2 Ziel und Zweck

2.1 Ausgangslage

Momentan gibt es, wie in der Einleitung beschrieben, für das WSCAR ein einfaches Support- Ticket-System, daswelches rudimentärsten Ansprüchen genügt und auf Win32- Delphi basiert. Ein wichtiges strategisches Ziel der Firma besteht darin, den derzeit verwendeten Delphi-Technologiestack in den nächsten Jahren durch denmit dem .NET- Technologiestack abzulösen. In einer vorhergehenden Bachelor-Diplomarbeit wurde darum ein Prototyp für ein Service-Desk Modul auf Basis von .NET entwickelt. Dieser Prototyp wurde nach dem Modellierungsprinzip des DDD erstellt. Dabei war es möglich, Tickets zu erstellen und diesen einfache Attribute wie Titel oder ein Fälligkeitsdatum zuzuordnen. Das Ticket wurde dann in einer relationalen In-Memory Datenbank (Firebird) abgespeichert. Welche Ticketfunktionen und Attribute bereits implementiert wurden, ist im Dokument [ANFORDERUNGEN] detailliert festgehalten.

Eine wichtige Anforderung an das Service-Desk Modul besteht darinist, dass alle Änderungen eines Tickets festgehalten werden und einsehbar sind. Es soll nachvollziehbar sein, wer wann und was an einem Ticket gemacht hat. An diesem Problem knüpft diese Arbeit an und willmöchte dieses Problemstellung lösen.

2.2 Problemstellung

In einem ersten Schritt sollen die zukünftig geforderten Funktionen für das SDM aufgenommen werden und weiter ausgearbeitet werden. Wie unterin der Ausgangslage erwähnt, gibt es dieeine Anforderung, dass jede Änderung eines Tickets persistiert werden soll, um auch im Nachhinein Analysen und Auswertungen zu erstellen zu können [besser so?]. Ebenfalls soll auch nachvollziehbar sein, wer wann welche Änderungen vorgenommen hat. Der Prototyp des SDMs[das s ist bei Abkürzungen nicht unbedingt nötig] wurde bereits mit dem Ansatz des Domain-Driven Design modelliert. Im Zusammenhang mit dieser Methodik bieten sich die zwei Architekturprinzipien CQRS und Event Sourcing an, die der Problematik der Nachvollziehbarkeit von Änderungen Abhilfe verschaffen sollen. Durch den Event- basierten Ansatz sollen ebenfalls die Flexibilität und Skalierbarkeit der Applikation verbessert werden.

Im Rahmen dieser BDA sollen die Vor- und Nachteile von CQRS und Event Sourcing analysiert werden sowie die Richtlinien für ihrenderen Einsatz festgelegt werden. Falls die Prinzipien sich als geeignet herausstellen, soll die bestehende Architektur angepasst und die Patterns umgesetzt werden. Weiter soll die Applikation später der Firma als Referenzarchitektur für neue Module auf Basis von .NET dienen. Daher soll der Fokus auf die Architektur der Software gelegt werden. Ebenfalls soll der Prototyp des SDM um zusätzliche Funktionen ergänzt werden. Die offizielle Beschreibung der Aufgabenstellung ist im Dokument [AUFGABENSTELLUNG] zu finden.

2.3 Abgrenzung

Ziel dieser Arbeit ist es nicht, ein Service-Desk System zu entwickeln, daswelches bereit ist für den produktiven Einsatz bereit ist. Der Fokus wird bewusst auf die Engineering- Aspekte gelegt, damit eine (wiederverwendbare) Architektur geschaffen wird, die für den Product- Owner auch in später zu entwickelnden Modulen wiederverwendet werden kann. Auf die dadurch zurückgestellten Anforderungen wird im Dokument [ANFORDERUNGEN] im Kapitel 4.6 genauer eingegangen.

2.4 Vorgehen

Um dasdie gestellte Problemstellung optimal zu lösen, wird die Aufgabenstellung wie folgt angegangen:

Zu Beginn werden zusammen mit dem Supportleiter und dem Chef der Firma die Anforderungen an das SDM festgehalten. Die Anforderungen werden in Muss, Soll und Wunsch-Anforderungen eingeteilt. Die festgehaltenen Anforderungen sind im Dokument [ANFORDERUNGEN] aufgeführt.

Parallel wird dazu eine Literaturrecherche durchgeführt. Diese legt den Hauptfokus auf Informationen über DDD, CQRS, Event Sourcing und Hexagonale Architektur. Weiter werden Frameworks festgehalten, die für die Entwicklung hilfreich sein könnten, festgehalten, wie auch Code Beispiele für Event Sourcing und im speziellen dern Event Store angeschaut. Die jeweiligen Quellen werden dann nach Umfang und Nutzen beurteilt. Die gesamte Liste ist im Dokument [LITRECH] zu finden.

Mit der Literaturrecherche einher geht ebenfalls das Einlesen in die Thematik. Dafür wird einerseitszum einen die analysierte Literaturliste zur Hilfe genommen, andererseitszum anderen wird vor allem ein bestimmtes Buch für die Modellierung mit Domain-Driven Design hinzugezogen. Dies ist Vaughn Vernon's „Implementing Domain- Driven Design".

Anschliessend wird die Systemidee ausgearbeitet, diewelche nachfolgend im Kapitel 2.5 beschrieben und in der [SYSSPEZ] noch detaillierter ausgearbeitet ist. Die Lösungsstrategie wird hier aufgezeigt und die Architekturpatterns wie Event Sourcing und CQRS sowie ihr Einsatz im SDM werden im Detail erklärt. Dabei wird nach dem Ansatz von DDD vorgegangen.

[…]

Nachdem die Systemidee entwickelt ist, geht es an die Umsetzung des geplanten Systems. Zuerst werden der Event Store sowie CQRS implementiert. Falls sich herausstellen sollte, dass sich die Architekturpatterns nicht eignen für die Applikation nicht eignen, wäre immer noch genügend Zeit vorhanden, um nach einer Alternative zu suchen. Danach werden die Muss--Anforderungen umgesetzt. Falls noch genügend Zeit vorhanden ist, können auchnoch Soll- bzw. Wunsch-Anforderungen implementiert werden. Welche Anforderungen effektiv umgesetzt werden konnten, ist im Dokument [ANFORDERUNGEN] festgehalten.

Weiter werden in der letzten Phase die Business Aspekte von DDD, Event Sourcing und CQRS im Dokument [BUSINESSASPEKTE] untersucht. Diese werden am Ende der Arbeit festgehalten, weil der Erfahrungswert mit diesen Themen zu diesem Zeitpunkt grösser ist als am Anfang. Weiter wird dieser Bericht verfasst, der alswie ein Leitfaden durch das Projekt führen soll.