Abschlussprüfung Sommer 2024

Dokumentation zur betrieblichen Projektarbeit

Automatisierte Bewertung von Auswirkungen auf die Sicherheit und Stabilität auf Kunden Systemen

Überprüfen von Kundenänderungen an Serverbetriebssystemen mittels KI

Zeitraum der Projektdurchführung:
Köln, 21.03.2024 - 29.03.2024

Von
Yahya Güzide
Fachinformatiker für Systemintegration
Prüflingsnummer: 5002354471

Ausbildungsbetrieb:
SVA GmbH
Borsigstraße 26
65205 Wiesbaden

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Allgemeine Informationen

2 Projektdefinition

2.1 Projektumfeld organisatorisch und technisch

Das Projekt wird als internes Projekt bei der System Vertriebe Alexander GmbH, im Folgenden als SVA abgekürzt, durchgeführt.

Die SVA ist Deutschlandweit eine der führenden IT-Dienstleistern dessen Hauptgeschäft der Verkauf und das Aufsetzen von Serversystemen ist. Ergänzend hierzu bietet die SVA auch andere Dienstleistungen rund um Entwicklung, Instandhaltung, Verwaltung Security sowie Prozessoptimierung an. Es werden circa 2.700 Mitarbeiter an 27 Standorten deutschlandweit, in Frankreich und den USA beschäftigt. Seit 2021 bietet die SVA ihren Kunden Managed Services in verschiedene Bereichen wie Cloud, M365, Virtual Desktop etc. an. Die Zielgruppe des Unternehmens besteht aus mittelständigen bis hinzu großen Unternehmen.

Die Durchführung des Projekts findet in der Abteilung MCMS im Team CALMA statt. Das Team CALMA ist zuständig für die Administration von Linux System, verwalten von den Azure Umgebungen, erstellen sowie bereitstelle von Container Resourcen und verwalten der Monitoring Infrastruktur. Das Projekt wird von Marco Wester begleitet, dieser ist SCRUM Master im Team und verwaltet das Security Produkt SentinelOne, sowie Administriert Linux Serversysteme.
Für das Projekt wurden mir mehrere Evaluationsdaten von Microsoft sowie Linux Kunden Systemen bereitgestellt. Diese Evalutaionsdaten wurden im verlauf des Projektes verwendet um mein Skript Testen zu können. Weiterhin musste ich anhand der Struktur dieser Evaluationsdaten einen Kontext zusammenzustellen, die der KI als prompt übergeben wird bevor die Evaluationsdaten an die KI gesendet werden. Desweiterem wurde mir eine Chat-GPT 4 Instanz, die über API erreicht werden kann, bereitgestellt (die Auswahl der KI wird in <füge link zum text> genauer beschrieben).

Für mein Projekt wurden als Budget <BUDGET> festgelegt. Dieses Budget sollte die Arbeitskosten, Kosten für die im Schnitt benötigten Tokens und ggfs. anfallende Lizenzkosten abdecken.

2.2 Projektvorfeld (Ist-Analyse)

Im Rahmen von Managed Service Verträgen betreibt und verwaltet die SVA hunderte von Servern für ihre Kunden. Da die SVA jederzeit compliant sein und die vereinbarten SLAs erfüllen muss, sind die Privilegierten Rechte ausschließlich den SVA-Administratoren vorbehalten. Damit unsere Kunden aber dennoch ihre Individuellen Applikationen einrichten und betreuen können, muss ihnen ermöglicht werden temporär mit administrativen Rechten auf ihre Systeme zuzugreifen. Die Befehlshistorie der Kundensysteme wird derweil zentralisiert gesammelt und den Administratoren bereitgestellt. Aktuell werden die Befehlshistorie und Konfigurationsdateien manuell von den SVA-Administratoren genauestens geprüft, um sicherzustellen, dass kein ungewollten oder gefährdenden Änderungen an den Kundensystemen vorgenommen wurden. Dieser Prozess beansprucht sehr viel Zeit und kann schnell Monoton werden, die dann zu Unachtsamkeit und damit einhergehend zu Fehlern führen können.

2.3 Problembeschreibung

Mit einer stetig wachsenden Anzahl an Kundensysteme steigt der Arbeitsaufwand den die SVA-Administratoren aufbringen müssen, um stets die vereinbarten SLAs einhalten zu können. Außerdem ist der Aufwand der Analyse so beträchtlich, dass diese die SVA-Administratoren zeitlich bindet und zeitgleich kein Entwicklungstätigkeiten etc. ermöglicht. Um weiterhin die von unseren Kunden erwartete Qualität anbieten zu können und somit auf dem Markt konkurrenzfähig zu bleiben, muss dieser Prozess vereinfacht oder wenn möglich Automatisiert werden.

2.4 Anforderungen (Soll-Konzept)

Das Projekt zielt darauf ab den derzeitig sehr aufwändigen Prozess zu verbessern. Basieren auf einer Nutzwertanalyse (<hier ein link zu nutzwertanalyse>) soll ein KI-Modell gewählt werden. Daraufhin soll für das Projekt dies intern bereitgestellt werden. Die KI wird über die API angesprochen. Das zu entwickelnde Skript soll den Kontext und die Befehlshistorie sowie die Änderungen der Konfigurationsdateien über die API an die KI übergeben. Der KI wird außerdem noch die gewünschte Struktur der Ausgabe mitgeteilt. Die Rückgabe der KI soll von meinem Skript ausgewertet und eine Rückgabe ausgegeben werden, die vom Automationssystem weiterverarbeitet werden kann. Ergänzend dazu soll eine Analyse der durchschnittlichen kosten für die Anfragen an die KI erstellt werden.

2.5 Auftragsbeschreibung und Pflichtenheft

Die Auftragsbeschreibung wurde während des Kick-off-Meetings mündlich von meinem Teamleiter Ruben Meichsner mitgeteilt. Die mir mitgeteilten Projektziele wurden in einem Lastenheft zusammen gefasst (Anhang A.2). Anschließen wurde auf Basis des Lastenhefts und dem ermitteltem Soll-Konzept ein Pflichtenheft (Anhang A.3) zusammengestellt.
Die Wichtigsten im Pflichtenheft festgelegten Kriterien sind, dass die KI die von der SVA gestellte Compliance mit den relevanten Sicherheitsstandards und -vorschriften gewährleistet. Des Weiteren muss das Skript mit der KI kompatibel sein, die Rückgabe muss so strukturiert sein, das dieses in die aktuelle IT-Infrastruktur integriert werden kann. Hinzu kommt das die Betriebskosten für die KI nicht das Monatliche Budget von <Betriebskosten> überschreiten darf.

2.6 Projektabgrenzung

Da es sich bei diesem Projekt um eine PoC (Proof of Concept) handelt wurden mir folgende Abgrenzungskriterien vorgeschrieben, die Anbindung an die IT-Systeme ist nicht in diesem Projekt vorgesehen. Des weiterem ist eine Umfangreiche Auswertung und Bewertung der Rückgaben nicht Bestandteil dieses Projektes.

Mit der Definition des Pflichtenheftes ist die Phase der Projektdefinition abgeschlossen und der erste Meilenstein erreicht.

3 Projektplanung

3.1 Projektphasen

Beispiel Tabelle 1 zeigt ein Beispiel für eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	9 h
Entwurfsphase	19 h
Implementierungsphase	29 h
Abnahmetest der Fachabteilung	1 h
Einführungsphase	1 h
Erstellen der Dokumentation	9 h
Pufferzeit	2 h
Gesamt	**70 h**

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1.

3.2 Abweichungen vom Projektantrag

- Sollte es Abweichungen zum Projektantrag geben (z. B. Zeitplanung, Inhalt des Projekts, neue Anforderungen), müssen diese explizit aufgeführt und begründet werden.

3.3 Ressourcenplanung

- Detaillierte Planung der benötigten Ressourcen (Hard-/Software, Räumlichkeiten usw.).
- Ggfs. sind auch personelle Ressourcen einzuplanen (z. B. unterstützende Mitarbeiter).
- Hinweis: Häufig werden hier Ressourcen vergessen, die als selbstverständlich angesehen werden (z. B. PC, Büro).

3.4 Entwicklungsprozess

- Welcher Entwicklungsprozess wird bei der Bearbeitung des Projekts verfolgt (z. B. Wasserfall, agiler Prozess)?

4 Durchführungsphase

4.1 Implementierung der Datenstrukturen

- Beschreibung der angelegten Datenbank (z. B. Generierung von **SQL!** aus Modellierungswerkzeug oder händisches Anlegen), **XML!**-Schemas usw..

4.2 Implementierung der Benutzeroberfläche

- Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Implementierung der Geschäftslogik erfolgt (z. B. bei **HTML!**-Oberflächen und Stylesheets).
- Ggfs. Beschreibung des Corporate Designs und dessen Umsetzung in der Anwendung.
- Screenshots der Anwendung

Beispiel Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang ??.

4.3 Implementierung der Geschäftslogik

- Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfsmuster zeigen.
- Quelltextbeispiele zeigen.
- Hinweis: Wie in Kapitel **??**: **??** zitiert, wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurchführung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

Beispiel Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang A.5.

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 ??. Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

6 Fazit

6.1 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 2 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 2: Soll-/Ist-Vergleich

6.2 Lessons Learned

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

6.3 Ausblick

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

Eidesstattliche Erklärung

Ich, Yahya Güzide, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

> *Automatisierte Bewertung von Auswirkungen auf die Sicherheit und Stabilität auf Kunden Systemen – Überprüfen von Kundenänderungen an Serverbetriebssystemen mittels KI*

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.

Köln, den 14.03.2024

Yahya Güzide

A Anhang

A.1 Glossar

Glossar

API Aplication Programming Interface, eine Software Schnittstelle zu einer Applikation mit der man dieses an andere Applikationen anbinden kann und so Befehle und Daten an dieses übergeben kann.

CLI Command Line Interface, textbasierte Benutzeroberfläche, mit der man direkt mit dem Betriebssystem interagieren kann, indem man Befehle eintippt.

Kontext Zusätzliche Informationen, die ein KI-System benötigt, um ein Situation oder Aufgabe korrekt zu verstehen und zu interpretieren.

prompt Eine Anweisung an ein KI-System, die vorgibt, welche Aufgaben es zu erledigen hat und wie es diese lösen soll.

Token Kleinste Einheit, die von einem Sprachmodell verarbeitet werden kann.

A.1 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	
3.1. Verarbeitung der CSV-Daten	1 h		
3.2. Verarbeitung der SVN-Daten	1 h		
3.3. Verarbeitung der Sourcen der Programme	2 h		
4. Benutzeroberflächen entwerfen und abstimmen		2 h	
5. Erstellen eines UML-Komponentendiagramms der Anwendung		4 h	
6. Erstellen des Pflichtenhefts		4 h	
Implementierungsphase			**29 h**
1. Anlegen der Datenbank		1 h	
2. Umsetzung der HTML-Oberflächen und Stylesheets		4 h	
3. Programmierung der PHP-Module für die Funktionen		23 h	
3.1. Import der Modulinformationen aus CSV-Dateien	2 h		
3.2. Parsen der Modulquelltexte	3 h		
3.3. Import der SVN-Daten	2 h		
3.4. Vergleichen zweier Umgebungen	4 h		
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h		
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h		
3.7. Anzeigen einer Liste mit den Modulen und geparsten Metadaten	3 h		
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h		
4. Nächtlichen Batchjob einrichten		1 h	
Abnahmetest der Fachabteilung			**1 h**
1. Abnahmetest der Fachabteilung		1 h	
Einführungsphase			**1 h**
1. Einführung/Benutzerschulung		1 h	
Erstellen der Dokumentation			**9 h**
1. Erstellen der Benutzerdokumentation		2 h	
2. Erstellen der Projektdokumentation		6 h	
3. Programmdokumentation		1 h	
3.1. Generierung durch PHPdoc	1 h		
Pufferzeit			**2 h**
1. Puffer		2 h	
Gesamt			**70 h**

A.2 Lastenheft

Die Projektziele wurden während des Kick-off-Meetings vom Auftraggeber Ruben Meichsner mir mündlich mitgeteilt. Das folgende Lastenheft nach Balzert fasst alle punkte aus dem Meeting zusammen und dient zur besseren Übersicht.

1. **Zielbestimmung**
   Mit diesem Projekt soll ein PoC vorgeführt werden, wie eine KI-Basierte Lösung zur Entlastung der SVA-Administratoren aussehen könnte. Des weiterem soll die Möglichkeit geschaffen werden diesen und ähnliche Prozesse, anhand von KI, in der Zukunft automatisieren zu können.

2. **Produkteinsatz**
   Das Produkt soll SVA-Administratoren entlasten und den Prozess nachdem Kunden der privilegierte zugriff entzogen wurde, vereinfachen.

3. **Produktübersicht**
   Es soll ein Skript entwickelt werden, welches per Schnittstelle Daten an eine KI übergibt und die Antwort klar strukturiert ausgibt. Diese soll SVA-Administratoren entlasten und dabei unterstützen um Sicherheits- und Stabilitätsrisiken nach Privilegierten zugriffen auf Kundensysteme festzustellen.

4. **Produktfunktion**
   Das Produkt soll Automatisiert oder Manuell von SVA-Administratoren gestartet werden können. Dabei ist es wichtig, dass die zu analysierenden Daten als Text an das Skript übergeben werden können. Das Skript fast die Daten in eine für die KI leicht verständliche Struktur zusammen, übergibt der KI den nötigen Kontext und prompted die Daten. Die Rückgabe erfolgt als text in einer Strukturierten Form, aus der schnell die Befunde ausgelesen werden können. Die Bewertung der KI bezüglich der Sicherheit und Stabilität muss anhand der Befunde begründbar sein.

5. **Qualitätsanforderungen**

   - Die Skriptsprache muss von vielen Teammitgliedern beherrscht werden
   - Sollte sich Problemlos in das vorhandene Automatisierungsumgebung integrieren lassen
   - Die Operativen Kosten dürfen den Monatlichen Budget nicht überschreiten
   - Die KI muss unsere Compliance einhalten
   - Der Kontext sollte alle wichtigen fälle beinhalten

6. **Ergänzungen**
   Um die KI für Reale Vorfälle vorzubereiten, werden ausschließlich Evaluierungsdaten verwendet die über die Zeit von unseren Kundensystemen gesammelt wurden. Weiterhin soll das Skript zum späteren Zeitpunkt einfach über die CLI getestet werden können.

Anhand des Projektergebnisses und der Wirtschaftlichen Betrachtung wird die Anbindung an die Produktive Automatisierungsumgebung in Erwägung gezogen.

A.3 Pflichtenheft (Auszug)

Zielbestimmung

1. Musskriterien
   1.1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninformationen sowie Symbolen zur Einhaltung des Entwicklungsprozesses an
      - In der Liste wird der Name, die Bibliothek und Daten zum Source und Kompilat eines Moduls angezeigt.
      - Ebenfalls wird der Status des Moduls hinsichtlich Source und Kompilat angezeigt. Dazu gibt es unterschiedliche Status-Zeichen, welche symbolisieren in wie weit der Entwicklungsprozess eingehalten wurde bzw. welche Schritte als nächstes getan werden müssen. So gibt es z. B. Zeichen für das Einhalten oder Verletzen des Prozesses oder den Hinweis auf den nächsten zu tätigenden Schritt.
      - Weiterhin werden die Benutzer und Zeitpunkte der aktuellen Version der Sourcen und Kompilate angezeigt. Dazu kann vorher ausgewählt werden, von welcher Umgebung diese Daten gelesen werden sollen.
      - Es kann eine Filterung nach allen angezeigten Daten vorgenommen werden. Die Daten zu den Sourcen sind historisiert. Durch die Filterung ist es möglich, auch Module zu finden, die in der Zwischenzeit schon von einem anderen Benutzer editiert wurden.

   1.2. Tag-Liste: Bietet die Möglichkeit die Module anhand von Tags zu filtern.
      - Es sollen die Tags angezeigt werden, nach denen bereits gefiltert wird und die, die noch der Filterung hinzugefügt werden könnten, ohne dass die Ergebnisliste leer wird.
      - Zusätzlich sollen die Module angezeigt werden, die den Filterkriterien entsprechen. Sollten die Filterkriterien leer sein, werden nur die Module angezeigt, welche mit einem Tag versehen sind.

   1.3. Import der Moduldaten aus einer bereitgestellten **CSV!**-Datei
      - Es wird täglich eine Datei mit den Daten der aktuellen Module erstellt. Diese Datei wird (durch einen Cronjob) automatisch nachts importiert.
      - Dabei wird für jedes importierte Modul ein Zeitstempel aktualisiert, damit festgestellt werden kann, wenn ein Modul gelöscht wurde.
      - Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Programmtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
      - Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.

   1.4. Import der Informationen aus **SVN!** (**SVN!**). Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein **PHP!**-Script auf der Konsole aufgerufen, welches die

Informationen, die vom **SVN!**-Kommandozeilentool geliefert werden, an **NatInfo!** übergibt.

1.5. Parsen der Sourcen
   - Die Sourcen der Entwicklungsumgebung werden nach Tags, Links zu Artikeln im Wiki und Programmbeschreibungen durchsucht.
   - Diese Daten werden dann entsprechend angelegt, aktualisiert oder nicht mehr gesetzte Tags/Wikiartikel entfernt.

1.6. Sonstiges
   - Das Programm läuft als Webanwendung im Intranet.
   - Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwicklungsprozessen ausgehen können.
   - Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

Produkteinsatz

1. Anwendungsbereiche
   Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informationen für die Module an einer Stelle gesammelt. Vorher getrennte Anwendungen werden ersetzt bzw. verlinkt.
2. Zielgruppen
   NatInfo wird lediglich von den **Natural!** (**Natural!**)-Entwicklern in der EDV-Abteilung genutzt.
3. Betriebsbedingungen
   Die nötigen Betriebsbedingungen, also der Webserver, die Datenbank, die Versionsverwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Webanwendung ist jederzeit aus dem Intranet heraus erreichbar.

A.4 Nutzwertanalyse

Mögliche Kandidaten an KI-Modellen habe ich mittels ausführlicher Recherche, sowohl im Internet und durch befragung von Arbeitskollegen ermittelt. Es sind folgende KI-Modelle als resultat der Recherche erfasst worden:

Chat-GPT 4 / Chat-GPT 4 Turbo / Chat-GPT 3.5 Turbo, Azure Mistral Large, Azure, AWS Bedrock Cloud 2 / Cloud 3 Opus / Sonnet / Haiku, AWS Bedrock Google Gemini Ollama, AWS Bedrock

A.5 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```php
<?php
class ComparedNaturalModuleInformation
{
  const EMPTY_SIGN = 0;
  const SIGN_OK = 1;
  const SIGN_NEXT_STEP = 2;
  const SIGN_CREATE = 3;
  const SIGN_CREATE_AND_NEXT_STEP = 4;
  const SIGN_ERROR = 5;

  private $naturalModuleInformations = array();

  public static function environments()
  {
    return array("ENTW", "SVNENTW", "QS", "PROD");
  }

  public static function signOrder()
  {
    return array( self :: SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP, self::
        SIGN_CREATE, self::SIGN_OK);
  }

  public function ___construct(array $naturalInformations)
  {
    $this->allocateModulesToEnvironments($naturalInformations);
    $this->allocateEmptyModulesToMissingEnvironments();
    $this->determineSourceSignsForAllEnvironments();
  }

  private function allocateModulesToEnvironments(array $naturalInformations)
  {
    foreach ($naturalInformations as $naturalInformation)
    {
      $env = $naturalInformation->getEnvironmentName();
      if (in_array($env, self :: environments()))
      {
        $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
      }
    }
  }

  private function allocateEmptyModulesToMissingEnvironments()
  {
    if (array_key_exists(0, $this->naturalModuleInformations))
    {
      $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
    }

    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              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 1: Klasse: ComparedNaturalModuleInformation