beteiligten Komponenten sicher und nicht manipulierbar zu erfassen. Darunter wird die erste vertrauenswürdige Komponente verstanden, welche eine Messung ausführt und integritätsgeschützt im RTS speichert.

Root of Trust for Reporting – RTR

Es beschreibt die Fähigkeit einer IT-Plattform, gesicherte Systemeigenschaften zuverlässig und überprüfbar bereitzustellen.

II.3 Konzept einer hardwarebasierten Vertrauenspolitik

Das Konzept einer hardwarebasierten Vertrauenspolitik (*Root of Trust*) ermöglicht die stufenweise Absicherung darauf aufbauender Plattformbausteine und bescheinigt die Konformität für dynamisch hinzugefügte Betriebssystemkomponenten und Anwendungen. Zur Bestimmung der Vertrauenswürdigkeit sind messbare und und vor allem überprüfbare Eigenschaften erforderlich. Nachfolgend werden Grundlagen beschrieben, um Eigenschaften der IT-Plattform-Bestandteile sicher ermitteln und überprüfen zu können.

II.3.1 Sichere Mikroarchitektur

TPMs führen hardwarebasierte kryptographische Funktionen für die Verschlüsselung oder für die elektronische Signatur von Daten aus. In dem hier vorgestellten Ansatz bilden ein TPM, der Systemprozessor und der zugehörige Chipsatz eine ganzheitliche hardwareintegrierte Sicherheitsarchitektur. Das TPM besitzt eine Immunität bezüglich der Veränderbarkeit gespeicherter Plattformwerte. Ein erweiterter Sicherheitsbefehlssatz des Prozessors und der zugehörige Chipsatz mit spezialisierten Registern (Platform Configuration Register – PCR) steuern die Absicherung kryptographisch ermittelter Statuswerte im TPM. Die integrierte Sicherheitsarchitektur ist Ausgangspunkt für ein Vorgehensmodell zur sicheren Bestimmung der Vertrauenswürdigkeit aller Systembausteine der IT-Plattform. Ausgehend von einem vertrauenswürdigen Mikrocode des Chipherstellers erfolgt eine kooperative Etablierung von weiterem Vertrauen (*Chain of Trust*). Das Gesamtkonzept zur Überprüfung der Systemeigenschaften einer IT-Plattform wird in zwei Phasen vorgenommen:

1. Messung der statischen Eigenschaften der Hardwareinfrastruktur.
2. Kontrollierter Systemstart durch Messung der dynamischen Eigenschaften des Systems.

   (In Kebbedies et al. [Keb+15] erfolgt eine Darstellung der sicherheitsrelevanten Abhängigkeiten in Form eines Verhaltensdiagramms.)

II.3.2 Messung statischer Systemeigenschaften

Nach Aktivierung des Mikrocodes (*Root of Trust for Measurements*) erfolgt die kryptographische Überprüfung der integrierten Komponenten einer IT-Plattform bezüglich ihrer Korrektheit und Konfigurationen vor ihrer Aktivierung. Für die Prüfung stehen elektronisch signierte Prüfmodule (*Authenticated Code Module – ACM*) einer IT-Plattform zur Verfügung.
Signierte Prüfmodule besitzen kryptographische Eigenschaften, um ihre Herkunft (*Authentizität*) und ihre Unversehrtheit (*Integrität*) überprüfen zu können. Es ist Aufgabe des Mikrocodes, die Vertrauenswürdigkeit des ersten Prüfmoduls zu bestätigen. Anschließend steuert das Prüfmodul die Überprüfung der Eigenschaften von weiteren IT-Plattformbestandteilen. Das Prüfmodul verifiziert einen entsprechend signierten Teil des BIOS, der nach erfolgreicher Überprüfung und eigener Aktivierung den restlichen BIOS-Code überprüft.

Die Messwerte jeder Überprüfung werden als Hash-Wert in einer nicht veränderbaren Form in kryptographisch gesicherte PCRs des TPMs gespeichert (*Root of Trust for Storage*) und beschreiben einen bis zu diesem Zeitpunkt ermittelten Systemzustand der IT-Plattform. Nachfolgende Prüfschritte setzen auf die gespeicherten Messwerte auf und erweitern kryptographisch den bestehenden Zustand. Für alle nachfolgenden Messungen wiederholt sich diese Form der kryptographischen Erweiterung.
Die kryptographische Zustandserweiterung verhindert, dass kryptographische Werte vorhersagbar sind und darüber ein Angriffsvektor zur Manipulation der Messwerte entstehen kann. Nach jedem System-Neustart findet unter Anwendung der beteiligten PCRs des TPMs eine kryptographische Neuberechnung des statischen Systemzustands Z_{pconf} statt, um die Unversehrtheit der IT-Plattform durch einen Wertvergleich nachzuweisen.
Konformitätsanforderungen zur IT-Plattformkonfiguration werden über eine Policy P_{pconf} eingeführt:

Definition II.1 *Policy zur Beschreibung einer Plattformkonfiguration*
P_{pconf} = $\{c_{i \in N} \mid$ *Plattform Konfigurationen* $c_i \in C_P$*, die einzuhalten sind*$\}$

Die Policy enthält Vorgaben über einen festen Satz an Konfigurationen c_i einer Plattformkonfiguration C_P (z. B. BIOS Version, ROM Versionen, MBR Konfiguration) [Tru11a].

II.4 Kontrollierter Systemstart

Ein kontrollierter Systemstart (Measured Launch) verwendet dieselben Prinzipien zur Messung von Sicherheitsbedingungen während der Initialisierung, der Startphase eines Host Operating Systems (Host-OS) und der dazugehörigen Virtualisierungssoftware (Virtual Machine Managers – VMM).
Host-OS und VMM-Komponenten starten auf einer überprüften und gültigen Plattformkonfiguration (*Measured Launch Environment* – MLE). Der Start selbst erfolgt im *Protected Mode* des Prozessors.
Es steht wie in Phase 1 ein ACM bereit (Secure Initialization ACM), dessen Echtheitsbestätigung schon während der Überprüfung der Plattformkonfiguration erfolgte. Das SINIT-ACM überprüft die OS/VMM-Softwarekomponenten, ihre Konfigurationen und die Ausführungskomponenten vor dem Systemstart. Die Überprüfung der Korrektheit des *Master Boot Records* (MBR) ist bereits Teil der IT-Plattformüberprüfung. Jeder Überprüfungsschritt speichert und erweitert kryptographisch die Messwerte in gesonderte PCRs des TPMs für den Systemstart.
Die Policy $P_{plaunch}$ definiert Vorgaben zur Regulierung des Systemstarts. Beispiel dafür sind Festlegungen über zugelassene OS/VMM-Softwarekomponenten (elektronisch signierte OS/VMM-Softwarekomponenten). Die Vorgaben entstehen in enger Zusammenarbeit mit den Herstellern der Softwarekomponenten.
Nach jedem System-Neustart findet unter Anwendung der beteiligten PCRs des TPMs eine kryptographische Neuberechnung des dynamischen Laufzeitzustands $Z_{plaunch}$ statt, um den korrekten Systemstart durch Wertvergleich nachzuweisen. Mit der Bestätigung erhält die Laufzeitumgebung die technische Autorisierung, um auf Bereiche des TPMs zugreifen zu können. Sie ist berechtigt, Aufgaben einer sicheren Attestierung zu übernehmen und steuert die sichere Kommunikation zwischen einer Cloud-Plattform und einem System des Cloud-Anwenders.

II.4.1 Identifizierbarer Plattform-Eigentümer

Für die Übernahme einer Plattform in den Verantwortungsbereich eines Cloud-Anbieters werden kryptographische Methoden zur Aktivierung des integrierten TPMs benutzt. Die Anwendung dieser Funktion kennzeichnet eine bewusste und nachweisbare Inbesitznahme des Systems als Plattform-Eigentümer.
Das TPM-System enthält einen asymmetrischen Aktivierungsschlüssel K_E (*Endorsement Key – EK*) mit seinen Bestandteilen $K_{E_{pub}}$ und $K_{E_{priv}}$, um die Plattform gegen unberechtigte Manipulation zu schützen. Im Prozess zur Autorisierung des Betreibers der Plattform wird unter Nachweis seiner physischen Präsenz ein Autorisierungsschlüssel (*Credential*) zur Anwendung des Schlüssels K_E festgelegt. Im Ergebnis führt dieser Prozessschritt zur Aktivierung und Freischaltung des TPMs. Die darüber hergestellte Bindung zwischen Betreiber und dem Schlüssel $K_{E_{pub}}$ schafft die Grundlage zur Überprüfbarkeit der Eigentümer-Identität.

II.4.2 Versiegeln von Systemwerten (*Sealing*)

Das Versiegeln von Geheimnissen (*Secrets*) ist eine Eigenschaft des TPMs, um eine hardwarebasierte Ablaufsteuerung durchsetzen zu können. Geheimnisse sind zu schützende Datenwerte, die im TPM verschlüsselt gespeichert werden.
Ausgehend von einer allgemeinen Abbildung $f : P \times K_1 \rightarrow C$ zur Beschreibung der Verschlüsselung mit P als Klartextmenge, C als Geheimtextmenge und K_1 als eine im TPM gebildete Schlüsselmenge, wird bei der Versiegelung der Schlüssel $k_i \in K_1$ aus den bereits hardwarebasierten Zuständen Z_{pconf} und $Z_{plaunch}$ abgeleitet. Als verschlüsselte Geheimnisse können z. B. wiederum kryptographische Schlüssel $k_i \in K_2$ dienen, mit K_2 als eigene Schlüsselmenge.
Das Konzept der Ableitung kryptographischer Schlüssel aus nicht veränderbaren kryptographisch erzeugten Systemzuständen ermöglicht einen technischen Nachweis eingehaltener Konformitätsansprüche an eine konkrete Cloud-Plattform. Nur unter der Bedingung, dass die korrekten Zustände Z_{pconf} und $Z_{plaunch}$ vorliegen, lässt sich das Geheimnis entschlüsseln und eine kontrollierte Anschlusshandlung ausführen.

II.5 Konzept der Attestierung

Die sichere Attestierung ist ein Prozess zur Bescheinigung, dass die übermittelten Eigenschaften (*TPM-Quote*) einer IT-Plattform und ihrer gestarteten OS/VMM-Komponenten unversehrt und authentisch sind. Dabei ist eine Überprüfung der Vertrauenswürdigkeit einer *TPM-Quote* möglich.

II.5.1 Attestierungs-Schlüssel

Für ein flexibles Konzept zur Attestierung von IT-Plattformeigenschaften sind Attestierungs-Identifikationsschlüssel K_A erforderlich. Attestierungs-Identifikationsschlüssel bilden das Bindeglied zwischen einer identifizierbaren IT-Plattform und Aufgaben der IT-Plattform-Attestierung mit unterschiedlichen Cloud-Anwendern.
Attestierungs-Identifikationsschlüssel sind asymmetrische Schlüsselpaare $K_{A_{pub}}$ und $K_{A_{priv}}$, die im TPM der entsprechenden IT-Plattform erzeugt werden. Es können mehrere solcher Schlüssel erzeugt und

verwendet werden. Der Einsatz von $K_{A_{priv}}$ in digitalen Signaturen erlaubt eine spätere kryptographische Überprüfbarkeit der übermittelten IT-Plattformeigenschaften.

II.5.2 Zertifizierung des Attestierungs-Identifikationsschlüssels

Die Nutzung des Schlüssels K_A erfordert den Nachweis seiner Bindung an ein konkretes TPM. Die Identifizierbarkeit eines TPM ist allein über den Aktivierungsschlüssel K_E möglich. Die Zertifizierung von K_A bestätigt seine Zugehörigkeit zur IT-Plattform mit dem Aktivierungsschlüssel K_E.
Die Zertifizierung verfolgt zwei Zielstellungen:

1. Nachweis einer kryptographischen Bindung zwischen dem Aktivierungsschlüssel K_E und einem Attestierungs-Identifikationsschlüssel $K_{A_{pub}}$
2. Bestätigung und Bereitstellung einer TPM-basierten IT-Plattformidentität

Eine CA (*Certificate Authority*) bescheinigt, dass der Schlüssel $K_{A_{pub}}$ von einem identifizierbaren TPM erzeugt wurde. Dafür sendet das TPM, neben dem Schlüssel $K_{A_{pub}}$, auch seinen öffentlichen Aktivierungsschlüssel $K_{E_{pub}}$ an die CA.
Wenn M_{KA} eine Menge der Attestierungs-Identifikationsschlüssel K_A bildet, die von einem TPM mit dem Aktivierungsschlüssel K_E ausgestellt wurde, so bildet der Prozess der Zertifizierung formal eine Abbildung $c : K_E, M_{KA} \rightarrow M_{ZA}$ für die Ausstellung einer Menge M_{ZA} von Attestierungszertifikaten. Die Ausstellung eines Attestierungszertifikats $z_i \in M_{ZA}$ ist der Nachweis darüber, dass ein Attestierungs-Identifikationsschlüssel K_A an ein identifizierbares TPM mit dem Aktivierungsschlüssel K_E gebunden ist.
Mit dem Besitz des öffentlichen Schlüssels Z_A sind Cloud-Anwender in der Lage, eine konkrete IT-Plattform hinsichtlich ihrer Echtheit zu verifizieren.

II.5.3 Attestierungs-Modul

Für die Übermittlung einer *TPM-Quote* an einen externen Attestierungs-Service (*Remote Attestation*) sind Attestierungs-Module erforderlich.
Ein Attestierungs-Modul erhält seine Vertrauenswürdigkeit, da es sich als Bestandteil einer überprüften OS/VMM-Umgebung ausweisen kann und selbst einen Teil der bestehenden IT-Plattform-Vertrauenskette bildet. Das Attestierungs-Modul besitzt die Autorisierung, eine gesicherte Datenaustauschkommunikation mit dem TPM (*Root of Trust for Reporting*) der IT-Plattform auszuführen.
Für die Integration von Diensten und Anwendungen, die außerhalb der IT-Plattform die Vertrauenskette erweitern, stellt das Attestierungs-Modul eine Schnittstelle bereit.

II.5.4 Attestierungs-Service

Der externe Attestierungs-Service bildet eine zentrale Schnittstelle zwischen dem Cloud-Anwender und einer Cloud-IT-Plattform. Seine Funktionsweise ist nicht allein auf eine IT-Plattform beschränkt, sondern kann eine größere Anzahl registrierter IT-Plattformen mit unterschiedlichen OS/VMM-Softwarekomponenten umfassen.
Seine Hauptaufgabe liegt in der Attestierung von gesicherten Systemzuständen der entsprechenden IT-Plattformen und der sicheren Weiterleitung an entsprechende Endgeräte der Cloud-Anwender.

Der Attestierungs-Service verwaltet eine zentrale Repository mit registrierten gültigen IT-Plattformeigenschaften (*Whitelists*), um die Konformität der IT-Plattform und der eingesetzten OS/VMM-Softwarekomponenten bewerten zu können. Weitere Funktionen sichern die Aufbereitung und den Vergleich (*Compliance Verification*) von IT-Plattformeigenschaften für den anfragenden Cloud-Anwender. Der Attestierungs-Service kann optional eine Zertifizierungskomponente (*Certificate Authority*) bereitstellen.

II.5.5 Hardwarebasierte Geo-Lokalisierung

Die Bestimmung und der Nachweis geografischer Koordinaten in Bezug auf den Standort einer installierten IT-Plattform beeinflussen den Ausführungsrahmen von Cloud-Services. Der Grad der Vertrauenswürdigkeit steht in enger Beziehung mit der Bestimmbarkeit geografischer Koordinaten.
Auf Grundlage TPM-gesicherter geografischer Identifikatoren (*Geo-Tagging*) kann über die Mechanismen der Attestierung Auskunft gegeben werden, in welchem geografischen Bereich Dienste einer IT-Plattform angeboten werden.

III | Übersicht der Anforderungen

Die Erhebung der nachfolgenden Anforderungen erfolgte in einer frühen Phase der Analyse und diente der ersten Identifizierung von möglichen Regulierungszielen. Die Vorgehensweise orientiert sich an den Architekturebenen eines Cloud-Systems. Die weitere Systematisierung und Beschreibung von Regulierungszielen ist Gegenstand von Abschnitt 3.1.

III.1 Anforderungen an die Cloud-Infrastruktur-Plattform-Ebene

Req001 Integrität der Cloud-Infrastruktur-Plattform überprüfen

Die Integrität der Hardware-Struktur, der Konfiguration und des Hypervisors muss gewährleistet sein.

Req002 Zugriff auf Infrastruktur-Plattformbereiche protokollieren

Zugriffe von der Cloud-Laufzeitumgebung auf Infrastruktur-Plattformbereiche müssen protokolliert werden. Es muss daraus ersichtlich werden, welcher Prozess oder Akteur welchen Zugriff ausgeführt hat.

Req003 Cloud-Laufzeitumgebung geregelt starten

Der Start von mehreren virtualisierten Cloud-Laufzeitumgebungen, inklusive der darin ausgeführten Services, muss bezüglich der bereitgestellten Ressourcen geregelt werden.

Req004 Separierung ausführen

Für unterschiedliche Benutzergruppen muss eine sichere Separierung der Cloud-Laufzeitumgebung gewährleistet werden. Die Separierung kann sowohl auf der Cloud-Laufzeitumgebungs- als auch auf der Netzwerkebene erfolgen.

Req005 Geo-Koordinaten ermitteln

Für den Start einer Cloud-Laufzeitumgebung müssen die geografischen Koordinaten bestimmbar sein.

Req006 Verteilung autorisiert durchführen

Die Verteilung von virtualisierten Cloud-Laufzeitumgebungen darf nur für autorisierte Akteure möglich sein.

Req007 Kommunikation absichern

Die Kommunikation zwischen einer Cloud-Infrastruktur-Plattform und virtualisierten Cloud-Laufzeitumgebungen muss über einen abgesicherten Kanal erfolgen.

Req008 Vertrauenswürdige Cloud-Laufzeitumgebungen zusichern

Für den Einsatz in einem vertrauenswürdigen Cloud-System dürfen nur authentische und evaluierte Cloud-Service-Laufzeitumgebungen zum Einsatz kommen.

Req009 Verfügbarkeit gewährleisten

Es müssen Verfahren zum Einsatz kommen, um eine hohe Verfügbarkeit von Cloud-Plattform-Ressourcen zu gewährleisten.

Req010 Dynamische Ressourcenverwaltung ausführen

Es sind Maßnahmen vorzusehen, die Cloud-Laufzeitumgebungen in Abhängigkeit vom Bedarf der Zugriffe mit ausreichenden Ressourcen bereitstellen.

Req011 Sicherheitsniveau (Attestierung) nachweisen

Einem Cloud-Anwender müssen die Eigenschaften einer Cloud-Plattform in einer vertrauenswürdigen Form übermittelt werden.

Req012 Netzwerk-Kommunikation sichern

Auf der Netzwerkebene ist eine sichere Kommunikation zwischen unterschiedlichen Cloud-Service-Laufzeitumgebungen sicherzustellen (Firewalls, Netzwerkverschlüsselung).

Req013 Schlüssel- und Zertifikatsmanagement integrieren

Über ein integriertes Schlüssel- und Zertifikatsmanagement muss die Identität einer Cloud-Infrastruktur-Plattform sicher festgestellt werden können.

III.2 Anforderungen an die Cloud-Laufzeitebene

Req100 Services sicher starten

Der Start von Services in verschiedenen Cloud-Laufzeitumgebungen darf sich nicht auf das Ressourcenangebot anderer Services auswirken.

Req101 Speicherung vertraulich ausführen

Benutzerdaten, Metadaten und Konfigurationen müssen vertraulich übertragen und gespeichert werden. Der lesende Zugriff ist nur autorisierten Cloud-Anwendern erlaubt.

Req102 Daten klassifizieren

Der Cloud-Anwender muss die Möglichkeit besitzen, Festlegungen zur Datenklassifizierung bestimmen zu können.

Req103 Datenposition bestimmen

Ein Cloud-System muss Funktionen zur sicheren Bestimmung seiner geografischen Position erhalten, um einen Nachweis erbringen zu können, wo Daten eines Cloud-Anwenders gespeichert werden.

Req104 Cloud-Service-Konfigurationen schützen

Es müssen Maßnahmen vorgesehen werden, die eine Modifikation von Konfigurationsdaten feststellen können.

Req105 Services und Daten trennen

Es sind Maßnahmen zu treffen, um die Bereiche zur Ausführung von Services von den Bereichen zur Speicherung der Daten zu trennen.

Req106 Services separieren

Es sind Maßnahmen vorzusehen, die eine gezielte Separierung der Services innerhalb einer Cloud-Laufzeitumgebung ermöglichen. Die Separierungskonzepte beschreiben auch notwendige Kommunikationen zwischen den Services.

Req107 Vertrauenswürdige Services zusichern (Evaluierung)

Für den Einsatz in einem vertrauenswürdigen Cloud-System dürfen nur authentische und evaluierte Services zum Einsatz kommen. Diese Services besitzen eine verifizierbare Zulassungsbestätigung.

Req108 Services dynamisch bereitstellen

Es sind Maßnahmen vorzusehen, die Services in Abhängigkeit vom Bedarf der Zugriffe bereitstellen, so dass immer ausreichende Service-Ressourcen zur Verfügung stehen.

Req109 Administration autorisieren

Es muss sichergestellt werden, welche Akteure eine Berechtigung besitzen, um eine Cloud-Laufzeitumgebung zu verwalten. Die Verwaltung umfasst die Prozesse zum Starten, Beenden und für die Verschiebung einer Cloud-Laufzeitumgebung.

Req110 Services zuverlässig neu starten

Es muss sichergestellt werden, dass Services in Ausnahmebedingungen die Lauffähigkeit anderer Services nicht beeinträchtigen. Ein sicherer Wiederanlauf nach einer Störung ist zu gewährleisten.

III.3 Anforderungen an die Cloud-Service-Ebene

Req200 Datenverteilung nachweisen

Es muss eine Protokollierung über die Datenverteilung und Datenauslagerung (Archivierung, Datensicherung) sichergestellt werden.

Req201 Datenverteilung autorisieren (Regulierung)

Der Prozess für das Verteilen, Löschen und Auslagern von Daten muss festgelegt und nachvollziehbar sein.

Req202 Prinzip *Kenntnis nur, wenn nötig* anwenden (Regulierung)

Die Autorisierung eines Cloud-Anwenders für den Datenzugriff muss nach dem Prinzip *Kenntnis, nur wenn nötig* erfolgen.

Req203 Daten vollständig löschen

Es muss sichergestellt werden, dass bei einer Datenlöschung auch alle Kopien und Ausfertigungen bestehender Daten gelöscht werden.

Req204 Verschlüsselungsverfahren auswählen

Es muss sichergestellt werden, dass für Daten mit einer Klassifikation angemessene Verschlüsselungsverfahren zum Einsatz kommen.

Req205 Daten vertraulich austauschen

Das System muss sicherstellen, dass die Kommunikation zwischen einem Cloud-Anwender und dem Cloud-System-Service vertraulich erfolgt.

Req206 Daten authentisch austauschen

Es sind Maßnahmen für eine sichere Cloud-Anwender-Identifizierung vorzusehen. Auf Grundlage der Identitätsbestimmung kann festgestellt werden, welche Cloud-Anwender für welche Service-Ressourcen autorisiert sind.

Req207 Daten integritätsgeschützt austauschen

Es muss sichergestellt werden, dass die Datenübertragung zwischen einem Cloud-Anwender und einem Cloud-System unverfälscht erfolgt.

Req208 Datenprozesse nachweisen

Der Import und Export von Daten als Austauschkonzept einer Trusted-Domain ist nachweispflichtig auf Grundlage einer Verteilungsregulierung zu gewährleisten (Regulierung).

Req209 Daten sicher rückführen

Es sind Funktionen für die sichere und vollständige Rückübertragung von Daten von einem Cloud-System auf das System eines Cloud-Anwenders sicherzustellen.

Req210 Aktivitäten sicher nachweisen

Es sind Maßnahmen vorzusehen, die die Kommunikationen zwischen einem Cloud-Anwender und dem Cloud-System im Sinne der Zurechenbarkeit protokollieren und für die Nachweisführung bereitstellen. Welche Aktivitäten nachgewiesen werden müssen, wird im Rahmen einer Regulierung festgelegt.

Req211 Kooperative Aktivitäten nachweisen

Für die Ausführung von Geschäftsgängen müssen Methoden bereitstehen, die eine regulierte und nachweisgeführte Abarbeitung von Aktivitäten zur Erfüllung von Geschäftszielen gewährleisten.

Req212 Daten sicher vernichten

Daten müssen so vernichtet werden, dass diese danach technisch nicht reproduziert und wiederhergestellt werden können.

III.4 Anforderungen an operatives Management

Req300 Daten wiederherstellen

Die Verfügbarkeit von Daten muss nach einem Systemausfall durch die Anwendung von Recovery-Funktionen gewährleistet werden.

Req301 Datenerhalt kryptographisch sichern

Verschlüsselte Daten müssen nach dem Verlust des Schlüssels durch Anwendung einer kryptographischen Key-Recovery wieder verfügbar gemacht werden können.

Req302 Management autorisiert durchführen (Regulierung)

Die Ausführung von Einstellungen für die Netzwerkebene von Cloud-Laufzeitumgebungen darf nur für autorisierte Akteure möglich sein.

Req303 Sicherheitsstatus übergreifend ermitteln

Es sind Maßnahmen zu treffen, um die Sicherheit der Cloud-Infrastruktur-Plattform, der Cloud-Laufzeitumgebung und der Cloud-Service-Ebene zyklisch zu überprüfen.

Req304 Berechtigungsdaten schützen

Es sind Maßnahmen zur Registrierung, Verwaltung und für das Berechtigungsmanagement von Cloud-Anwender-Identitäten vorzusehen. Die sichere Verwaltung umfasst ebenfalls die vertrauliche Speicherung von elektronischen Cloud-Identitätsdaten.

Req305 Datenschutz regulieren

Es sind Funktionen und Technologien anzubieten, die dem Cloud-Anwender eine Umsetzung seiner eigenen Sicherheitsstrategien und Datenschutzkonzepte erlaubt.

Req306 Serviceeigenschaften attestieren

Es muss für einen Cloud-Anwender überprüfbar sein, welches Niveau an Sicherheitskonzepten durch einen Cloud-Anbieter bereitgestellt wird. Auf dieser Grundlage ist ein Cloud-Anwender in der Lage, eine vertrauenswürdige Basis zu finden, um seine Serviceleistungen gestalten zu können.

Req307 Sicherheitseigenschaften nachweisen

Ein Cloud-Anwender muss in der Lage sein, formale Nachweise in Bezug auf bereitgestellte Sicherheitskonzepte überprüfen zu können.

Req308 Nutzungsbedingungen aushandeln

Für einen Cloud-Anwender müssen Prozesse bereitgestellt werden, so dass er gemeinsam mit dem Cloud-Anbieter die Nutzungsbedingungen (SLAs) aushandeln und technisch manifestieren kann.

In die Aushandlung gehen ein:

- rechtliche Aspekte
- geografische Aspekte
- Ausnahmen, die nicht erfüllt werden können
- Vertragsstrafen bei Abweichungen
- Informationen über den Cloud-Anbieter (Reputation)

Req309 Prozesse für Nutzungsvorbereitung anbieten

Einem Cloud-Anwender müssen Prozesse angeboten werden, so dass er die Cloud-Nutzungsphase gemäß der vereinbarten Zielstellungen vorbereiten und ausführen kann.

Req310 Prozesse für Überwachung anbieten

Einem Cloud-Anwender müssen Prozesse angeboten werden, so dass er während der Cloud-Nutzungsphase gemäß den Ablaufvorgaben entsprechende Ergebnisse überprüfen kann. Es stehen folgende Funktionen bereit:

- Bewertung der Sicherheitsqualität der Plattform und Cloud-Laufzeitumgebung
- Bewertung der Cloud-Servicequalität
- Funktionen zur Darstellung von Abweichungen

Req311 Prozesse für Abrechnung anbieten

Einem Cloud-Anwender müssen Prozesse angeboten werden, worüber eine Abrechnung seiner in Anspruch genommenen Dienstleistungen erfolgen kann.

Req312 Prozesse für Nutzungsabschluss anbieten

Einem Cloud-Anwender müssen Prozesse angeboten werden, die es ihm erlauben, die Ergebnisse zu überprüfen, die Abrechnung vorzunehmen und vereinbarte Nachbedingungen zu schaffen (z. B. Rückübertragung von Daten, Löschung von Zwischendaten).

Req313 Identitäten kontrolliert austauschen (Regulierung)

Der Austausch von Benutzer-Identitäten muss Anforderungen an die Privatsphäre sicherstellen. Der Austausch von Identitätsinformationen zwischen Organisationseinheiten kann nur auf Grundlage von Regelungen der betroffenen Cloud-Anwender erfolgen.

III.5 Anforderungen an Cloud-Anwender-Nutzungsebene

Req400 Schnittstellen für Systemauswahl anbieten

Einem Cloud-Anwender müssen Schnittstellen angeboten werden, um unterschiedliche Kriterien der Cloud-Nutzung bewerten zu können.

Dazu gehören:

- Vertrauensniveau der Cloud-Plattform
- Konfiguration der Cloud-Service-Laufzeitumgebung und Services
- Übersicht zu Service-Ausführungen und Aktivitäten
- Abrechnungen
- Cloud-Nutzungs-Nachbedingungen

Req401 Schnittstellen für Policy-Management anbieten

Einem Cloud-Anwender müssen Schnittstellen angeboten werden, um Cloud-Aufträge vorbereiten zu können und seine Regulierungsziele zu formulieren.

Dazu gehören:

- Definition von Cloud-Service-Verträgen
- Bereitstellung von Nutzungsbedingungen (Policies)
- Steuerung der sicheren Bereitstellung von Cloud-Services

- Steuerung der Datenübertragung zwischen Cloud-Anwender und Cloud-Anbieter
- Cloud-Nutzungs Nachbdedingungen

Req402 Schnittstellen für Nutzungs-Steuerung anbieten

Einem Cloud-Anwender müssen Schnittstellen angeboten werden, um Cloud-Services nutzen zu können und um auf die Service-Steuerung Einfluss zu nehmen.

Req403 Instrumente zum Sicherheitsmanagement bereitstellen

Einem Cloud-Anwender müssen Instrumente bereitgestellt werden, um Aufgaben für das Sicherheitsmanagement vorzubereiten.

Dazu gehören:

- Management von Cloud-Anwender-Accounts
- Management von Cloud-Anwender-Berechtigungen
- Management von Verteilungskonzepten für Identitätsdaten
- Management von Verteilungskonzepten für Anwendungsdaten
- Management von Verteilungskonzepten für virtuelle Service-Laufzeitumgebungen

Req404 Zertifikatsmanagement integrieren

Einem Cloud-Anwender und Cloud-Anbieter müssen Funktionen für die Erstellung und Verwaltung von kryptographischen Schlüsseln und Zertifikaten bereitgestellt werden.

Req405 Kryptographische Schlüssel sicher verteilen

Cloud-Anwender und Cloud-Anbieter müssen Funktionen für die sichere Speicherung und Verteilung von kryptographischen Schlüsseln anwenden.

IV | Spezifikation Ontologie

Die Ontologie *Trusted-Regulation* besteht aus den Dateien:

- Domain.obl
- Regulation.obl
- Security.obl

Dropbox Freigabe : https://www.dropbox.com/sh/7tp9ba1cgu9n5jc/AAC9cdYCFBWQ1S6SnkVYbHd3a?dl=0

Kontakt bei Fragen:

Joerg.Kebbedies@mailbox.tu-dresden.de
alternativ
info@kebbedies.eu