



Abschlussprüfung Sommer 2020

Fachinformatiker für Systemintegration  
Dokumentation zur betrieblichen Projektarbeit

# Evaluierung von „Icinga“

## Open Source Monitoring

Abgabetermin: Augsburg, den 21.05.2020

**Prüfungsbewerber:**

Andreas Germer  
Hagelbach 9  
86316 Bachern



**Ausbildungsbetrieb:**

KUKA AG  
Zugspitzstraße 144  
86163 Augsburg

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

## Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>Listings</b>	<b>V</b>
<b>Abkürzungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Projektumfeld . . . . .	1
1.2 Projektziel . . . . .	1
1.3 Projektbegründung . . . . .	1
1.4 Projektabgrenzung . . . . .	1
<b>2 Projektplanung</b>	<b>2</b>
2.1 Projektphasen . . . . .	2
2.2 Abweichungen vom Projektantrag . . . . .	2
2.3 Ressourcenplanung . . . . .	2
<b>3 Planungsphase</b>	<b>3</b>
3.1 Ist-Analyse . . . . .	3
3.2 Soll-Analyse . . . . .	3
3.2.1 Befragung der Fachabteilungen . . . . .	3
3.2.2 Kriterienkatalog . . . . .	4
3.3 Wirtschaftlichkeitsanalyse . . . . .	4
3.3.1 Projektkosten . . . . .	4
3.3.2 Betriebskosten des Monitorings . . . . .	5
3.3.3 Amortisationsdauer . . . . .	5
<b>4 Durchführungsphase</b>	<b>6</b>
4.1 Vorbereitung der Tests . . . . .	6
4.1.1 Vorbereiten der Hardware . . . . .	6
4.1.2 Installation des Hypervisors . . . . .	6
4.1.3 Installation der virtuellen Maschinen . . . . .	7
4.1.4 Installation von „Icinga“ . . . . .	7
4.2 Durchführung der Tests . . . . .	9
4.2.1 Betriebssystemkompatibilität . . . . .	9
4.2.2 Überwachung von Leistungsparametern . . . . .	9
4.2.3 Webserverüberwachung . . . . .	9
4.2.4 Grafische Aufbereitung . . . . .	10

<b>5</b>	<b>Abnahmephase</b>	<b>10</b>
<b>6</b>	<b>Einführungsphase</b>	<b>10</b>
<b>7</b>	<b>Dokumentation</b>	<b>10</b>
<b>8</b>	<b>Fazit</b>	<b>11</b>
8.1	Soll-/Ist-Vergleich . . . . .	11
8.2	Lessons Learned . . . . .	11
8.3	Ausblick . . . . .	11
	<b>Quellenverzeichnis</b>	<b>12</b>
	<b>Eidesstattliche Erklärung</b>	<b>13</b>
<b>A</b>	<b>Anhang</b>	<b>i</b>
A.1	Berechnung des Stundensatzes von Mitarbeitenden . . . . .	i
A.2	Installationsskript für Ubuntu 18.04 . . . . .	ii
A.3	Screenshots . . . . .	iii
A.4	Detaillierte Zeitplanung . . . . .	vi
A.5	Use Case-Diagramm . . . . .	ix
A.6	Datenbankmodell . . . . .	ix
A.7	Oberflächenentwürfe . . . . .	xi
A.8	Entwicklerdokumentation . . . . .	xiii
A.9	Testfall und sein Aufruf auf der Konsole . . . . .	xv
A.10	Klasse: ComparedNaturalModuleInformation . . . . .	xvi
A.11	Klassendiagramm . . . . .	xix
A.12	Benutzerdokumentation . . . . .	xx

## Abbildungsverzeichnis

1	Netzwerkconfiguration in VMware ESXi. Rechts der physische Netzwerkport, links die virtuellen Ports der virtuellen Maschinen. Beides verbunden durch einen virtuellen Switch (mitte). . . . .	iii
2	Anpassung der Parameter bei Erstellung einer virtuellen Maschine in VMware ESXi .	iii
3	Ausführen des Skripts „mysql_secure_installation“ zur Absicherung eines MySQL-Systems . . . . .	iv
4	Angezeigter PHP-Fehler nach Installation des Icinga-Webfrontends . . . . .	v
5	Willkommens-Bildschirm des Konfigurationsassistenten . . . . .	v
6	Einrichtung der Datenbank für Webfrontend-Benutzer . . . . .	v
7	Startseite von „Icinga 2“ nach der Erstkonfiguration. Der Server, auf dem die Instanz von „Icinga 2“ läuft, ist automatisch als erster Server hinzugefügt . . . . .	vi
8	Zwei neue Server werden durch Anhängen dieser Zeilen an <code>/etc/icinga2/conf.d/hosts.conf</code> dem Monitoring hinzugefügt . . . . .	vi
9	Beispiel für ein definiertes „CheckCommand“-Objekt . . . . .	vi
10	Beispiel für einen definierten Dienst . . . . .	vii
11	Detailansicht eines überprüften Dienstes; hier die Webserverüberwachung . . . . .	vii
12	Detailansicht eines überprüften Dienstes; hier die Webserverüberwachung . . . . .	viii
13	Use Case-Diagramm . . . . .	ix
14	Datenbankmodell . . . . .	x
15	Liste der Module mit Filtermöglichkeiten . . . . .	xi
16	Anzeige der Übersichtsseite einzelner Module . . . . .	xii
17	Anzeige und Filterung der Module nach Tags . . . . .	xii
18	Aufruf des Testfalls auf der Konsole . . . . .	xvi
19	Klassendiagramm . . . . .	xix

## **Tabellenverzeichnis**

1	Zeitplanung . . . . .	2
2	Kostenaufstellung Projekt . . . . .	4
3	Kostenaufstellung Monitoring . . . . .	5
4	Soll-/Ist-Vergleich . . . . .	11

Listings

1	Testfall in PHP . . . . .	xv
2	Klasse: ComparedNaturalModuleInformation . . . . .	xvi

## **Abkürzungsverzeichnis**

<b>API</b>	Application Programming Interface
<b>UML</b>	Unified Modeling Language

## 1 Einleitung

### 1.1 Projektumfeld

Die KUKA AG ist ein international tätiges Unternehmen in der Maschinenbau- und Automatisierungsbranche mit rund 14.200 Mitarbeitern. Zum Produktportfolio zählen neben Industrierobotern auch die Planung und der Bau von Produktionsstraßen.

Die IT-Infrastruktur am Standort Augsburg besteht aus ca. 1.200 größtenteils virtualisierten Servern; an anderen Standorten befindet sich vereinzelt eine kleine Anzahl an Servern. Es kommen alle gängigen Betriebssysteme zum Einsatz. Auftraggeber dieses Projekts ist die Abteilung „Datacenter & Network“, die den Betrieb der Rechenzentren sowie des internen IT-Netzwerks der KUKA AG überwacht und koordiniert.

### 1.2 Projektziel

Momentan wird kein einheitliches Monitoring der IT-Systeme durchgeführt. Die verschiedenen Standorte und Abteilungen setzen auf unterschiedliche und teils veraltete Lösungen; zur besseren Administration sollen diese durch ein einheitliches und zentrales System ersetzt werden.

Es soll evaluiert werden, ob die freie Monitoringsoftware „Icinga“ für den firmeninternen Einsatz geeignet ist. Dazu werden in einem ersten Schritt Anforderungen der Process Owner und Systemadministratoren gesammelt. Anschließend wird eine Testumgebung eingerichtet, um zu prüfen, inwieweit die Ansprüche des Unternehmens durch „Icinga“ erfüllt werden können. Abschließend werden die Ergebnisse analysiert und eine Empfehlung ausgesprochen.

### 1.3 Projektbegründung

Durch die zunehmende Digitalisierung aller Geschäftsprozesse sind Unternehmen äußerst abhängig von Computersystemen. Wird das Schutzziel der Verfügbarkeit nicht ausreichend gut verfolgt, kommt es zu Systemausfällen und der Geschäftsbetrieb ist nicht mehr möglich - mit unabsehbaren wirtschaftlichen Folgen. Um Ausfallsicherheit zu gewährleisten, ist ein umfangreiches und zuverlässiges Monitoring aller Ressourcen unerlässlich.

### 1.4 Projektabgrenzung

Als Monitoringsystem interagiert „Icinga“ potenziell mit allen Geräten und Diensten im Netzwerk. Für dieses Projekt wird das Zusammenspiel auf virtuelle Maschinen mit ausgewählten Betriebssystemen und Diensten eingegrenzt.



## 2 Projektplanung

### 2.1 Projektphasen

Das Projekt wird innerhalb einer 35-Stunden-Arbeitswoche durchgeführt. Die einzelnen Phasen werden sich teils überschneiden; beispielsweise können virtuelle Maschinen bereits eingerichtet werden, während noch auf Rückmeldungen bezüglich der Anforderungen an das System gewartet wird.

Planungsphase		Durchführungsphase		Projektabschluss	
Ist-Analyse	2h	Einrichtung Wirt und VMs	6h	Ergebnisbewertung	1h
Soll-Analyse	2h	Installation Icinga	2h	Soll-Ist-Vergleich	1h
Definierung Kriterienkatalog	3h	Installation Testsysteme	3h	Empfehlungsformulierung	1h
		Testdurchführung	7h	Dokumentation	7h

Tabelle 1: Zeitplanung

### 2.2 Abweichungen vom Projektantrag

Es gab keine Abweichungen vom Projektantrag.

### 2.3 Ressourcenplanung

Folgende Hardware wird zur Durchführung des Projekts benötigt:

- Notebook mit RJ-45 Port und USB Port
- Workstation mit mindestens: 2x RJ-45 Port, 2x USB Port, 16 GB Arbeitsspeicher, 500 GB HDD
- Software: Windows 10 (min. Professional), Windows Server (min. 2012), Debian (aktuelle Version), Ubuntu (aktuelle Version), ESXi (aktuelle Version), aktueller Webbrowser, SSH-Client
- 2x Netzkabel, min. 1,5m
- USB-Datenträger, 8GB

## 3 Planungsphase

### 3.1 Ist-Analyse

Zu Projektbeginn existiert im Unternehmen keine einheitliche Lösung zur Serverüberwachung. Für den Großteil der Server am Standort Augsburg wird der „Advanced Host Monitor“ eingesetzt. Diese Lösung erhält trotz ihres Alters noch regelmäßige Updates, aufgrund des veralteten User Interfaces und einem beschränkten Funktionsumfang, sowie der nicht ausreichenden Leistungsfähigkeit, wird seit mehreren Jahren der Wunsch nach einem neuen Monitoring-System geäußert.

Die Entwicklungsabteilungen überwachen die von ihnen betreuten Server mit einer auf „Elasticsearch“ und „Kibana“ basierenden Lösung. Um Know-how zu bündeln und Personalkosten zu sparen, signalisierten die Verantwortlichen eine Bereitschaft zur Zusammenlegung des Servermonitoring.

Am Standort Bremen wird seit mehreren Jahren auf die Open-Source-Anwendung „Icinga“ gesetzt. Die Systembetreuer haben mit dieser Lösung positive Erfahrungen gemacht, und loben insbesondere die einfache Erweiterbarkeit durch Plugins, sowie die Möglichkeit ohne Aufwand optisch ansprechende Dashboards zu erstellen. Aufgrund der Komplexität dieser Software und einem anstehenden Update auf „Icinga 2“ wurde auch von dieser Seite der Wunsch nach einem einheitlichen und zentral verwaltetem Servermonitoring geäußert.

### 3.2 Soll-Analyse

#### 3.2.1 Befragung der Fachabteilungen

Es wurden die entsprechenden Verantwortlichen via E-Mail, Telefonaten und Meetings zu den betreuten Servern und deren Monitoring befragt. Hierbei zeigten sich große Überschneidungen bei den Serverumgebungen sowie den Anforderungen an deren Überwachung, was eine Zusammenlegung logisch erscheinen lässt.

Über alle Abteilungen hinweg werden verschiedenste Windows-Versionen sowie Linux-Distributionen eingesetzt, weswegen ausschließlich ein plattformunabhängiges Monitoring-System wie „Icinga“ eingesetzt werden kann. Zwar sind die meisten Server virtualisiert, allerdings werden gerade bei Datenbanken auch dedizierte Rechner eingesetzt, welche selbstverständlich ebenfalls überwacht werden müssen.

Die zu überwachenden Parameter beziehungsweise Dienste ähneln sich zwischen den Abteilungen ebenfalls sehr stark. Neben Leistungsmetriken wie Prozessorauslastung, Arbeitsspeicherbelegung oder freiem Massenspeicher wird viel Wert auf die Überprüfung der Erreichbarkeit im Netzwerk gelegt. Aufgrund der zunehmenden Verbreitung von webbasierten Anwendungen soll auch der Betrieb von Webservern überwacht werden. Weiterhin wurde der Wunsch nach einer grafischen Aufbereitung geäußert.

### 3.2.2 Kriterienkatalog

Auf Basis der Befragungen wurde folgender Kriterienkatalog aufgestellt:

- Überwachbare Betriebssysteme: Alle gängigen Windows Server Versionen und Linux-Distributionen
- Überwachbare Systemparameter: CPU-Auslastung, RAM-Belegung, Massenspeicherbelegung
- Überwachung von Webservern
- Ansprechende und anpassbare grafische Aufbereitung

## 3.3 Wirtschaftlichkeitsanalyse

### 3.3.1 Projektkosten

Die Kosten, die durch das Projekt verursacht werden, setzen sich sowohl aus Personal- als auch aus Ressourcenkosten zusammen. Die Berechnung der Stundensätze findet sich im Anhang A.1: Berechnung des Stundensatzes von Mitarbeitenden auf Seite i.

Die Kosten für die 35-Stunden-Woche des Auszubildenden belaufen sich auf 350 €. Die Arbeitszeit, die bei mitarbeitenden Personen zur Durchführung des Projekts angefallen ist (z.B. für Befragungen, Konfiguration der Netzwerkkomponenten, Anpassungen der Firewall für Updates, Abnahme) wurde auf vier Stunden geschätzt. Dafür fielen Personalkosten in Höhe von 160 € an. Die für das Projekt eingesetzte Hardware ist bereits abgeschrieben, und wird mit einer Pauschale von 360 € pro Jahr verrechnet. Auf die Projektdauer von fünf Tagen ergibt das Betriebskosten von 4,93 €.

Eine Aufstellung der Kosten befindet sich in Tabelle 2. Sie betragen insgesamt 514,93 €.

Vorgang	Zeit	Kosten	Gesamtkosten
Projektdurchführung	35 h	10 € / h	350,00 €
Kollegiale Unterstützung	4 h	40 € / h	160,00 €
Betriebskosten Server	5 d	360 € / a	4,93 €
			<b>514,93 €</b>

Tabelle 2: Kostenaufstellung Projekt

### 3.3.2 Betriebskosten des Monitorings

Die Betriebskosten für ein Monitoringsystem umfassen Personalkosten für Wartungsarbeiten und Anpassungen, sowie die Kosten die für zwei redundant ausgelegte Hardware-Server anfallen. Diese werden benötigt, da das Monitoring nicht innerhalb der Virtualisierungsumgebung betrieben werden sollte, um eine funktionierende Serverüberwachung auch bei Ausfall der VM-Infrastruktur zu gewährleisten.

Für das Einpflegen neuer Server oder Funktionen wird eine Arbeitsstunde pro Woche veranschlagt. Weiterhin wird der Arbeitsaufwand für Updates und Fehlerbehebungen nach Erfahrungsberichten auf acht Stunden pro Quartal geschätzt. Auf ein Jahr summiert ergibt beides einen Arbeitsaufwand von insgesamt 84 Stunden.

Die geschätzten Kosten für einen Hardware-Server belaufen sich auf 4.000 € jährlich. Die Gesamtkosten für den Betrieb einer Monitoring-Instanz belaufen sich, wie in Tabelle 3 dargelegt, auf 11.360,00 € pro Jahr.

Beschreibung	Einzelkosten	Einheiten	Jahreskosten
Wartungs- und Pflegearbeiten	40,00 € / h	84 h / a	3.360,00 €
Hardwarekosten	4.000,00 € / a	2	8.000,00 €
			<b>11.360,00 €</b>

Tabelle 3: Kostenaufstellung Monitoring

### 3.3.3 Amortisationsdauer

Sollte die Evaluation durch diese Projektarbeit ergeben, dass „Icinga“ für den firmenweiten Einsatz geeignet ist, werden drei momentan im Betrieb befindliche Monitoringinstanzen (siehe 3.1 Ist-Analyse) durch ein zentrales „Icinga“-System ersetzt (siehe 1.2 Projektziel). Die Betriebskosten für die derzeitigen Überwachungssysteme entsprechen etwa den in 3.3.2 Betriebskosten des Monitorings berechneten 11.360,00 € pro Jahr; die jährliche Einsparung, wenn statt drei Systemen nur noch eines betrieben wird, beläuft sich also auf:

$$11.360,00 \text{ €} \cdot 2 \text{ Tage/Jahr} = 22.720,00 \text{ €} \quad (1)$$

Für die Ersteinrichtung des neuen Systems werden etwa zwei Wochen benötigt, also insgesamt 70 Arbeitsstunden. Das verursacht einmalige Kosten in Höhe von:

$$70 \text{ h} \cdot 40,00 \text{ €/h} = 2.800 \text{ €} \quad (2)$$

Die Amortisationszeit beträgt also  $\frac{2.800 \text{ €}}{22.720,00 \text{ €/a}} \approx 0,123 \text{ Jahre} \approx 45 \text{ Tage}$ .

## 4 Durchführungsphase

### 4.1 Vorbereitung der Tests

#### 4.1.1 Vorbereiten der Hardware

Eine ausrangierte DELL Precision Workstation dient als Hardwareplattform für das Projekt. Die Ausstattung von 32 Gigabyte Arbeitsspeicher und einem Intel Xeon Hochleistungsprozessor ist ausreichend für den Betrieb von mehreren virtuellen Maschinen.

Um den Bedingungen in einem „echtem“ Rechencenter möglichst nahe zu kommen, wird der VM-Hypervisor mit zwei Netzwerkschnittstellen ausgestattet. Ein Interface ist für die Kommunikation mit dem Hypervisor selbst vorgesehen (sogenanntes Management-LAN), und eine weitere Netzwerkschnittstelle wird von den virtuellen Maschinen benutzt, um im Netzwerk erreichbar zu sein.

Da die vorgesehene Workstation mit nur einem RJ45 Port ausgestattet war, wurde aus einer anderen die zusätzliche Netzwerkkarte entfernt und in diese eingebaut. Der Umbau gestaltete sich dank PCIe-Steckplatz als unkompliziert und war ohne Werkzeuggebrauch möglich.

#### 4.1.2 Installation des Hypervisors

Um die gegebenen Ressourcen optimal auszunutzen, wird ein sogenannter Typ 1-Hypervisor eingesetzt. Dieser kommuniziert direkt mit der Hardware, ohne dass ein anderes Betriebssystem zum Einsatz kommt. Der Markt an VM-Hypervisoren ist stark unkämpft; neben den bekannten Lösungen von den „Platzhirschen“ Microsoft, VMware und Citrix existiert eine Vielzahl an kleinen, teils auch quelloffenen, Virtualisierungsplattformen. Für dieses Projekt wird das System „ESXi“ von VMware in der Version 6.7 verwendet, das auch in den firmeneigenen Rechenzentren zum Einsatz kommt.

Die Installation gestaltet sich als simpel. Nachdem wichtige UEFI-Optionen angepasst wurden (ausschließliche Verwendung von UEFI statt legacy-BIOS; Aktivierung der hardwareseitigen Unterstützung für Virtualisierung „Intel VT-x“) wird das System von einem USB-Datenträger, der zuvor mit dem ESXi-Image geflasht wurde, gestartet. Nachdem die für die Installation zu benutzende Festplatte ausgewählt, und ein Root-Passwort vergeben wurde, startet der Installationsvorgang. Nach abgeschlossener Installation muss noch eine IP-Adresse für das Managementnetzwerk vergeben werden.

Die restliche Konfiguration geschieht bequem über eine Weboberfläche. Der Lizenzschlüssel wird hinterlegt, die zweite Festplatte als Datastore eingebunden, und das Netzwerk für die virtuellen Maschinen konfiguriert (siehe Abbildung 1). Hierzu wird ein neuer virtueller Switch für den zweiten Netzwerkport (der, der nicht mit dem Management-LAN belegt ist) erstellt und mit einer neuen Port Group versehen. Diese Port Groups werden in ESXi verwendet, um logische Netzwerkschnittstellen bereit zu stellen und zu verwalten.

### 4.1.3 Installation der virtuellen Maschinen

Für virtuelle Maschinen muss in der ESXi-Weboberfläche zunächst die Systemkonfiguration festgelegt werden. Hierfür werden die für die VM vorgesehene CPU-Kerne, Arbeits- und Massenspeicherspeicherkapazität (siehe Abbildung 2) sowie die zu benutzenden Port Groups eingestellt. Abschließend wird in ein virtuelles DVD-Laufwerk das ISO-Abbild für das entsprechende zu installierende Betriebssystem eingehängt, und die virtuelle Maschine gestartet. Die anschließende Betriebssysteminstallation kann über eine emulierte Browser-Konsole durchgeführt werden.

Um in diesem Testsystem eine möglichst nah an das Firmennetzwerk heranzukommen, wurden drei Betriebssysteme für die Server ausgewählt: Windows Server 2019, Ubuntu 18.04 LTS sowie Debian 10.3. Die Installation der Systeme verlief ohne Probleme.

### 4.1.4 Installation von „Icinga“

Die Installation von „Icinga“ unter Ubuntu beginnt mit dem Aufruf der Rootshell und einem anschließendem kompletten Systemupdate. Danach werden mittels Paketverwaltung die wichtigsten Ressourcen installiert; die Pakete `icinga2` und `icingacli` sind für die Monitoring-Engine an sich sowie die Verwaltung über Kommandozeile zuständig. `monitoring-plugins` installiert die grundsätzlichen Monitoringmethoden (z.B. Ping) für Nagios-kompatible Systeme wie „Icinga“.

Zusätzlich muss noch eine passende Datenbank aufgesetzt werden, hier fiel die Entscheidung aufgrund der firmeninternen Verbreitung sowie der In-Memory Unterstützung auf MySQL. Durch Installation der Pakete `mysql-server` und `mysql-client` werden ein MySQL-Server und ein MySQL-Client bereitgestellt, das Skript `mysql_secure_installation` (siehe Abbildung 3) verbessert die Sicherheit der Installation durch Maßnahmen wie das Entfernen von anonymen Accounts oder die Absicherung des root-Accounts.

Im nächsten Schritt muss die Schnittstelle zwischen „Icinga Data Output“ (Exportfunktion für Monitoringdaten) und MySQL geschaffen werden, damit die gesammelten Daten auch gespeichert werden können. Dies geschieht durch Installation des Pakets `icinga2-ido-mysql`. Anschließend kann das Webinterface `icingaweb2` installiert werden.

Beim Versuch, die über die Paketverwaltung verfügbare Version des Web-Frontends aufzurufen, wird lediglich ein PHP-Fehler (siehe Abbildung 4) angezeigt. Nach einer umfangreichen Recherche ergab sich, dass dieser Fehler durch einen Bug verursacht wird, wodurch „Icinga“ nicht mit bestimmten PHP 7.2 Plugins kompatibel ist. Dieser Fehler wurde bereits Mitte 2018 behoben, allerdings ist der Bugfix anscheinend nicht in das Paket eingespielt worden. Das Problem wurde gelöst, indem die aktuelle Version von „Icinga Web 2“ vom öffentlichen Github-Repository nach `/usr/share/icingaweb2` geklont wurde.

Ist „Icinga Web 2“ fertig installiert, muss noch ein Benutzer für die MySQL-Datenbank erstellt werden. Abschließend wird IDO mittels `icinga2 feature enable command ido-mysql` aktiviert, und

#### 4 Durchführungsphase

---

ein Setup-Token mit `icingacli setup token create` generiert werden. Nach einem Neustart des icinga2-Dienstes (`systemctl restart icinga2`) ist „Icinga 2“ bereit über die Webschnittstelle eingerichtet zu werden. Eine komplette (nachträglich optimierte) Liste an Befehlen, die für eine komplette Installation unter Ubuntu 18.04 ausgeführt werden müssen, findet sich im Anhang A.2: Installationskript für Ubuntu 18.04 auf Seite ii.

**Einrichtung über Webfrontend** Im Browser wird nun (IP-Adresse des Servers)/icingaweb2/setup aufgerufen, um den Konfigurationsassistenten (siehe Abbildung 5) zu starten. Dieser verlangt nach dem setup token, welcher im vorherigen Schritt generiert wurde. Nach Eingabe dieses tokens wird im ersten Schritt geprüft, ob alle notwendigen PHP-Plugins vorhanden sind. In der für diese Projekt durchgeführten Installation fehlten die Plugins „PDO-PostgreSQL“, welches aufgrund der Verwendung von MySQL nicht benötigt wird, sowie „cURL“, das mittels `apt install php7.2-curl` nachinstalliert wurde.

Im Abschnitt „Configuration“ werden noch kleinere, weitere Einstellungen getroffen, hierbei bietet sich aber meistens an, die Standardeinstellungen beizubehalten. Eine Ausnahme stellt die bevorzugte Authentifizierungsmethode für Benutzer des Frontends dar; hier kann zwischen LDAP und einer Datenbank mit Benutzern gewählt werden. Für dieses Projekt wurde aufgrund des geringeren Umfangs letztere Option gewählt, die Konfiguration hierfür findet sich in Abbildung 6.

**Einpflegen von Servern** Ist der Konfigurationsassistent abgeschlossen, müssen die zu überwachenden Server eingepflegt werden. Durch Installation des Plugins „Icinga Director“ kann dies auch über die Webschnittstelle erledigt werden, für dieses Projekt wird darauf verzichtet und die weitergehende Konfiguration geschieht über Anpassung der Konfigurationsdateien unter `/etc/icinga2/conf.d/`. Dort können in der Datei `hosts.conf` neue Server hinzugefügt werden (Syntax siehe Abbildung 8). Abschließend kann die Syntax der Konfigurationsdateien mittels `icinga2 daemon -validate` überprüft werden; nach einem Neustart des „Icinga 2“ Dienstes sind die neuen Hosts im Monitoring verfügbar.

**Installation des Agents** Mit der bisherigen Konfiguration können einfache Checks (z.B. Ping) auf die entsprechenden Server durchgeführt werden. Für aufwendigere Überprüfungen wie Speicherplatzkontrolle muss auf den Servern eine Software installiert werden. Hierfür muss zunächst der Monitoringserver als „Master“ deklariert werden, dies geschieht mittels des Befehls `icinga2 node wizard`. Anschließend wird mit `sudo icinga2 pki ticket -cn (FQDN)` noch ein „Ticket“ für den angegebenen FQDN erstellt, mithilfe dessen der Host die Verbindung zum Master herstellen kann.

Der Host selbst installiert die Software ebenfalls, auf Linux-basierten Systemen besteht dies einfach aus dem Paket `icinga2`, für Windows existieren herunterladbare Installationsdateien. Mit Durchführung des Befehls `icinga2 node wizard` wird die Konfiguration des Agents abgeschlossen, bei Windows geschieht dies über ein grafisches Userinterface. Nachdem die Adresse des Masters sowie die

eben erstellte Ticketnummer angegeben und eventuell Ports angepasst wurden, stellt der Agent eine Verbindung zum Monitoring her.

## 4.2 Durchführung der Tests

### 4.2.1 Betriebssystemkompatibilität

Es konnten keine Probleme bei der Integration von anderen Betriebssystemen festgestellt werden. Sowohl unter Ubuntu 18.04, Debian 10 und Windows Server 1809 läuft „Icinga 2“ ohne Probleme. Auch eine Online-Recherche förderte keine Probleme zu Tage, weswegen davon ausgegangen werden kann, dass eine umfassende Betriebssystemkompatibilität gewährleistet ist.

### 4.2.2 Überwachung von Leistungsparametern

Das installierte Pluginpaket `monitoring-plugins` bringt die benötigten Plugins zur Überwachung der Leistungsparametern (CPU-Auslastung, RAM-Belegung und belegter Massenspeicher) mit. Für diese Überprüfungen müssen in `/etc/icinga2/conf.d/commands.conf` Definitionen erstellt werden. Am Beispiel in Abbildung 9 ist erkenntlich, dass diese Definitionen neben dem auszuführendem Befehl (diese sind als Bash-Skript abgelegt; das Verzeichnis hierfür ist in der Konstante „PluginDir“ gespeichert) auch die zu übergebenden Argumente enthält. Im Beispiel der CPU-Auslastung wird mitgegeben, dass der Durchschnitt aller CPU-Kerne berechnet (-r) und bei bestimmten Schwellenwerten (-critical) ein Alarm gegeben werden soll-

Ist die Definition erfolgreich validiert, muss sie noch den entsprechenden Hosts zugewiesen. In der Datei `/etc/icinga2/conf.d/services.conf` wird hierzu ein Eintrag, wie in Abbildung 10 gezeigt, erstellt; dieser gibt den entsprechenden Befehl (`check_command`) und die zu überprüfende Instanz (`command_endpoint`) an, und weist das Überprüfungskommando zum Schluss allen Hosts mit einer Adresse zu (`assign where host.address`).

Die Überwachung der nach dem Kriterienkatalog benötigten Parameter funktioniert problemlos und die Werte decken sich mit den von den Systemen ermittelten Auslastungen. Somit ist „Icinga 2“ auch hierfür einsatzfähig.

### 4.2.3 Webserverüberwachung

Die Überwachung von Webservern muss analog zu den vorhergehenden Checks eingerichtet werden. Das Plugin „http“ übernimmt diese Aufgabe. In der Abbildung 12 ist gut erkenntlich, wie die Antwort der HTTP-Abfrage (Statuscode 200 OK) ausgewertet und dargestellt wird. Für die Antwortzeit lassen sich auch hier Schwellenwerte konfigurieren, sodass nicht nur Erreichbarkeit, sondern auch Performance überprüft werden kann. Die Überwachung von Webdiensten ist somit nach den in der Planungsphase aufgestellten Kriterien möglich.



#### 4.2.4 Grafische Aufbereitung

Auf der Startseite lassen sich mehrere Dashboards erstellen, die mithilfe sogenannter „Dashlets“ befüllt werden können. Alle über die Weboberfläche aufrufbaren Menüs haben eine eindeutige URL, mit der ein solches Dashlet erstellt werden kann. Somit ist es einfach, sehr angepasste und frei konfigurierbare Übersichten zu erstellen (siehe Abbildung ??).

Sollte dies nicht ausreichen, bietet die Entwicklercommunity viele weitere Plugins für diesen Zweck an. Als wichtigstes ist hier `dashin-dashboard` vertreten, das schon am Standort Bremen eingesetzt wird, und nach Auskunft der Mitarbeitenden durch die Unterstützung von Hintergrundbildern und Animationen in der Lage ist, noch ansprechendere Dashboards zu erstellen. Auch hier konnten also die vorher definierten Kriterien erfüllt werden.

## 5 Abnahmephase

- Welche Tests (z. B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z. B. Logs von Unit Tests, Testprotokolle der Anwender)?
- Wurde die Anwendung offiziell abgenommen?

**Beispiel** Ein Auszug eines Unit Tests befindet sich im Anhang A.9: Testfall und sein Aufruf auf der Konsole auf Seite xv. Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

## 6 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

## 7 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, API-Dokumentation)?
- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z. B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf jeden Fall in einer Dokumentation für den IT-Bereich).

**Beispiel** Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang A.12: Benutzerdokumentation auf Seite xx. Die Entwicklerdokumentation wurde mittels PHPDoc<sup>1</sup> automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang A.8: Entwicklerdokumentation auf Seite xiii.

## 8 Fazit

### 8.1 Soll-/Ist-Vergleich

- Wurde das Projektziel erreicht und wenn nein, warum nicht?
- Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?
- Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?
- Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet werden (z. B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

**Beispiel (verkürzt)** Wie in Tabelle 4 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

Phase	Geplant	Tatsächlich	Differenz
Entwurfsphase	19 h	19 h	
Analysephase	9 h	10 h	+1 h
Implementierungsphase	29 h	28 h	-1 h
Abnahmetest der Fachabteilung	1 h	1 h	
Einführungsphase	1 h	1 h	
Erstellen der Dokumentation	9 h	11 h	+2 h
Pufferzeit	2 h	0 h	-2 h
Gesamt	70 h	70 h	

Tabelle 4: Soll-/Ist-Vergleich

### 8.2 Lessons Learned

- Was hat der Prüfling bei der Durchführung des Projekts gelernt (z. B. Zeitplanung, Vorteile der eingesetzten Frameworks, Änderungen der Anforderungen)?

<sup>1</sup>Vgl. PHPDOC.ORG [2010]

### **8.3 Ausblick**

- Wie wird sich das Projekt in Zukunft weiterentwickeln (z. B. geplante Erweiterungen)?

## Quellenverzeichnis

### Dr. Peter Hoberg

DR. PETER HOBERG: *Vollständige Ermittlung von Personalkosten.* <https://www.controllingportal.de/Fachinfo/Kostenrechnung/Vollstaendige-Ermittlung-von-Personalkosten.html>, Abruf: 22.04.2020

### igmetall-bayern.de

IGMETALL-BAYERN.DE: *Tarifinfos Metall- und Elektroindustrie Bayern.* <https://www.igmetall-bayern.de/metall-elektro/>, Abruf: 22.04.2020

### igmetall.de

IGMETALL.DE: *Metall- und Elektroindustrie ERA – Ausbildungsvergütungen.* [https://www.igmetall.de/download/docs\\_MuE\\_ERA\\_Ausbildung\\_Juni2018\\_9bdc083c9c0ed885c63bd1b076830a817eaec814.pdf](https://www.igmetall.de/download/docs_MuE_ERA_Ausbildung_Juni2018_9bdc083c9c0ed885c63bd1b076830a817eaec814.pdf), Abruf: 22.04.2020

### phpdoc.org 2010

PHPDOC.ORG: *phpDocumentor-Website2.* Version: 2010. <http://www.phpdoc.org/>, Abruf: 20.04.2010

### Sensio Labs 2010

SENSIO LABS: *Symfony - Open-Source PHP Web Framework.* Version: 2010. <http://www.symfony-project.org/>, Abruf: 20.04.2010

## Eidesstattliche Erklärung

Ich, Andreas Germer, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

*Evaluierung von „Icinga“ – Open Source Monitoring*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Augsburg, den 21.05.2020

---

ANDREAS GERMER

## A Anhang

### A.1 Berechnung des Stundensatzes von Mitarbeitenden

Nach der Entgelttabelle der IG Metall Bayern für Metall und Elektro<sup>2</sup> erhalten Angestellte der Entgeltgruppe EG 08 ein Monatsbruttogehalt von 3.800,00 €. Zusätzlich dazu müssen ungefähr ein Fünftel für die Arbeitgeberanteile der Sozialversicherung und Mehrleistungen wie Urlaubsgeld hinzugerechnet werden.<sup>3</sup> Aufsummiert ergibt das jährliche Personalkosten von 54.720,00 €.

$$3.800,00 \text{ €} \cdot 1,2 \cdot 12 \text{ m} = 54.720,00 \text{ € /a} \quad (1)$$

Eine Arbeitszeit von 35 Stunden pro Woche bedeutet bei 52 Wochen eine Jahresarbeitszeit von 1820 Stunden. Abzüglich der Urlaubstage (30), Feiertage (ca. 11) und Fehltage durch Krankheit und Fortbildungen (ca. 15) ergibt das eine Jahresarbeitszeit von 1498 Stunden pro Jahr.

$$35 \text{ h/w} \cdot 52 \text{ m} = 1820 \text{ h/a} \quad (2)$$

$$1820 \text{ h/a} - [(30 + 11 + 15) \text{ d} \cdot 35 \text{ h/d}] = 1498 \text{ h/a} \quad (3)$$

Werden die Jahrespersonalkosten von 54.720,00 € durch die Jahresarbeitszeit von 1498 Stunden dividiert, ergibt dies Stundenkosten für den Arbeitgeber in Höhe von 36,53 €. Da es sich hierbei um eine Schätzung handelt, wird vereinfacht von 40,00 € pro Stunde ausgegangen.

$$54.720,00 \text{ € /a} \div 1498 \text{ h/a} = 36,53 \text{ € /h} \approx 40,00 \text{ € /h} \quad (4)$$

Für Auszubildende mit einer Ausbildungsvergütung in Höhe von 1.207,00 €<sup>4</sup> ergeben sich nach selber Kalkulation Personalkosten in Höhe von 10,57 €, also vereinfacht 10,00 € pro Stunde.

---

<sup>2</sup>Vgl. IGMETALL-BAYERN.DE

<sup>3</sup>Vgl. DR. PETER HOBERG

<sup>4</sup>Vgl. IGMETALL.DE

## A.2 Installationsskript für Ubuntu 18.04

```
# Systemupdate
apt update
apt dist-upgrade

# Grundinstallation icinga und MySQL
apt install icinga2 icingacli monitoring-plugins mysql-server mysql-client
mysql_secure_installation

# Installation Icinga IDO und Web-Interface
apt install icinga2-ido-mysql
apt install icingaweb2

# Klonen Github-Repository
cd /usr/share
mv icingaweb2 old.icingaweb2
git config --global http.proxy http://webproxy1.kuka.int.kuka.com:80
git clone https://github.com/Icinga/icingaweb2.git

# Installation fehlendes PHP-Plugin
apt install php7.2-curl
systemctl restart apache2

# Hinzufügen MySQL User für icingaweb2
mysql -e "CREATE USER 'icingaweb'@'localhost' IDENTIFIED BY '....';"
mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'icingaweb'@'localhost';"
mysql -e "FLUSH PRIVILEGES;"

# Einrichtungstoken erstellen
icinga2 feature enable command ido-mysql
icingacli setup token create
systemctl restart icinga2
```

### A.3 Screenshots

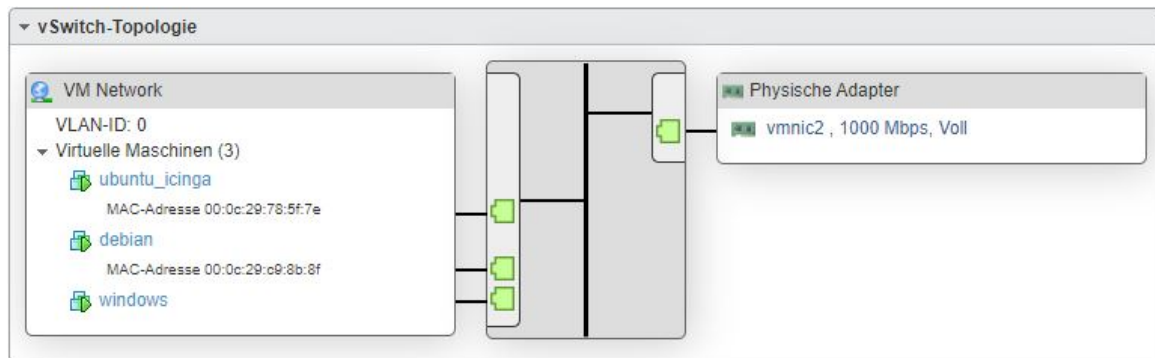


Abbildung 1: Netzwerkkonfiguration in VMware ESXi. Rechts der physische Netzwerkport, links die virtuellen Ports der virtuellen Maschinen. Beides verbunden durch einen virtuellen Switch (mitte).

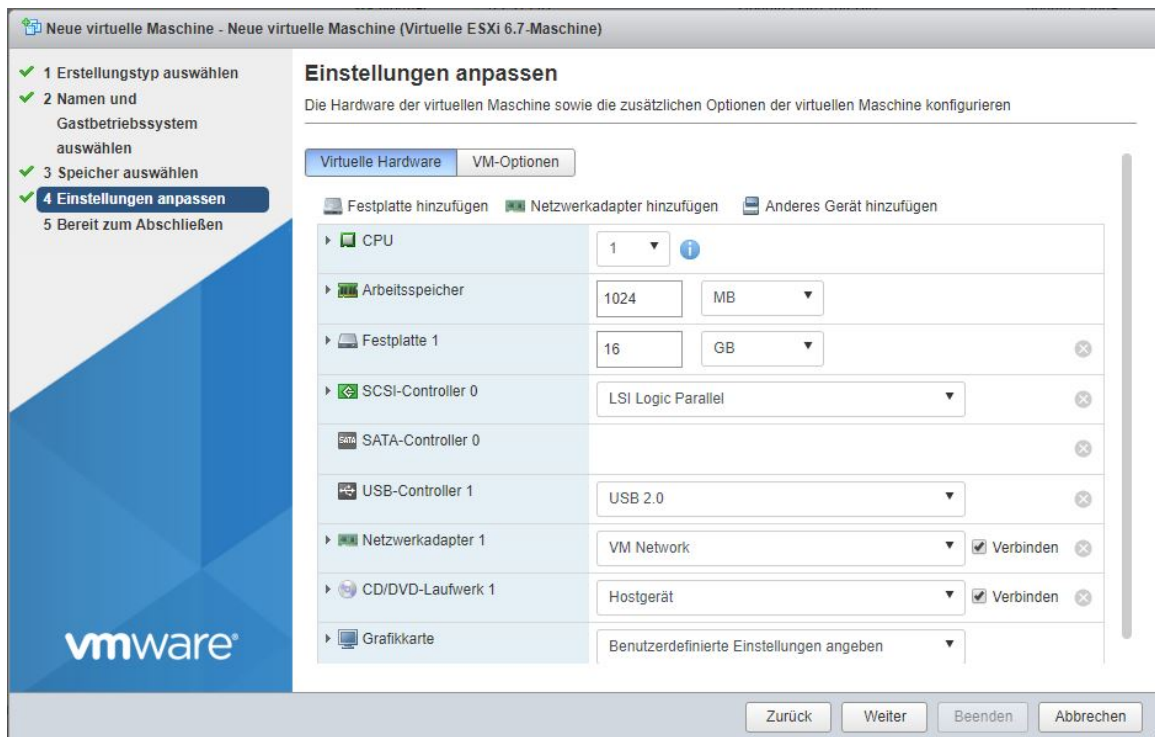


Abbildung 2: Anpassung der Parameter bei Erstellung einer virtuellen Maschine in VMware ESXi



```
administrator@ubuntu_icinga:~$ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW      Length >= 8
MEDIUM  Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary      file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 2
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
- Dropping test database...
Success.

- Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Abbildung 3: Ausführen des Skripts „mysql\_secure\_installation“ zur Absicherung eines MySQL-Systems



Abbildung 4: Angezeigter PHP-Fehler nach Installation des Icinga-Webfrontends

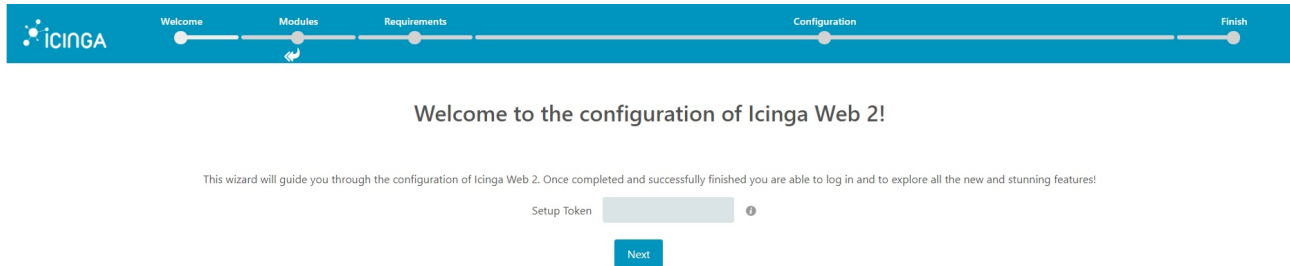


Abbildung 5: Willkommens-Bildschirm des Konfigurationsassistenten

**Database Resource**

Now please configure the database resource where to store users and user groups.  
Note that the database itself does not need to exist at this time as it is going to be created once the wizard is about to be finished.

The configuration has been successfully validated.

Resource Name *	icingaweb_db	i
Database Type *	MySQL	MySQL i
Host *	localhost	i
Port		i
Database Name *	web2_users	i
Username *	icinga2	i
Password *	.....	i
Character Set		i

Use SSL ☐ i

[Back](#) [Next](#) [Validate Configuration](#)

Abbildung 6: Einrichtung der Datenbank für Webfrontend-Benutzer

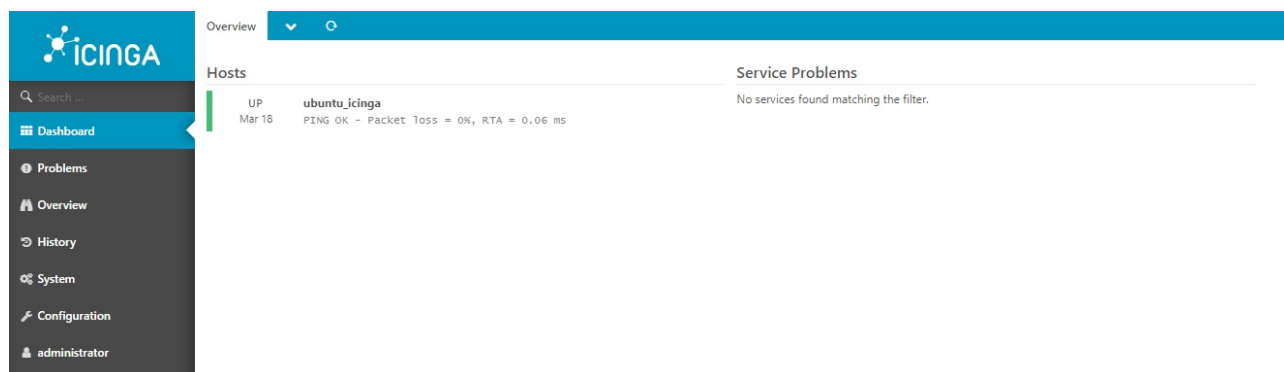


Abbildung 7: Startseite von „Icinga 2“ nach der Erstkonfiguration. Der Server, auf dem die Instanz von „Icinga 2“ läuft, ist automatisch als erster Server hinzugefügt

```
object Host "debian" {                                // Name des Hosts, am besten FQDN
    import "generic-host"                             // Vorlage für Host
    address = "10.129.37.217"                          // IP-Adresse oder FQDN
    vars.os = "Linux"                                 // Betriebssystem-Variable
    vars.agent_endpoint = "debian"                   // Endpunkt auf dem die Checks durchgeführt werden
                                                    // In diesem Fall das hiermit erstellte Host object "debian"
}

object Host "windows" {
    import "generic-host"
    address = "10.129.37.76"
    vars.os = "Windows"
    vars.agent_endpoint = "windows"
}
```

Abbildung 8: Zwei neue Server werden durch Anhängen dieser Zeilen an `/etc/icinga2/conf.d/hosts.conf` dem Monitoring hinzugefügt

```
object CheckCommand "cpuload" {
    command = [ PluginDir + "/check_load" ]
    arguments = {
        "-r" = { }

        "--critical" = {
            value = "0.8,0.8,0.8"
        }
    }
}
```

Abbildung 9: Beispiel für ein definiertes „CheckCommand“-Objekt

## A.4 Detaillierte Zeitplanung

Analysephase			9 h
1. Analyse des Ist-Zustands			3 h
1.1. Fachgespräch mit der EDV-Abteilung			1 h
1.2. Prozessanalyse			2 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse			1 h
3. Erstellen eines „Use-Case“-Diagramms			2 h
4. Erstellen des Lastenhefts mit der EDV-Abteilung			3 h
Entwurfsphase			19 h
1. Prozessentwurf			2 h
2. Datenbankentwurf			3 h
2.1. ER-Modell erstellen			2 h
2.2. Konkretes Tabellenmodell erstellen			1 h
3. Erstellen von Datenverarbeitungskonzepten			4 h

```
apply Service "loadcheck" {  
    import "generic-service"  
  
    check_command = "cpuload"  
  
    command_endpoint = host.vars.client_endpoint  
  
    assign where host.address  
}
```

Abbildung 10: Beispiel für einen definierten Dienst

The screenshot displays the Icinga 2 web interface for a service named 'ubuntu\_icinga'. At the top, a status bar shows 'UP' with a green bar, 'since Mar 18', and '127.0.0.1'. Below this, it shows 'OK' with a green bar, 'since Mar 18', and 'Service: http'. A row of icons for 'Check now', 'Comment', 'Notification', and 'Downtime' is present. The 'Plugin Output' section shows 'HTTP OK: HTTP/1.1 200 OK - 166 bytes in 0.001 second response time'. The 'Problem handling' section includes links for 'Add comment', 'Schedule downtime', and 'HTTP Checks'. The 'Performance data' section shows a table with 'time' (799.00 µs, 10.00 s) and 'size' (166.00 B, -). The 'Notifications' section shows links for 'Send notification', 'Icinga 2 Admin', and 'Icinga 2 Admin Group'. The 'Check execution' section shows details for the 'http' command, including 'Check Source', 'Reachable', 'Last check', 'Next check', 'Check attempts', 'Check execution time', and 'Check latency'.

Label	Value	Max
time	799.00 µs	10.00 s
size	166.00 B	-

Abbildung 11: Detailansicht eines überprüften Dienstes; hier die Webserverüberwachung

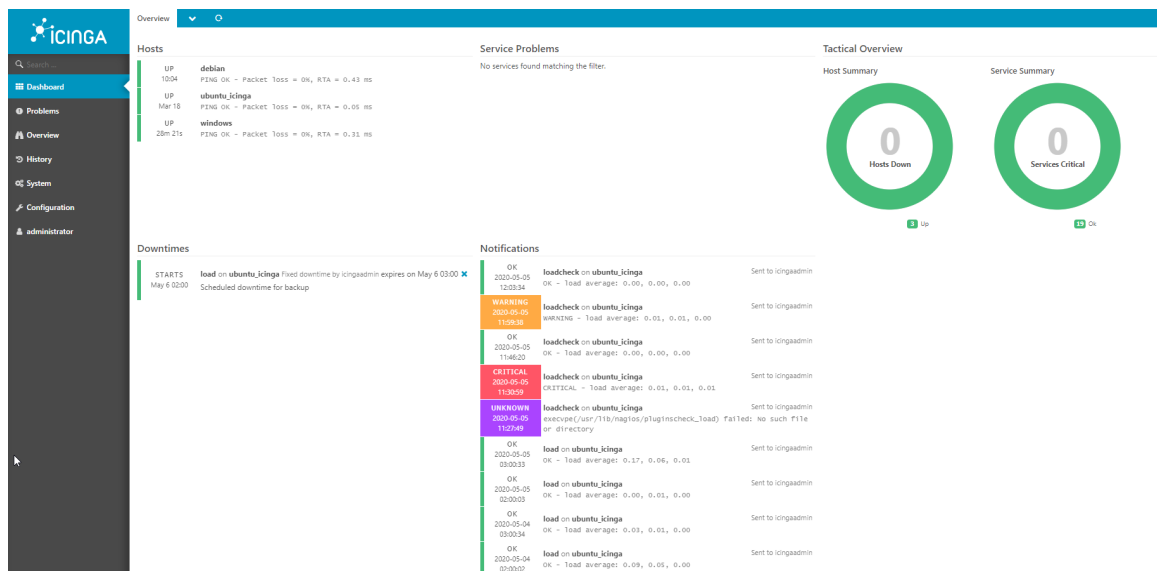


Abbildung 12: Selbsterstelltes Dashboard

## A.5 Use Case-Diagramm

Use Case-Diagramme und weitere UML-Diagramme kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/usecase-diagram.html>.

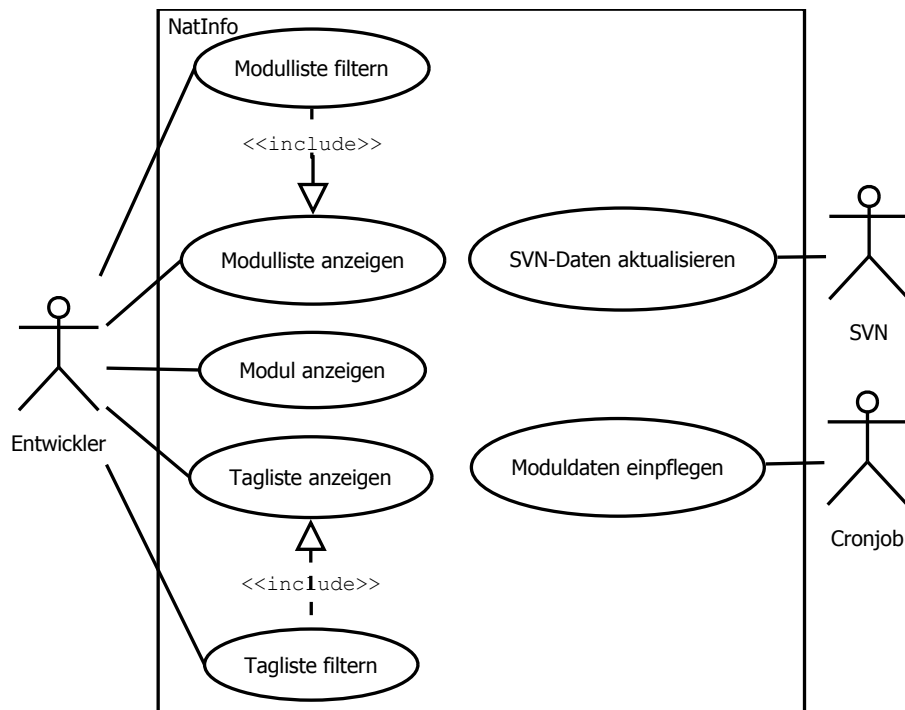


Abbildung 13: Use Case-Diagramm

## A.6 Datenbankmodell

ER-Modelle kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

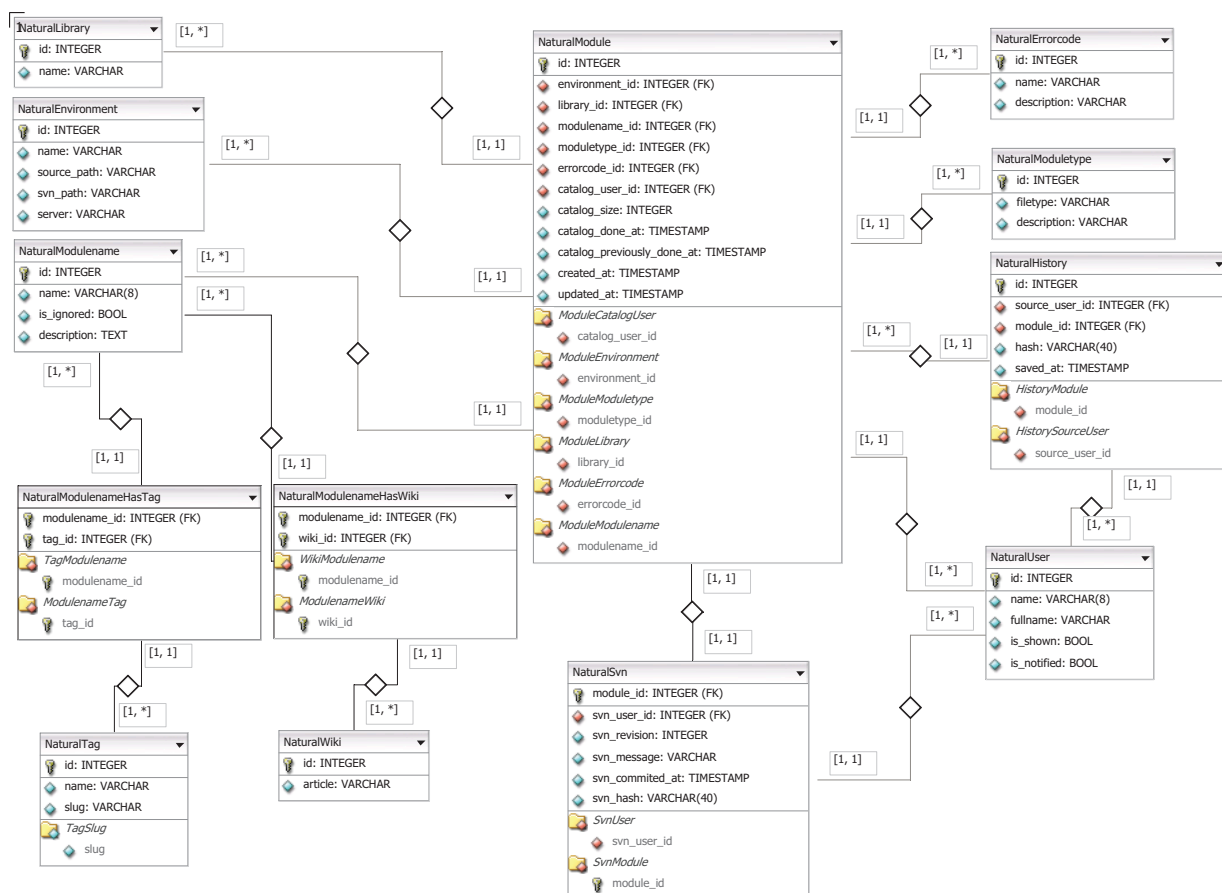


Abbildung 14: Datenbankmodell

## A.7 Oberflächenentwürfe

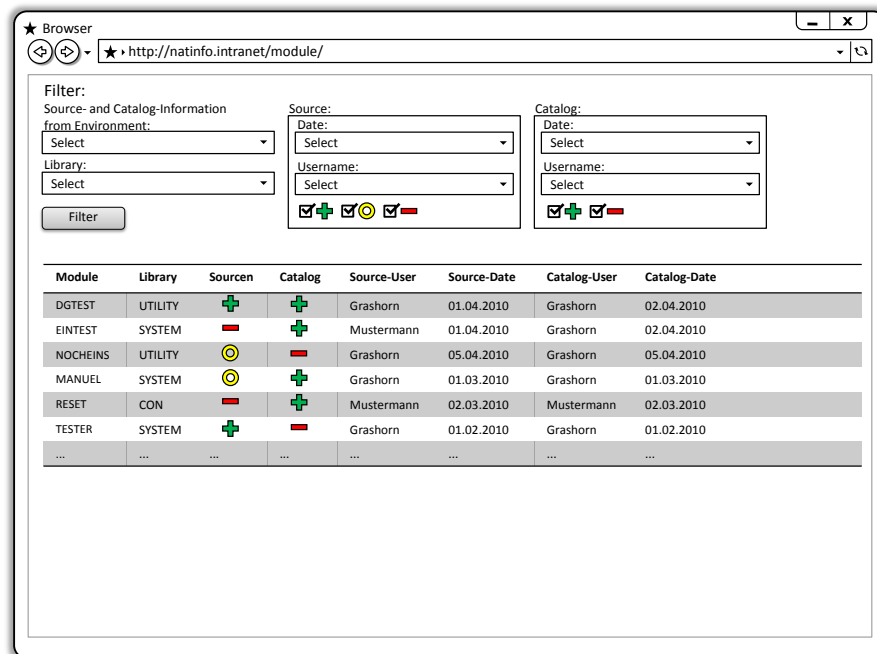


Abbildung 15: Liste der Module mit Filtermöglichkeiten



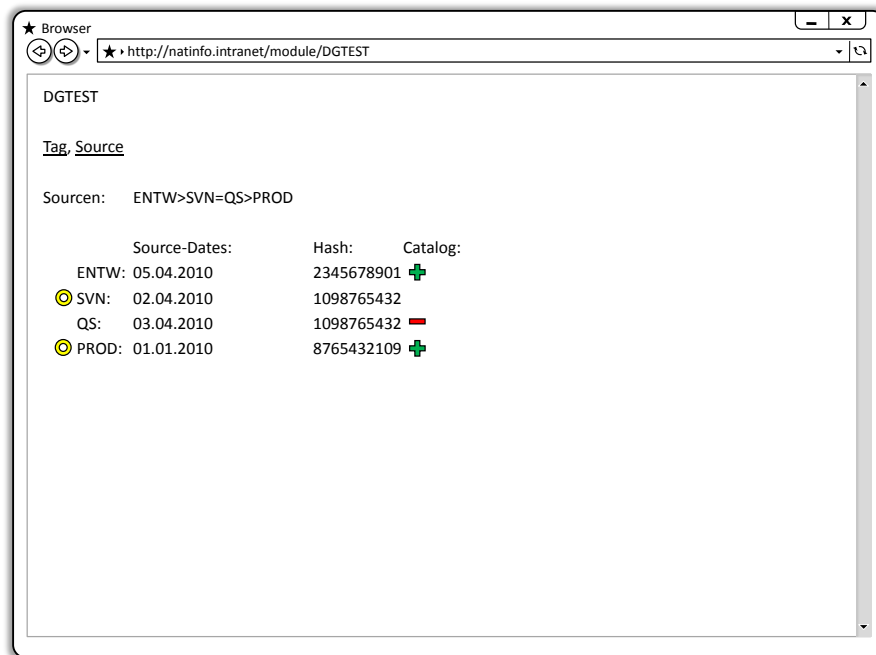


Abbildung 16: Anzeige der Übersichtsseite einzelner Module

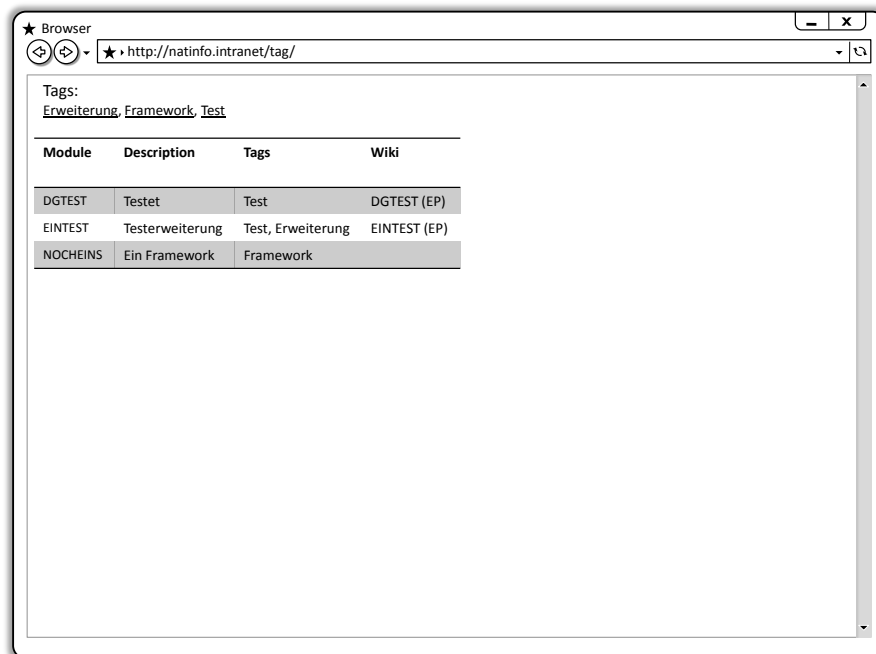


Abbildung 17: Anzeige und Filterung der Module nach Tags

## A.8 Entwicklerdokumentation

lib-model

[ class tree: lib-model ] [ index: lib-model ] [ all elements ]

**Packages:**  
lib-model

**Files:**  
Naturalmodulename.php

**Classes:**  
Naturalmodulename

# Class: Naturalmodulename

Source Location: /Naturalmodulename.php

### Class Overview

```
BaseNaturalmodulename
|
--Naturalmodulename
```

Subclass for representing a row from the 'NaturalModulename' table.

### Methods

- [\\_\\_construct](#)
- [getNaturalTags](#)
- [getNaturalWikis](#)
- [loadNaturalModuleInformation](#)
- [\\_\\_toString](#)

---

### Class Details

[line 10]  
Subclass for representing a row from the 'NaturalModulename' table.

Adds some business logic to the base.

[ [Top](#) ]

---

### Class Methods

#### constructor [\\_\\_construct](#) [line 56]

```
Naturalmodulename __construct( )
```

Initializes internal state of Naturalmodulename object.

**Tags:**  
**see:** parent::\_\_construct()  
**access:** public

[ [Top](#) ]

#### method [getNaturalTags](#) [line 68]

```
array getNaturalTags( )
```

Returns an Array of NaturalTags connected with this Modulename.

**Tags:**

**return:** Array of NaturalTags  
**access:** public

[\[ Top \]](#)

---

**method getNaturalWikis** [line 83]

```
array getNaturalWikis( )
```

Returns an Array of NaturalWikis connected with this Modulename.

**Tags:**

**return:** Array of NaturalWikis  
**access:** public

[\[ Top \]](#)

---

**method loadNaturalModuleInformation** [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

**Tags:**

**access:** public

[\[ Top \]](#)

---

**method \_\_toString** [line 47]

```
string __toString( )
```

Returns the name of this NaturalModulename.

**Tags:**

**access:** public

[\[ Top \]](#)

---

Documentation generated on Thu, 22 Apr 2010 08:14:01 +0200 by [phpDocumentor 1.4.2](#)

## A.9 Testfall und sein Aufruf auf der Konsole

```

1 <?php
2 include(dirname(__FILE__).'/../bootstrap/Propel.php');
3
4 $t = new lime_test(13);
5
6 $t->comment('Empty Information');
7 $emptyComparedInformation = new ComparedNaturalModuleInformation(array());
8 $t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, '
    Has no catalog sign');
9 $t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_CREATE, '
    Source has to be created');
10
11 $t->comment('Perfect Module');
12 $criteria = new Criteria();
13 $criteria->add(NaturalmodulePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulePeer::doSelectOne($criteria);
15 $t->is($moduleName->getName(), 'SMTAB', 'Right module name selected');
16 $comparedInformation = $moduleName->loadNaturalModuleInformation();
17 $t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign
    shines global');
18 $t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign
    shines global');
19 $infos = $comparedInformation->getNaturalModuleInformations();
20 foreach($infos as $info)
21 {
22     $env = $info->getEnvironmentName();
23     $t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines at ' . $env);
24     if ($env != 'SVNENTW')
25     {
26         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines at ' .
            $info->getEnvironmentName());
27     }
28     else
29     {
30         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is empty
            at ' . $info->getEnvironmentName());
31     }
32 }
33 ?>

```

Listing 1: Testfall in PHP

```

ao-suse-ws1.ao-dom.alte-oldenburger.de - PuTTY
ao-suse-ws1:/srv/www/symfony/natural # ./symfony test:unit ComparedNaturalModuleInformation
1..13
# Empty Information
ok 1 - Has no catalog sign
ok 2 - Source has to be created
# Perfect Module
ok 3 - Right modulename selected
ok 4 - Source sign shines global
ok 5 - Catalog sign shines global
ok 6 - Source sign shines at ENTW
ok 7 - Catalog sign shines at ENTW
ok 8 - Source sign shines at QS
ok 9 - Catalog sign shines at QS
ok 10 - Source sign shines at PROD
ok 11 - Catalog sign shines at PROD
ok 12 - Source sign shines at SVNENTW
ok 13 - Catalog sign is empty at SVNENTW
# Looks like everything went fine.
ao-suse-ws1:/srv/www/symfony/natural #

```

Abbildung 18: Aufruf des Testfalls auf der Konsole

## A.10 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```

1 <?php
2 class ComparedNaturalModuleInformation
3 {
4     const EMPTY_SIGN = 0;
5     const SIGN_OK = 1;
6     const SIGN_NEXT_STEP = 2;
7     const SIGN_CREATE = 3;
8     const SIGN_CREATE_AND_NEXT_STEP = 4;
9     const SIGN_ERROR = 5;
10
11     private $naturalModuleInformations = array();
12
13     public static function environments()
14     {
15         return array("ENTW", "SVNENTW", "QS", "PROD");
16     }
17
18     public static function signOrder()
19     {
20         return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP, self::SIGN_CREATE, self::SIGN_OK);
21     }
22
23     public function __construct(array $naturalInformations)
24     {
25         $this->allocateModulesToEnvironments($naturalInformations);

```

```

26     $this->allocateEmptyModulesToMissingEnvironments();
27     $this->determineSourceSignsForAllEnvironments();
28 }
29
30 private function allocateModulesToEnvironments(array $naturalInformations)
31 {
32     foreach ($naturalInformations as $naturalInformation)
33     {
34         $env = $naturalInformation->getEnvironmentName();
35         if (in_array($env, self::environments()))
36         {
37             $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
38         }
39     }
40 }
41
42 private function allocateEmptyModulesToMissingEnvironments()
43 {
44     if (array_key_exists(0, $this->naturalModuleInformations))
45     {
46         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
47     }
48
49     for ($i = 0; $i < count(self::environments()); $i++)
50     {
51         if (!array_key_exists($i, $this->naturalModuleInformations))
52         {
53             $environments = self::environments();
54             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);
55             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
56         }
57     }
58 }
59
60 public function determineSourceSignsForAllEnvironments()
61 {
62     for ($i = 1; $i < count(self::environments()); $i++)
63     {
64         $currentInformation = $this->naturalModuleInformations[$i];
65         $previousInformation = $this->naturalModuleInformations[$i - 1];
66         if ($currentInformation->getSourceSign() <> self::SIGN_CREATE)
67         {
68             if ($previousInformation->getSourceSign() <> self::SIGN_CREATE)
69             {
70                 if ($currentInformation->getHash() <> $previousInformation->getHash())
71                 {
72                     if ($currentInformation->getSourceDate('YmdHis') > $previousInformation->getSourceDate('YmdHis'))
73                     {
74                         $currentInformation->setSourceSign(self::SIGN_ERROR);
75                     }
76                 }
77             }
78         }
79     }
80 }

```

```
76         else
77         {
78             $currentInformation->setSourceSign(self::SIGN_NEXT_STEP);
79         }
80     }
81     else
82     {
83         $currentInformation->setSourceSign(self::SIGN_OK);
84     }
85 }
86 else
87 {
88     $currentInformation->setSourceSign(self::SIGN_ERROR);
89 }
90 }
91 elseif ($previousInformation->getSourceSign() <> self::SIGN_CREATE && $previousInformation->
    getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP)
92 {
93     $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
94 }
95 }
96 }
97
98 private function containsSourceSign($sign)
99 {
100     foreach($this->naturalModuleInformations as $information)
101     {
102         if($information->getSourceSign() == $sign)
103         {
104             return true;
105         }
106     }
107     return false ;
108 }
109
110 private function containsCatalogSign($sign)
111 {
112     foreach($this->naturalModuleInformations as $information)
113     {
114         if($information->getCatalogSign() == $sign)
115         {
116             return true;
117         }
118     }
119     return false ;
120 }
121 }
122 ?>
```

Listing 2: Klasse: ComparedNaturalModuleInformation

## A.11 Klassendiagramm

Klassendiagramme und weitere UML-Diagramme kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

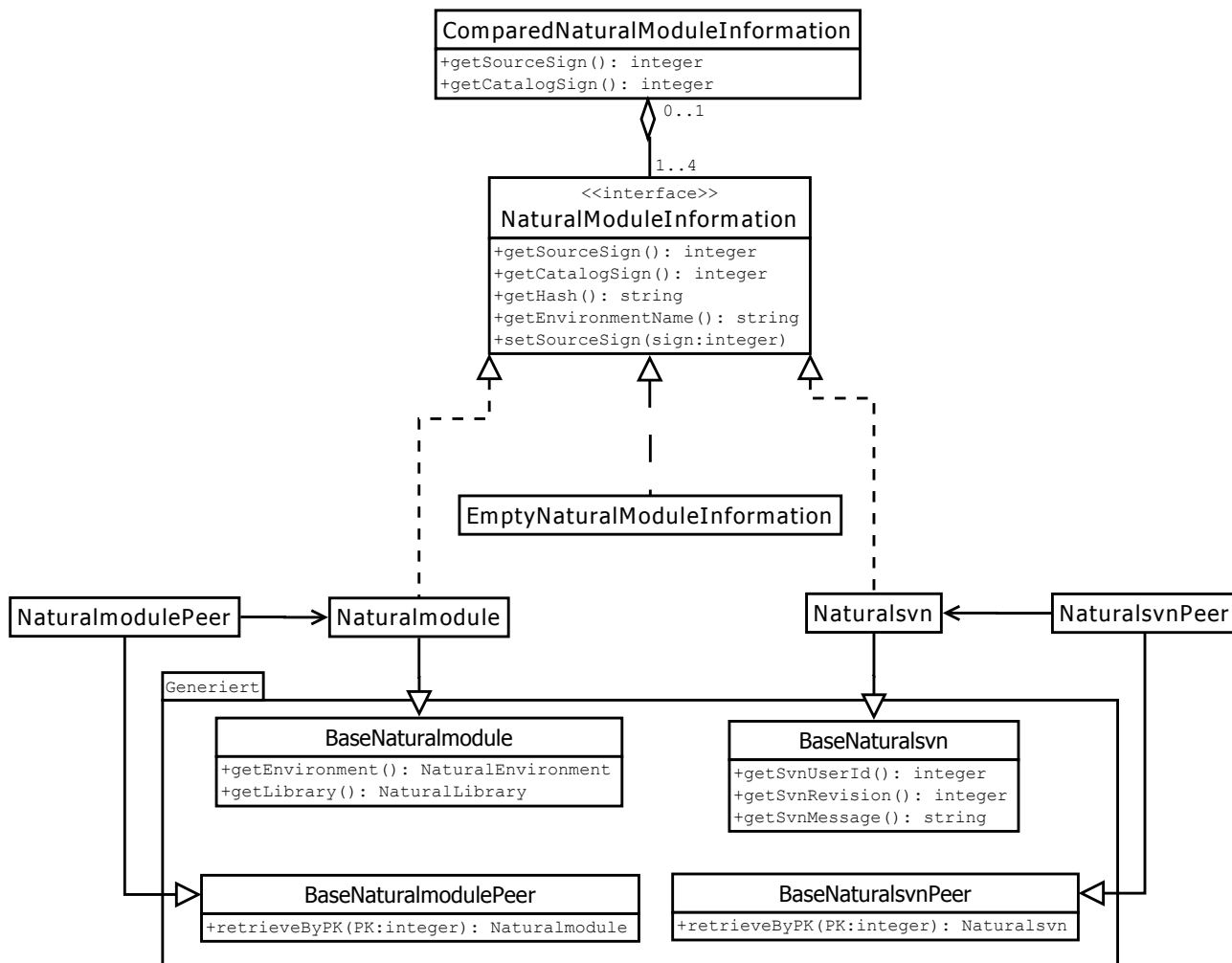







Abbildung 19: Klassendiagramm



## A.12 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

Symbol	Bedeutung global	Bedeutung einzeln
	Alle Module weisen den gleichen Stand auf.	Das Modul ist auf dem gleichen Stand wie das Modul auf der vorherigen Umgebung.
	Es existieren keine Module (fachlich nicht möglich).	Weder auf der aktuellen noch auf der vorherigen Umgebung sind Module angelegt. Es kann also auch nichts übertragen werden.
	Ein Modul muss durch das Übertragen von der vorherigen Umgebung erstellt werden.	Das Modul der vorherigen Umgebung kann übertragen werden, auf dieser Umgebung ist noch kein Modul vorhanden.
	Auf einer vorherigen Umgebung gibt es ein Modul, welches übertragen werden kann, um das nächste zu aktualisieren.	Das Modul der vorherigen Umgebung kann übertragen werden um dieses zu aktualisieren.
	Ein Modul auf einer Umgebung wurde entgegen des Entwicklungsprozesses gespeichert.	Das aktuelle Modul ist neuer als das Modul auf der vorherigen Umgebung oder die vorherige Umgebung wurde übersprungen.