Die zentrale Modellierung von Domainmodellen ist sehr verbreitet in der Entwicklung von großen Software-Projekten und zentraler Bestandteil von Product Line Engineering. Sie fällt damit in den Bereich der Model Driven Software Development (MDSD). Dabei stellt die Modellierung des Domainmodells einen Kompromiss zwischen der kompletten Modellierung einer Software und der klassischen Entwicklung ohne Modelle dar.
Ziel dieses Kompromisses ist die Effizienz und Sicherheit der Codegenerierung für das Datenmodell einzusetzen, um die Entwicklung der restlichen Software zu vereinfachen. Bevor wir in Kapitel 0.3 die Anforderungen an ein flexibles Framework zur Generierung von Datenmodellen beschreiben, werden wir zunächst grundlegende Themen behandeln.

0.1 Model Driven Software Development (MDSD)

Modellgetriebene Softwareentwicklung (Model Driven Software Development, MDSD) ist ein Oberbegriff für Techniken, die aus formalen Modellen automatisiert lauffähige Software erzeugen. [**modellbuch**]

Zu diesem Oberbegriff gehören Frameworks mit unterschiedlicher Granularität. Komplett modellierte Projekte nutzen grafische Modellierung für Daten, Prozesse und Schnittstellen. Hierfür werden häufig UML-Diagramme genutzt. Die UML Spezifikation enthält Diagramme für Datenmodelle, Abläufe und die Strukturierung von Komponenten.
Modelle für einzelne Komponenten stellen das Gegenteil dar. Sie basieren häufig auf einer Datei für jedes Modell. Diese Dateien können auch direkt im Texteditor bearbeitet werden. Die Auswahl zwischen den Strategien basiert häufig auf den beabsichtigten Nutzer*innen. Modelle für einzelne Softwarekomponenten eigenen sich nur zur vereinfachung der Arbeit von Entwickler*innen. Grafische Modellierung hingegen eignet sich auch für Nutzer*innen mit geringer technischer Kompetenz, jedoch auch nur für eine eingeschränkte Menge aller Software.

0.1.1 Product Line Engineering (PLE)

PLE befasst sich mit der Entwicklung von mehreren verwandten Softwareprodukten. Dabei handelt es sich häufig um Software für Teilaufgaben und angepasste Kundenversionen der Standardsoftware.
Hierbei besteht für eine Organisation die Gefahr, viele Komponenten mehrfach zu entwickeln und zu verwalten. Durch gemeinsam genutzte Komponenten (Assets) wird die Entwicklung vereinfacht und die Software verhält sich beim Kunden einheitlich. Datenmodelle stellen im PLE wichtige Assets dar. Einheitliche Modelle verhindern das Übersetzen zwischen verschiedenen Produkte.

0.2 EMF

Eclipse Modelling Framework (EMF) ist ein häufig eingesetztes Framework zur Modellierung von Modellen in Java. Es lassen sich große Modelle darstellen und mithilfe von Maven Workflows können diese durch das Build Tool übersetzt werden.
EMF bietet dabei jedoch keine Wahl bei der IDE oder der Programmiersprache. Dies führt dazu, dass Projekte und ganze Firmen bei ihren bisherigen Technologien stehen bleiben. Es wird bei Neuentwicklungen nicht mehr die gefragt, was wären die besten Technologien um das Problem zu lösen, sondern es wird gefragt, wie lösen wir das mit unserer bisherigen Architektur.

0.2.1 Effekte eines unflexiblen Frameworks

Ein wenig flexibles Framework wie EMF beeinträchtigt das Projekt auf mehreren Ebenen:

1. **Konzentrierung von Wissen und Erfahrung**
   Da nur eine Architektur in Betracht gezogen wird, hat jedes Mitglied des Teams nur Erfahrung mit der aktuellen Architektur und jegliche Erfahrung mit anderen Technologien verfällt mit der Zeit. Dies schränkt die die Perspektiven auf Probleme sehr stark ein und macht einen Wechsel sehr aufwendig.
2. **sinkende Bewerberzahl**
   Da nur Bewerber für die gewählten Technologien in Betracht gezogen werden, verringert sich die Anzahl stark. Der Effekt wird verstärkt, wenn die Technologien als veraltet gelten. Eine kleinere Bewerberanzahl zwingt Unternehmen auch Bewerber, die andernfalls nicht beachtet worden wären, in Betracht zu ziehen. Dies führt zu weiteren negativen Effekten, da einige schlechte Angestellte die Produktivität vieler guter Angestellter stark senken können.
3. **Anfälligkeit gegenüber Sicherheitslücken**
   Eine starke Festlegung auf Technologien führt dazu, dass Sicherheitslücken gleich jedes Projekt betreffen. So könnten bei einem Zero-Day-Exploit direkt mehrere Schichten im “Schweizer Käse Modell” wegfallen.

Ein Zero-Day-Exploit beschreibt einen Angriffsweg, bei dem den Betreibern eines Systems keine Reaktionszeit bleibt. [**ibmZeroDay**] Diese Angriffswege nehmen verschiedene Formen an. Die Unbekanntheit der Zero-Day-Exploits bis zur Ausnutzung, stellt besondere Schwierigkeiten bei der Vorhersage dar, weshalb neue Metriken entwickelt wurden. [**wang2013k**]

Das Schweizer Käse Modell wurde durch die amerikanische Behörde FAA für die Analyse von Verkehrsunglücken entwickelt. Es bildet die Redundanzen die bei einem Unfall fehlschlagen als Schichten, durch deren Löcher ein spezifischer (Un-)Fall passt, ab. Diese Schichten existieren auch in der Software Entwicklung.[**bergeon2009swiss**]

Diese Anfälligkeit wird stark erhöht, sobald eine Technologie nicht mehr aktiv weiterentwickelt wird. Dies führt häufig dazu, dass andere Updates auch nicht genutzt werden können.

0.3 Anforderungen

Das Domain Modell Framework (DMF) soll es ermöglichen Datenmodelle zentral zu modellieren, sodass diese von verschiedenen Software-Projekten genutzt werden können. Die Grundvoraussetzung ist, dass ein Modell die Arbeit gegenüber der manuellen Implementierung erleichtert. Dabei soll die Flexibilität besonders beachtet werden, um die bisher bestehenden Nachteile zu vermeiden. Zur Flexibilität gehört die (möglichst) freie Wahl der Programmiersprache und die freie Wahl der Entwicklungsumgebung.

Konkret soll das DMF im ersten Schritt die Generierung von Java und Typescript Code aus einer Modelldatei ermöglichen.
Es soll ein Datenbankschema zur Persistierung des Modells generiert werden.
Während der Entwicklung soll die Generation automatisch mit dem Bauvorgang der Anwendung durchgeführt werden können.
Die Generationsausgabe soll auf einer allgemeinen Abstraktion basieren. Mithilfe dieser Abstraktion werden Inhalt und Aufbau im Modell festgelegt und in die generierten Dateien übersetzt.
Die Übersetzung vom Modell soll für einzelne Elemente überschrieben werden können. Dadurch sollen die Einschränkungen und Annahmen der Abstraktion umgangen werden können.
Um die Entwicklung in einem PLE Kontext zu vereinfachen, soll es die Möglichkeit geben Modelle miteinander zu vereinen und zu erweitern.
Code welcher ein erweitertes Modell nutzt soll nach der Erweiterung weiter genutzt werden können.
Die Entwicklung in verschieden Entwicklungsumgebungen soll mit den IDEs Intellij und Visual Studio Code validiert werden.