DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

GEBERIT

Zentrale Plattform für die Analyse und Verwaltung der Code-Sicherheit in Unternehmenssoftwaresystemen

Konzept und Implementierung einer zentralen Plattform für das Ausführen und Verwalten von automatisierten Sicherheits-Checks auf Eigenentwicklungen in einem hybriden SAP-Umfeld

Bachelorarbeit

in der sechsten Praxisphase

an der Fakultät Technik
im Studiengang Informationstechnik

an der DHBW Ravensburg
Campus Friedrichshafen

von
Johannes Brandenburger

25.09.2023

Bearbeitungszeitraum:	03.07.2023 - 25.09.2023
Matrikelnummer, Kurs:	1688304, Informationstechnik (B. Sc.) TIT20
Dualer Partner:	Geberit Verwaltungs GmbH
Betreuer im Partnerunternehmen:	Christian Reutebuch
Zweitgutachter:	Steffen Uhlig, IBM Deutschland Research & Development GmbH

Selbstständigkeitserklärung

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017.

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema:

Zentrale Plattform für die Analyse und Verwaltung der Code-Sicherheit in Unternehmenssoftwaresystemen

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Pfullendorf, 17.04.2023

Ort, Datum

Unterschrift

Gendererklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Bachelorarbeit auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Formulierungen gelten gleichermaßen für alle Geschlechter.

Abstract (English)

The company Geberit has several corporate software systems in use that are extended and adapted by employees. These developments are made using the ABAP programming language and include employee applications, the illustration of business processes and the customization of system functions for specific equipment and technologies.

Given the growing importance of enterprise software systems, it is critical to protect them as best as possible against attacks and malicious intentions. This also applies to the customization developments, which is why the implementation of a static code analysis was decided. More precisely, the SAP product „Code Vulnerability Analyzer“ will be used as a test tool in order to identify security vulnerabilities. Since several systems are in use in the corporate landscape, it is planned to make this analysis accessible via a central platform. In particular, developers will use the platform to view, fix and manage their own potential security vulnerabilities.

First, the bachelor project plans to collect the exact requirements for the code security housekeeping platform. It also explains the choices already made by the company (such as using the Code Vulnerability Analyzer). Between these two fixed points of view (determinations and requirements), a suitable solution is developed in the form of a concept.

To do this, a utility analysis is performed to choose between two fundamentally different approaches (running the analysis on on-premise or cloud). The system architecture is then designed to create the basis for implementation.

After the concept has been created, the development of the platform takes place. During the bachelor thesis, the system is fully implemented so that at the end of the processing period, there is a finished platform with automated security checks and a user interface for administration by developers.

In addition, once the implementation is complete, a workshop will be held to introduce the platform and provide an opportunity for developers to give feedback and ask questions about the use of the application. This feedback will then be incorporated into the platform after the completion of the bachelor thesis, which will result in iterative development.

In summary, the bachelor thesis will show how a housekeeping platform can be implemented to identify and fix security vulnerabilities in a multi-system landscape and what aspects need to be addressed.

Abstract (Deutsch)

Die Firma Geberit hat mehrere Unternehmenssoftwaresysteme im Einsatz, die von Mitarbeitern erweitert und angepasst werden. Diese Entwicklungen erfolgen in der Programmiersprache ABAP und umfassen unter anderem Anwendungen für Mitarbeiter, die Abbildung von Geschäftsprozessen und die Anpassung von Systemfunktionen an spezifische Anlagen und Technologien.

Angesichts der wachsenden Bedeutung von Unternehmenssoftwaresystemen ist es entscheidend, diese bestmöglich vor Angriffen und böswilligen Absichten zu schützen. Dies gilt auch für die Erweiterungsentwicklungen, weshalb die Implementierung einer statischen Code-Analyse dafür beschlossen wurde. Genauer soll hierfür das SAP-Produkt „Code Vulnerability Analyzer“ als Testwerkzeug eingesetzt werden, um Sicherheitslücken zu identifizieren. Da in der Unternehmenslandschaft mehrere Systeme im Einsatz sind, ist geplant, diese Analyse über eine zentrale Plattform zugänglich zu machen. Insbesondere Entwickler sollen die Plattform nutzen, um ihre eigenen potenziellen Sicherheitslücken einzusehen, zu beheben und zu verwalten.

Die Bachelorarbeit sieht vor zunächst die genauen Anforderungen an die Code-Sicherheits-Plattform zu erheben. Zudem werden die bereits vom Unternehmen getroffenen Entscheidungen (wie das Verwenden des Code Vulnerability Analyzer) erläutert. Zwischen diesen beiden festen Standpunkten (Festlegungen und Anforderungen) wird eine geeignete Lösung in Form eines Konzepts erarbeitet.

Hierzu erfolgt zunächst eine Nutzwertanalyse, um zwischen zwei grundlegend verschiedenen Ansätzen (Betrieb der Analyse auf On-premise oder Cloud) zu wählen. Anschließend wird die Systemarchitektur konzipiert, um die Grundlage für die Implementierung zu schaffen.

Nach dem Erstellen des Konzepts erfolgt die Entwicklung der Plattform. Im Rahmen der Bachelorarbeit wird das System vollständig implementiert, sodass am Ende des Bearbeitungszeitraums eine fertige Plattform mit automatisierten Tests und einer Benutzeroberfläche zur Verwaltung durch Entwickler vorhanden ist.

Darüber hinaus wird nach Abschluss der Implementierung ein Workshop stattfinden, bei dem die Plattform eingeführt wird und die Entwickler nach der Erklärung und Benutzung der Applikation Gelegenheit haben, Feedback zu geben und Fragen zur Nutzung der Anwendung zu stellen. Diese Rückmeldung wird dann nach Abschluss der Bachelorarbeit in die Plattform einfließen, was zu einer iterativen Weiterentwicklung führen soll.

Zusammenfassend soll die Bachelorarbeit zeigen, wie eine Code-Sicherheits-Plattform zur Identifizierung und Behebung von Sicherheitslücken in einer Mehr-System-Landschaft implementiert werden kann und welche Aspekte dabei zu beachten sind.

Inhaltsverzeichnis

Listingsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

1 Einleitung

1.1 Hinführung zum Thema

Die Digitalisierung schreitet mit großen Sprüngen voran und immer mehr Unternehmen unterstützen oder verlagern ihre Geschäftsprozesse und Firmendaten in IT-Systeme. Sei es in der Fertigung, im Vertrieb oder in der Verwaltung, überall werden Daten gesammelt, verarbeitet und ausgewertet. Bei allen Vorteilen, die dies mit sich bringt, wächst jedoch auch die Angriffsfläche für beispielsweise Cyberangriffe. Täglich erreichen Nachrichten über erfolgreiche System-Übernahmen oder komprimierte Passwörter die Medien. Um sich vor ungewollten Zugriffen zu schützen, müssen Unternehmen ihre Systeme so sicher wie möglich gestalten.

Unternehmenssoftwaresysteme werden aufgrund der großen Anforderungen an sie bei den allermeisten Unternehmen von Herstellerfirmen bezogen und nicht intern entwickelt. Allerdings müssen die Anwendungen angepasst werden, um die Geschäftsprozesse des Unternehmens abzubilden und Mitarbeiter bei ihrer Arbeit zu unterstützen. Diese Anpassungen werden bei vielen Firmen, so auch bei Geberit, zum Großteil von eigenen Entwicklern durchgeführt. Wie bei jeder Softwareentwicklung können auch hier Sicherheitslücken entstehen, die von Angreifern ausgenutzt werden können. So könnte beispielsweise ein bösartiger Mitarbeiter eine Sicherheitslücke in einer Anwendung ausnutzen, um sich Zugriff auf Daten zu verschaffen, die bei Veröffentlichung einen großen Schaden anrichten würden.

Um dies zu verhindern, müssen die Anwendungen auf Sicherheitslücken überprüft werden. Dies kann unter anderem durch statische Code-Analyse geschehen, bei der der Quellcode auf bekannte Muster Sicherheitslücken untersucht wird.

Diese Bachelorarbeit beschäftigt sich mit der Konzeption und Entwicklung einer zentralen Plattform, die solche statischen Code-Analysen auf Anwendungen in verschiedenen Unternehmenssoftwaresystemen durchführt und die gefundenen Sicherheitslücken verwaltet lässt. Die konkrete Problemstellung und das Ziel der Arbeit werden im folgenden Abschnitt definiert.

1.2 Problemstellung und Ziel

Entwickler von Geberit erweitern eingesetzte Unternehmenssoftwaresysteme durch viele Eigenentwicklungen. Diese Entwicklungen können Sicherheitslücken enthalten, welche von Angreifern oder Mitarbeiter mit böswilligen Absichten ausgenutzt werden können, um großen Schaden anzurichten. Es gibt bisher keine Lösung, die Eigenentwicklungen systematisch auf Sicherheitsrisiken zu überprüfen.

Das Ziel dieser Bachelorarbeit ist es, eine zentrale Plattform zu konzipieren und zu entwickeln, die automatisierte Sicherheits-Checks auf Eigenentwicklungen in verschiedenen Unternehmenssoftwaresystemen durchführt. Die Checks sollen in Form von statischer Code-Analyse in einem SAP-Umfeld umgesetzt werden. Die gefundenen Sicherheitslücken sollen mittels einer Benutzeroberfläche von den Entwicklern verwaltet und mithilfe der Plattform behoben werden können. Für das Ausführen der Sicherheits-Checks sollen zwei Ansätze evaluiert werden: On-Premise und Cloud. Am Ende der Bachelorarbeit soll die Plattform in Betrieb genommen werden und die Entwickler aufgefordert werden, diese zu nutzen, um ihre Eigenentwicklungen auf Sicherheitslücken zu überprüfen und diese bei Bedarf zu beheben.

Die Plattform soll speziell das oben beschriebene Problem für Geberit lösen und für die Anforderungen des Unternehmens ausgelegt sein. Das Konzept soll allerdings auch aufzeigen, wie eine solche Plattform allgemein aussehen könnte. Es könnte also auch in anderen Unternehmen Anwendung finden.

Eine Technologieentscheidung ist nicht Bestandteil dieser Arbeit. Lediglich die Entscheidung, ob die Analyse On-Premise oder in der Cloud ausgeführt werden soll, wird getroffen.

Die zentrale Plattform für die Analyse und Verwaltung der Code-Sicherheit in Unternehmenssoftwaresystemen wird in den folgenden Kapiteln „Code-Sicherheits-Plattform“ genannt.

1.3 Methodisches Vorgehen

Dieses Kapitel beschreibt den Aufbau der Arbeit und soll einen Überblick über die einzelnen Abschnitte verschaffen.

Zunächst wird in Kapitel 2 auf die Grundlagen eingegangen, die für das Verständnis der Arbeit notwendig sind. Dabei werden fachliche Themen wie Unternehmenssoftwaresysteme und statische Code-Analyse behandelt und es werden die verwendeten Methoden vorgestellt und definiert. Das Kapitel 3 beschreibt das Projektumfeld, in dem die Arbeit durchgeführt. Es werden die betroffenen Instanzen in Beziehung zueinander gesetzt und bereits eingesetzte Systeme vorgestellt.

Die Konzeption in Kapitel 4 konzentriert sich zunächst auf die Anforderungserhebung mittels Requirements Engineering. Zudem werden die bereits getroffenen Entscheidungen erläutert. Danach folgt eine Nutzwertanalyse, die über die Wahl des Betriebsmodells entscheidet, sodass dies in der Systemarchitektur berücksichtigt werden kann. Das erarbeitete mögliche Konzept zur Systemarchitektur bildet die Grundlage für die Implementierung, die in Kapitel 5 beschrieben wird.

Das Konzept und die Implementierung, welche die Hauptbestandteile der Arbeit sind, werden in Kapitel 6 evaluiert. Hierfür wird ein Workshop mit den Entwicklern durchgeführt, um die Plattform vorzustellen und Feedback zu erhalten. Abschließend wird in Kapitel 7 ein Fazit gezogen und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

2 Theoretische Grundlagen

2.1 Unternehmenssoftwaresysteme

2.1.1 Allgemein

Der Begriff „Unternehmenssoftwaresystem“ wird in dieser Bachelorarbeit als Oberbegriff für IT-Anwendungen verwendet, die in Unternehmen in den Bereichen Materialwirtschaft, Produktion, Personalwesen, Finanz- und Rechnungswesen, Controlling, Einkauf, Vertrieb, Marketing, Lagerhaltung, Logistik und Business Intelligence eingesetzt werden. Oft wird hierfür auch der Begriff „Enterprise-Resource-Planning-System“ verwendet. Bei Enterprise-Ressource-Planing-Systeme, kurz ERP-Systeme, handelt es sich aber genau genommen um IT-Systeme, die Unternehmen beim Steuern und Planen von Ressourcen, also bei Geschäftsprozessen, unterstützen. Darunter fällt also beispielsweise nicht das Business Intelligence. Die Literatur verwendet die Begriffe aber oft synonym und es wird keine strikte Trennung vorgenommen. Auch gibt es Unternehmenssoftwaresysteme, die nur ganz spezifische Teile eines Bereiches abdecken und andere, die Daten zu vielen Betriebsfeldern wie Produktion, Personalwesen, Rechnungswesen und Vertrieb in einer gemeinsamen Datenbasis halten. Mit diesen Daten in Kombination mit dem Abbilden von Unternehmensprozessen steigert ein ERP zum Beispiel vor allem die Effizienz der Planung und dem Controlling. (vgl. Chies 2016, S. 3; vgl. Schubert und Winkelmann 2023, S. 11)

2.1.2 On-Premise-Betrieb

Die klassische Art ein Unternehmenssoftwaresystem zu betreiben ist der On-Premise-Betrieb, bei dem die Software auf eigenen Servern im Unternehmen installiert und betrieben wird. Das System läuft also lokal und wird von den eigenen Mitarbeitern oder einem externen Dienstleister betreut. Ein Vorteil ist hierbei, dass das Unternehmen selbst die volle Kontrolle über die Daten hat, was gerade beim Thema Datenschutz ein wichtiger Punkt ist. Außerdem kann das Unternehmen die Software nach eigenen Wünschen anpassen und erweitern. Der entscheidende Nachteil ist, dass das Unternehmen selbst für die Wartung und die Verfügbarkeit des Systems verantwortlich ist. Zudem sind hohe Investitionskosten nötig, welche gerade in einer sich schnell entwickelnden IT-Branche schnell veraltet sein können. (vgl. Stefan Grieß 2022; vgl. Oracle 2023)

2.1.3 Cloud-Betrieb

Der cloudbasierte Betrieb von Unternehmenssoftware ist eine moderne Alternative zum On-Premise-Betrieb. Hierbei wird das System nicht lokal auf eigenen Servern betrieben, sondern auf Servern eines Cloud-Anbieters. Es gibt mehrere Möglichkeiten, in welcher Abstraktionsebene die Software betrieben wird. Bei allen Stufen wird allerdings die Verantwortung über gewisse Aufgaben, wie Wartung und Betrieb, an den Cloud-Anbieter abgegeben. Dadurch entfallen die hohen Investitionskosten für die Hardware, was gerade für kleine und mittelständische Unternehmen ein großer Vorteil ist. Außerdem ist das System immer auf dem neusten Stand und kann bei Bedarf schnell und einfach erweitert werden. Dazu kommt, dass die Leistung der Systeme bei Bedarf schnell und einfach skaliert werden kann und das Unternehmen nur für die Leistung bezahlt, die es auch wirklich benötigt. Insgesamt ist das Einrichten und Betreiben der Software wesentlich dynamischer und flexibler. Ein Nachteil ist, dass das Unternehmen die Kontrolle über die Daten an den Cloud-Anbieter abgibt, was gerade bei sensiblen Daten ein Problem sein kann. Außerdem können Anpassungen an spezifische Unternehmensprozesse schwieriger sein. (vgl. Stefan Grieß 2022; vgl. Oracle 2023)

2.1.4 SAP-Systeme und -Services

Da diese Bachelorarbeit in einem hybriden SAP-Umfeld stattfindet, werden in dem folgenden Abschnitt die wichtigsten verwendeten SAP-Systeme und SAP-Cloud-Services kurz vorgestellt.

SAP R/3 Enterprise Resource Planning (ERP) Das SAP R/3 ERP ist ein klassisches ERP-System, welches viele Jahre lang das Flaggschiff von SAP war. Das System kann mittels der Advanced Business Application Programming (ABAP) Programmiersprache erweitert und angepasst werden und somit für die verschiedensten Geschäftsprozesse eingesetzt werden. Es wurde in anderen Versionen auch unter den Namen „SAP ECC“ oder „SAP ERP“ vertrieben. (vgl. GAMBIT Consulting GmbH 2023)

SAP NetWeaver Gateway Ein SAP NetWeaver Gateway System ist eine Art Webserver, der das Erstellen und Betreiben von OData-Services ermöglicht. Diese werden dafür verwendet, Frontend-Applikationen mit SAP-Backend-Systemen zu verbinden. Das Gateway kann auch direkt auf einem SAP-System installiert werden, allerdings ist es empfohlen, das Produkt als eigenständiges System zu betreiben. (vgl. Menken 2016)

SAP Extended Warehouse Management (EWM) Das Unternehmenssoftwaresystem SAP EWM ist ein Warehouse-Management-System, welches die Verwaltung von Lagerbeständen und die Steuerung von Lagerprozessen ermöglicht. Es ist ein Teil des Produkts SAP Supply Chain Management (SCM), kann aber auch als eigenständiges System betrieben werden. (vgl. SAP 2023b)

SAP High Performance Analytic Appliance (HANA) SAP HANA ist eine relationale Datenbank-Technologie, die von SAP für die schnelle Verarbeitung großer Datenmengen entwickelt wurde. Dabei nutzt das Datenbanksystem die In-Memory-Technologie, bei der zu analysierende Daten im Arbeitsspeicher gehalten werden, um die Verarbeitungsgeschwindigkeit zu erhöhen. Zudem wird Parallelisierung eingesetzt, um von modernen Mehrkernprozessoren zu profitieren (vgl. Computerworld.ch 2011). Die Datenbank kann als On-Premise-Lösung betrieben werden oder als Database as a Service (DBaaS) von SAP bezogen werden (SAP HANA Cloud). (vgl. SAP 2023f)

SAP Business Suite 4 SAP HANA (S4/HANA) SAP S4/HANA ist die aktuelle ERP-Lösung von SAP und basiert auf der HANA-Datenbank-Technologie. Auch dieses System kann On-Premise installiert oder als Cloud-ERP genutzt werden. Das System kann mittels einer GUI verwalten werden. (vgl. SAP 2023c)

SAP S4/HANA EWM Seit 2016 ist SAP EWM auch als Teil von SAP S4/HANA verfügbar. Somit nutzt es die neue Datenbank-Technologie und kann auch als Cloud-Service bezogen werden. (vgl. Vanam 2020)

SAP Human Capital Management (HCM) for S4/HANA SAP HCM for S4/HANA ist eine On-Premise-Lösung für das Personalmanagement. Mit dem System können Mitarbeiterdaten verwaltet werden, die Unternehmenshierarchie abgebildet werden und Beziehungen zwischen Organisationseinheiten definiert werden. (vgl. Danielle Larocca 2021)

SAP Business Technology Platform (BTP) Die SAP BTP ist eine Cloud-Computing-Plattform, von SAP, die eine Reihe an modularen Services und Produkten an einem Ort zusammenfasst. Die Plattform bietet eine Infrastruktur, für das Entwickeln, Integrieren und Betreiben von Anwendungen und Prozessen, die auf SAP-Technologien basieren. Hierfür gibt Produkte in den Cloud-Computing-Modellen: CaaS, PaaS, FaaS

und SaaS. Für die Bereitstellung der Cloud-Services nutzt SAP die Infrastruktur von den IaaS-Anbietern Amazon Web Services (AWS) und Microsoft Azure. (vgl. Steckenborn 2022)

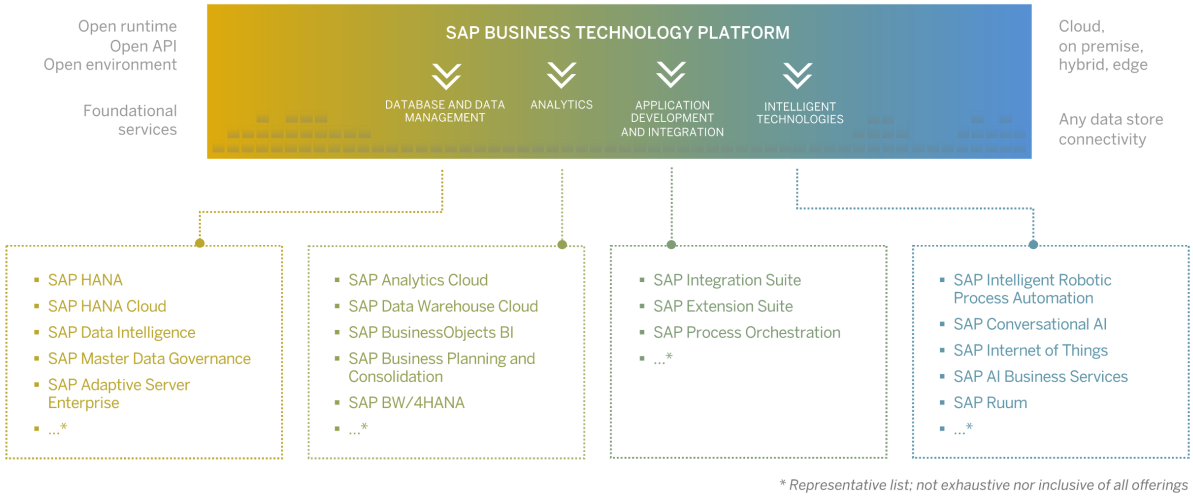


Abbildung 2.1: SAP BTP Produkte (Quelle: Feldherr 2023)

SAP Analytics Cloud (SAC) Das Produkt SAC ist eine Software-as-a-Service (SaaS)-Lösung von SAP, die für die Datenanalyse und -visualisierung verwendet werden kann und in der SAP BTP integriert ist. Der Vorteil von SAC ist, dass es eine Vielzahl an Datenquellen nativ unterstützt und somit eine zentrale Anlaufstelle für Analyse und Planung darstellt. Mit dem Produkt können Daten in Dashboards visualisiert werden und so bei Entscheidungen unterstützen. (vgl. SAP 2023d)

SAP Data Intelligence (SAP DI) Auch SAP DI ist ein Cloud-Produkt in der SAP BTP und dient der Datenintegration und -management. Dafür bietet es verschiedene Möglichkeiten, Daten aus unterschiedlichen Quellen zu extrahieren, zu transformieren und weiterzuverarbeiten. Auch im Machine Learning Bereich kann das Produkt eingesetzt werden. Die Transformationen können mittels einer grafischen Oberfläche und dem Bauen von Graphen erfolgen. Die Graphen bestehen aus mehreren Operatoren, welche die einzelnen Transformationsschritte darstellen. Es gibt Standard-Operatoren, die von SAP bereitgestellt werden, aber auch die Möglichkeit, eigene Operatoren in den Programmiersprachen Python, Go und JavaScript zu entwickeln. (vgl. SAP 2023e)

SAP Cloud Application Programming Model (SAP CAP) SAP CAP ist ein Framework für die Entwicklung von Backend-Services und Webanwendungen. Es verbindet mehrere OpenSource-Technologien, wie Node.js, Express.js und OData mit SAP-Technologien, wie die Modellierungssprache Core Data Services (CDS). Das Ziel beim Entwickeln mit SAP CAP ist es eine ganzheitliche Cloud-Native-Anwendung zu entwickeln, die (unter anderem) in der SAP BTP betrieben werden kann. Dafür bietet das Framework Werkzeuge für die Integration mit anderen Komponenten der SAP BTP, wie beispielsweise einer HANA Datenbank. (vgl. SAP 2023a)

SAP UI Development Toolkit for HTML5 (SAPUI5) Um User-Interfaces für beispielsweise SAP CAP Anwendungen zu entwickeln, entwickelt SAP das Development Toolkit SAPUI5. Es basiert auf der Webtechnologie HTML5 und bietet eine Reihe an UI-Elementen, die für die Entwicklung von Benutzeroberflächen verwendet werden können. Das Framework trennt die Definition der UI in drei Schichten: Model, View und Controller, wobei die Programmiersprachen JavaScript, XML und CSS verwendet werden. Ein SAPUI5-Frontend verbindet sich über Daten-Bindungs-Definitionen (Data-Bindings) mit den OData-Services einer SAP CAP App oder anderen Backend-Services. (vgl. SAP 2023g)

2.2 Statische Code-Analyse

2.2.1 Allgemein

Statische Code-Analyse ist ein Software-Testverfahren, das den Quellcode eines Programms analysiert, ohne diesen auszuführen. Alexandru G. Bardas definierte einen statischen Code-Analyzer wie folgt:

> Ein statischer Analysator oder Checker ist ein Programm, das geschrieben wurde, um andere Programme auf Fehler zu untersuchen. Diese Art von Programmen scannt den Quellcode, den Bytecode oder die Binärdateien eines Programms, um nach Mustern zu suchen. Statische Analysatoren haben das Potenzial, seltene Fehler oder versteckte Hintertüren zu finden. (übersetzt aus Bardas 2010)

Grundsätzlich kann die statische Code-Analyse auch manuell durchgeführt werden, allerdings ist dies bei großen Projekten mit vielen Codezeilen sehr aufwendig und fehleranfällig, weshalb die werkzeuggestützte statische Code-Analyse in der Praxis verwendet wird. Diese prüfen den Code auf spezielle Muster und können so Fehler entdecken. (vgl. Liggesmeyer 2009, S. 270)

Statische Code-Analyse kann direkt in einer Entwicklungsumgebung durchgeführt werden und so dem Programmierer direkt Feedback geben. Dieser Anwendungsfall wird oft als „Linting“ bezeichnet. Die Analyse kann aber auch beispielsweise periodisch auf eine große Code-Basis angewendet werden, um Fehler zu finden. (vgl. Dana Wilson (IBM) 2023)

2.2.2 ABAP Test Cockpit (ATC)

Das ABAP Test Cockpit (ATC) ist ein Analyse-Framework von SAP, welches in die SAP-GUI und in die SAP-Entwicklungsumgebung integriert ist. Das Framework wird dazu genutzt, durch statische Code-Analyse Probleme in ABAP-Code zu erkennen und bestimmte Programmierstile zu erzwingen. ABAP ist eine SAP-eigene Programmiersprache, die für die Entwicklung und Erweiterung von SAP-Systemen verwendet wird. ATC beinhaltet eine Reihe von Prüfvarianten, kann aber auch durch die Aktivierung von Erweiterungen oder durch das Hinzufügen von eigenen Prüfvarianten erweitert werden. Jede

Prüfvariante hat dann spezielle Muster, nach denen der Code überprüft wird. (vgl. DSAG 2020)

Die Tests im ATC beziehen sich unter anderem auf die Themen Sicherheit, Code-Robustheit, Update-Readiness, Unittests, Benutzbarkeit und Performance (vgl. Peter Barker (SAP) 2020).

2.2.3 Code Vulnerability Analyzer (CVA)

Der Code Vulnerability Analyzer (CVA) ist SAP Produkt, welches durch statische Code-Analyse ABAP-Code auf Sicherheitslücken prüft. Dabei ist es als Prüfvariante in das ATC integrierbar und kann mittels einer kostenpflichtigen Lizenz aktiviert werden. (vgl. Peter Barker 2017a)

Der Analysator prüft den Code auf Muster in den folgenden Kategorien:

- SQL Injection
- Code Injection
- Call Injection
- OS Command Injection
- Directory Traversal
- Backdoors & Authentications
- Web Exploitability

(vgl. Peter Barker (SAP) 2020)

In den meisten Fällen geht es bei den Sicherheits-Checks um Code-Stellen, bei denen Benutzereingaben nicht ausreichend validiert werden, bevor sie in SQL-Statements oder anderen kritischen Stellen verwendet werden (vgl. Peter Barker 2017a). Um diese Stellen zu finden und dabei so wenig Falschmeldungen wie möglich zu produzieren, führt der CVA eine Datenfluss-Analyse anhand des Quellcodes durch (vgl. Peter Barker 2017b).

2.3 Methodik

2.3.1 Requirements Engineering

Requirements Engineering ist eine Methodensammlung zur ingenieurmäßigen, systematischen Verarbeitung von Anforderungen. Der Prozess dient vor allem der Berücksichtigung von Bedürfnissen der Kunden. Dies geschieht durch das Entwickeln und Definieren von qualitativ guten Anforderungen, also dem Spezifizieren eines Systems, bevor dieses existiert. Das Vorgehen hat somit direkten Einfluss auf die Projektkosten. (vgl. Herrmann 2022, S. 1)

Eine Anforderung ist per Definition:

(1) „Ein notwendiges Bedürfnis eines Stakeholders."
(Problemraum, Fachsicht)
(2) „Eine Fähigkeit oder Eigenschaft, die ein System erfüllen muss."
(Lösungsraum, technische Sicht)
(3) „Eine dokumentiere Repräsentation eines Bedürfnisses, einer Fähigkeit oder Eigenschaft."
((1) oder (2))

(Pohl und Rupp 2021, S. 1).

Requirements Engineering findet in vielen verschiedenen Arten von Projekten Anwendung. In den nachfolgenden Erklärungen wird sich aber auf die Entwicklung von IT-Systemen beschränkt, da dies der Kontext dieser Bachelorarbeit darstellt.

Die Methode beinhaltet die Aktivitäten *Ermittlung*, *Dokumentation*, *Analyse*, *Prüfung*, *Abstimmung* und *Verwaltung* von Anforderungen und schafft dadurch eine gemeinsame, dokumentierte Wissensbasis zwischen Benutzern und Entwickler (vgl. Ebert 2022, S. 50).

Das Problem, das Requirements Engineering angeht, ist folgendes: Softwarearchitekten und -entwickler müssen genug Wissen darüber haben, was entwickelt werden soll. Wenn sie dieses Wissen nicht besitzen, wird das Produkt, was am Ende der Entwicklung vorliegt, höchstwahrscheinlich nicht nützlich sein (vgl. Herrmann 2022, S. 13). Es wird lediglich eine „Sammlung irrelevanter Funktionen" (Ebert 2022, S. 34).

Dieses Schaffen einer guten Wissensgrundlage von Entwicklungsteams, welche zu einer stabileren Umsetzung führt, ist das Ziel der Methode (vgl. Herrmann 2022, S. 14). Zudem ist ein wichtiger Aspekt die Erstellung einer Spezifikation als Output des Prozesses,

der „bindend“ zwischen Benutzer und Entwickler agiert. Es geht also bei dem Requirements Engineering nicht nur um das Finden von Anforderungen, sondern auch um deren Dokumentation.

Die konkrete Durchführung von Requirements Engineering variiert je nach Literaturquelle. Der IREB-Lehrplan[1] sieht folgende Vorgehensweise mit einer Unterteilung in 4 Hauptaufgaben vor.

1. Erhebung
2. Dokumentation
3. Validierung
4. Verwaltung

(vgl. Pohl und Rupp 2021, S. 13)

In der **Erhebungsphase** werden die Stakeholder des Projekts über ihre Ideen und Anforderungen an das System befragt. Da die Antworten auf solche Anfragen oft nicht sehr detailliert sind, werden häufig Workshops organisiert oder Interviews geführt, bei denen den Projektbeteiligten klargemacht werden muss, in welchem Detailgrad die Anforderungen kommuniziert werden müssen.

Häufig kommt es bei den Gesprächen mit den Stakeholdern zu Anforderungen, die in Konflikt zueinander stehen. Diese müssen dann durch Gesprächen bzw. Verhandlungen in generell akzeptierte Kompromisse umgewandelt werden. (vgl. Rupp und SOPHIST-Gesellschaft für Innovatives Software-Engineering 2021, S. 130)

Nach dem Kommunizieren mit den Stakeholdern müssen die Anforderungen analysiert, spezifiziert und priorisiert werden. Dies passiert in der **Dokumentationsphase**. Das Ergebnis der Dokumentation ist ein Spezifikationsentwurf, der die Anforderungen strukturiert und priorisiert in einem formellen Dokument definiert. Dies kann rein textuell oder durch Unterstützung von grafischen Darstellungen realisiert werden (vgl. Bron 2020, S. 10). Hier können die Anforderungen in funktionale und nicht-funktionale/qualitative Anforderungen aufgeteilt werden. Funktionale Anforderungen beschreiben kundenspezifische, zweckbestimmte Anforderungen an die Funktionen eines Systems. Qualitätsanforderung hingegen beschreiben, „wie“ das System sich in Bezug auf Zeitverhalten, Robustheit, Ressourcenverbrauch, Schönheit, Sicherheit, etc. verhalten soll und sind somit oft nicht produktspezifisch. (vgl. Herrmann 2022, S. 37)

[1] Die IREB (International Requirements Engineering Board) ist eine Non-Profit-Organisation, die Standards zum Thema Requirements Engineering definiert und Zertifikate ausstellt.

In der **Validierungsphase** wird der Spezifikationsentwurf von den Stakeholdern überprüft und bestätigt. Dafür wird das Dokument den Beteiligten präsentiert und wenn nötig modifiziert, bis alle mit den exakten Anforderungen einverstanden sind.

Die **Verwaltungsaufgabe** ist ein Prozess, der ständig während den anderen Phasen stattfindet. Ziel dieser Aufgabe ist es die gesammelten Anforderungen während einer Phase zu koordinieren und gegebenenfalls anzupassen. (vgl. Pohl und Rupp 2021, S. 13)

Die Hauptaufgaben können zwar grob in der oben angeführten Reihenfolge bearbeitet werden, allerdings ist es nicht möglich diese streng zu trennen, da beispielsweise beim Auffallen eines Fehlers bei der Validierung erneut ermittelt bzw. dokumentiert werden muss.

Das Requirements Engineering bringt eine Struktur in einen Projektstart und ist heutzutage essenziell für fast jede große Entwicklung, an der mehrere Kunden und Entwickler beteiligt sind.

2.3.2 Nutzwertanalyse

Die Nutzwertanalyse ist ein Bewertungsverfahren, das mehrere Alternativen anhand unterschiedlicher Kriterien bewertet. Die Methode kann zum Lösen von Auswahlproblemen („entweder ... oder ...“) oder für ein Priorisierungsproblemen genutzt werden, in dem es für jede Alternative einen Nutzwert berechnet. Dieser Nutzwert kann für Vergleiche verwendet werden. (vgl. Kühnapfel 2019, S. 6f)

Nutzwertanalysen werden zum Beispiel bei Investitions- oder Standortentscheidungen, beim Priorisieren von Agendapunkten benutzt (vgl. Nagel, Mieke und Teuber 2020, S. 55).

Die Durchführung einer Nutzwertanalyse lässt sich in mehrere Schritte unterteilen, die im Folgenden erläutert werden.

1. **Entscheidungsproblem beschreiben**
   Zunächst muss das Entscheidungsproblem beschrieben werden. Der Kontext muss dargelegt werden und es muss genau geklärt werden, zu welchem Zweck die Entscheidung dienen soll. Dabei wird unterschieden, ob die Nutzwertanalyse für ein Auswahlproblem oder für ein Priorisierungsproblem genutzt wird.
2. **Alternativen darlegen**
   Die Alternativen, zwischen denen mittels der Nutzwertanalyse entschieden werden

soll, bzw. die priorisiert werden sollen, müssen erhoben und erklärt werden. Gefunden werden diese häufig durch vorangehende Überlegungen, Recherchen oder Ideenfindungs-Verfahren.

3. **Bewertungskriterien definieren**
   In diesem Schritt werden Kriterien gesucht und definiert, die zur Bewertung dienen sollen. Vorzugsweise sollten dies metrische Messgrößen sein, um eine möglichst objektive Bewertung zu gewährleisten. Wenn sonstige „weiche“ Kriterien wichtig sind, so müssen diese anhand detaillierter Beschreibungen nachvollziehbar dargestellt werden.
4. **Bewertungskriterien gewichten**
   Das Gewichten der Bewertungskriterien ist entscheidend für das Ergebnis der Analyse und somit nicht zu vernachlässigen. Es dient dazu, bestimmte Kriterien in ihrer Entscheidungsrelevanz hervorzuheben, bzw. abzuschwächen. Die Summe aller Gewichte muss 100 % bzw. 1 sein. Um die Gewichtung zu vereinfachen, können mehrere Kriterien zu Kriterienkategorien zusammengefasst werden, sodass eine Struktur entsteht. Zudem können Paarvergleiche genutzt werden, um realistischere Gewichtungen zu generieren.
5. **Merkmale auf Skalen abbilden**
   Um eine einheitliche Bewertung zu erreichen, werden Transformationsvorschriften definiert, welche die Merkmale eines Kriteriums auf eine einheitliche Skala abbilden, dessen Intervall für alle Kriterien gleich ist. Typische Skalen wären „0-10“, „Schweizer Schulnoten (1-6)“ und „Oberstufen-Punkte (0-15)“. Quantitative, metrische Messgrößen können durch eine Transformationsfunktion auf die Skala abgebildet werden. Bei qualitativen, „weichen“ Kriterien können Intervalle bestimmt werden (Beispiel: *Kriterium ist hinreichend erfüllt, es gibt aber Mängel* $\rightarrow$ 6/10 Punkte). Hier muss dann aber ganz strikt definiert sein, was Einstufungen wie „erfüllt“ für das jeweilige Kriterium bedeutet, was sich oft als schwierig erweist. Negative Kriterien, wie beispielsweise Investitions-Kosten, müssen über invertierende Transformationen umgerechnet werden (gering Kosten $\rightarrow$ hoher Skalenwert)
6. **Alternativen bewerten**
   Nach den Vorbereitungsschritten, ist dies die eigentliche Bewertung der Entscheidungsobjekte. Je Alternative werden die Kriterien zunächst betrachtet und mittels der Transformationsvorschriften feste Skalenwerte bestimmt. Anschließend wird durch eine Multiplikation mit der Gewichtung der Teilnutzwert bestimmt. Die Teilnutzwerte werden aufsummiert und ergeben somit den Gesamtnutzwert der Alternative. Das Bewerten der Alternativen wird typischerweise in einer Matrix durchgeführt. Ein kurzes Beispiel hierfür zeigt die Abbildung 2.1.

(vgl. Nagel, Mieke und Teuber 2020, S. 54ff; vgl. Kühnapfel 2021, S. 18ff; vgl. Kühnapfel 2019, S. 6ff)

Kriterium	Gewichtung	Alternative 1		...		Alternative n	
		Skalenwert	Teilnutzwert	Skalenwert	Teilnutzwert	Skalenwert	Teilnutzwert
Bewertungs-kriterium 1	0,5	8	4	...	...	3	1,5
...	...	...	...	...	...	...	...
Bewertungs-kriterium m	0,2	2	0,4	...	...	9	0,45
Summe	1		Gesamt-nutzwert 1: 4,4		...		Gesamt-nutzwert n: 1,95

Tabelle 2.1: Matrix Nutzwertanalyse

Bei allen Schritte, vor allem aber beim Definieren und Gewichten von Bewertungskriterien, ist eine Absprache innerhalb dem Projektumfeld (beispielsweise in der Fachabteilung) sehr wichtig und essenziell für eine gute Akzeptanz und Rechtfertigung der Analyse (vgl. Nagel, Mieke und Teuber 2020, S. 56).

2.3.3 Workshop

Ein Workshop ist ein interaktives Arbeitstreffen mit einer bestimmten Personengruppe zur intensiven Auseinandersetzung zu einem bestimmten Thema. Das Ziel der Methode ist die Entwicklung von Ideen und konkreter Maßnahmen, welche dann in Zukunft umgesetzt werden sollen (vgl. Beermann und Schubach 2019, S. 6). Die Teilnehmergruppe setzt sich aus einem Moderator, den Stakeholdern des Projekts und Experten zusammen. Es ist also wichtig, zuvor alle Stakeholder zu identifizieren und diese (oder Vertreter von diesen) soweit das möglich ist in das Arbeitstreffen miteinzubeziehen. (vgl. Beermann und Schubach 2019, S. 8)

Die Methode hat zwei große Vorteile: Die entspannte Atmosphäre bietet, dass die Teilnehmer in kreativer und offener Art und Weise miteinander arbeiten, was oft ein Gegensatz zum alltäglichen Abteilungsumfeld ist und zu besserer Ideenfindung führt. Außerdem steigt die Akzeptanz und Motivation gegenüber der beschlossenen Ergebnissen und Maßnahmen bei den Teilnehmern. (vgl. Beermann und Schubach 2019, S. 6f)

Zu Beginn eines Workshops sollte der Ausgangspunkt, mit einem Problem oder Ziel, definiert und kommuniziert werden. Im Verlauf des Workshops erarbeiten die Teilnehmer dann Maßnahmen, um diesen Zustand zu verändern, wofür ausreichend Zeit eingeplant

werden muss. Zudem sollte darauf geachtet werden, dass die Teilnehmer direkt vom Thema betroffen sind bzw. Spezialisten sind, sodass auch wirklich nur projektrelevante Personen Lösungen betragen. (vgl. Beermann und Schubach 2019, S. 7f)

Ein Workshop ist nicht zu verwechseln mit einer Tagung oder eines Seminars, da anders als bei diesen Veranstaltungen, die Interaktivität und die Maßnamensfindung im Fokus steht (vgl. Beermann und Schubach 2019, S. 8).

Die Vorbereitung für einen Workshop beinhaltet das Identifizieren und Einladen der Teilnehmer und das Formulieren eines Problems, bzw. eines Ziels. Zudem wird der Ablauf festgelegt und die Arbeitsmaterialien vorbereitet. (vgl. Bohinc 2019, S. 60)

Der Workflow sollte mit einer Vorstellungsrunde und einer Erläuterung über Ziel und Ablauf starten. Dann werden in offenen Gesprächen Interessen, Anforderungen, Maßnahmen, Ideen und Wünsche die Beteiligten an das Projekt und das Projektziel gesammelt. Hierfür werden dann festen Formulierungen erarbeitet, die von allen Teilnehmer akzeptiert werden. (vgl. Bohinc 2019, S. 60)

Bei der Nachbearbeitung müssen die Ergebnisse in geeigneter Form dokumentiert werden. Gegebenenfalls folgt eine schriftliche Erklärung der Ergebnisse an die Teilnehmer des Workshops und den restlichen Stakeholdern. (vgl. Bohinc 2019, S. 60)

Wie bei vielen anderen Methoden ist auch der Workshop nicht ganz klar definiert, sondern individuell auf eine Situation anzupassen.

3 Projektumfeld

Dieses Kapitel beschreibt in welchem Umfeld die Bachelorarbeit entstanden ist. Dazu wird zunächst das Unternehmen Geberit vorgestellt. Danach werden die Stakeholder, die von der Code-Sicherheits-Plattform betroffen sind beschrieben und in einem System-Kontext-Diagramm in Beziehung gesetzt. Abschließend wird die Unternehmenssoftwaresystem-Landschaft beschrieben, in der die Plattform eingesetzt wird, da dies ein wichtiger Aspekt für die Konzeption und Umsetzung darstellt.

3.1 Geberit

Die Geberit Gruppe ist der europäische Marktführer im Bereich der Sanitärprodukte (Geberit 2022a). Der Hauptsitz befindet sich in Rapperswil-Jona (Schweiz), wo das Unternehmen 1817 von Caspar Melchior Albert Gebert gegründet wurde (Geberit 2022c). Die Gruppe hat Standorte in 50 Ländern und beschäftigt rund 12.000 Mitarbeiter. Sie ist seit 1999 erfolgreich an der SIX Swiss Exchangtonotiert und erzielte im Jahr 2022 einen Umsatz von CHF 3,39 Milliarden (Geberit 2022b).

Die global-agierende IT der Geberit-Gruppe sitzt überwiegend in Rapperswil-Jona und in Pfullendorf (Deutschland). Die IT ist in verschiedene Bereiche und Abteilungen aufgeteilt, die sich mit unterschiedlichen Themen beschäftigen.

3.2 Stakeholderbeschreibung

Entwickler Das Unternehmen Geberit beschäftigt ca. 50 Mitarbeiter in unterschiedlichen Ländern, die Erweiterungen für die verschiedenen Unternehmenssoftwaresysteme entwickeln. Dies geschieht mitunter durch das Entwickeln von Applikationen für Mitarbeiter, dem Abbilden von Geschäftsprozessen und dem Zuschneiden von System-Funktionen auf eingesetzte Anlagen und Technologien. Für diese Erweiterungs-Entwicklungen wird die Programmiersprache ABAP genutzt. Die Code-Sicherheits-Plattform analysiert den Erweiterungscode der Entwickler. Diese müssen die gefundenen Sicherheitslücken anschließend beheben und dies auf der Plattform vermerken. Sollten Unterstützung oder Anleitungen zur Fehlerbehebung benötigt werden, stehen entsprechende Ressourcen auf der Plattform zur Verfügung. Zudem können die Entwickler Berichte über Sicherheitslücken einsehen und so beispielsweise abschätzen, wo ihre Abteilung im Vergleich zu anderen Abteilungen bei der Code-Sicherheit steht.

Management Die Übersicht-Berichte über die gefundenen Sicherheitslücken können auch in der Management-Ebene genutzt werden, um Abteilungen oder sogar einzelne Entwickler zu bewerten. Möglicherweise kann sogar Druck ausgeübt werden, dass die Sicherheitslücken bis zu einem gewissen Zeitpunkt behoben sein müssen. Insgesamt verspricht sich das Management eine bessere Codequalität der Eigenentwicklungen und somit eine geringere Angreifbarkeit des Unternehmens.

SAP-Dev-Abteilung Die SAP-Development-Abteilung ist die Haupt-Entwicklungsabteilung der Geberit-IT und zählt somit auch zu den Entwicklern (3.2). Sie agiert vor allem als Dienstleister für die anderen Entwicklungsabteilungen, wobei sie größere Programmierungen umsetzen. Allerdings entwickelt die Abteilung auch eigene Applikationen, wie die dieser Bachelorarbeit. Die SAP-Dev-Abteilung ist verantwortlich für die Entwicklung und den Betrieb der Code-Sicherheits-Plattform. Außerdem sollten laufend Inhalte zum Beheben der (neuen) Sicherheitslücken zur Applikation hinzugefügt werden, um den Entwicklern Hilfestellungen zu geben.

SAP Auch SAP kann als Stakeholder gesehen werden, da die Plattform auf SAP-Produkte aufbaut und von diesen abhängig ist. Zum einen sind die zu analysierenden Systeme von SAP. Zum anderen baut die Code-Sicherheits-Plattform auf ein SAP-Produkt für statische Code-Analyse, dem „Code Vulnerability Analyzer“, auf.

3.3 System-Kontext-Diagramm

Um die Beziehungen zwischen den Stakeholdern und den Systemen zu visualisieren, wird in Abbildung 3.1 ein System-Kontext-Diagramm gezeigt.

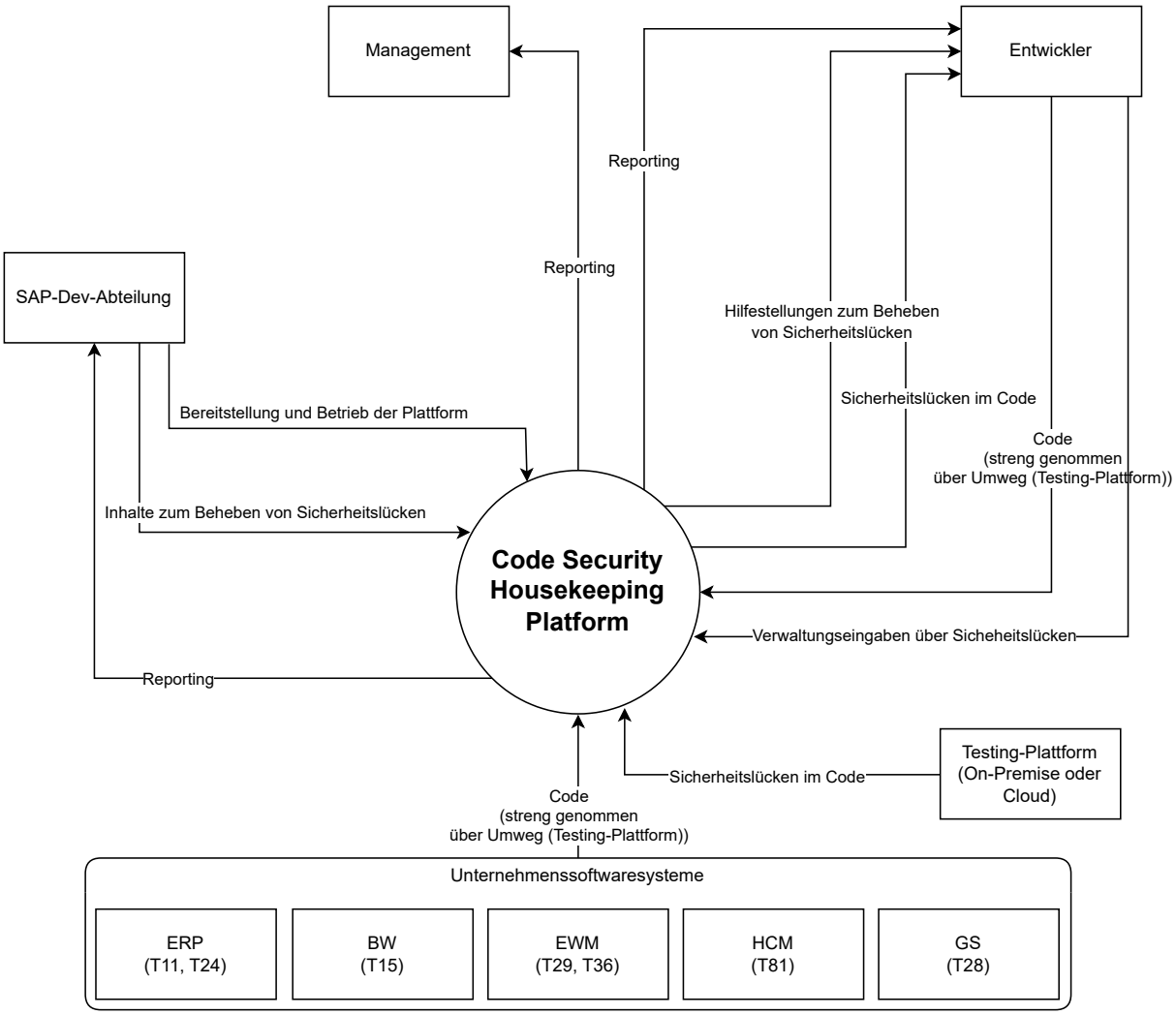


Abbildung 3.1: System-Kontext-Diagramm

3.4 Unternehmenssoftwaresystem-Landschaft

Die Geberit-Gruppe setzt mehrere On-Premise-Unternehmenssoftwaresysteme ein, die von Services in der Cloud unterstützt werden. Das wichtigste System ist ein SAP S/4HANA-System, welches als globales ERP-System dient. Zudem wird aktuell noch ein älteres SAP R/3-System als lokales ERP-System für manche Standorte eingesetzt, dessen Funktionalitäten aber in den nächsten Jahren auf das S/4HANA-System migriert werden sollen. Für das Bereitstellen von Mitarbeiter-Applikationen verwendet Geberit ein SAP Netweaver Gateway-System. Für die Lagerverwaltung wird gerade von einem SAP EWM-System auf ein S/4HANA EWM-System migriert, um auch hier die Technologie zu vereinheitlichen und das neueste System zu verwenden. Im Bereich Data Intelligence wird ein SAP BW 4/HANA-System als Data Warehouse betrieben. Für das Personalwesen wird ein SAP HCM for S/4HANA-System (H4SF) verwendet.

Zusätzlich zu den On-Premise-Systemen werden auch Cloud-Services von SAP genutzt. Diese werden zentral von der SAP BTP bereitgestellt und dort verwaltet. Als zentrale Datenbank-Lösung, beispielsweise für das Halten von Applikations-Daten, wird SAP HANA Cloud verwendet. Für das Reporting wird SAC genutzt. Zudem wird SAP DI eingesetzt, um Daten aus verschiedenen Quellen zu integrieren und transformieren.

4 Konzeption

Die Konzeption dient dazu, die Implementierung vorzubereiten und ist somit ein essenzieller Schritt im Entwicklungsvorgang. Es werden zunächst die Anforderungen an die Plattform erhoben und definiert. Anschließend wird dargelegt, was bereits im Voraus vom Unternehmen festgelegt wurde, um die Möglichkeiten an die Umsetzung weiter einzugrenzen. Um zu entscheiden, ob ein On-Premise- oder Cloud-Betrieb der Plattform bei dem Anwendungsfall besser geeignet ist, werden die beiden Lösungen mittels einer Nutzwertanalyse verglichen. In Bezug auf diese Vorarbeit wird ein Konzept präsentiert: eine Code-Sicherheits-Plattform, die dazu fähig ist, Fehler in Unternehmenssoftwaresystemen zu analysieren und bei deren Behebung zu unterstützen.

4.1 Anforderungen an die Plattform (Requirements Engineering)

Vor der Entwicklung der Plattform müssen zunächst die Anforderungen an das System festgelegt werden. Es müssen also die Wünsche und Bedürfnisse der Kunden aufgenommen und verarbeitet werden. Um dies durchzuführen wird in diesem Kapitel Requirements Engineering betrieben, wie es im Kapitel 2.3.1 definiert wurde. Bei dieser Bachelorarbeit steht nicht die Anforderungserhebung, sondern das Konzept und die Entwicklung der Plattform im Mittelpunkt. Daher wird in diesem Kapitel nicht jedes Detail granular beleuchtet, sondern lediglich ein Überblick über den Prozess gegeben.

4.1.1 Erhebung

Für die Erhebungen der Anforderungen werden Experteninterviews mit den Kunden geführt. Hierfür werden zunächst die Kunden ermittelt. Aus der Stakeholderbeschreibung in Kapitel 3.2 lässt sich ableiten, dass die Kunden die Entwickler, das Management und die SAP-Dev-Abteilung sind. Da selbstverständlich nicht alle Personen der Interessensgruppen interviewt werden können, werden Repräsentanten ausgewählt. Um eine gewisse Struktur in die Interviews zu bekommen, wird ein Leitfaden mit Fragen erstellt. Die Interviews werden persönlich, mündlich durchgeführt und nicht transkribiert, sondern informell mittels Notizen dokumentiert. Folgende System-Beschreibung resultiert nach der Zusammenfügung der Notizen:

Es wird ein System benötigt, das Eigenentwicklungen auf verschiedenen Unternehmenssoftwaresystemen analysiert. Die Analyse soll eine statische Code-Analyse und auf Sicherheitslücken ausgerichtet sein. Die Tests sollen automatisiert jeden Tag durchgeführt werden. Die Sicherheitslücken sollen an einem zentralen Ort gespeichert werden. Dabei soll erfasst werden, um was für eine Art Sicherheitslücke. es sich handelt, an welcher Stelle (System, Paket, Objekt und Zeile) der Fehler liegt, welcher Entwickler diesen Code zuletzt bearbeitet hat und welches „Gewicht“/welche „Priorität“ der Fehler hat. (Eine Art von Sicherheitslücke wäre beispielsweise „Potenzielle SQL-Injektion“.)

Für die Entwickler soll eine Benutzeroberfläche erstellt werden. Hier sollen die Fehler tabellarisch dargestellt werden. Die Felder sollen dabei sein:

1. Art der Sicherheitslücke
2. Status
3. Verantwortlicher Entwickler
4. Notiz
5. Paket
6. Objekt
7. Zuständiger Bereich
8. Link zum Sourcecode
9. Zuletzt bearbeitet am
10. Zuletzt bearbeitet von

Die Tabelle soll filterbar, sortierbar und durch einen Kopfdruck als Excel-Datei exportierbar sein. Außerdem sollen die Felder 2., 3., und 4. vom Benutzer modifizierbar sein.

Der Link zum Sourcecode soll den Entwicklern ermöglichen, direkt in eine Entwicklungsumgebung abspringen zu können, wo sie dann den Fehler beheben können.

Neben der tabellarischen Übersicht soll für jede Art von Sicherheitslücke eine Erklärungsseite erstellt werden. Diese soll geöffnet werden, wenn ein Nutzer auf die Art der Sicherheitslücke klickt. Die Erklärungsseite soll den Entwicklern behilflich sein, die Sicherheitslücke zu beheben. Die Entwickler geben dazu vor, welche Informationen sie für die Behebung brauchen:

Zunächst soll das Problem, also wieso der Code eine Sicherheitslücke darstellt und warum der Test angeschlagen ist, kurz textuell beschrieben werden. Dann soll ein Negativ-Code-Beispiel aufgeführt werden, zusammen mit einer Erklärung, wie die Sicherheitslücke an diesem Beispiel ausgenutzt werden kann. Zusätzlich soll ein Vorher-Nachher-Code-Beispiel mit Erklärung dargestellt werden, sodass die Entwickler im besten Fall keine anderen Quellen zur Behebung brauchen.

Zudem soll ein Reporting-Tool genutzt werden, das die Daten zu den Fehlern aufbereitet und in „nützlicher Weise“ darstellt, sodass dies von den Entwicklern und dem Management genutzt werden kann, ein Überblick über die Code-Sicherheits-Situation zu bekommen. Genauer soll eine Grafik darstellen, wie viele Fehler es über einen gewissen Zeitraum pro Bereich[2] gibt. Diese Darstellung wurde bereits in einer vorangegangenen Bachelorarbeit als nützlich und akzeptiert erachtet. Es soll allerdings einfach möglich sein, weitere Diagramme hinzuzufügen. Die Reporting-Oberfläche soll in die Benutzeroberfläche integriert sein.

Generell soll die Benutzerschnittstelle einfach von Mitarbeitern in Entwicklung und Management erreichbar sein. Alle anderen Zugriffe müssen verweigert werden, da die Plattform sicherheitskritische Informationen beinhaltet.

Diese Zusammenfassung der Interviews gilt als Grundlage für die weitere Verarbeitung, welche in den nächsten Kapiteln beschrieben wird.

4.1.2 Dokumentation

In der Dokumentationsphase werden die einzelnen Anforderungen, die aus den durchgeführten Interviews gewonnen wurden, systematisch extrahiert und in einer strukturierten Form, dem Spezifikationsentwurf, dargestellt.

Eine Priorisierung der Anforderungen wird ebenfalls durchgeführt. Da die Bachelorarbeit aber darauf abzieht, alle gesammelten Anforderungen umzusetzen und die Priorisierung daher eine weniger bedeutende Rolle spielt, wird auf eine aufwändige Methode wie die Nutzwertanalyse verzichtet. Stattdessen wird die Priorisierung in Absprache mit den Kunden festgelegt. Eine Einteilung in funktionale und nicht-funktionale Anforderungen

ist aufgrund der sehr wenigen Qualitätsanforderungen nicht nötig.

Der priorisierte Spezifikationsentwurf sieht wie folgt aus:

Spezifikationsentwurf Code-Sicherheits-Plattform

- **Analyse auf Eigenentwicklungen** (Prio: 1)
  - in mehreren verschiedenen Unternehmenssoftwaresystemen
  - statische Code-Analyse
  - auf Sicherheitslücken ausgerichtet
  - automatisierte Durchführung jeden Tag
  - Speicherung der Sicherheitslücken an zentralem Ort
    - Erfassung von:
      - Art der Sicherheitslücke
      - Stelle des Fehlers (System, Paket, Objekt, Zeile)
      - Ersteller des Codes
      - Gewicht/Priorität
- **Benutzeroberfläche für Entwickler**
  - 1. View: Sicherheitslücken tabellarisch dargestellt (Prio 2)
    - Felder:
      - Art der Sicherheitslücke
        - Absprung in Erklärungsseite
      - Status
      - Verantwortlicher Entwickler
      - Notiz
      - System
      - Paket
      - Objekt
      - Zuständiger Bereich
      - Link zum Sourcecode
        - Absprung in Entwicklungsumgebung
      - Zuletzt bearbeitet am
      - Zuletzt bearbeitet von
    - filterbar
    - sortierbar

        - durch Kopfdruck als Excel-Datei exportierbar
        - Status, Verantwortlicher Entwickler und Notiz vom Benutzer modifizierbar
    - 2. View: Erklärungsseite für jede Art von Sicherheitslücke (Prio 3)
        - durch Klicken auf Art in Tabelle öffnen
        - Inhalt:
            - Beschreibung des Problems (warum ist die Sicherheitslücke schlecht, warum schlägt der Test an)
            - Negativ-Code-Beispiel mit Erklärung, wie die Sicherheitslücke ausgenutzt werden kann
            - Vorher-Nachher-Code-Beispiel mit Erklärung
    - einfache Erreichbarkeit von Mitarbeitern in Entwicklung und Management
    - Zugriffsverweigerung für alle anderen, da Plattform sicherheitskritische Informationen beinhaltet
- **Reporting-Tool** (Prio 4)
    - Daten zu Fehlern aufbereiten und in nützlicher Weise darstellen
    - Grafik: Wie viele Fehler über gewissen Zeitraum pro Bereich
    - Möglichkeit, einfach weitere Diagramme hinzuzufügen
    - in Benutzeroberfläche integriert

4.1.3 Validierung

Für die Validierung des Entwurfs und somit der Anforderungs-Abnahme wird der Entwurf nochmals den Verantwortlichen der Kundengruppen vorgelegt und um Feedback gebeten. Der Spezifikationsentwurf wird hierbei von allen Kunden akzeptiert und es müssen keine Änderungen vorgenommen werden.

Zudem wurde der komplette Pozess, wie die Entwickler mit der Plattform interagieren, mit den Kunden durchgesprochen und abgenommen.

Als dritte Validierungsmaßnahme wird ein UI-Mockup (4.1) von der Benutzeroberfläche erstellt, welches den Kunden vorgelegt wird.

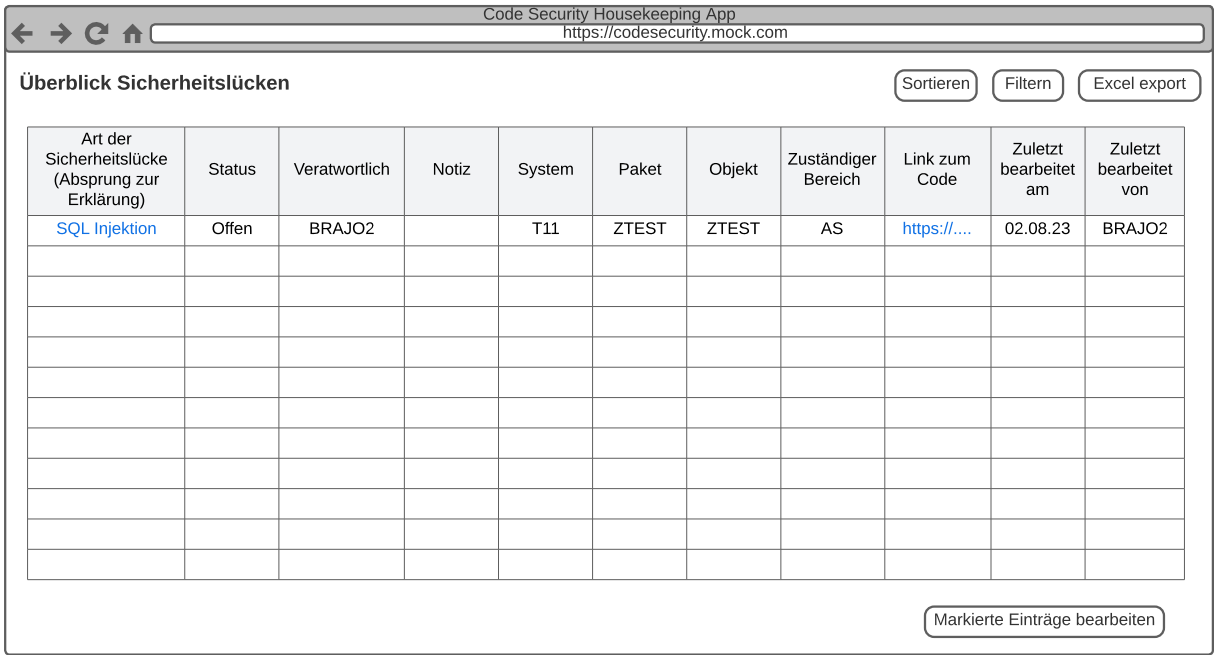


Abbildung 4.1: UI-Mockup der Benutzeroberfläche für Entwickler

Auch dieses wird von den Kunden akzeptiert und es müssen keine Änderungen vorgenommen werden.

Somit sind die Anforderungen an die Plattform definiert und können für das Systemkonzept genutzt werden.

4.2 Vorausgegangene Festlegungen

In diesem Kapitel werden die Festlegungen, die bereits im Voraus vom Unternehmen getroffen wurden, dargelegt. Diese Bachelorarbeit trägt nicht zur Entscheidung über diese Festlegungen bei, sondern muss diese berücksichtigen, was eine ausführliche Technologieentscheidung hinfällig macht. Im Groben geht es um die Entscheidung, welche Systeme bereits genutzt werden, welches Analyse-Produkt verwendet werden soll und welche Technologie für die Benutzeroberfläche benutzt werden soll.

Wie im Kapitel 3.4 beschrieben, werden bei Geberit verschiedene Unternehmenssoftwaresysteme genutzt. Diese sind alle von dem Softwarehersteller SAP. Damit sind auch die zu analysierenden Eigenentwicklungen in SAP-Systemen geschrieben und somit in der Programmiersprache ABAP.

Für die Analyse dieser Eigenentwicklungen soll das SAP-Produkt „Code Vulnerability Analyzer“ (CVA) genutzt werden. Dieses Produkt wurde von SAP entwickelt und ist speziell für die statische Code-Analyse von ABAP-Code ausgelegt.

Weitere Festlegungen betreffen Software-Komponenten, für die die Firma Geberit Standards festgelegt hat. Diese wurden in Kapitel 2.1.4 beschrieben. So soll, wenn Datenbanktabellen benötigt werden, die SAP HANA Datenbank genutzt werden. Für die geforderte Benutzeroberfläche soll die SAPUI5-Technologie verwendet und konzernweite Design-Richtlinien eingehalten werden. Dies soll sicherstellen, dass die Apps für die Benutzer einheitlich zu bedienen sind und ein einheitliches Design haben. Als Reporting-Lösung wird im Unternehmen SAC genutzt. Diese soll auch für die Reporting-Funktion der Code-Sicherheits-Plattform verwendet werden. Für Datenflüsse zwischen den Komponenten soll SAP DI Plattform genutzt werden.

4.3 Vergleich zwischen On-Premise- und Cloud-Betrieb (Nutzwertanalyse)

Im vorherigen Kapitel wird deutlich, das der CVA benutzt werden soll. Dieses SAP-Tool kann entweder kostenpflichtig auf einem On-Premise-System installiert werden oder auf einem Cloud-System betrieben werden. Es ist möglich von einem zentralen On-Premise-System aus, mehrere On-Premise-Systeme zu analysieren. Dies wäre der von SAP empfohlene Betrieb. Allerdings ist es über Umwege auch möglich, On-Premise-Systeme über eine Remote-Verbindung von dem CVA auf einem Cloud-System zu analysieren. Deshalb

sind beide Betriebsarten für die Umsetzungen der Anforderungen möglich und es muss entschieden werden, welche implementiert wird.

Für diese Entscheidung wird eine Nutzwertanalyse genutzt, wie sie in Kapitel 2.3.2 beschrieben wurde.

4.3.1 Beschreibung des Entscheidungsproblems und der Alternativen

Der erste Schritt einer Nutzwertanalyse ist die Beschreibung des Entscheidungsproblems. Bei diesem Anwendungsfall wird die Nutzwertanalyse für ein Auswahlproblem zwischen zwei Alternativen genutzt. Hierbei geht es, wie zuvor bereits erwähnt, um die Betriebsart des CVAs, welcher essenziell für die Umsetzung der Code-Sicherheits-Plattform ist.

Bei beiden Alternativen werden die Ergebnisse des CVAs genutzt, um die Code-Sicherheits-Plattform zu betreiben. Hierfür müssen allerdings noch weitere Entwicklungen erfolgen, welche auch von der Entscheidung abhängig sind, weshalb auch diese in die Nutzwertanalyse einfließen. Um die Alternativen genauer bewerten zu können, wird bereits vor der eigentlichen Konzeption eine grobe Architektur der Plattform mit beiden Alternativen erstellt.

Die **Alternativen** sind Folgende:

- **Installieren des CVAs auf einem zentralen On-Premise-System**
  Es wird ein neues On-Premise-SAP-Sytem aufgesetzt, auf dem der CVA installiert wird. Die zu analysierenden Unternehmenssoftwaresysteme werden über Remote Function Call (RFC)-Verbindungen mit dem zentralen System verbunden und können so analysiert werden. Die Checks können direkt über die ATC-Transaktion in der SAP-GUI automatisiert geplant werden. Die Ergebnisse werden in einer zentralen Datenbanktabelle auf dem zentralen Check-System gespeichert. Diese Lösung kostet einmalig etwa 5.000.000 € und pro Jahr 1.100.000 € Lizenzkosten. Aufwand für die Einrichtung der Checks und die Entwicklung der Code-Sicherheits-Plattform ist vergleichsweise gering, da die Ergebnisse direkt aus der Datenbanktabelle ausgelesen werden können und die Applikation darauf basieren kann. Eine mögliche Architektur ist in Abbildung 4.2 dargestellt.

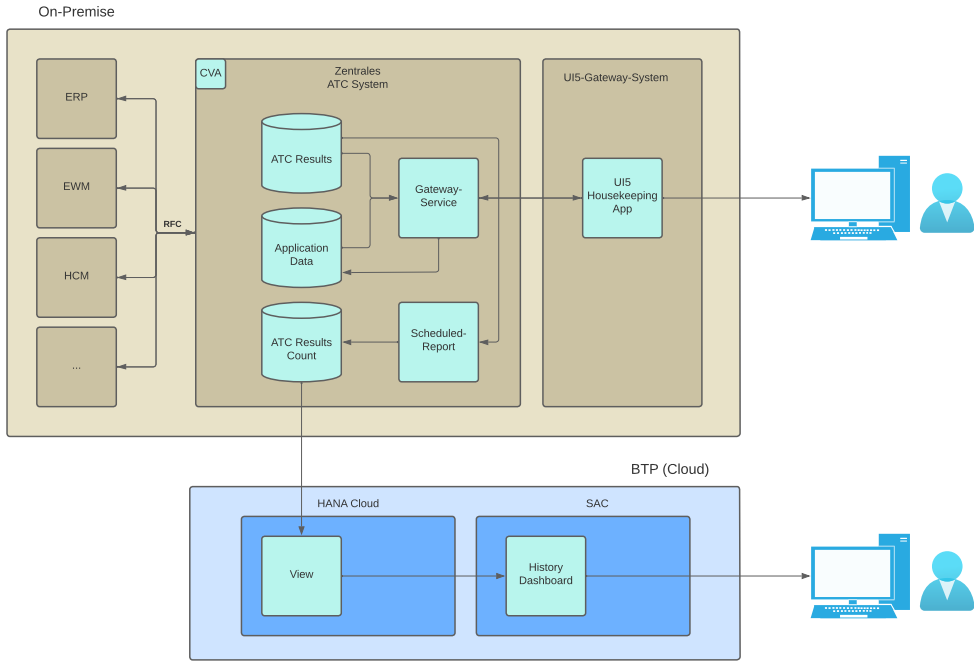


Abbildung 4.2: Mögliche Architektur der Code-Sicherheits-Plattform mit zentralem On-Premise-System

Auf die jeweiligen Aufwände und Komponenten wird in der Bewertung der Alternative genauer eingegangen.

- **Installieren des CVAs auf einem Cloud-System**
  Der CVA wird auf einem extra dafür aufgesetztem Cloud-System installiert. Die Analyse wird über Remote-Verbindungen von dem Cloud-System auf die zu analysierenden Unternehmenssoftwaresysteme durchgeführt. Diese Verbindung muss zunächst konfiguriert werden. Die Checks können über die Custom Code Migration App (CCMA) eingerichtet, allerdings nicht automatisiert werden. Die Ergebnisse werden in einer großen Datenbank-Tabelle gespeichert, welche dann für die Auswertung und Verwaltung genutzt werden kann. Diese Lösung kostet lediglich für den Betrieb des Cloud-Systems, da der CVA in der Cloud kostenlos ist. Es wird allerdings viel Aufwand benötigt, den ATC-Check zu konfigurieren, die Remote-Verbindung zu allen Systemen einzurichten und die Ergebnisse aus der internen Datenbank zu extrahieren und zu verwalten, da hier zwischen mehreren Systemen und Technologien Daten ausgetauscht werden müssen. Die höhere Komplexität lässt sich auch an der Architektur in Abbildung 4.3 erkennen.

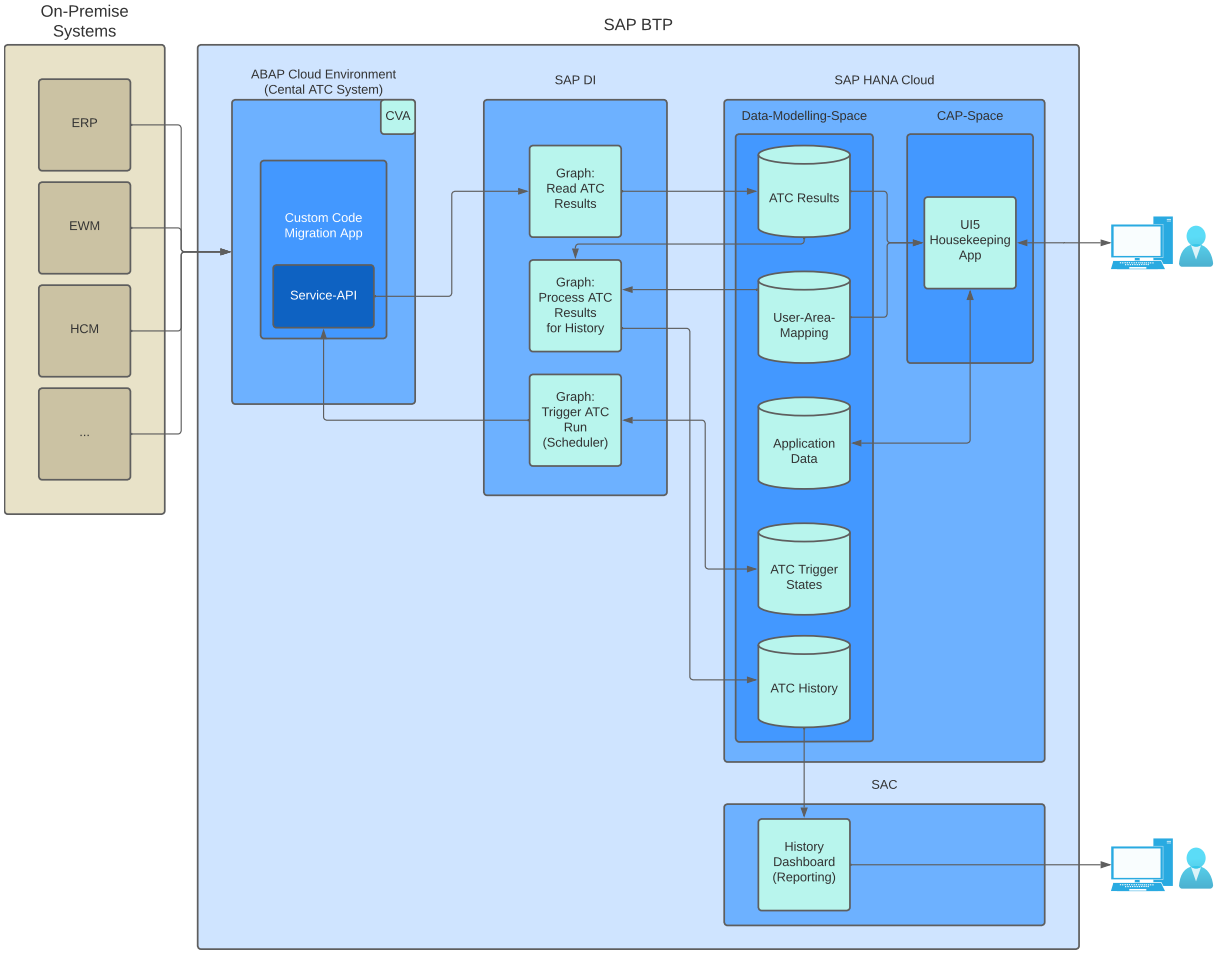


Abbildung 4.3: Mögliche Architektur der Code-Sicherheits-Plattform mit Cloud-System

4.3.2 Festlegung der Bewertungskriterien und Transformationsvorschriften

Im nächsten Schritt werden die Kriterien festgelegt, anhand derer die Alternativen bewertet werden sollen. Hierfür werden zunächst grundlegende Kriterien definiert, welche typisch für eine Nutzwertanalyse in der Softwareentwicklung sind (vgl. Herczeg 2018, S. 216), die anschließend in Unterkriterien aufgeteilt werden, um die Bewertung differenzierter und genauer durchführen zu können.

- Kosten und Aufwände
  - Kosten
    - **Anschaffungskosten**
    - **Fortlaufende Kosten**
  - Aufwände
    - **Aufwand für Einrichtung**
    - **Aufwand zu Weiterverarbeitung der Daten**
- **Funktionsumfang**
- **Performance**
- Wartung und Support
  - **Aktualität (Updates)**
  - **Wartungsaufwand bei Updates**
  - **Support von SAP**
- **Datensicherheit**
- **Komplexität**
- **Skalierbarkeit**

In den folgenden Abschnitten werden die Kriterien genauer erläutert und Transformationsvorschriften festgelegt. Bei dieser Nutzwertanalyse wird eine Skala von 0 bis 10 (10 = sehr gut, 0 = sehr schlecht) genutzt. Da nur zwei Alternativen bewertet werden, sind die Transformationsvorschriften auf diese Anzahl ausgelegt.

Anschaffungskosten (AC - Acquisition Costs) Die monetären Kosten für eine (Software-)Lösung, bestehen aus Anschaffungskosten (Fix-Kosten) und fortlaufende Gebühren. Da ein Unternehmen selbstverständlich gerne so wenig Kosten wie möglich hätte, sind alle anfallenden Kosten wichtige Entscheidungsfaktoren. Die Anschaffungskosten sind

einmalig und beinhalten alle Kosten, die anfallen, bis das System in einem betriebsbereiten Zustand ist. Hierzu zählen nur monetäre Kosten, die Aufwände für die Installation und Einrichtung werden erst in einem späteren Kriterium betrachtet.

Um das quantitative Kriterium der Anschaffungskosten zu bewerten, wird die folgende Transformationsvorschrift genutzt:

$$O_{1,\mathrm{AC}} = (1 - \frac{I_{2,\mathrm{AC}}}{I_{1,\mathrm{AC}} + I_{2,\mathrm{AC}}}) * 10 \tag{1}$$

$$O_{2,\mathrm{AC}} = (1 - \frac{I_{1,\mathrm{AC}}}{I_{1,\mathrm{AC}} + I_{2,\mathrm{AC}}}) * 10 \tag{2}$$

Die Kosten werden auf eine lineare Skala transformiert, die abhängig von den Kosten der Alternative ist. Durch diese Formel wird beachtet, dass sich die Höhe der Preisunterschiede zwischen den beiden Alternativen in dem Skalenwert widerspiegelt. Diese Transformationsvorschrift wird für viele der folgenden Kriterien genutzt, da sie gerade bei zwei Alternativen und quantitativen Kriterien sehr gut funktioniert.

Fortlaufende Kosten (OC - Ongoing Costs) Bei einer Softwarelösung fallen neben den Anschaffungskosten oft auch fortlaufende Kosten, für beispielsweise Lizenzen und den Betrieb an. Gerade bei Cloud-Produkten und bei Softwarelösungen mit häufigen Updates sind diese Kosten oft sehr hoch. Auch bei dieser Art von Kosten werden zunächst nur die monetären Faktoren betrachtet.

Für die fortlaufenden Kosten wird dieselbe Transformationsvorschrift, wie für die Anschaffungskosten genutzt. Hierbei beziehen sich die Kosten auf einen Zeitraum von einem Jahr.

$$O_{1,\mathrm{OC}} = (1 - \frac{I_{2,\mathrm{OC}}}{I_{1,\mathrm{OC}} + I_{2,\mathrm{OC}}}) * 10 \tag{3}$$

$$O_{2,\mathrm{OC}} = (1 - \frac{I_{1,\mathrm{OC}}}{I_{1,\mathrm{OC}} + I_{2,\mathrm{OC}}}) * 10 \tag{4}$$

Aufwand für Einrichtung (SE - Setup Effort) Der Einrichtungsaufwand setzt sich aus allen Aufwänden (Beschaffung, Einrichtung, Konfiguration, Programmierungen, etc.) zusammen, die anfallen, bis das System in einem betriebsbereiten Zustand ist. Der Auf-

wand wird in Personentagen gemessen, kann aber vor einem Projektstart nur geschätzt werden.

In diesem genauen Fall, geht es bei dem Aufwand darum, den CVA auf den Systemen zu installieren und einzurichten, sodass die Analyse für alle Systeme automatisiert durchgeführt werden kann und Ergebnisse sammelt.

Auch für Aufwände kann die Kosten-Transformationsvorschrift genutzt werden, da auch hier gilt, dass ein geringerer Aufwand besser ist.

$$O_{1,\mathrm{SE}} = (1 - \frac{I_{2,\mathrm{SE}}}{I_{1,\mathrm{SE}} + I_{2,\mathrm{SE}}}) * 10 \tag{5}$$

$$O_{2,\mathrm{SE}} = (1 - \frac{I_{1,\mathrm{SE}}}{I_{1,\mathrm{SE}} + I_{2,\mathrm{SE}}}) * 10 \tag{6}$$

Aufwand zu Weiterverarbeitung der Daten (FE - Further Effort) Da das Gesamtsystem noch aus zwei weiteren Anforderungskomponenten (Benutzeroberfläche und Reporting-Tool) besteht, müssen die Daten aus dem CVA weiterverarbeitet werden. Hierfür muss eine Schnittstelle zwischen dem Analysetool und den anderen Komponenten entwickelt werden, was einen gewissen Aufwand bedeutet. Dieser Aufwand wird ebenfalls in Personentagen gemessen und kann vor einem Projektstart nur geschätzt werden.

Auch für diesen Aufwand kann die bekannte, quantitative Transformationsvorschrift genutzt werden.

$$O_{1,\mathrm{FE}} = (1 - \frac{I_{2,\mathrm{FE}}}{I_{1,\mathrm{FE}} + I_{2,\mathrm{FE}}}) * 10 \tag{7}$$

$$O_{2,\mathrm{FE}} = (1 - \frac{I_{1,\mathrm{FE}}}{I_{1,\mathrm{FE}} + I_{2,\mathrm{FE}}}) * 10 \tag{8}$$

Funktionsumfang (RF - Range of Functions) Der Funktionsumfang beschreibt die Menge der bewältigbaren Aufgaben, die ein System erfüllen kann. Dieser ist, auch wenn sich die Systeme sehr ähneln, bei beiden Alternativen unterschiedlich und muss daher bewertet werden. Hierfür wird ein Punktesystem entwickelt: Um die Funktionen eines Systems zu bewerten, werden zunächst alle Funktionen aufgelistet und mit Punkten (0-10) bewertet. Diese Bewertung findet in Absprache mit den Kunden statt, ist allerdings nur eine grobe Einschätzung. Die Summe der Bewertungspunkte ergibt den Funktionsumfang

des Systems, welcher dann mit folgender Transformationsvorschrift auf eine Skala von 0 bis 10 transformiert wird.

$$O_{1,\mathrm{RF}} = \frac{I_{2,\mathrm{RF}}}{I_{1,\mathrm{RF}} + I_{2,\mathrm{RF}}} * 10 \tag{9}$$

$$O_{2,\mathrm{RF}} = \frac{I_{1,\mathrm{RF}}}{I_{1,\mathrm{RF}} + I_{2,\mathrm{RF}}} * 10 \tag{10}$$

Diese Formel ist ähnlich zu der Kosten-Formel, allerdings führt so eine hohe Punktzahl zu einem hohen Skalenwert.

Performance (P) Da sich die Systeme in ihrer Betriebsform unterscheiden, ist auch die Performance unterschiedlich, weshalb diese bewertet werden muss. Bei einem Code-Analysetool wie dem CVA spiegelt sich die Performance in der Dauer wider, die benötigt wird, um eine Analyse auf eine bestimmte Anzahl Objekten durchzuführen. Hierbei wird angenommen, dass die Objekte in ihrer Größe im Durchschnitt etwa gleich sind. Die Performance wird in Minuten gemessen und anschließend wieder mit der bekannten Transformationsvorschrift auf eine Skala von 0 bis 10 transformiert.

$$O_{1,\mathrm{P}} = (1 - \frac{I_{2,\mathrm{P}}}{I_{1,\mathrm{P}} + I_{2,\mathrm{P}}}) * 10 \tag{11}$$

$$O_{2,\mathrm{P}} = (1 - \frac{I_{1,\mathrm{P}}}{I_{1,\mathrm{P}} + I_{2,\mathrm{P}}}) * 10 \tag{12}$$

Aktualität (U - Updates) Die Aktualität eines Tools für statische Code-Analyse ist wichtig, da sich die Sicherheitslücken in der Softwarewelt ständig weiterentwickeln und somit auch die Analyse-Tools auf dem neusten Stand sein müssen. Die Aktualität hat somit direkt mit der Sicherheit, für die das System sorgen soll, zu tun. Um auf dem neusten Stand zu sein, müssen die Tools regelmäßig aktualisiert werden. Die Aktualität kann mittels der durchschnittlichen Dauer zwischen zwei Updates gemessen werden. Auch hier wird wieder die bekannte Transformationsvorschrift genutzt.

$$O_{1,\mathrm{U}} = (1 - \frac{I_{2,\mathrm{U}}}{I_{1,\mathrm{U}} + I_{2,\mathrm{U}}}) * 10 \tag{13}$$

$$O_{2,\mathrm{U}} = (1 - \frac{I_{1,\mathrm{U}}}{I_{1,\mathrm{U}} + I_{2,\mathrm{U}}}) * 10 \tag{14}$$

Wartungsaufwand bei Updates (UE - Update Effort) Um die Aktualität zu gewährleisten, müssen die Tools regelmäßig aktualisiert werden, was eventuell zu einem gewissen Aufwand führen kann. Dieser Aufwand kann in Personenstunden geschätzt und wieder mit der bekannten Transformationsvorschrift auf eine Skala von 0 bis 10 transformiert werden.

$$O_{1,\mathrm{UE}} = (1 - \frac{I_{2,\mathrm{UE}}}{I_{1,\mathrm{UE}} + I_{2,\mathrm{UE}}}) * 10 \tag{15}$$

$$O_{2,\mathrm{UE}} = (1 - \frac{I_{1,\mathrm{UE}}}{I_{1,\mathrm{UE}} + I_{2,\mathrm{UE}}}) * 10 \tag{16}$$

Support von SAP (MS - Manufacturer Support) Um die Systeme einzurichten und zu warten, ist es wichtig, dass es genügend Materialien und Support von dem Hersteller gibt. Hierfür wird der Support von SAP bewertet. Da dies ein qualitatives Kriterium ist, kann es nicht mit einer einfachen Transformationsvorschrift bewertet werden. Stattdessen werden zwei Bewertungsverfahren durchgeführt und anschließend gemittelt.

Das erste Verfahren ist ein Punktesystem für die Anzahl und Qualität der durch eine Internet-Recherche gefundenen Materialien. Hierfür wird folgende Tabelle genutzt:

Beschreibung	Punkte
Keine Materialien und Services verfügbar	+0 Punkte
Grundlegende Informationen verfügbar	+1 Punkt
Grobe Installationsanleitung verfügbar	+2 Punkt
Detaillierte Installationsanleitung verfügbar	+4 Punkte
Blog-Artikel zu verwandten Themen verfügbar	+2 Punkte
Ausführliche FAQs verfügbar	+3 Punkte

Tabelle 4.1: Support von SAP - Internet-Recherche

Die Punkte werden aufsummiert und ergeben somit einen Wert zwischen 0 und 10.

Das zweite Verfahren ist die Befragung von Experten. Hierzu wird die Frage generalisiert: „Wie schneidet die Support-Qualität von SAP für ein On-Premise-System im Vergleich zu einem Cloud-System ab?“ Es werden Mitarbeiter um eine Bewertung gebeten, die schon längere Zeit im SAP-On-Premise bzw. SAP-Cloud-Bereich arbeiten. Auch hier erfolgt die Bewertung von 0 bis 10.

Die beiden Bewertungen werden anschließend gemittelt, was den Skalenwert ergibt.

Datensicherheit (S - Security) Die Bewertung der Datensicherheit konzentriert in diesem Kontext sich auf die Sicherheit der Datenübertragung und -speicherung innerhalb des Systems. Die Sicherheit der Daten ist von Bedeutung, da sie potenzielle Sicherheitslücken beinhalten, deren Offenlegung von Angreifern ausgenutzt werden könnte, um gezielte Angriffe gegen das Unternehmen durchzuführen. Es geht bei dem Kriterium also um die Datensicherheit im System und nicht um die Sicherheit/der Genauigkeit der Checks, da diese bei den beiden Alternativen grundsätzlich gleich ist.

Es ist anzumerken, dass die Sicherheit nicht leicht durch quantitative Metriken erfasst werden kann und im Allgemeinen schwer vergleichbar ist. Aus diesem Grund wird hier lediglich die Anzahl der Schnittstellen betrachtet, die von den Alternativen (einschließlich der Weiterverarbeitung der Daten) benötigt werden, um eine grobe Einschätzung der Sicherheit zu ermöglichen. Die Anzahl der Schnittstellen allein liefert zwar keine umfassende Sicherheitsbewertung, kann jedoch als grobe Indikation dienen, da eine höhere Anzahl von Schnittstellen potenziell mehr Angriffspunkte bietet. Diese begrenzte Aussagekraft wird in der Gewichtung des Kriteriums berücksichtigt.

Die Anzahl der Schnittstellen wird anschließend mithilfe der bekannten Transformationsvorschrift auf eine Skala von 0 bis 10 übertragen.

$$O_{1,\mathrm{S}} = (1 - \frac{I_{2,\mathrm{S}}}{I_{1,\mathrm{S}} + I_{2,\mathrm{S}}}) * 10 \tag{17}$$

$$O_{2,\mathrm{S}} = (1 - \frac{I_{1,\mathrm{S}}}{I_{1,\mathrm{S}} + I_{2,\mathrm{S}}}) * 10 \tag{18}$$

Komplexität (C - Complexity) Die Komplexität eines Systems beeinflusst die Wartbarkeit und die Performance eines Systems und kann somit ebenfalls zur Bewertung der Alternativen genutzt werden. Das Kriterium wird in der Softwareentwicklung oft mit der Anzahl der Codezeilen, Komponenten oder Schnittstellen gemessen. Eine Bewertung an-

hand Code-Zeilen ist in diesem Fall nicht möglich, da die Systeme nicht in ihrem Quellcode vorliegen. Auch die Anzahl der Komponenten und Schnittstellen innerhalb der Systeme ist nicht bekannt. Allerdings kann bewertet werden, was für Komponenten und Schnittstellen gebraucht werden, um das System für die Code-Sicherheits-Plattform zu nutzen und genau diese Komplexität ist die ausschlaggebende, da hier der Aufwand für die Einrichtung und Wartung entsteht.

Die Bewertung erfolgt über eine Auflistung der zu entwickelnden Komponenten und Schnittstellen und einer Bewertung dessen Komplexität auf einer Skala von 1 bis 3 (1 = gering, 3 = hoch). Anschließend werden die Teilkomplexitäten aufsummiert und mit der bekannten Transformationsvorschrift auf eine Skala von 0 bis 10 transformiert. Auch bei der Komplexität kann vor dem Projektstart nur eine grobe Einschätzung vorgenommen werden, welche aber durch die Differenzierung in Teilkomplexitäten genauer wird.

$$I_{n,\mathrm{C}} = \sum_{i=1}^{m} \mathrm{Teilkomplexität}_{n,\mathrm{C},i} \tag{19}$$

$$O_{1,\mathrm{C}} = (1 - \frac{I_{2,\mathrm{C}}}{I_{1,\mathrm{C}} + I_{2,\mathrm{C}}}) * 10 \tag{20}$$

$$O_{2,\mathrm{C}} = (1 - \frac{I_{1,\mathrm{C}}}{I_{1,\mathrm{C}} + I_{2,\mathrm{C}}}) * 10 \tag{21}$$

Skalierbarkeit (SC - Scalability) Die Skalierbarkeit eines Systems ist wichtig, da sich die Anforderungen an das System im Laufe der Zeit ändern können und dieses dann erweitert werden muss. In diesem Fall geht es bei der Skalierbarkeit darum, welche Kosten und Aufwände benötigt werden, das System um weitere Unternehmenssoftwaresysteme zu erweitern.

Hierfür werden die Kosten und Aufwände jeweils beschrieben, bewertet und anschließend gemittelt. Dies geschieht, wie bei beispielsweise den Anschaffungskosten und dem Aufwand für die Einrichtung, mittels der bekannten Transformationsvorschrift.

$$O_{1,\mathrm{SC}} = (1 - \frac{I_{2,\mathrm{SC,Kosten}}}{I_{1,\mathrm{SC,Kosten}} + I_{2,\mathrm{SC,Kosten}}}) * 5 + (1 - \frac{I_{2,\mathrm{SCAufwand}}}{I_{1,\mathrm{SC,Aufwand}} + I_{2,\mathrm{SC,Aufwand}}}) * 5 \tag{22}$$

$$O_{2,\mathrm{SC}} = (1 - \frac{I_{1,\mathrm{SC,Kosten}}}{I_{1,\mathrm{SC,Kosten}} + I_{2,\mathrm{SC,Kosten}}}) * 5 + (1 - \frac{I_{1,\mathrm{SCAufwand}}}{I_{1,\mathrm{SC,Aufwand}} + I_{2,\mathrm{SC,Aufwand}}}) * 5 \tag{23}$$

4.3.3 Gewichtung der Kriterien

Nach der Festlegung der Kriterien müssen diese gewichtet werden. Um die Gewichtung strukturierter durchzuführen, werden zunächst die Ober-Kriterien (Kosten und Aufwände, Funktionsumfang, Performance, Wartung und Support, Sicherheit, Komplexität und Skalierbarkeit) gewichtet und anschließend weiter unterteilt. Die Gewichtung findet in einer Absprache mit der Abteilung statt, um ein möglichst reflektiertes Ergebnis zu erzielen.

Die fertige Gewichtung wird in der folgenden Tabelle (4.2) dargestellt:

(Ober-)Kriterium	**Gewichtung**
Kosten und Aufwände	51 %
Kosten	27 %
Anschaffungskosten (AC)	13 %
Fortlaufende Kosten (OC)	14 %
Aufwände	24 %
Aufwand für Einrichtung (SE)	12 %
Aufwand zu Weiterverarbeitung der Daten (FE)	12 %
Funktionsumfang (RF)	21 %
Performance (P)	3 %
Wartung und Support	11 %
Aktualität (Updates) (U)	6 %
Wartungsaufwand bei Updates (UE)	3 %
Support von SAP (MS)	2 %
Datensicherheit (S)	4 %
Komplexität (C)	5 %
Skalierbarkeit (SC)	5 %

Tabelle 4.2: Gewichtung der Kriterien

Die größte Gewichtung fällt auf die Kosten und Aufwände, da diese für ein wirtschaftlich arbeitendes Unternehmen sehr wichtig sind. Hierbei sind vor allem die fortlaufenden Kosten sehr relevant, da das System für eine lange Zeit genutzt werden soll. Die Gewichtung bei den Aufwänden wurde aufgeteilt, da es für das Gesamtsystem, also bis der CVA auch sinnvoll genutzt werden kann beide Aufwände bedarf. Der Funktionsumfang ist ebenfalls ein wichtiges Unterscheidungsmerkmal der beiden Alternativen und daher hoch gewichtet. Die Performance ist bei dieser Anwendung nicht wichtig, da die Anforderungen lediglich einen täglichen Durchlauf der Analyse vorsehen und hierfür beide Alternativen

ausreichend performant sind. Beim Wartung und Support ist die Aktualität der Updates am wichtigsten, da diese die Sicherheit des Systems gewährleisten. Wenn SAP durch neue Checks neue Sicherheitslücken erkennt, müssen diese schnellstmöglich in das System integriert werden. Die Datensicherheit wird nicht hoch gewichtet, da sie für das System schlecht zu beurteilen ist und durch das Nutzen von SAP-Tools als gegeben angenommen wird. Die Komplexität und Skalierbarkeit sind ebenfalls nur bedingt beurteilbar und werden daher nur gering gewichtet.

4.3.4 Bewertung der Alternativen

Da alle Kriterien, wie auch deren Transformationsvorschriften und Gewichtung festgelegt sind, kann die Bewertung der Alternativen durchgeführt werden.

Anschaffungskosten (AC - Acquisition Costs) Für On-Premise-Systeme wird der CVA mit einer einmaligen Lizenzgebühr pro fünf Benutzer verkauft. Als Benutzer zählt jeder Mitarbeiter, der entwickelt, die Ergebnisse der Analyse nutzt oder sie weiterverarbeitet. Pro fünf Benutzer kostet das Paket 250.000 €, hat allerdings eine Obergrenze von 5.000.000 €, welche bei der Firma Geberit erreicht wird[3]. Zu der CVA-Lizenz kommen noch Kosten für den Server und Lizenzgebühren für das System, auf dem der CVA installiert wird. Diese Kosten können allerdings vernachlässigt werden, da sie im Vergleich zu den 5.000.000 € sehr gering sind und für jedes Unternehmen unterschiedlich sind[4].

Die Anschaffungskosten für die Cloud-Lösung sind 0 €, da der CVA in der Cloud kostenlos ist und der Betrieb der Cloud-Instanz fortlaufend bezahlt wird.

Mit der Transformationsvorschrift für die Kosten resultieren folgende Skalenwerte:

$$O_{\text{On-Premise,AC}} = (1 - \frac{5.000.000}{5.000.000 + 0}) * 10 = 0 \tag{24}$$

$$O_{\text{Cloud,AC}} = (1 - \frac{0}{5.000.000 + 0}) * 10 = 10 \tag{25}$$

Fortlaufende Kosten (OC - Ongoing Costs) Auch bei den fortlaufenden Kosten des On-Premise-Betriebs ist die Lizenzgebühr für den CVA der größte Kostenfaktor. So

[3]Diese Werte stammen aus der SAP-Preistabelle, welche allerdings nicht öffentlich verfügbar ist, sondern als Unternehmen angefordert werden muss.

[4]In diesem Fall würde bei einer On-Premise-Lösung ein bereits existierendes System genutzt werden.

müssen für den Betrieb (Support und Wartung) des CVAs pro Jahr 22 % der Lizenzgebühr bezahlt werden. Dies ergibt bei der maximalen Lizenzgebühr von 5.000.000 € jährliche Kosten von 1.100.000 €. Auch hier kommen noch Betriebskosten für den Server hinzu, die vernachlässigt werden.

In der Cloud fallen ebenfalls Kosten für den Betrieb an. Zwar ist der CVA kostenlos, allerdings muss das ABAP Cloud Environment bezahlt werden. Die Kosten hierfür werden je nach Nutzung berechnet und werden von den verantwortlichen Mitarbeitern auf maximal 50.000 € pro Jahr geschätzt.

$$O_{\text{On-Premise,OC}} = (1 - \frac{1.100.000}{1.100.000 + 50.000}) * 10 \approx 0,43 \tag{26}$$

$$O_{\text{Cloud,OC}} = (1 - \frac{50.000}{1.100.000 + 50.000}) * 10 \approx 9,57 \tag{27}$$

Aufwand für Einrichtung (SE - Setup Effort) Der Aufwand für die Einrichtung des CVAs setzt sich bei der On-Premise-Lösung aus dem Aufsetzen des Systems, der Installation des CVAs und der Konfiguration der Checks und Verbindungen zu den Systemen zusammen. Das Aufsetzen des Systems wird auf drei Personentage geschätzt. Die Installation und Konfiguration des CVAs ist auf einem On-Premise-System nicht aufwändig, da lediglich ein Report ausgeführt werden muss. Zusammen mit der Konfiguration der Verbindungen zu den Systemen, welche hingegen sehr zeitintensiv sein kann, wird der Aufwand auf drei Personentage geschätzt. Somit ergibt sich ein Aufwand von sechs Personentagen.

Bei der Cloud-Lösung muss zunächst das SAP BTP aufgesetzt werden, wenn dies noch nicht geschehen ist. Außerdem muss ein ABAP Cloud Environment aufgesetzt werden, auf dem der CVA installiert wird.

Da die Cloud-Lösung keine Automatisierung unterstützt, muss dies zusätzlich entwickelt werden, was mit einem Aufwand von sechs Personentagen geschätzt wird.

Zusammen mit der umständlichen Konfiguration des CVAs und der Verbindungen zu den Systemen, wird auf 18 Personentage geschätzt

Das Einrichten des CVAs ist bei der On-Premise-Lösung also deutlich weniger aufwändig, was sich auch in der Bewertung widerspiegelt.

$$O_{\text{On-Premise,SE}} = (1 - \frac{6}{18+6}) * 10 = 7,5 \tag{28}$$

$$O_{\text{Cloud,SE}} = (1 - \frac{18}{18+6}) * 10 = 2,5 \tag{29}$$

Aufwand zu Weiterverarbeitung der Daten (FE - Further Effort) Ähnlich wie bei der Einrichtung des CVAs ist auch der Aufwand für die Weiterverarbeitung der Daten bei der On-Premise-Lösung deutlich geringer, wie in dem Konzept in Abbildung 4.2 zu sehen ist. Das Backend für die SAPUI5 App kann direkt auf dem System entwickelt werden, womit die Daten direkt aus der ATC-Datenbank-Tabelle gelesen werden können. Durch das Verbinden des On-Premise-Systems mit dem SAP BTP Cockpit, kann hier eine automatisch generierte Schnittstellen-View genutzt werden, um die Daten an das Reporting-Tool zu übertragen. Insgesamt wird der Aufwand mit dem Entwickeln der Benutzeroberfläche auf 10 Personentage geschätzt.

Bei der Cloud-Lösung kann nicht direkt auf die Tabelle des CVAs zugegriffen werden, weshalb eine Schnittstelle entwickelt werden muss, die die Daten aus der Tabelle ausliest und in eine Custom-Tabelle schreibt. Da hierfür ein neues System (HANA Cloud) und die Schnittstelle über das SAP DI benötigt wird, wird der Aufwand für die Benutzeroberfläche auf 24 Personentage geschätzt.

$$O_{\text{On-Premise,FE}} = (1 - \frac{10}{10+24}) * 10 \approx 7,06 \tag{30}$$

$$O_{\text{Cloud,FE}} = (1 - \frac{24}{10+24}) * 10 \approx 2,94 \tag{31}$$

Funktionsumfang (RF - Range of Functions) Der Funktionsumfang lässt sich mit der Tabelle 4.3 beschreiben und bewerten:

Funktion	**Bewertung**
Grundfunktionen (erfüllen beide Systeme)	
Statische Codeanalyse auf Sicherheitslücken durch Datenflussanalyse	10
Analyse von mehreren Systemen per RFC Verbindungen	8
Funktionen, die nur auf einem On-Premise-System verfügbar sind	
Automatisierung der Test-Ausführungen (Scheduler)	7
Ausführen der Analyse direkt in Entwicklungsumgebung Eclipse (Developer-Szenario)	6
Integration in Solution Manager	4
Ausnahme-Anfragen	3
Baseline Konzept (Unterdrücken von Fehlerquellen in Legacy Code)	3
ATC Oberfläche in SAP GUI	5
Funktionen, die nur auf einem Cloud-System verfügbar sind	
CCMA (Web-Schnittstelle zu ATC-Checks)	5

Tabelle 4.3: Vergleich der Funktionalitäten

Somit ergibt sich für die On-Premise-Möglichkeit ein Funktionsumfang von 41 und für die Cloud-Möglichkeit ein Funktionsumfang von 23, was mittels der vorgesehenen Transformationsvorschrift folgende Skalenwerte ergibt:

$$O_{\text{On-Premise,RF}} = \frac{41}{23+41} * 10 \approx 6,41 \tag{32}$$

$$O_{\text{Cloud,RF}} = \frac{23}{23+41} * 10 \approx 3,59 \tag{33}$$

Performance (P) Die Performance der beiden Alternativen lässt sich anhand der Laufzeit der Analyse auf eine gleiche Anzahl an Zeilen Code bewerten. Selbstverständlich ist die Analysezeit bei einer On-Premise-Lösung von der Hardware abhängig, weshalb der Test mit einer Standardkonfiguration durchgeführt wird. Die Analysezeit für die On-Premise-Lösung beträgt 2,5 Stunden und für die Cloud-Lösung 5,5 Stunden.

$$O_{\text{On-Premise,P}} = (1 - \frac{2,5}{2,5 + 5,5}) * 10 \approx 6,88 \tag{34}$$

$$O_{\text{Cloud,P}} = (1 - \frac{5,5}{2,5 + 5,5}) * 10 \approx 3,13 \tag{35}$$

Aktualität (U - Updates) Über den zeitlichen Abstand zwischen den Updates lässt sich eine Aussage über die Aktualität der Systeme treffen. Es wird angenommen, dass für die On-Premise-Lösung das System „S4HANA 2023“, welches das neueste Produkt ist, genutzt wird. Dieses hat einen Release-Zyklus von 2 Jahren und alle 6 Monate wird ein Feature Pack veröffentlicht, welcher neue Funktionen (und somit eventuell auch neue Checks) und Fehlerbehebungen enthält. Somit ist die Dauer zwischen zwei Updates bei der On-Premise-Lösung 6 Monate.

Bei der Cloud-Lösung wird der CVA in einem ABAP Cloud Environment ausgeführt. Dieses hat einen Release-Zyklus von 3 Monaten, wodurch die Dauer zwischen zwei Updates 3 Monate beträgt.

$$O_{\text{On-Premise,U}} = (1 - \frac{6}{6 + 3}) * 10 \approx 3,33 \tag{36}$$

$$O_{\text{Cloud,U}} = (1 - \frac{3}{6 + 3}) * 10 \approx 6,67 \tag{37}$$

Wartungsaufwand bei Updates (UE - Update Effort) Der Wartungsaufwand bei Updates liegt laut den Verantwortlichen bei rund 40 Personenstunden für die On-Premise-Lösung, da hier das Update manuell durchgeführt werden muss und anschließend auf Fehler geprüft werden muss.

Bei einem ABAP-Cloud-System werden die Updates automatisch eingespielt, wodurch kein Aufwand entsteht. Allerdings muss die Funktionalität des Gesamtsystems nach dem Update überprüft werden, wodurch ein Aufwand von 5 Personenstunden entsteht.

$$O_{\text{On-Premise,UE}} = (1 - \frac{40}{40 + 5}) * 10 \approx 1,11 \tag{38}$$

$$O_{\text{Cloud,UE}} = (1 - \frac{5}{40 + 5}) * 10 \approx 8,89 \tag{39}$$

Support von SAP (MS - Manufacturer Support) Bei einer Internet-Recherche wurden folgende Materialien gefunden:

Beschreibung	Punkte	On-Premise	Cloud
Keine Materialien und Services verfügbar	+0 P.		
Grundlegende Informationen verfügbar	+1 P.	✓	✓
Grobe Installationsanleitung verfügbar	+2 P.		✓
Detaillierte Installationsanleitung verfügbar	+4 P.	✓	
Blog-Artikel zu verwandten Themen verfügbar	+2 P.	✓	✓
Ausführliche FAQs verfügbar	+3 P.	✓	✓
	Summe:	10	8

Tabelle 4.4: Support von SAP - Ergebnisse Internet-Recherche

Bei der Befragung von Mitarbeitern die im Bereich SAP-On-Premise bzw. SAP-Cloud-Bereich arbeiten, wird folgendes Ergebnis erzielt: Für On-Premise-Support werden 8 und für den Cloud-Support 9 Punkte vergeben.

Auf die Skala projiziert ergeben sich folgende Mittelwerte:

$$O_{\text{On-Premise,MS}} = \frac{10+8}{2} = 9 \tag{40}$$

$$O_{\text{Cloud,MS}} = \frac{8+9}{2} = 8,5 \tag{41}$$

Datensicherheit (S) Die Datensicherheit wird mit der Anzahl an Schnittstellen zwischen Systeme gemessen, die die Alternativen benötigen.

Das On-Premise-Konzept beinhaltet 3 Schnittstellen:

- Zwischen dem Zentralen ATC-System und dem SAPUI5-Gateway-System
- Zwischen dem Zentralen ATC-System und der HANA Cloud (im SAP BTP)
- Zwischen der HANA Cloud und dem SAC

Das Cloud-Konzept beinhaltet ebenfalls 3 Schnittstellen:

- Zwischen dem ABAP Cloud Environment und dem SAP DI
- Zwischen dem SAP DI und der HANA Cloud

- Zwischen der HANA Cloud und dem SAC

Somit ergeben sich folgende Skalenwerte:

$$O_{\text{On-Premise,S}} = (1 - \frac{3}{3+3}) * 10 = 5 \tag{42}$$

$$O_{\text{Cloud,S}} = (1 - \frac{3}{3+3}) * 10 = 5 \tag{43}$$

Komplexität (C) Um die Komplexität zu messen, werden die zu entwickelnden Komponenten und Schnittstellen aufgelistet und bewertet. Das Ergebnis ist in Tabelle 4.5 zu sehen.

(Unter-)Komponente	Geschätzte Komplexität
On-Premise	
Gateway-Service (Backend)	2
Scheduled-Report	2
UI5 Housekeeping App (Frontend)	1
Tabelle: ATC Results	1
Tabelle: ATC History	1
Tabelle: Application Data	1
ATC History View	1
SAC History Dashboard	1
Schnittstelle: Gateway-Service - SAPUI5 App	1
Summe:	11
Cloud	
DI-Graph: Read ATC Results	2
DI-Graph: Process ATC Results for History	3
DI-Graph: Trigger ATC Run (Scheduler)	3
CAP - UI5 Housekeeping App (Backend + Fronend)	3
Tabelle: ATC Results	1
Tabelle: User-Area-Mapping	1
Tabelle: Application Data	1
Tabelle: ATC Trigger States	1
Tabelle: ATC History	1
SAC History Dashboard	1
Schnittstelle: ABAP Cloud Env - SAP DI	3
Summe:	20

Tabelle 4.5: Komplexität der Alternativen

Durch dieses Ergebnis resultieren folgende Skalenwerte:

$$O_{\text{On-Premise,C}} = (1 - \frac{11}{11+20}) * 10 = 6,45 \tag{44}$$

$$O_{\text{Cloud,C}} = (1 - \frac{20}{11+20}) * 10 = 3,55 \tag{45}$$

Skalierbarkeit (SC - Scalability) Um ein neues Unternehmenssoftwaresystem an ein zentrales On-Premise-ATC-System anzubinden, muss das System für die RFC-Verbindung

freigegeben werden und diese im ATC konfiguriert werden. Dieser Aufwand wird auf 4 Personenstunden geschätzt.

Bei der Cloud-Lösung muss ebenfalls die RFC-Verbindung konfiguriert werden. Allerdings muss auch ein neues Custom-Code-Migration-Projekt erstellt werden und der eigens entwickelte Report Graph im SAP DI geändert angepasst werden. Der Aufwand hierfür wird auf 6 Personenstunden geschätzt.

Bei beiden Alternativen fallen keine Kosten an, wodurch sich folgende Skalenwerte ergeben:

$$O_{\text{On-Premise,SC}} = (1 - \frac{4}{4+6}) * 10 = 6 \tag{46}$$

$$O_{\text{Cloud,SC}} = (1 - \frac{6}{4+6}) * 10 = 4 \tag{47}$$

4.3.5 Ergebnis

Kriterium	Gewichtung	**On-Premise**		**Cloud**	
		Skalenwert	Teilnutzwert	Skalenwert	Teilnutzwert
AC	13%	0,00	0,00	10,00	1,30
OC	14%	0,43	0,06	9,57	1,34
SE	12%	7,50	0,90	2,50	0,30
FE	12%	7,06	0,85	2,94	0,35
RF	21%	6,41	1,35	3,59	0,75
P	3%	6,88	0,21	3,13	0,09
U	6%	3,33	0,20	6,67	0,40
UE	3%	1,11	0,03	8,89	0,27
MS	2%	9,00	0,18	8,50	0,17
S	4%	5,00	0,20	5,00	0,20
C	5%	6,45	0,32	3,55	0,18
SC	5%	6,00	0,30	4,00	0,20
Summe	100%		**4,60**		**5,55**

Tabelle 4.6: Ergebnis Nutzwertanalyse

Das Ergebnis der Nutzwertanalyse in Tabelle 4.6 zeigt, dass der Betrieb des CVAs in der Cloud mit einem Nutzwert von 5,55 die bessere Alternative ist. Zusammengefasst benötigt diese Alternative zwar einen höheren Aufwand, ist allerdings wesentlich kostengünstiger als die Standard-On-Premise-Lösung von SAP.

4.4 Konzept zur Systemarchitektur

In der Nutzwertanalyse wurde der Cloud-Betrieb als die bessere Alternative bewertet. Daher wird im Folgenden ein Konzept für die Systemarchitektur der Cloud-Lösung vorgestellt, basierend auf den Anforderungen und den vorausgegangenen Festlegungen.

Die Konzeption der Architektur geschieht in drei Schritten, bei denen jeweils mehrere Komponenten zum System hinzugefügt und damit mehr Anforderungen erfüllt werden. Die einzelnen Schritte werden im Folgenden beschrieben und in den Abbildungen 4.4, 4.5 und 4.7 dargestellt. Schritt 3 ist dabei das fertige Konzept, welches schon bei der Beschreibung der Alternativen in der Nutzwertanalyse kurz definiert wurde und alle Anforderungen abdeckt.

Der erste Schritt des Konzepts erfüllt die grundlegenden Anforderungen, einen CVA zu nutzen, um Systeme auf Sicherheitslücken zu prüfen und dessen Ergebnisse darzustellen.

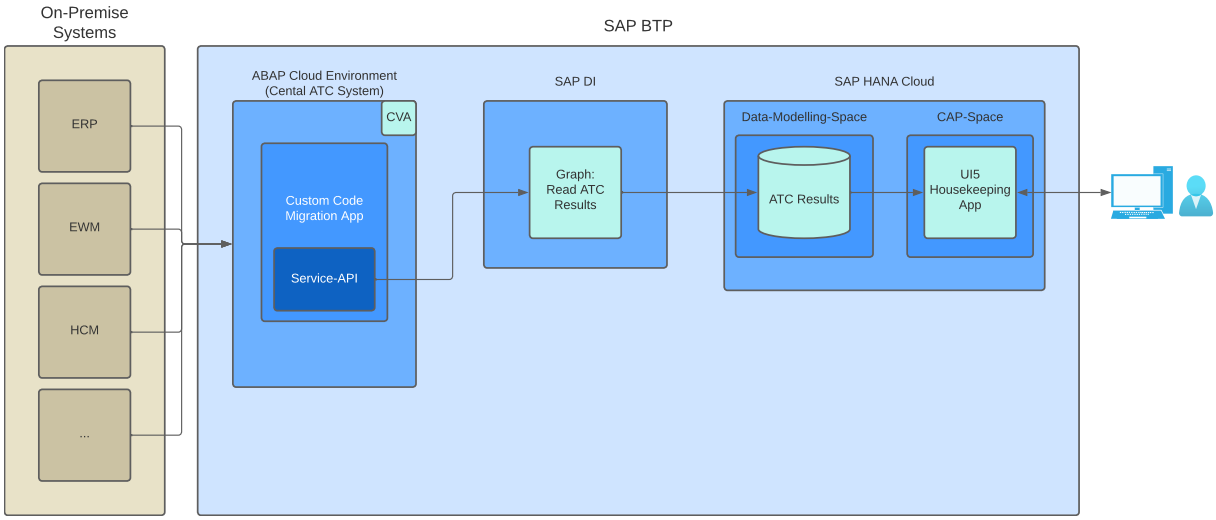


Abbildung 4.4: Konzept Schritt 1 (informelles Komponentendiagramm)

Hierfür befindet sich ein ABAP Cloud Environment innerhalb des SAP BTP, auf dem der CVA installiert ist. Das ABAP Cloud Environment hat im Standard die SAP-App „Custom Code Migration“ installiert. Diese ist eigentlich dafür da, ATC-Checks auszuführen, um Eigenentwicklungen für neue Versionen von SAP-Produkten vorzubereiten. Allerdings kann diese auch genutzt werden, um eigens definierte Check-Varianten auszuführen, wodurch auch der CVA genutzt werden kann. Die CCMA kann bereits als Benutzerschnittstelle genutzt werden, um die Ergebnisse der Analyse zu betrachten. Allerdings werden durch diese nicht annähernd alle Anforderungen abgedeckt, weshalb eine eigene Plattform entwickelt werden muss. Hierfür wird aber die Service-API der CCMA

genutzt, um mit dem ATC zu interagieren. Das ABAP Cloud Environment ist somit die erste Komponente im Datenfluss.

Die zweite Komponente, die es zu entwickeln gilt, ist ein Graph[5] im SAP DI, welches ebenfalls als Service im SAP BTP verfügbar ist. Dieser Graph soll die Service-API der CCMA nutzen, um die Ergebnisse der Analyse auszulesen. Hierfür muss er sich zunächst mit dem ABAP Cloud Environment verbinden, was zum Stand dieser Bachelorarbeit noch nicht nativ, ohne eine Eigenentwicklung („einem Umweg“) möglich ist. Diese Verbindung ist ein schwieriger Teil des Konzepts und wird daher in der Implementierung (5) genauer beschrieben.

Nach dem Auslesen und Verarbeiten der Ergebnis-Daten, werden diese in eine Datenbanktabelle in der HANA Cloud geschrieben.

Die Datenbank kann dann von einer CAP-Anwendung konsumiert werden, welche die Daten in einer SAPUI5-App darstellt.

Mit diesem Stand des Konzepts, ist es möglich den CVA durchzuführen und die Ergebnisse einem Endnutzer darzustellen. Allerdings fehlen essenzielle Anforderungen, wie die Automatisierung der Analyse und das Bearbeiten von Ergebnissen durch den Nutzer, weshalb der zweite Schritt des Konzepts folgt.

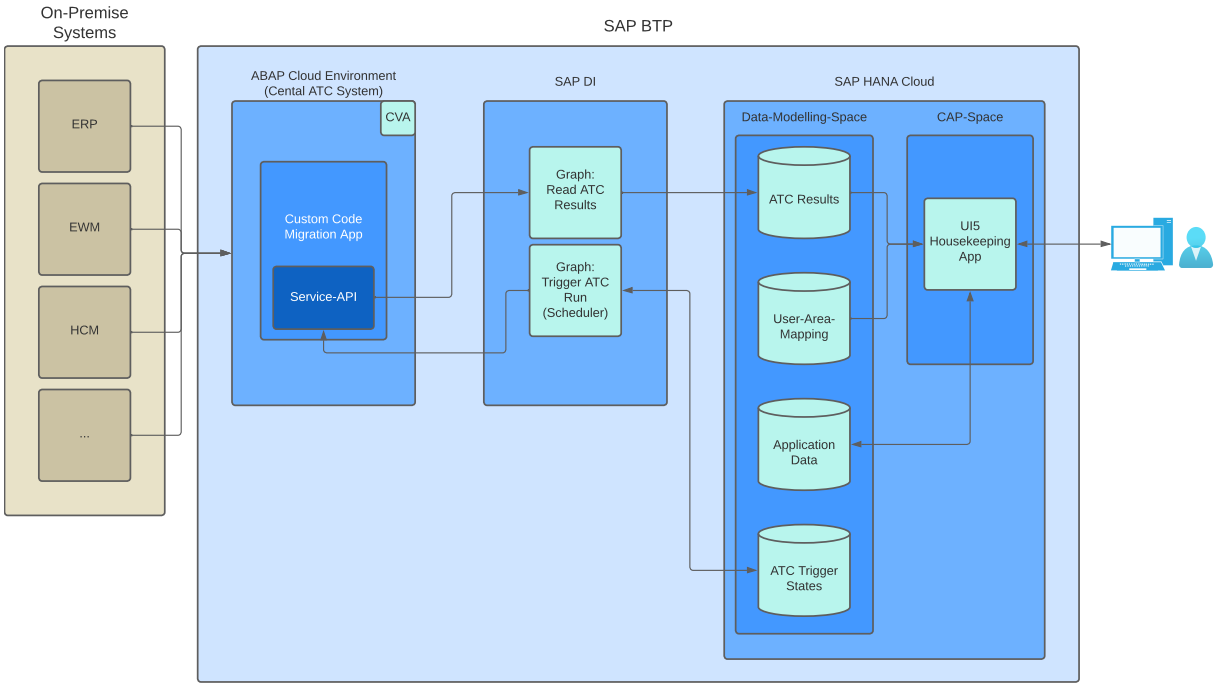


Abbildung 4.5: Konzept Schritt 2 (informelles Komponentendiagramm)

[5]Ein Graph ist eine Art Pipeline zur Datenverarbeitung, die aus mehreren Standart- oder Custom-Operatoren besteht.

Um die Automatisierung der Analyse zu ermöglichen, muss ein eigener Scheduler[6] entwickelt und ins Konzept integriert werden. Hierfür wird ein weiterer Graph im SAP DI entwickelt, welcher sich wieder über die Service-API mit dem ABAP Cloud Environment verbindet und die Analysen periodisch ausführt. Um dies für alle Systeme zu koordinieren, soll noch eine Datenbank-Tabelle in der HANA Cloud erstellt werden, in der die Systeme und deren Analyse-Zeitpunkte gespeichert werden.

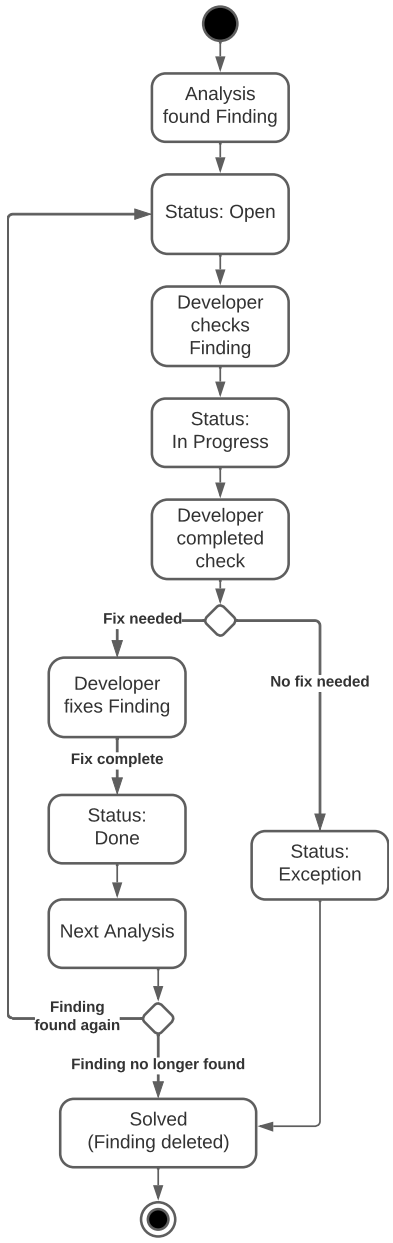


Abbildung 4.6: Abbildung des Lebenszyklus eines Eintrags mittels des Status-Felds

Um die Ergebnisse der Analyse bearbeiten zu können, soll eine „Application Data“-Tabelle erstellt werden, die die Änderungen der Einträge hält. Dies kann nicht direkt auf der Ergebnis-Tabelle geschehen, da diese bei jeder neuen Analyse überschrieben wird. Zudem ist gefordert, dass der verantwortliche Bereich für die Sicherheitslücken ersichtlich ist. Da aus dem CVA aber nur der verantwortliche Entwickler hervorgeht, muss dieser mit dem verantwortlichen Bereich verknüpft werden, wofür eine weitere Datenbanktabelle benötigt wird. Diese wird initial befüllt, muss anschließend bei Personaländerungen aber manuell gepflegt werden.

Mit dem Feld „Status“ soll der Lebenszyklus potenziellen Sicherheitslücke abgebildet werden. Es soll die Werte „Open“, „In Progress“, „Done“ und „Exception“ annehmen können. Der genaue Ablauf des Lebenszyklus wird mittels eines Zustandsdiagramms in Abbildung 4.6 dargestellt.

Mit diesem Stand des Konzepts, ist es möglich die Analyse zu automatisieren und die Ergebnisse zu bearbeiten.

Der letzte Schritt des Konzepts bezieht sich auf das Reporting der Ergebnisse.

[6]deutsch: Zeitplaner, Steuerer

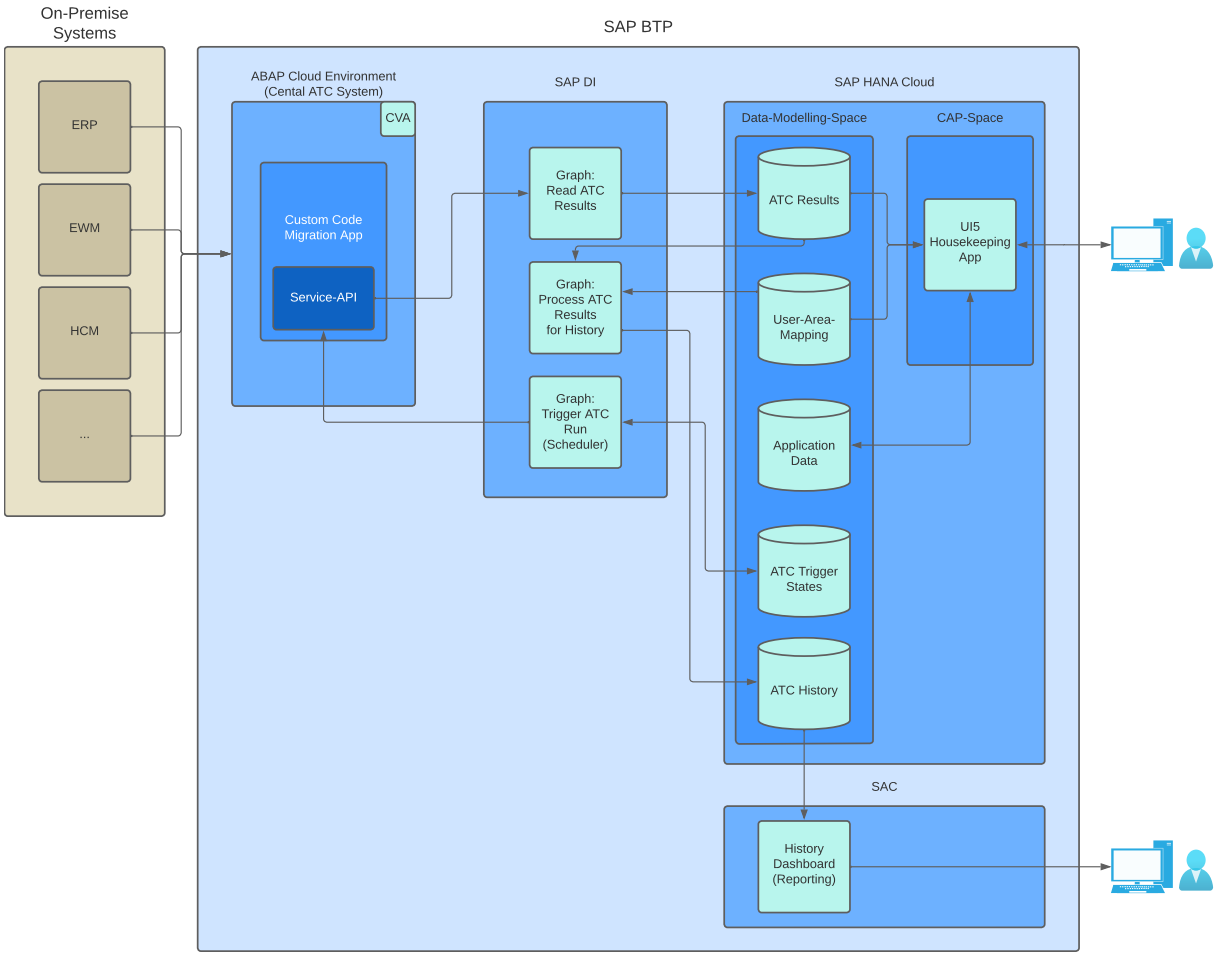


Abbildung 4.7: Konzept Schritt 3 (informelles Komponentendiagramm)

Hierfür muss eine Historie der Summe der Sicherheitslücken pro Bereich erstellt werden.

Dafür soll ein dritter SAP DI Graph entwickelt werden, der periodisch die Daten aus der Ergebnis-Tabelle ausliest. Diese werden mit den Einträgen der Entwickler-Bereichs-Tabelle und den Benutzereingaben (Application Data) verknüpft, sodass auch ein Verantwortlichen- bzw. Bereichswechsel durch einen Benutzer berücksichtigt wird. Anschließend sollen die aufsummierten Ergebnisse in eine weitere Datenbanktabelle der HANA Cloud geschrieben werden. Diese kann dann einfach von einem SAC-Dashboard konsumiert werden, welches die Ergebnisse in der geforderten Historien-Ansicht darstellt.

Mit diesem finalen Stand des Konzepts sind alle Anforderungen abgedeckt.

Da der Datenfluss angesichts der vielen verschiedenen Komponenten relativ komplex ist, wird dieser in Abbildung 4.8 mittels eines Datenflussdiagramms noch einmal detailliert dargestellt.

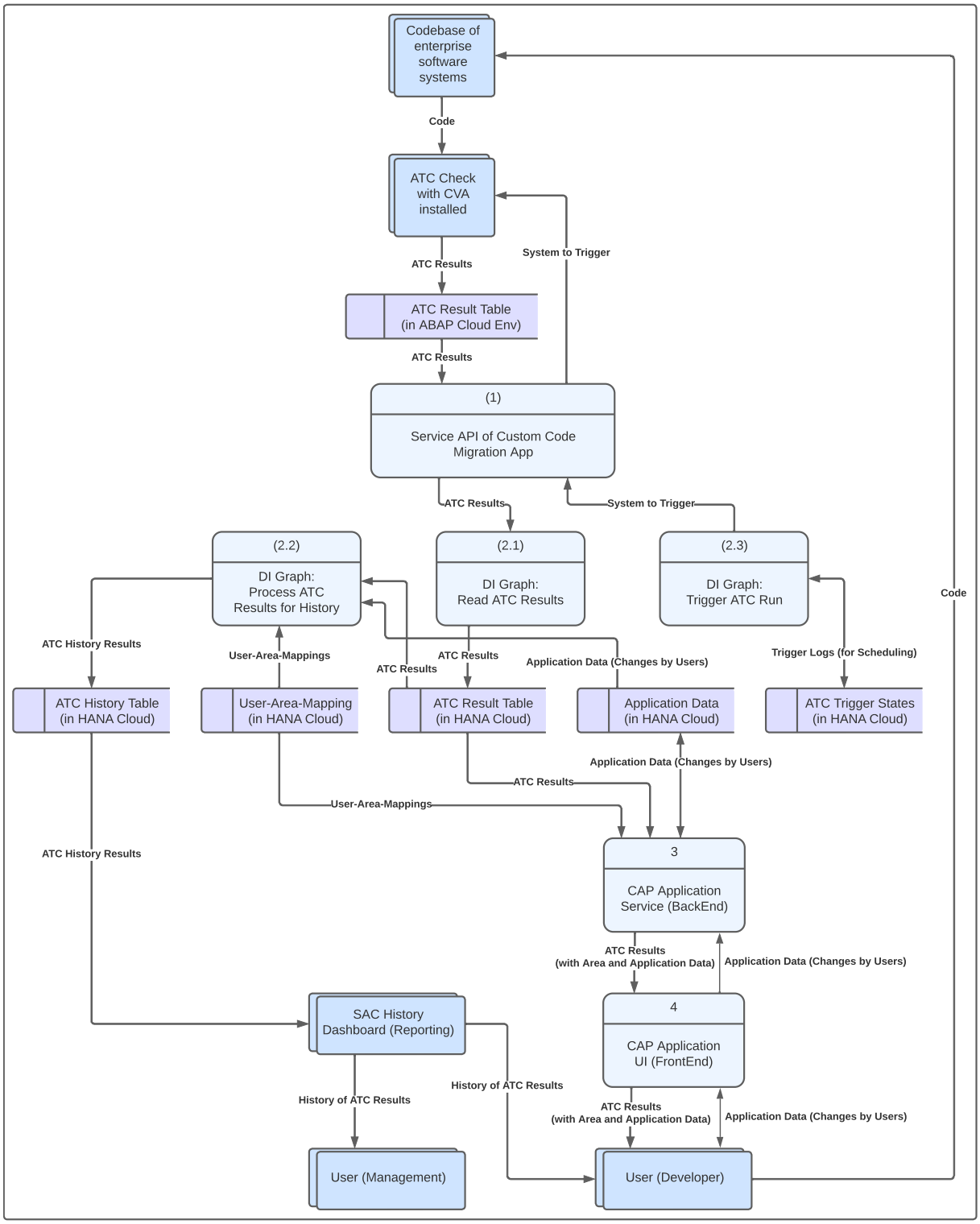


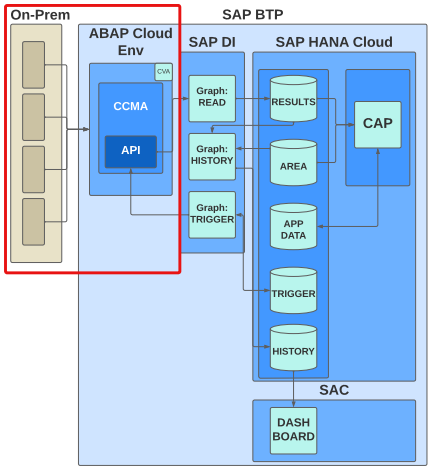
Abbildung 4.8: Datenflussdiagramm der Code-Sicherheits-Plattform

5 Implementierung

Dieses Kapitel beschreibt die Umsetzung des in Kapitel 4.4 erarbeiteten Konzepts. Dabei werden verstärkt Herausforderungen und Dinge, die besonders beachtet werden mussten, hervorgehoben. Die Umsetzung bezieht sich auf das Unternehmensumfeld der Geberit-Gruppe, ist aber auch auf andere Unternehmen, die mehrere SAP-Systeme analysieren wollen, übertragbar.

Die Strukturierung des Implementierungskapitels folgt grob dem Datenfluss der Plattform. Um beim Lesen den Überblick zu behalten, um welchen Teil des Konzeptes es sich gerade handelt, wird eine Art „Minimap“ mit Markierungen eingeführt, die sich auf das Komponentendiagramm des Konzepts 4.7 bezieht.

5.1 Einrichten des CVAs im ABAP Cloud Environment



Die Firma Geberit hat bereits ein bestehendes SAP BTP im Einsatz, welches für die Entwicklung von Cloud-Anwendungen und mehr genutzt wird. Der erste Schritt der Implementierung ist daher die Erstellung eines neuen Subaccounts im BTP, der für die ABAP Cloud Environment genutzt wird. Im Subaccount muss ein Cloud Foundry Environment installiert werden, welches dann wiederum den ABAP Cloud Environment Space beinhaltet. Dies wird mit dem Entitlement „ABAP Environment“ und „Web access for ABAP“ freigeschaltet.

Um die verschiedenen On-Premise-Systeme mit dem ABAP Cloud Environment zu verbinden, muss zudem ein „Cloud Connector“[7] vorhanden sein werden. Die Verbindungen können dann als „Destinations“ im SAP BTP konfiguriert werden, wodurch die RFC-Verbindungen zwischen den Systemen und dem ABAP Cloud Environment hergestellt werden.

Um eine ATC Check Variante mit dem CVA zu erstellen, wird zum Stand der Bachelorarbeit ein ABAP-Cloud-Projekt in Eclipse benötigt, da dies noch nicht über die UI des ABAP Cloud Environments möglich ist. Es muss dabei beachtet werden, eine aktuelle

[7] Ein SAP Cloud Connector ist ein Server, der als Verbindung zwischen BTP-Services und On-Premise-Systemen dient. Er wird im Firmennetz aufgesetzt und fungiert als Reverse Proxy zwischen dem lokalen Netzwerk und dem SAP BTP. (vgl. SAP 2023e)

Version von Eclipse und den ABAP Development Tools zu nutzen, da es sonst zu Kompatibilitätsproblemen kommen kann. In dem ABAP-Cloud-Projekt wird dann eine neue Check Variante erstellt, die die CVA-Checks enthält.

Mit dieser Check Variante kann dann in der CCMA ein „Custom Code Analysis Project“ erstellt werden, welches die CVA-Checks auf einem Remote-System ausführt. Es muss pro System ein Analyse-Projekt erstellt werden. Die ATC-Checks können nicht auf mehreren Systemen gleichzeitig ausgeführt werden. Dies muss beim Entwickeln des Scheduler-Graphs beachtet werden.

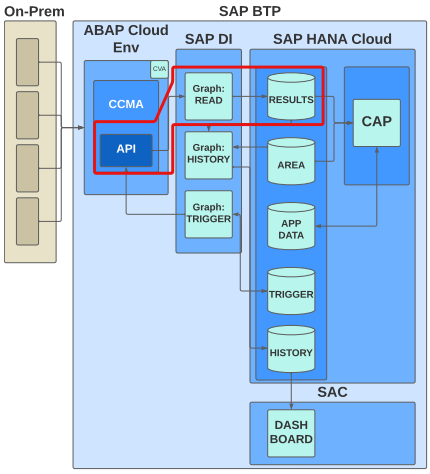
5.2 Laden der Ergebnisse der CVA-Analysen

Die Custom Code Migration App (CCMA) ist eine Fiori-Anwendung, die auf einen OData-Service[8] im Backend zugreift. Dieser OData-Service kann über die richtige Adresse auch von externen Anwendungen genutzt werden, um beispielsweise die Ergebnisse der Analyse auszulesen. Der Service und die für die Code-Sicherheits-Plattform relevanten Endpunkte werden in Abbildung 5.1 dargestellt.

`https://<host>/sap/opu/odata/sap/SYCM_APS_C_PROJECT_CDS`

- `/sycm_aps_c_atc_find_alp` - ATC Ergebnisse
- `/sycm_aps_c_project` - CCMA Projekte (und deren Eigenschaften)
- `/sycm_aps_c_projectRun_atc` - Start einer ATC Analyse

Abbildung 5.1: OData-Service der CCMA



Um die Ergebnisse der Analysen zu laden wird ein SAP DI Graph entwickelt, der sich mit dem OData-Service verbindet und die Daten ausliest („Read ATC Results“). Diese Verbindung ist, wie im Konzept bereits erwähnt, nicht nativ möglich, weshalb hier ein komplexerer Umweg entwickelt werden muss.

Das Problem bei der Datenabfrage ist die Authentifizierung. Wenn ein Benutzer sich auf dem BTP und somit auch auf dem ABAP Cloud Environment anmeldet, wird ein „Access Token“ generiert, welches für die Authentifizierung genutzt wird. Dies wird beim direkten Abfragen des OData-Services im Browser beim HTTP Request an den OData-Service übermittelt, wodurch der Benutzer authentifiziert ist. Wenn allerdings von einer neuen Session, wie beispielsweise von einem SAP DI Graph oder einem lokal ausgeführten Skript, auf den OData-Service zugegriffen wird, ist kein Access Token vorhanden, wodurch die Anfrage abgelehnt wird. Um dieses Problem zu lösen, wird eine Zentrale Python-Klasse erstellt, die als Interface zu den OData-Endpunkten dient und die Authentifizierung übernimmt. Zudem wird ein technischer User im BTP angelegt, der für den programmatischen Zugriff auf

[8] OData ist ein Protokoll, welches auf Representational State Transfer (REST) basiert und es ermöglicht, über das HTTP-Protokoll Anfragen zu senden und Daten zu empfangen. Es wird dafür genutzt, um RESTful APIs zu erstellen (vgl. OData.org 2023).

die OData-Endpunkte genutzt wird. Für die Erstellung des Authentifizierungs-Interfaces wurde der Authentifizierungsprozess im Browser analysiert und „reverse-engineered“. Der Prozess wird in Abbildung 5.2 dargestellt und im Folgenden beschrieben.

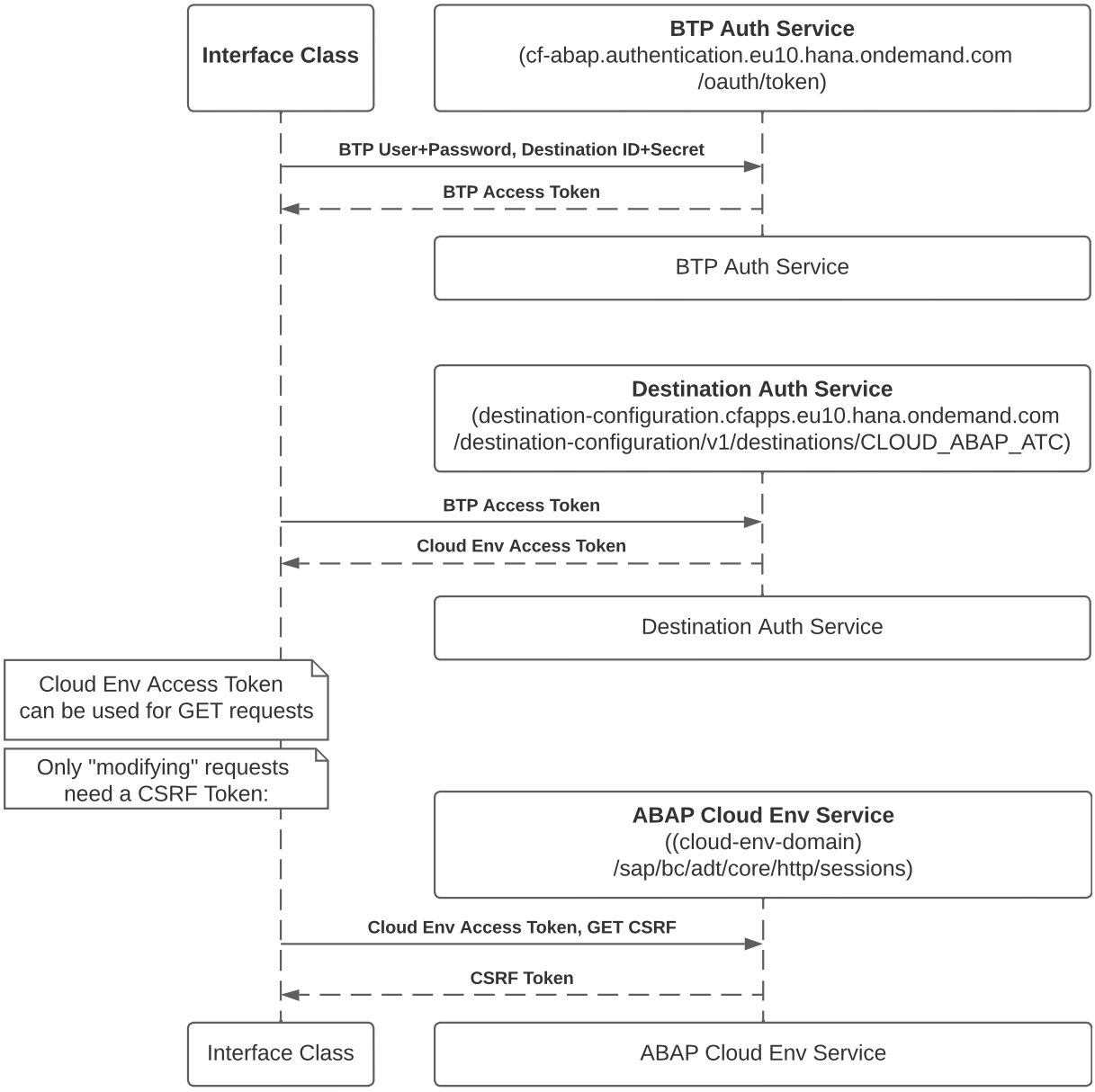


Abbildung 5.2: Sequenzdiagramm der Authentifizierung

Zunächst wird eine Anfrage mit den Credentials des technischen Users an den Authentifizierungs-Service des BTP gesendet. Dieser Service generiert da BTP Access Token und sendet diesen zurück. Dieses Token kann verwendet werden, um sich bei dem sogenannten Destination-Service zu authentifizieren. Eine Destination ist eine Konfiguration im BTP, die beispielsweise eingehende Verbindungen zu einem bestimmten System oder Service definiert. Der Destination-Service bündelt dann die Anmeldungsinformationen und

generiert das Token für das gewünschte System, hier das ABAP Cloud Environment. Dieses Token wird bei den OData-Requests an die API mitgesendet und authentifiziert den Benutzer erfolgreich. Für modifizierende Requests, also `POST`, `PUT` und `DELETE`, wird zusätzlich ein „CSRF-Token“[9] benötigt, welches mittels einer Anfrage an den dafür vorgesehenen Endpunkt generiert wird.

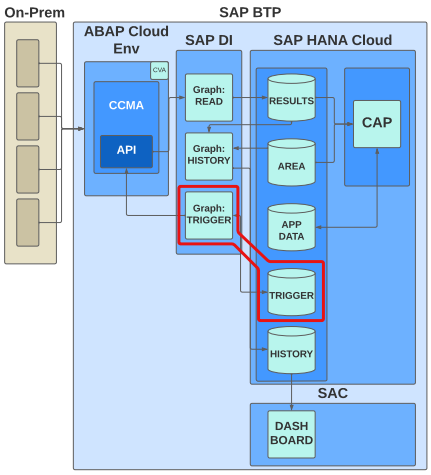
Die Interface-Klasse wird mit den Credentials des technischen Users und den System-Informationen initialisiert und hat zwei öffentliche Methoden, eine für das Ausführen von `GET` Requests und eine für das Ausführen von `POST` Requests. Die Authentifizierungs-Token werden in der Klasse gespeichert und nur einmal pro Instanz erstellt, um unnötige Anfragen an die Token-Services zu vermeiden. Aus demselben Grund wird auch nur bei einem `POST`-Request das CSRF-Token generiert, da dies nur bei Änderungen an den Daten notwendig ist. Die Klasse kann dann von anderen Skripten oder Graphen genutzt werden, um Daten aus einem OData-Service eines ABAP Cloud Environments auszulesen oder zu bearbeiten.

Der Graph „Read ATC Results“ lädt mithilfe der Interface-Klasse die Analyse-Ergebnisse für jedes System aus dem ABAP Cloud Environment (`Endpunkt: sycm_aps_c_atc_find_alp`) und schreibt diese in eine HANA Cloud Tabelle. Dieser Prozess wird jeden Tag um eine bestimmte Uhrzeit ausgeführt, wobei die Daten der letzten Analyse überschrieben werden. Dies stellt sicher, dass behobene Sicherheitslücken nicht mehr in der Tabelle vorhanden sind.

In den Anforderungen ist das Datenfeld „System“ gefordert. Dieses Feld wird nicht von dem OData-Service bereitgestellt, weshalb es in dem Graph ergänzt werden muss. Hierfür wird eine Mapper-Klasse erstellt, die die Konvertierung von der `project_id` des CCMA-Projekts zu dem Systemnamen (`system_id`) übernimmt. Es muss beachtet werden, dass die in der Klasse gespeicherten Zuordnungen beim Ändern von CCMA-Projekten manuell angepasst werden müssen.

[9] CSRF steht für „Cross-site request forgery“ und bezeichnet eine Web-Sicherheitslücke, bei dem ungewollte Anfragen mit bereits authentifizierten Usern von Angreifer ausgeführt werden. Um dies zu verhindert, werden CSRF-Tokens benutzt, die einen geheimen Wert zwischen User und Server darstellen und bei jeder sensitiven Anfrage mitgeschickt werden müssen. (vgl. Cloudflare 2023).

5.3 Automatisierung der CVA-Analysen



Da im ABAP Cloud Environment zum Stand dieser Bachelorarbeit noch keine Möglichkeit besteht, die ATC-Analysen zu automatisieren, muss ein Scheduler entwickelt werden, der die Analysen periodisch ausführt. Hierfür wird ein weiterer SAP DI Graph entwickelt, der sich mit dem OData-Service-Endpunkt `/sycm_aps_c_projectRun_atc` verbindet und die Analysen startet („Trigger ATC Run“).

Der Graph verwendet hierfür ebenfalls die Authentifizierungs-Interface-Klasse und ist im grafischen Editor des SAP DIs wie in Abbildung 5.3 dargestellt aufgebaut.

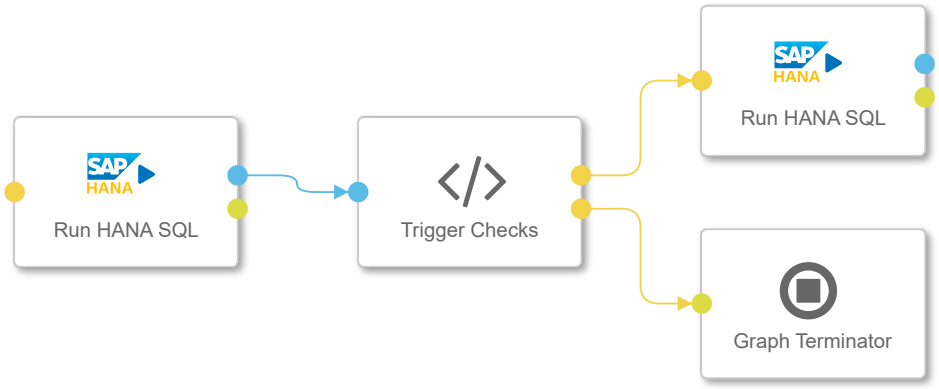


Abbildung 5.3: Trigger ATC Run Graph

Die Analysen der Systeme können nicht parallel zueinander ausgeführt werden. Aus diesem Grund müssen die einzelnen `project_ids` der CCMA-Projekte nacheinander ausgeführt werden. Dies muss aufgrund der unterschiedlichen benötigten Zeitspannen der Analysen koordiniert werden. Hierfür wird in einer weiteren Datenbank-Tabelle in der HANA Cloud protokolliert, wann die Analysen der Systeme starten und als beendet markiert werden („Trigger States“).

Der Graph „Trigger ATC Run“ startet das erste System um 0 Uhr und prüft dann alle 30 Minuten, ob die Analyse beendet ist. Dies geschieht auch über einen API-Request an den OData-Service, wobei der Endpunkt `/sycm_aps_c_project` genutzt wird. Ist die Analyse beendet, wird dies in der Tabelle vermerkt und das nächste System gestartet, bis die Reihe an Systemen komplett analysiert ist. Die Analyse aller fünf angeschlossenen

Systeme benötigt circa 6,5 Stunden.

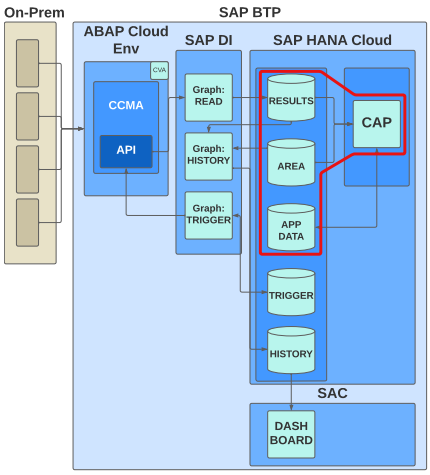
Der Python-Operator „Trigger Checks“ in der Mitte hält die zuvor beschriebene Logik des Schedulers, welche im Listing 5.1 nochmals dargestellt ist. Die Protokoll-Daten aus der „Trigger States“ Tabelle werden dabei in einem Dataframe in den Python-Operator übertragen und in den Variablen `current_system`, `current_event` und `current_timestamp` gespeichert.

```
if current_event == "finished":

    if is_last_system(current_system) and not is_today(current_timestamp):
        start_analysis_of_first_system()
    elif not is_last_system(current_system):
        start_analysis_of_next_system(current_system)

elif current_event == "started":

    if not check_if_analysis_is_running():
        finish_system(current_system)

        if is_last_system(current_system) and not is_today(current_timestamp):
            start_analysis_of_first_system()
        elif not is_last_system(current_system):
            start_analysis_of_next_system(current_system)

    # else: analysis is still running -> do nothing
```

Listing 5.1: Trigger Checks Python-Operator

Der Graph „Trigger ATC Run“ deckt somit die Anforderung der Automatisierung der Analyse ab.

5.4 Benutzeroberfläche



Die Benutzeroberfläche ist die zentrale Schnittstelle, mit der Enduser mit der Plattform interagieren. Es wurde bereits festgelegt, dass diese in einer CAP-Anwendung entwickelt wird, welche auf dem SAP BTP läuft und als Frontend eine SAPUI5-App nutzt. Wie zuvor bereits erwähnt, wird die Datenbank-Tabelle mit den Ergebnissen der Analysen (Findings) jeden Tag überschrieben, wodurch die Notwendigkeit einer weiteren Datenbank-Tabelle entsteht, die die Änderungen der Einträge hält. Diese Tabelle (`Application Data`) wird ebenfalls in der HANA Cloud erstellt und hat die Felder `item_id`, `status`, `remark`, `new_responsible`, `last_modified_at` und `last_modified_by`. Neben der Änderungstabelle wird eine Tabelle erstellt, die die Zuordnung der Entwickler zu den Bereichen hält (`Developer Areas`), sodass dies auch dem Enduser angezeigt werden kann. Diese Tabelle muss manuell gepflegt werden, könnte aber in Zukunft aus Personaldaten automatisch aktualisiert werden.

Die Applikation besteht aus dem CAP-Backend, das auf Node.js basiert und dem SAPUI5-Frontend. Die Ordnerstruktur ist in Abbildung 5.4 dargestellt.

```
/
├── app/
│   ├── findings-list/ ..... SAPUI5-Frontend
│   │   └── webapp/
│   │       ├── model/
│   │       ├── view/
│   │       └── controller/
│   └── fiori.html/ ..... SAPUI5-App-Container (Einstieg)
├── db/
│   └── schema.cds/ ..... Datenbank-Modell (Tabellen-Definitionen)
└── srv/ ..... Backend-Service
    ├── serviz.cds/ ..... Datentypen und Funktionsheader des Services
    └── serviz.js/ ..... Node.js-Implementierung des Services
```

Abbildung 5.4: Ordnerstruktur der CAP-Anwendung (Ausschnitt)

Zunächst wird das Datenbank-Modell definiert. Da die CAP App aber auf bereits existierenden Tabellen in der HANA Cloud zugreift, ist dies in erster Linie für die lokale Entwicklung notwendig, bei der mit Mock-Daten gearbeitet wird. Der Backend-Service

kann bei einer CAP-Anwendung in Java oder JavaScript (Node.js) entwickelt werden. Da JavaScript auch im Frontend genutzt wird, wird für die Implementierung des Services ebenfalls JavaScript genutzt.

Zunächst werden die benötigten Entitäten definiert, was in der Datei `serviz.cds` geschieht. Hierbei wird die CDS-Syntax genutzt, die eine eigene Sprache für die Definition von Datenmodellen ist. Die Haupt-Entität ist `Data`, welches die Verknüpfungen aus den zuvor beschriebenen Tabellen enthält, wie in Listing 5.2 dargestellt.

```
entity Findings          as projection on db.ATC_RESULTS;
entity CustomChanges     as projection on db.APP_DATA;
entity UserAreaMappings as projection on db.USER_AREA;
entity CustomAreas       as SELECT from CustomChanges
    LEFT JOIN UserAreaMappings on CustomChanges.new_responsible = UserAreaMappings.user {
        item_id,
        new_responsible,
        area as areaCustom
    };
entity Data              as SELECT from Findings
    LEFT JOIN CustomChanges              on Findings.item_id = CustomChanges.item_id
    LEFT JOIN UserAreaMappings as Areas on Findings.person_responsible = Areas.user
    LEFT JOIN CustomAreas                on Findings.item_id = CustomAreas.item_id   {
        key Findings.item_id,
        // ...
    };
    action changeCustomData(aItemId : array of String,sResponsible : String,sStatus :
    String,sRemark : String);
}
```

Listing 5.2: Datenverknüpfung im CAP-Backend

Um die Tabelle im Frontend zu füllen und sortieren und filtern zu können, müssen keine eigenen OData-Services entwickelt werden, da diese von CDS automatisch generiert werden. Lediglich die Änderungs-Aktion `changeCustomData` muss implementiert werden. Diese wird aufgerufen, wenn der Benutzer einen oder mehrere Einträge in der Tabelle auswählt und deren veränderbare Felder bearbeitet. Die Aktion muss dann die Änderungs-Einträge in der Datenbank-Tabelle `Application Data` aktualisieren oder neue Einträge erstellen. Für die Implementierung der Aktion wird die Standard-Klasse des generierten Services überladen und das Event `changeCustomData` hinzugefügt. Hier wird über alle im Request mitgegebenen Einträge iteriert und die Änderungsdaten in die Datenbank geschrieben, wobei geprüft wird, ob der Eintrag bereits existiert oder nicht. Die Implementierung der Aktion ist in Listing 5.3 dargestellt.

```
class atcfindingserviz extends cds.ApplicationService { async init() {
  this.on('changeCustomData', async req => {
    // loop over all items
    for (let i = 0; i < req.data.aItemId.length; i++) {
      const sItemId = req.data.aItemId[i].split("'")[1];
```

```
6       const aCustomChange = [{
7         item_id: sItemId,
8         new_responsible: req.data.sResponsible,
9         status: req.data.sStatus,
10        remark: req.data.sRemark,
11        last_modified_at: new Date(),
12        last_modified_by: req.user.id
13      }];
14      // check if item already exists in app data table and update or insert accordingly
15      let aCheckExists = await SELECT`*`.from(cds.entities.APP_DATA)
16                                        .where({ item_id: sItemId });
17      if (aCheckExists.length !== 0) {
18        await UPDATE(cds.entities.APP_DATA).set(aCustomChange[0]
19                                              .where({ item_id: sItemId });
20      } else {
21        await INSERT(aCustomChange).into(cds.entities.APP_DATA);
22      }
23    };
24  });
25  super.init()
26 }}
```

Listing 5.3: Implementierung der Änderungs-Aktion im CAP-Backend

Bei der Entwicklung von SAPUI5-Apps wird die Applikation definiert, indem Views, Models und Controller erstellt werden. Die Views definieren die Benutzeroberfläche, die Models verwalten die Daten und die Controller halten die Logik. Dafür reagieren die Controller auf Interaktionen des Benutzers mit der View und modifizieren Views und Models.

Die reine UI-Programmierung ist kein Schwerpunkt dieser Bachelorarbeit und wird aufgrund der zeitlichen Beschränkungen unter Unterstützung eines Mitarbeiters des SAP-Development-Teams vorgenommen. Deshalb werden hier nur die wichtigsten Punkte der Umsetzung beschrieben werden.

Die Einstiegs-View der App ist die Tabelle, die die Ergebnisse der Analysen darstellt. Diese wird in der Datei `findinglist.view.xml` definiert und hat als Hauptbestandteil eine `sap.m.Table`, die mittels eines Data-Bindings an den OData-Service gebunden wird.

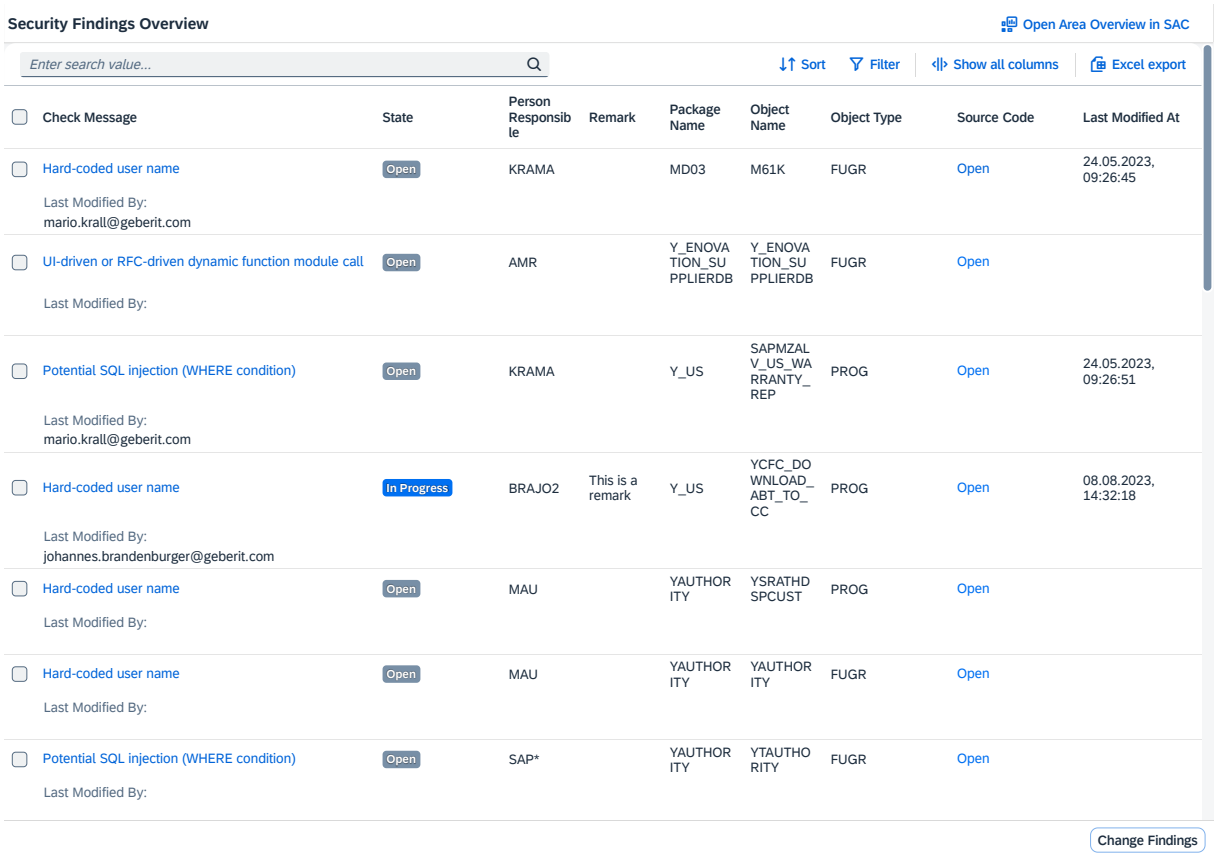


Abbildung 5.5: Benutzeroberfläche: 1. View: Sicherheitslücken tabellarisch dargestellt

Um weniger wichtigere Spalten auszublenden, wird ein Button implementiert, der diese der Tabelle hinzufügt bzw. entfernt. Zudem werden Buttons für das Sortieren, Filtern und Exportieren der Daten hinzugefügt. Um schnell Einträge wiederzufinden, wird zusätzlich zu den in den Anforderungen festgelegten Filtern eine Volltextsuche implementiert. Das Ändern der Einträge ist durch das Markieren und dem Klicken auf den „Changes Finding“-Button möglich. Dieser öffnet den in Abbildung 5.6 dargestellten Dialog, in dem die veränderbaren Felder bearbeitet werden können und die Änderungen durch die zuvor beschriebene Aktion in der Datenbank gespeichert werden.

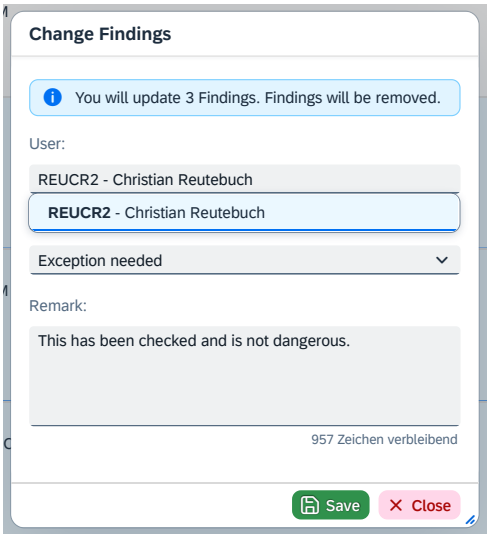


Abbildung 5.6: Benutzeroberfläche: Dialog zum Bearbeiten der Einträge

Durch Klicken auf das „Check Message“-Feld eines Eintrags wird auf die 2. View der App gewechselt, die Details zu der Art Sicherheitslücke darstellt und helfen soll, diese zu beheben. Dabei wird vom Einstieg auf `/Explanation/<check_message>` geroutet. Wenn in der URL eine unbekannte Art von Sicherheitslücke angegeben wird, wird eine Fehlerseite angezeigt.

Die Daten für die Erklärungsseiten werden in einer JSON-Datei gespeichert, die als Model von der View konsumiert wird. Sie werden händisch erstellt, wobei ein SAP-Blog als Basis dient, der die CVA-Checks beschreibt (vgl. Peter Barker (SAP) 2017). Dieser Aufwand wird zunächst nur für Sicherheitslücken mit der Priorität 1 betrieben, da diese am kritischsten sind, kann aber in Zukunft auf alle Sicherheitslücken ausgeweitet werden. Die Information werden in der View mittels Text und Codeblöcken dargestellt. Zusätzlich wird der passende Teil des SAP-Blogs eingebettet und verlinkt.

Die Abbildung 5.7 zeigt am Beispiel der Sicherheitslücke „Potential SQL injection (SET clause)“, wie die Erklärungsseiten aufgebaut sind.

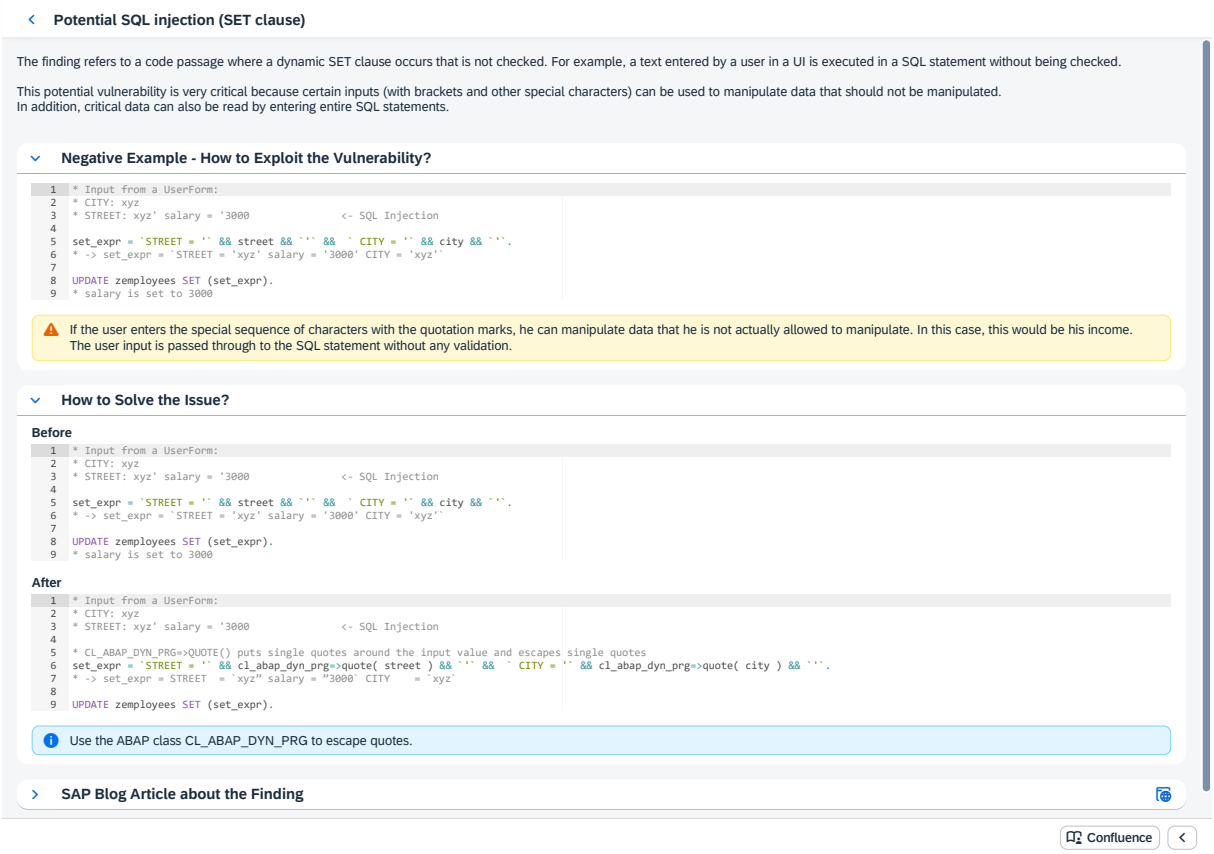


Abbildung 5.7: Benutzeroberfläche: 2. View: Erklärung der Sicherheitslücken

Die wichtigsten Bestandteile der SAPUI5-App werden nochmals im Komponentendiagramm 5.8 in Bezug zueinander gestellt.

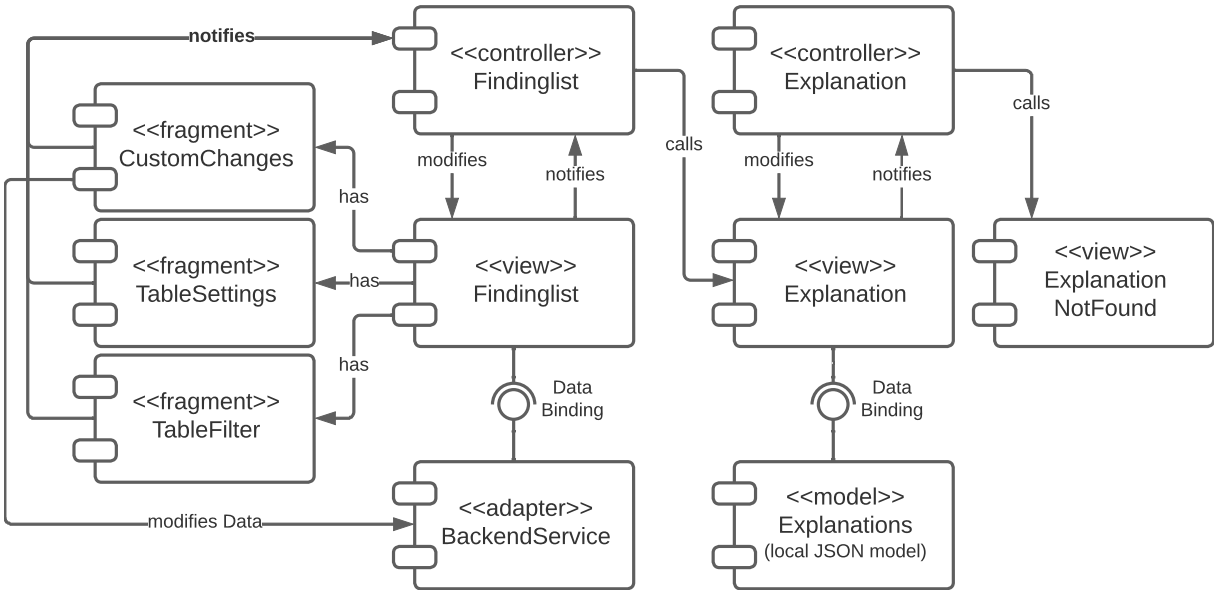
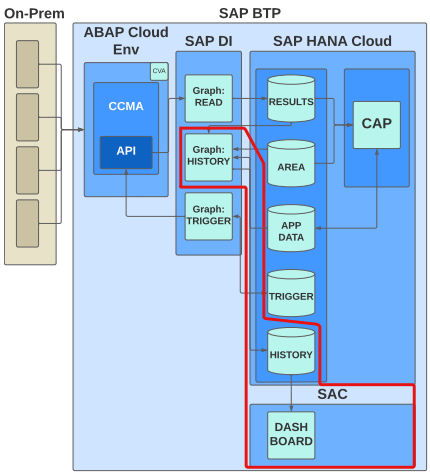


Abbildung 5.8: Komponentendiagramm der Benutzeroberfläche

Durch die Implementierung der Benutzeroberfläche kann ein Enduser die Ergebnisse der CVA-Analysen einsehen, bearbeiten und sich über die Sicherheitslücken informieren und diese beheben. Dadurch sind alle Anforderungen abgedeckt, bis auf die Historie der Sicherheitslücken, die in einer Übersicht über die Bereiche in dem Reporting-Tool SAC dargestellt werden soll.

5.5 Übersichts-Dashboard für das Reporting



Für die Umsetzung der Reporting-Funktionalität wird wie im Konzept beschrieben ein neuer Graph in SAP DI benötigt, welcher ebenfalls täglich eingeplant wird. Der Graph verknüpft bei jedem Lauf die Daten aus drei der zuvor erstellten Tabellen und speichert die aufsummierten Werte in einer neuen Tabelle. Da die ATC-Ergebnis-Tabelle jeden Tag überschrieben wird, braucht es diesen Schritt um eine Historie der Sicherheitslücken zu erstellen. Die neue Tabelle hält die Anzahl der Sicherheitslücken pro Bereich und Tag fest.

Da die Tabelle ebenfalls in der HANA Datenbank liegt, kann sie in SAC nativ als Datenquelle verwendet werden. Dadurch ist das Erstellen des eigentlichen Dashboards in SAC sehr einfach. Es wird eine neue „Story“ erstellt, die ein Linien-Diagramm enthält, welches die geforderten Daten darstellt. Die Abbildung 5.9 zeigt das fertige Dashboard.

Abbildung 5.9: Reporting in SAC: Fehler pro Bereich[10]

Mit der Fertigstellung des Dashboards sind alle Anforderungen abgedeckt und die Implementierung abgeschlossen.

[10] Der Sprung vom 03. auf den 04. August ist auf die Anbindung von 2 weiteren Systemen zurückzuführen. Viele der gefundenen Sicherheitslücken können nicht einem Bereich zugeordnet werden („Not found“), da die Ersteller beispielsweise nicht mehr im Unternehmen arbeiten.

6 Evaluation

Für die Evaluation der Plattform wird ein Workshop mit den Stakeholdern durchgeführt. Dieser ist nicht strikt nach den in Kapitel 2.3.3 beschriebenen Vorgaben aufgebaut, sondern wird an die Situation angepasst. So wird die Entwicklung von Ideen und konkreter Maßnahmen, welche in Zukunft umgesetzt werden sollen, nur bedingt durchgeführt. Workshop soll dazu dienen, die Plattform vorzustellen, Feedback zu erhalten und Ideen für weitere Funktionen zu sammeln. Auf eine Vorstellungsrunde wird verzichtet, da die Stakeholder sich bereits kennen.

Zu Beginn des Workshops wird die Ausgangssituation und die Motivation für die Entwicklung der Plattform vorgestellt. Anschließend wird erklärt, wie die Plattform funktioniert und wie die Entwickler mit ihr interagieren sollen. Dabei wird nur beschränkt auf die technischen Hintergründe eingegangen, da diese für den Großteil der Stakeholder nicht relevant sind. Im Anschluss werden die Kollegen aufgefordert, die Plattform selbst zu testen und sich mit ihr vertraut zu machen. Dabei soll der Prozess klargestellt werden, wie die Entwickler die Plattform in ihren Arbeitsalltag integrieren sollen: Die Entwickler sollen regelmäßig die Benutzeroberfläche aufrufen und die ihnen zugeordneten Sicherheitslücken überprüfen. Sie können die Findings entweder beheben, einem Kollegen zuweisen oder die Lücke als nicht relevant markieren (siehe Abbildung 4.6).

Abschließend wird die Plattform in einer offenen Diskussion evaluiert. Dabei werden die Stakeholder aufgefordert, ihre Meinung zu der Plattform zu äußern. Das Feedback ist durchweg positiv - die Anforderungen an die Plattform sind erfüllt. Auch werden bei diesem Schritt weitere Ideen für Funktionen gesammelt, welche in Zukunft implementiert werden sollen. Die Ideen werden dokumentiert und von jedem Mitglied des Workshops akzeptiert. Die gewünschten Funktionen werden im Folgenden aufgelistet und in den Ausblick in Kapitel 7.2 einfließen.

- **Anzahl der Findings**: Bei der tabellarischen Übersicht der gefundenen Sicherheitslücken soll die Anzahl der Einträge, die die Filter erfüllen, angezeigt werden.
- **Eingabehilfe bei Filtern**
  - **Bereich, Priorität, Status und System**: Bei den Filtern Bereich, Priorität, Status und System soll eine Auswahlmöglichkeit angeboten werden, da diese Felder nur bestimmte Werte annehmen können.
  - **Verantwortlicher**: Beim Filtern nach dem verantwortlichen Entwickler sollen die Benutzereingaben direkt in Großbuchstaben umgewandelt werden.

- **Benachrichtigung bei neuen Findings**: Die Entwickler sollen per E-Mail benachrichtigt werden, wenn neue Sicherheitslücken gefunden wurden, die sie betreffen. Diese Funktion soll allerdings erst nach dem initialen Abarbeiten der Findings aktiviert werden.

7 Schlussbetrachtung

7.1 Fazit

Die IT-Landschaft eines Unternehmens ist nie zu 100 % vor Gefahren geschützt. Die voranschreitende Digitalisierung verlangt es aber, alles dafür zu tun, Unternehmenssoftwaresysteme so sicher wie möglich zu machen. Diesem Ziel dient auch die Entwicklung der in dieser Arbeit vorgestellten Code-Sicherheits-Plattform.

Die Plattform soll durch Analyse von ABAP-Eigenentwicklungen Sicherheitslücken aufdecken und die Entwickler bei der Behebung dieser unterstützen. Die Bachelorarbeit dokumentiert nach der Definition theoretischer Grundlagen die Anforderungsanalyse, die Konzeption und die Implementierung der Plattform. Den Ablauf und die Ergebnisse der einzelnen Phasen werden im Folgenden zusammengefasst.

Durch die Analyse der Anforderungen wurden die Erwartungen der Stakeholder an die Plattform konkretisiert und spezifiziert. Dies ergab zusammengefasst, dass die Plattform aus einer Analyse-Komponente, einer Benutzeroberfläche mit zwei Ansichten und einem Reporting-Dashboard bestehen soll. Der Spezifikationsentwurf diente als Grundlage für die weitere Konzeption. Neben den Anforderungen wurden die bereits festgelegten Entscheidungen dokumentiert. Hierbei wurde deutlich, dass das System mit SAP-Technologien entwickelt werden soll und der Code Vulnerability Analyzer als Analyse-Tool verwendet wird.

Eine herausfordernde Frage, die im Rahmen der Konzeption beantwortet werden musste, war die Wahl des Betriebsmodells des Analyse-Tools. Hier existieren zwei grundsätzliche Möglichkeiten, die sich explizit auf die Systemarchitektur auswirken: On-Premise- oder Cloud-Betrieb. Mittels einer Nutzwertanalyse wurde festgelegt, dass der Code Vulnerability Analyzer in der Cloud betrieben werden soll. Entscheidend für dieses Ergebnis waren vor allem die hohen Lizenzkosten für die On-Premise-Variante.

Anhand der Anforderungen, den vorausgegangenen Festlegungen und des Ergebnisses der Nutzwertanalyse als Informationsgrundlage, wurde die Systemarchitektur konzipiert. Das Konzept ist speziell an die Anforderungen und Systemlandschaft des Unternehmens Geberit angepasst, kann aber auch auf andere Unternehmen übertragen werden. Die Architektur besteht aus mehreren Komponenten und Systemen, die im Zusammenspiel die Code-Sicherheits-Plattform bilden und dabei alle Anforderungen erfüllen. Zusammengefasst beinhaltet das Konzept ein ABAP-Cloud-Environment, mehrere SAP DI-Graphen

und HANA-Datenbanktabellen, eine SAP CAP-Anwendung und ein SAC-Dashboard.

Die Implementierung der Plattform erfolgte unmittelbar nach der Konzeption und bestätigte die Machbarkeit des Konzepts. Herausfordernd war hierbei die Kommunikation zwischen den einzelnen Komponenten und Systemen, da es für manche Verbindungen keine nativen Schnittstellen gab. Durch die Programmierung einer Interface-Klasse und dem Reverse-Engineering eines Authentifizierungsprozesses, konnte diese Herausforderung gemeistert werden.

Mittels eines Workshops wurde das System den Stakeholdern vorgestellt und Feedback eingeholt, was die Evaluation der Plattform darstellt. Das Feedback an die Plattform war durchweg positiv und die Anforderungen somit erfüllt. Die Teilnehmer des Workshops sammelten weitere Ideen für Funktionen, welche in Zukunft implementiert werden sollen.

Die in dieser Bachelorarbeit dokumentierte Entwicklung der Code-Sicherheits-Plattform beantwortet die Frage, wie eine zentrale Plattform für die Analyse und Verwaltung der Code-Sicherheit in Unternehmenssoftwaresystemen aussehen kann. Mithilfe der Plattform können Sicherheitslücken in Eigenentwicklungen aufgedeckt und behoben werden. Dies bringt das große Ziel, die Sicherheit der Unternehmenssoftwaresysteme zu erhöhen, einen Schritt näher.

Im folgenden Kapitel wird ein Ausblick auf den weiteren Verlauf des Projekts gegeben und es werden mögliche Weiterentwicklungen der Plattform vorgestellt.

7.2 Ausblick

Die Code-Sicherheits-Plattform ist ein laufendes Projekt, welches in Zukunft weiterentwickelt wird. Wie in der Stakeholderbeschreibung in 3.2 beschrieben, wird die Plattform von der SAP-Development-Abteilung verantwortet. Das Team wird die Plattform in Zukunft betreiben und weiterentwickeln, weshalb ein Handover an das Team stattfindet. Hierbei wird nochmals die Funktionsweise der Plattform erklärt und das weitere Vorgehen besprochen: Die Plattform wird in fortlaufenden Weiterentwicklungen verbessert und die in der Evaluation gesammelten Ideen werden umgesetzt. Zudem müssen die Erklärungsinhalte, die in Form der Erklärungsseiten den Entwicklern dabei helfen, die Sicherheitslücken zu verstehen, aktualisiert werden. Wenn neue Sicherheitslücken in den CVA aufgenommen werden oder auch Priorität zwei und drei Lücken behoben werden sollen, bedarf es dieser Aktualisierung.

Auch Weiterentwicklungen seitens SAP beeinflussen die Zukunft der Plattform: Für die Kommunikation mit dem ABAP Cloud Environment wird ein OData-Service des Systems verwendet. Wenn sich die verwendeten API-Endpunkte in Zukunft ändern, muss die Plattform angepasst werden. Der Aufwand für diese Anpassung ist allerdings gering, da sich der komplexe Teil, die Authentifizierung, nicht verändern dürfte.

Ein Update des ABAP Cloud Environments, das zum Ende des Bearbeitungszeitraums dieser Bachelorarbeit freigeschaltet wurde, ermöglicht es, die Instanz nur zeitweise zu betreiben. So können in Zukunft Kosten gespart werden, da die Analysen nur etwa sieben Stunden am Tag beanspruchen.

Zudem soll Ende 2023 das „Developer Scenario“ veröffentlicht werden. Dieses erlaubt es, die in der Cloud ausgeführten ATC-Checks direkt in der Entwicklungsumgebung zu starten. So können Entwickler in Zukunft unmittelbar nach dem Programmieren eines neuen Features die Sicherheit des Codes überprüfen. Dann wird der Code Vulnerability Analyzer nur für das aktuell geöffnete Objekt ausgeführt und nicht mehr wie aktuell für ein komplettes System, was die Laufzeit entscheidend verkürzt.

Literatur

Bardas, Alexandru G. (2010). „STATIC CODE ANALYSIS“. In: URL: https://core.ac.uk/display/6552448?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1 (besucht am 28. 08. 2023).

Beermann, Susanne und Monika Schubach (2019). *Workshops: vorbereiten, durchführen, nachbereiten*. 4. Auflage. TaschenGuide 189. Freiburg: Haufe. 127 S. ISBN: 9783648134764 9783648134733.

Bohinc, Tomas (2019). *Grundlagen des Projektmanagements: Methoden, Techniken und Tools für Projektleiter*. OCLC: 1090497183. Offenbach: GABAL Verlag. ISBN: 9783956238512.

Bron, Jean-Yves (2020). *System requirements engineering: a SysML supported requirements engineering method*. OCLC: 1178650623. London, Hoboken, NJ: ISTE, Ltd. ; Wiley. ISBN: 9781119751557.

Chies, Sieglind (2016). „Definition ERP-System“. In: *Change Management bei der Einführung neuer IT-Technologien: Mitarbeiter ins Boot holen – mit angewandter Psychologie*. Hrsg. von Sieglind Chies. essentials. Wiesbaden: Springer Fachmedien, S. 3–7. ISBN: 9783658116354. DOI: 10.1007/978-3-658-11635-4_2. URL: https://doi.org/10.1007/978-3-658-11635-4_2 (besucht am 27. 04. 2023).

Cloudflare (2023). *What is cross-site request forgery?* Cloudflare. URL: https://www.cloudflare.com/learning/security/threats/cross-site-request-forgery/ (besucht am 29. 08. 2023).

Computerworld.ch (23. Feb. 2011). *SAPs In-Memory-Datenbank Sanssouci (HANA)*. Computerworld.ch. URL: https://www.computerworld.ch/business/forschung/saps-in-memory-datenbank-sanssouci-hana-1312143.html (besucht am 16. 08. 2023).

Dana Wilson (IBM) (2023). *Linting tools to identify and fix code mistakes - IBM Garage Practices*. IBM. URL: https://www.ibm.com/garage/method/practices/code/tool_lint/ (besucht am 28. 08. 2023).

Danielle Larocca (8. Nov. 2021). *What is SAP HCM for S/4HANA On-Premise (H4S4)? | SAP Blogs*. URL: https://blogs.sap.com/2021/11/08/what-is-sap-hcm-for-s-4hana-on-premise-h4s4/ (besucht am 16. 08. 2023).

DSAG (2020). *Leitfaden: Einsatz des ABAP Test Cockpit (ATC)*.

Ebert, Christof (2022). *Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten*. 7., überarbeitete und aktualisierte Auflage. Heidelberg: dpunkt.verlag. 461 S. ISBN: 9783969107683 9783864909191.

Feldherr, Tobias (2023). *SAP Business Technology Platform (SAP BTP)*. Mission Mobile. URL: https://mission-mobile.de/knowhow/sap-business-technology-platform/ (besucht am 31. 07. 2023).

GAMBIT Consulting GmbH (2023). *Unterschiede zwischen SAP R/3, ECC und S/4HANA | Gambit Consulting*. Unterschiede zwischen SAP R/3, ECC und S/4HANA | Gambit Consulting. URL: https://www.gambit.de/wiki/unterschiede-sap-r3-ecc-und-s4hana/ (besucht am 16. 08. 2023).

Geberit (2022a). *Company*. Geberit Group. URL: https://www.geberit.com/company/ (besucht am 10. 08. 2023).

— (2022b). *Geschäftsbericht | Geberit AG*. Geberit Geschäftsbericht 2022. URL: https://reports.geberit.com/geschaeftsbericht/2022 (besucht am 10. 08. 2023).

— (2022c). *Group history*. Geberit Group. URL: https://www.geberit.com/company/group-history/ (besucht am 10. 08. 2023).

Herczeg, Michael (2018). *Software-Ergonomie*. De gruyter studium. Boston: Walter de Gruyter. 1 S. ISBN: 9783110446869.

Herrmann, Andrea (2022). *Grundlagen der Anforderungsanalyse: standardkonformes Requirements Engineering*. Lehrbuch. Wiesbaden [Heidelberg]: Springer Vieweg. 420 S. ISBN: 9783658354602 9783658354596.

Kühnapfel, Jörg B. (2019). *Nutzwertanalysen in Marketing und Vertrieb*. 2. Auflage. essentials. Wiesbaden [Heidelberg]: Springer Gabler. 45 S. ISBN: 9783658251642 9783658251635.

— (2021). *Scoring und Nutzwertanalysen: ein Leitfaden für die Praxis*. Wiesbaden [Heidelberg]: Springer Gabler. 289 S. ISBN: 9783658348106 9783658348090.

Liggesmeyer, Peter (2009). *Software-Qualität*. Heidelberg: Spektrum Akademischer Verlag. ISBN: 9783827420565 9783827422033. DOI: 10.1007/978-3-8274-2203-3. URL: http://link.springer.com/10.1007/978-3-8274-2203-3 (besucht am 28. 08. 2023).

Menken, Moritz (11. Jan. 2016). *SAP Gateway – Die technische Grundlage für SAP Fiori*. CONET Technologie-Blog. URL: https://www.conet.de/blog/sap-gateway-grundlagen-sap-fiori/ (besucht am 16. 08. 2023).

Nagel, Michael, Christian Mieke und Stephan Teuber (2020). *Methodenhandbuch der Betriebswirtschaft*. 2., vollständig überarbeitete und erweiterte Auflage. UTB Betriebswirtschaftslehre 8564. München: UVK Verlag. 523 S. ISBN: 9783838587615 9783825287610.

OData.org (2023). *OData - the Best Way to REST*. URL: https://www.odata.org/ (besucht am 07. 08. 2023).

Oracle (2023). *Cloud-ERP vs. On-Premise-ERP*. URL: https://www.netsuite.de/portal/de/resource/articles/on-premise-cloud-erp.shtml (besucht am 14. 08. 2023).

Peter Barker (2017a). *Code Vulnerability Analyzer Checks | SAP Blogs*. URL: https://blogs.sap.com/2017/01/19/code-vulnerability-analyzer-checks/ (besucht am 08. 08. 2023).

Peter Barker (2017b). *SAP Code Vulnerability Analyzer (CVA) – FAQs | SAP Blogs*. URL: https://blogs.sap.com/2020/12/08/sap-code-vulnerability-analyzer-cva-faqs/ (besucht am 28. 08. 2023).

Peter Barker (SAP) (19. Jan. 2017). *Code Vulnerability Analyzer Checks | SAP Blogs*. URL: https://blogs.sap.com/2017/01/19/code-vulnerability-analyzer-checks/ (besucht am 16. 08. 2023).

— (2020). *Code Vulnerability Analyzer*. SAP. URL: https://www.sap.com/documents/2018/11/ec5d12c6-287d-0010-87a3-c30de2ffd8ff.html (besucht am 16. 08. 2023).

Pohl, Klaus und Chris Rupp (2021). *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. 5., überarbeitete und aktualisierte Auflage. Heidelberg: dpunkt.verlag. 241 S. ISBN: 9783969102473 9783864908149.

Rupp, Chris und SOPHIST-Gesellschaft für Innovatives Software-Engineering (2021). *Requirements-Engineering und -Management: das Handbuch für Anforderungen in jeder Situation*. 7., aktualisierte und erweiterte Auflage. München: Hanser. 584 S. ISBN: 9783446464308 9783446455870.

SAP (13. Juli 2023a). *About CAP | CAPire*. URL: https://cap.cloud.sap/docs/about/ (besucht am 16. 08. 2023).

— (2023b). *Extended Warehouse Management | WMS*. SAP. URL: https://www.sap.com/germany/products/scm/extended-warehouse-management.html (besucht am 16. 08. 2023).

— (2023c). *SAP Analytics Cloud | BI, Planning, and Predictive Analysis Tools*. SAP. URL: https://www.sap.com/products/technology-platform/cloud-analytics.html (besucht am 16. 08. 2023).

— (2023d). *SAP Data Intelligence | Features and Capabilities*. SAP. URL: https://www.sap.com/products/technology-platform/data-intelligence/features.html (besucht am 16. 08. 2023).

— (2023e). *SAP Help Portal: Cloud Connector*. URL: https://help.sap.com/docs/connectivity/sap-btp-connectivity-cf/cloud-connector (besucht am 04. 09. 2023).

— (2023f). *SAP S/4HANA Cloud, Public Edition*. SAP. URL: https://www.sap.com/germany/products/erp/s4hana.html (besucht am 16. 08. 2023).

— (2023g). *SAPUI5 SDK - Demo Kit*. URL: https://sapui5.hana.ondemand.com/ (besucht am 16. 08. 2023).

Schubert, Petra und Axel Winkelmann (2023). *Betriebswirtschaftliche Anwendungssysteme: Enterprise Resource Planning*. Wiesbaden: Springer Fachmedien. ISBN: 9783658409449 9783658409456. DOI: 10.1007/978-3-658-40945-6. URL: https://link.springer.com/10.1007/978-3-658-40945-6 (besucht am 29. 08. 2023).

Steckenborn, Tobias (2022). *Schnelleinstieg SAP Business Technology Platform (BTP) - Services und Integration*. OCLC: 1302012507. Gleichen: Espresso Tutorials. ISBN: 9783960127543.

Stefan Grieß (2022). *On-Premise, Hybrid ERP oder Cloud - Vorteile und Unterschiede*. URL: https://www.applus-erp.de/ressourcen/blog/on-premise-hybrid-erp-oder-cloud/ (besucht am 14. 08. 2023).

Vanam, Lingaiah (19. Jan. 2020). *S/4HANA Embedded Extended Warehouse Management (EWM) Overview | SAP Blogs*. URL: https://blogs.sap.com/2020/01/19/s-4hana-embedded-extended-warehouse-management-ewm-overview/ (besucht am 16. 08. 2023).