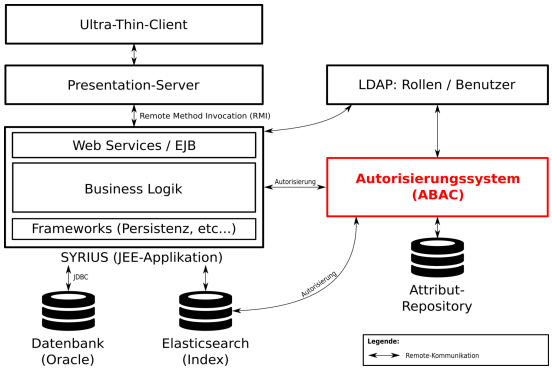
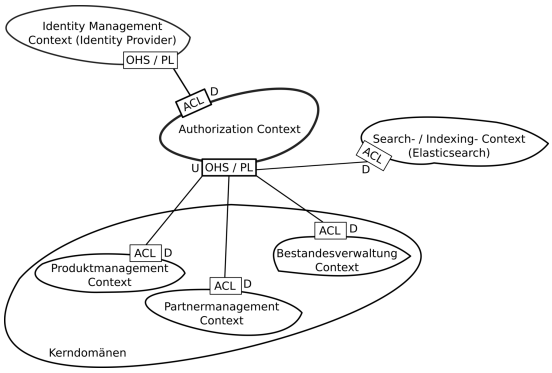
Stefan Kapferer	Studenten/-innen	Stefan Kapferer
	Dozenten/-innen	Prof. Dr. Olaf Zimmermann
	Co-Betreuer/-innen	- -
	Themengebiet	Software
	Projektpartner	Adcubum AG , St. Gallen , SG

Architectural Refactoring der Data Access Security

Attribut-basierte Zugriffskontrolle in einem verteilten Software-System



SYRIUS Systemlandschaft mit ext. Autorisierungssystem



«Domain-Driven Design» Context Map

Ausgangslage: In adcubum SYRIUS, einer geschichteten ERP-Lösung für Versicherungen, werden Zugriffsberechtigungen in der Datenbank gespeichert. Die Autorisierung der Daten wird bei jedem Datenzugriff in der Persistenzschicht durchgeführt. Solange nur Daten berechtigt werden müssen, welche in der eigenen Datenbank persistiert sind, ist diese Lösung genügend.

Vorgehen/Technologien: Strategisches Domain-Driven Design (DDD) ist eine Methodik, die zur Identifikation von Modulen oder Microservices dienen kann. In dieser Studienarbeit wurde ein Prototyp entwickelt, in welchem die Autorisierung einen eigenen Bounded Context nach DDD bildet. Dieser wird als eigener Microservice betrieben und von den Kerndomänen über eine Remote-Schnittstelle angesprochen. Dafür wurde im Rahmen dieser Studienarbeit eine Autorisierungsschnittstelle auf der Basis von RESTful HTTP entwickelt. Ausserdem wurde in adcubum SYRIUS ein Refactoring in der Persistenzschicht durchgeführt, um die alte Berechtigungslösung zu extrahieren und die neue Autorisierungsschnittstelle aufzurufen. Desweiteren wurde das Attribute-based Access Control (ABAC) Paradigma analysiert, welches die Wartung und Konfiguration der heutigen RBAC-Lösung vereinfachen soll. Mittels einer Mock-Implementation des Autorisierungssystems und dessen Schnittstelle wurde schliesslich die Umsetzbarkeit des Konzepts einer externen Autorisierungslösung überprüft.

Ergebnis: Für den Industriepartner Adcubum hat diese Studienarbeit das Wissen erarbeitet, an welchen Stellen in SYRIUS die Autorisierungsschnittstelle aufgerufen werden muss. Ausserdem dokumentiert die RESTful HTTP-Schnittstellendefinition, welche Daten dem Autorisierungssystem übergeben werden müssen. Aus den Überlegungen, wie das Zugriffskontrollmodell von RBAC auf ABAC umzustellen ist, wurde ein templatekonformes, wiederverwendbares Architectural Refactoring abgeleitet.

Architectural Refactoring: Change Access Control Model From RBAC To ABAC

Context:
- Logische Sicht

Stakeholder concerns and quality attributes:
- Flexibilität in der Gestaltung der Zugriffsregeln.
- Geringe Kopplung zwischen Benutzer und zu schützendem Objekt.

Architectural smell:
- Um die nötige Flexibilität zu erreichen, müssen häufig künstliche Pseudo-Rollen erstellt werden, was oft zu einer *Role Explosion* führt.
- Die starke Kopplung zwischen den Objekten und den Rollen bzw. Benutzern, führen dazu dass die Implementation der Autorisierung und die Businesslogik ebenfalls eine hohe Kopplung aufweisen.

Architectural decision(s):
- Wahl eines Zugriffkontrollparadigma's (ABAC-Paradigma).
- Wahl oder Implementation einer Autorisierungsschnittstelle welche das Paradigma unterstützt.
- Wahl oder Implementation eines Autorisierungssystems.

Initial position:
In einem Softwaresystem welches mit Rollen zur Autorisierung arbeitet (RBAC) ist die Granularität der Rollen, aufgrund der fehlenden Flexibilität des Zugriffkontrollmodels, nicht mehr adäquat um eine genügend feingranulare Autorisierung umzusetzen. Ausserdem referenzieren die Rollen und Benutzer *statisch* die zu schützenden Objekte innerhalb der Persistenz der Applikation.

Revised Design:
Ziel dieses Architectural Refactoring ist es, die Gestaltung der Autorisierungsregeln flexibler zu gestalten und dabei die Kopplung zwischen den Benutzern und deren Rollen und zu schützenden Objekten zu verringern. Die Regeln werden über Attribute definiert. Dabei ist ein Attribut ein beliebiges Key-Value-Pair und kann vom Benutzer, dem zu schützenden Objekt, oder auch vom System stammen. Die Autorisierungsregeln definieren mit welchen Werten die Attribute der Benutzer, Objekte und Systeme belegt sein müssen, damit ein Zugriff gewährt wird oder nicht. Dadurch können beliebig flexible Regeln definiert werden und es besteht keine statische Verbindung zwischen dem Benutzer und dem zu schützenden Objekt.

Pitfalls to avoid:
Die Flexibilität welche ABAC mit sich bringt, kann ebenfalls zu einer unübersichtlichen Anzahl und Komplexität von Regeln führen. Das *Attribute Engineering* muss wohlüberlegt durchgeführt werden, damit die neu gewonnene Flexibilität nicht zu einem potentiellen «Chaos» von komplexen Regeln führt.

Affected architectural elements:
- Autorisierungskomponente
- Applikations-Schicht (Datenzugriffsschicht) welche Autorisierungsaufruf durchführt.
- Weitere Systeme aus welchen das Autorisierungssystem Daten bezieht, wie zum Beispiel das «Identity Management» System.

Execution tasks:
- Implementiere ABAC-basiertes Autorisierungssystem, oder verwende ein Standard-Produkt.
- Entwerfe eine Remote-Schnittstelle, um Autorisierungsrequests durchführen zu können.
- Refaktoriere die Datenzugriffsschicht in welcher die Autorisierung der Daten durchgeführt wird. Rufe dort die Remote-Schnittstelle des ABAC-basierten Autorisierungssystems auf und filtere anschliessend die Daten auf welcher der Benutzer keinen Zugriff erhalten hat.
- Migriere die bestehenden Rollen zu Attribut-basierten Regeln.
- Prüfe ob Design-Ziele und Requirements mit dem Architectural Refactoring erreicht wurden.
- Prüfe ob Performance- und Audit-Anforderungen erfüllt werden.

Architectural Refactoring «From RBAC to ABAC» nach dem Template von Prof. Dr. O. Zimmermann.