CONTEXT MAPPER

OST
Ostschweizer Fachhochschule

Whiteboard-to-Model "Compiler" miro2cml

Studienarbeit

Studiengang Informatik
OST - Ostschweizer Fachhochschulen
Campus Rapperswil-Jona

Herbstsemester 2020

Autoren:	Timothée Moos, Saskia Stillhart
Betreuer:	Prof. Dr. Olaf Zimmermann
Ansprechpartner Context Mapper:	Stefan Kapferer

Aufgabenstellung

Aufgabenstellung Studienarbeit
Timothée Moos, Saskia Stillhart

Whiteboard-to-Model "Compiler" (miro2cml)

1. Betreuer

Diese Studienarbeit wird in Zusammenarbeit mit dem Institut für Software (IFS) durchgeführt.

Betreuer HSR/OST:

Prof. Dr. Olaf Zimmermann, Institut für Software, olaf.zimmermann@ost.ch

2. Ausgangslage

miro ist eine populäre "online collaborative whiteboard platform to bring teams together, anytime, anywhere". Wichtige Anwendungskonzepte in der miro-Plattform sind Boards mit ihren Widgets sowie Board-Templates. miro wird gern für Software-Design genutzt: es gibt Board-Templates für Story Maps und User Interface Mocking, aber auch Domain Modeling, insbesondere Domain-Driven Design (DDD).

Im Rahmen eines Open Source Projekts ist am IFS eine werkzeugunterstützte Context Mapping Language (CML) für DDD entstanden. CML unterstützt taktisches und strategisches DDD. Für diese plattformunabhängige, technikneutrale Sprache existieren zwar Editoren (Eclipse, Visual Studio Code), Validatoren sowie Generatoren z.B. für UML, aber noch keine Freiform-Visualisierungstools.

3. Ziele der Arbeit und Liefergegenstände

Es sollen noch näher zu spezifizierende Domain-Driven Design (DDD)-Boards und ihre Widgets aus miro ausgelesen und in formale, maschinenlesbare DDD-Modelle umgewandelt werden, die dann teilautomatisiert weiterentwickelt und -verarbeitet werden können. Als Zielsprache dient die CML (mit der enthaltenen Sculptor-DSL für taktisches DDD) des Context Mappers (https://contextmapper.org/).

Die folgenden Liefergegenstände (Deliverables) sollen erstellt werden:

- Mapping-Heuristiken und -Konzepte (Bsp. Name, Farbe, Form, Position, Beziehungen, Tags), deren Funktionsumfang in den ersten Projektwochen (Anforderungsanalyse) gemeinsam festgelegt wird.
- Prototyp, der die Mapping-Konzepte umsetzt.
- Technischer Bericht mit User-Tutorial und Beispielen.

Wichtige nichtfunktionale Anforderungen, die im Projektverlauf genauer erarbeitet, ergänzt, konkretisiert und dann gemeinsam festgelegt werden, sind:

- Robuster und benutzerfreundlicher Umgang mit Input-Varianz (Bsp. Korrekturvorschläge, wenn Input-Board nicht den Erwartungen der Mappings entspricht).
- Angemessener und zielgruppengerechter Abdeckungsgrad des CML-Sprachumfangs (und damit der DDD-Pattern), Bsp. User Stories, Subdomains, Aggregates mit Entities und Value Objects sowie Bounded Contexts und Ergebnisse von Event Storming Workshops (wie in Context Mapper Tutorials beschrieben).

- Domänen-Fachexperten ohne Erfahrung in CML sollen nach Einarbeitung mit Hilfe von Beispielen und ggfs. Tutorial Boards innerhalb von 5 bis 10 Minuten mappen können.
- Erweiterbar und modular im Hinblick auf neue Notationen und CML-Konzepte.
- Als Implementierungssprache soll Java verwendet werden.
- Als Lizenz soll Apache 2 verwendet werden analog zu Context-Mapper; externe direkte Abhängigkeiten müssen entsprechend kompatibel sein.

Dazu kommen weitere kritische Erfolgsfaktoren wie:

- Effiziente Nutzung in Analyse- und Design-Workshops (Ad Hoc, Ergebnissicherung).
- Robust genug, um im Beateralltag einsetzbar zu sein, d.h. insbesondere kein Datenverlust.
- Software-Engineering-Hygieneanforderungen wie Versionskontrolle, automatisiertes Testen, CI/CD (ggfs. unter Reuse der Context Mapper-Infrastruktur) sollen eingehalten werden.

4. Unterstützung

Die erwartete und effektiv erhaltene Unterstützung wird durch die Studenten in Sitzungsprotokollen definiert und im SA-Bericht dokumentiert.

5. Zur Durchführung

Mit dem Betreuer und Auftraggeber finden in der Regel wöchentlich Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf zu veranlassen. Alle Besprechungen, bei denen eine Vorbereitung durch den Betreuer nötig ist, sind von den Studenten mit einer Traktandenliste vorzubereiten. Beschlüsse sind in einem Protokoll zu dokumentieren.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. Arbeitszeiten sind zu dokumentieren. Die Zeiterfassung wird genauer im Projektplan erläutert.

Die Spezifikation der Anforderungen geschieht durch die Studenten in Absprache mit dem Betreuer. Bei Disputen entscheidet der Betreuer in Rücksprache mit den Studenten über die definitiv für die Studienarbeit relevanten Anforderungen.

Mapping- Konzept, Anforderungsdokumentation und Architekturdokumentation sollten im Laufe des Projektes mittels Milestones mit dem Betreuer in einem stabilen Zustand abgenommen werden. Die Termine für Milestones und welche Dokumente bis wann fertig zu stellen sind, werden im Projektplan definiert. Zu den abgegebenen Arbeitsresultaten wird ein vorläufiges Feedback abgegeben. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

Die Rechte an den Ergebnissen der Studienarbeit werden in einer separaten Vereinbarung definiert. (gemäss Vorlage Studiengang): die Ergebnisse der Arbeit dürfen sowohl von den Studenten wie von der HSR/OST und Prof. Zimmermann nach Abschluss der Arbeit verwendet und weiterentwickelt werden; evtl. können sie auch als Open Source Projekt veröffentlicht werden.

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Bei der Projektdokumentation und der Abgabe sind die Anleitungen des Studienganges inklusive Anhängen zu beachten.

7. Termine

Siehe HSR/OST-Webseiten. Allfällige weitere Termine sind am Sekretariat der Abteilung Informatik zu erfragen und sollten entsprechend in einem Sitzungsprotokoll dokumentiert werden.

8. Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS-Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert.

Siehe http://studien.hsr.ch/allModules/23498_M_SAI.html für die Modulbeschreibung der Studienarbeiten.

Für die Beurteilung ist der Betreuer verantwortlich unter Einbezug von User-Feedback.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Management Summary, technische u. persönliche Berichte) sowie Gliederung, Darstellung und Sprache der gesamten Dokumentation.	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 14. 09. 2020

Prof. Dr. Olaf Zimmermann
Institut für Software (IFS)
University of Applied Sciences of Eastern Switzerland (OST)

Abstract

Die vorliegende Studienarbeit beschreibt das Vorgehen, das Lösungskonzept und die Umsetzung eines Whitboard-to-Model "Compiler's". Als Whiteboard wird ein online Whiteboard Tool (Miro) verwendet. Das Model wird in der Zielsprache Context Mapper Language (CML) abgebildet.
Miro bietet diverse Vorlagen mit denen Konzepte aus dem Software Engineering modelliert werden können, wie beispielsweise User Stories. Einzelne Templates sind auch im Bereich Domain Driven Design (DDD) vorhanden. Die Brücke zwischen den Whiteboards und den analogen Abbildungen im Context Mapper bildet der miro2cml Compiler. Die Modellierer oder Domain Experts können mit dem Compiler ihre erstellten Whiteboards aus Miro in ein CML File umwandeln. Der Context Mapper bietet ihnen danach weitere Funktionen bei der Modellierung der Software Architektur.
Die Schnittmenge zwischen dem lose formatierten Whiteboard und dem streng formulierten Context Mapper Model wurde mithilfe von Mapping-Heuristiken definiert. Um die losen Inputelemente abzufangen, werden nur Whiteboards, die Templates enthalten, gemappt. Die Templates stellen ein gewisse Formatierung sicher, die dem Compiler ein sinnvolles Konvertieren nach CML erlauben. Die folgenden drei Templates werden unterstützt: User Story Map, Event Storming und The Bounded Context Canvas.

Management Summary

Ausgangslage

Seit dem Frühling 2020 wurde aufgrund der Coronapandemie in vielen Firmen das Homeoffice eingeführt. Weil das Arbeiten von Zuhause aus neue Herausforderungen mit sich bringt, wird vemehrt mit online Teamarbeit Tools gearbeitet. Eines davon ist das online Whiteboard Miro. Viele Designarbeiten im Bereich des Requirements Engineering und DDD werden neu im Miro durchgeführt.
Der Context Mapper, der am Institute for Software entwickelt wurde, ist ein modeling Framework für Strategic DDD. Das Ziel des Whitboard-to-Model "Compilersïst es eine Brücke zwischen dem Context Mapper und Miro zubauen. Miro soll als Einstiegsstelle für den Context Mapper dienen um neue Kunden für den Context Mapper zu gewinnen.

Vorgehen, Technologie

Als erstes arbeiteten wir uns in Miro und den Context Mapper ein. Wir mussten herausfinden, welche Möglichkeiten die Tools uns bieten und wo es Gemeinsamkeiten gibt. Es wurde festgelegt, dass in Miro Templates verwendet werden um den Nutzerinput möglichst einzuschränken. Anhand von drei Templates wurden gemeinsam mit unserem Betreuer und dem Context Mapper Experte Stefan Kapferer Mapping-Heuristiken definiert. Die Mapping-Heuristiken stellen eine sinngemässe Konvertierung vom Whiteboard zum Model sicher.
Da Miro meist über den Webbrowser genutzt wird, haben wir uns dazu entschieden den Konverter ebenfalls als Onlinetool zur Verfügung zu stellen. Im Auftrag war vorgegeben im Backend Java zu verwendlen, deshalb nutzten wir das Spring Boot Framework in Kombination mit Thymeleaf.

Ergebnisse

Das Resultat unserer Arbeit ist eine Webapplikation mit dem miro2cml Konverter. Der Benutzer kann sich auf unserer Seite gegenüber Miro authentisieren und seine Whiteboards konvertieren. Es werden drei verschiedene Templates unterstützt: User Story Map Framework, Event Storming und The Bounded Context.
Um den Nutzer zu instruieren, gibt es für jedes Template ein Tutorial. Die Mapping-Heuristiken sind ebenfalls für den Nutzer einsehbar, damit die Konvertierungen nachvollziehbar sind. Miro2cml bietet die Funktion EducatedGuessed, welche die Templates automatisch erkennt und entsprechend konvertiert.
Die Abbildung 0.1 zeigt ein Beispiel eines Event Storming Templates in Miro. Nach der Konvertierung des Event Storming Beispiels erscheint in der Webapplikation der Context Mapper Output, in der Abbildung 0.2 zu sehen. Der orange Bereich listet die Fehler und Empfehlungen für das Inputboard auf. Im Feld Preview ist der generierte Context Mapper Output angezeigt. Im Logfile werden weitere Details zur Konvertierung angegeben. Die Applikation bietet die Möglichkeit das CML und das Logile herunterzuladen.

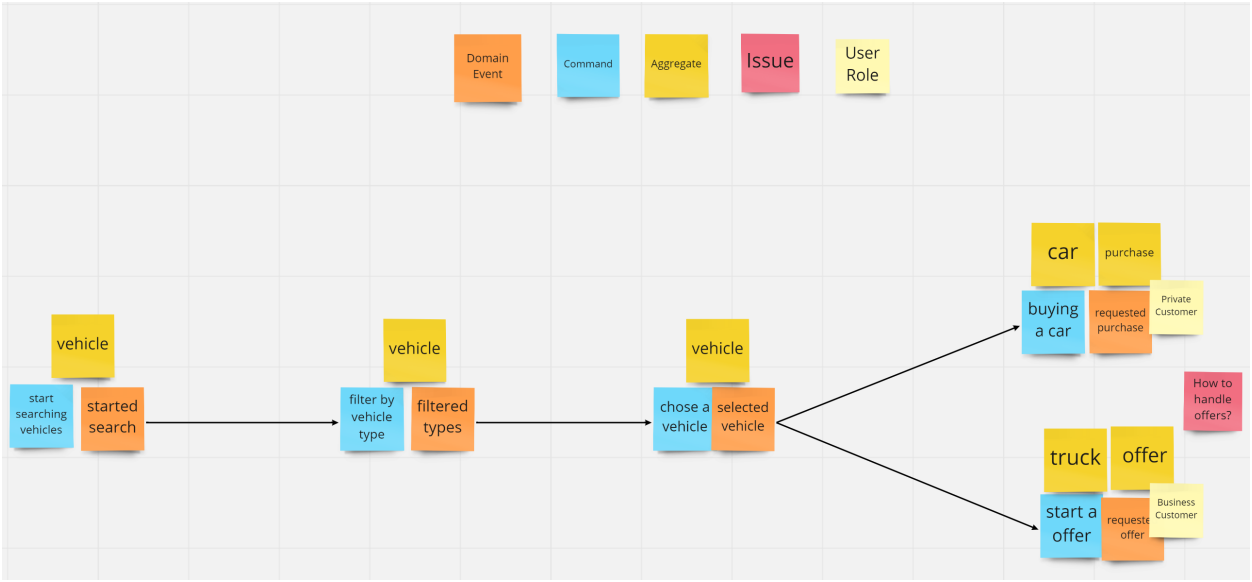


Abbildung 0.1.: Beispiel Event Storming Template in Miro

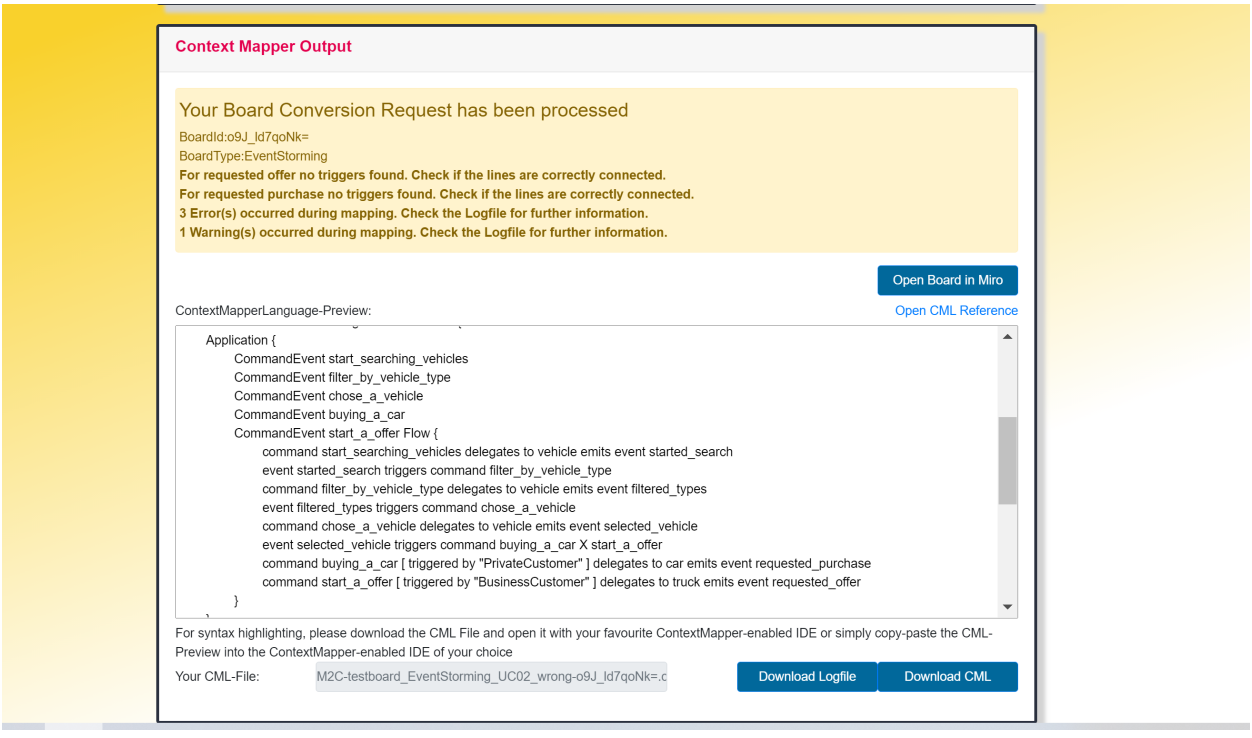


Abbildung 0.2.: Context Mapper Output von Event Storming Example

Glossar

DDD Domain Driven Design

Miro online Whiteboard Tool

CML Context Mapper Language

Widget Whiteboard Element in Miro

Sticker Widget, in Form von einem Sticky Note (Post-It)

Shape Widget, in beliebiger Form wie Rechteck, Kreis, etc.

Text Widget, Textfeld

Line Widget, in Form von Linien oder Pfeilen, kann ein Start- und Endwidget haben

Card Widget, ähnlich wie Textfeld mit Optionen zum Titel und Beschreibung setzten, sowie Anwendung von Tags

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1. Einleitung

1.1. Ausgangslage

Als Ausgangslage dienen zwei unabhängige Domänen. Zum einen die Domäne Miro. Miro ist ein online Whiteboard für kollaborative Arbeiten. Die Mission von Miro ist, einen Raum zu schaffen für gemeinsames Verständnis zwischen unterschiedlichen Rollen und Teams in einem Unternehmen, die unterschiedliche "Sprachen" sprechen[1]. Miro kann in verschiedenen Bereichen der Software Entwicklung eingesetzte werden. Beispielsweise für Event Storming Workshops oder das Erstellen von User Story Maps. Die zweite Domäne, der Context Mapper, bietet ähnliche Möglichkeiten im Bereich DDD. Der Context Mapper ist ein Modeling Tool für Strategic DDD, Bounded Context Modeling und Context Mapping [2].

1.2. Problembeschreibung

Das Ziel der Arbeit ist die semantische Lücke zwischen Miro und CML zu schliessen. Anhand der Schnittmenge soll ein Compiler entwickelt werden, der Miro Whiteboards zu CML Modellen konvertiert. Die besondere Herausforderung ist, dass in Miro Objekte nach belieben angeordnet und modelliert werden können, hingegen aber beim Context Mapper strenge semantische Regeln gelten. Um dieses Problem zu lösen, sollen Mapping-Heuristiken definiert werden, die sinngemässe Konvertierungen von Miro nach CML erlauben.

1.3. Übersicht

Der Bericht ist in drei Teile gegliedert. Als erstes wird das entwickelte Lösungskonzept erläutert. Dieser Teil beschreibt, welche Anforderungen definiert, wie die Templates bestimmt und die Mapping-Heuristiken definiert wurden. Im zweiten Teil ist die Umsetzung dokumentiert. Es wird erläutert, welche Architekturentscheide getroffen und welche Tools und Frameworks verwendet wurden. Am Ende werden die Ergebnisse vorgestellt und es wird eine Schlussfolgerung gezogen. Die jeweiligen Software-Engineering Dokumente sind im Anhang zu finden.

2. Lösungskonzept

Die folgenden Abschnitte beschreiben, welche Anfoderungen der "Compiler" erfüllen muss, weshalb Templates und welche Templates fürs Mapping evaluiert wurden und geben einen Einblick, wie wir die Mappingheuristiken spezifiziert haben.

2.1. Anfoderungen an den Compiler

Der "Compiler" soll im Bereich des User Requirements Engineering, Event Storming Workshops und bei der Modellierungsarbeiten mit DDD angewendet werden. Dadurch ergibt sich der Fokus auf die Benutzerfreundlichkeit bei den Anforderungen. Der Compiler soll schnell erlernbar sein, den Nutzer vor Inputfehler schützen und einfach zu bedienen sein. Durch die breite Anwendung des Compilers werden drei unterschiedliche Nutzergruppen angesprochen. Die Personas zu den Nutzergruppen sind im Abschnitt Personas A.4 zu finden. Die Anforderungsspezifikationen sind im Anhang A zu finden.

2.2. Verwendung von Templates

In Miro können Objekte frei gezeichnet und bearbeitet werden, dies widerspricht den strengen Regeln, die im Context Mapper für ein korrektes Model erfüllt sein müssen. Die Herausforderung, vom frei gezeichneten Whiteboard ein semantisch korrektes Context Mapper Model zu erzeugen, wird mithilfe von Templates in Miro gelöst. Miro stellt diverse Templates zur Verfügung und bietet die Möglichkeit Templates selbst zu erstellen mit Miroverse. Mit den Templates wird der Nutzer gezwungen gewisse Formate einzuhalten. Dies bietet den Vorteil, dass der Compiler Anforderungen an das Whiteboard stellen kann. Der Compiler konvertiert nur Boards, die von ihm als unterstütztes Template identifiziert werden.
Anhand der definierten Personas wurden drei bis vier Templates bestimmt, die unterstützt werden sollen. Die Templates sprechen jeweils eine Person aus den definierten Personas an. Die folgenden Abschnitte beschreiben, wie die Templates evaluiert wurden.

2.2.1. User Story Map

Der Context Mapper bietet die Möglichkeit unterschiedliche User Requirements zu definieren[3]. Miro bietet analog dazu zwei unterschiedliche User Story Map Templates an. Wir haben uns deshalb entschieden ein Mapping zu definieren, dass aus einer User Story Map die einzelnen User Stories in CML User Stories umwandelt. Diese Funktion richtet sich an die Requirements Analyst A.4.1 wie Ralf.

Template 1: User Story Map Framework

Das User Story Map Framework ist ein Raster, welches diverse Cards enthält. In den oberen beiden Zeilen sind Cards für die Activities und die Tasks reserviert und darunter folgen mehrere Zeilen mit Cards, die für die User Stories verwendet werden. Die User Stories können nach den Release Versionen geordnet werden. In der Abbildung 2.1 ist ein Beispiel eines User Story Map Framework Templates zu sehen.

- Vorteil: User Tasks und User Stories sind untereinander angeordnet, können nicht manuell verschoben werden, ansprechendes Design, Cards werden in keinem weiteren Mapping verwendet
- Nachteil: die Tags von einer Card können nicht ausgelesen werden

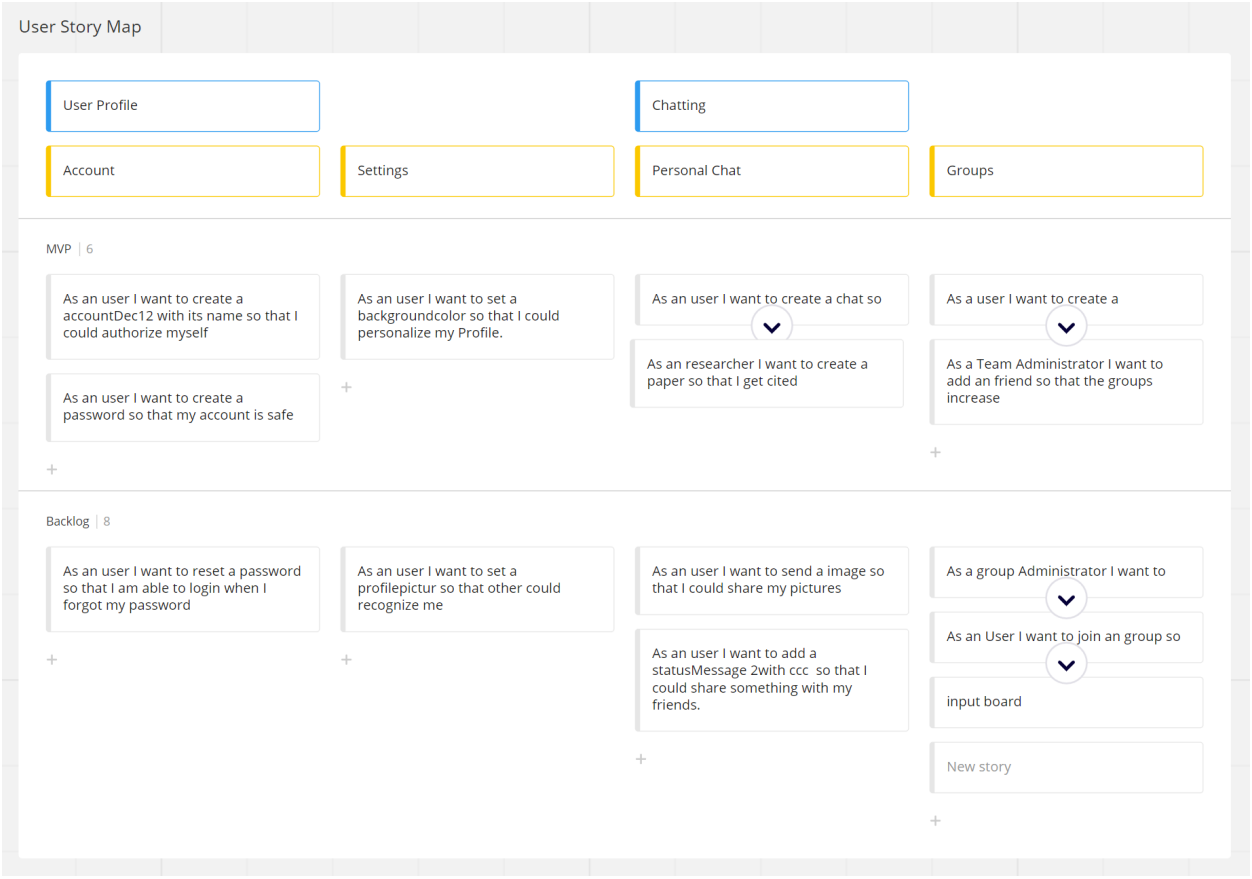


Abbildung 2.1.: Beipiel User Story Map Framework

Template 2: User Story Map

Die User Story Map ist ein Template, welches aus verschiedenen Stickers zusammengesetzt wird. Es ist ebenfalls eine Rasteranordnung, aber die Stickers können innerhalb des Boards

beliebig verschoben werden. Die oberen zwei Reihen von Stickers heben sich durch unterschiedliche Farben von den User Stories ab. Sie stellen ebenfall die Activities und Tasks dar.

- Vorteil: alle Funktionen der Stickers können ausgelesen werden, Stickers werden für Event Storming Mapping ebenfalls verwendet
- Nachteil: Stickers können einzeln verschoben werden. Vertikales auslesen von der passendem User Task zu User Story kann nicht mehr sichergestellt werden.

Weil die Funktion mit den Tags nicht notwendig/relevant ist für die UserStories und die Cards das Mapping von den anderen Templates unterscheiden lässt, haben wir uns für das User Story Map Framework entschieden.

2.2.2. Event Storming

Event Storming ist ein flexibles Workshop-Format für die kollaborative Erkundung komplexer Geschäftsdomänen [4]. Beim Event Storming wird die Domäne mithilfe von verschiedenen farbigen Stickers an einem Whiteboard modelliert. In der Abbildung 2.2 ist ein Beispiel von einem Event Storming zu sehen. Miro ist ideal geeignet für Event Storming Workshops im Online Format. Aus diesem Grund entschieden wir uns fürs Event Storming als zweites Template. Den Ursprung fürs Template war ein Beispiel Event Storming der Lakeside Mutual Application[1]. Das ursprüngliche Template haben wir auf die Bedürfnisse des Mappings angepasst und uns dabei am Cheatsheet der DDD-Crew[2] orientiert. Dieses Template soll besonders die Software Architekten A.4.2 ansprechen.

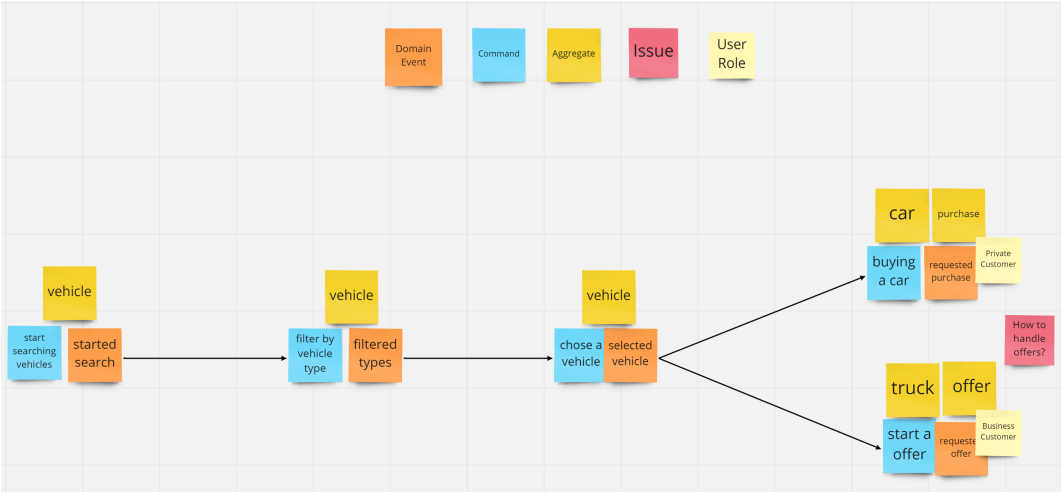


Abbildung 2.2.: Beispiel Event Storming

1 https://github.com/ContextMapper/context-mapper-examples/tree/master/src/main/cml/lakeside-mutual

2 https://github.com/ddd-crew/eventstorming-glossary-cheat-sheet

2.2.3. The Bounded Context Canvas

Bei der Suche nach Miro Templates sind wir auf ein Template der DDD-Crew getroffen, The Bounded Context Canvas[3]. Wie der Name schon sagt, wird in diesem Template ein Bounded Context sehr ausführlich modelliert. Ein Bounded Context beschreibt eine Grenze (typischerweise ein Teilsystem oder die Arbeit eines Teams) in der ein bestimmtes Modell definiert und anwendbar ist [5]. In der Abblidung 2.3 ist ein leeres Template vom Bounded Context Canvas zu sehen. Wie zu erkennen ist, können mit den Inbound und Outbound Communications die Beziehungen zu anderen Bounded Context modelliert werden. Es bieten sich ebenfalls Möglichkeiten die Strategic Classifications zu bestimmen. Dieses Template eignet sich besonders für die Domain Driven Design Experten A.4.3.

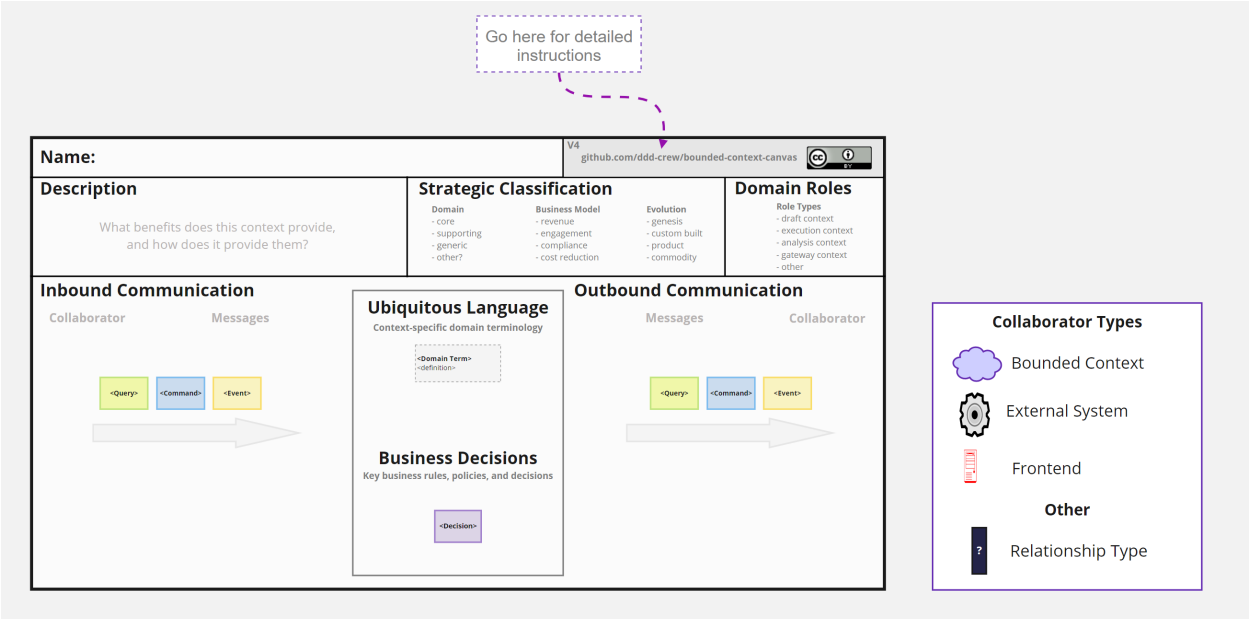


Abbildung 2.3.: Template: The Bounded Context Canvas

2.3. Mapping-Heuristiken

Die Mapping-Heuristiken haben wir anhand der drei Templates entwickelt. Als erstes wurden Gemeinsamkeiten zwischen den Boardelementen und der Context Mapper Language gesucht. Danach versuchten wir die Elemente mit gleicher Bedeutung, aber unterschiedliche Namen, wie Elemente im Context Mapper, zu identifizieren. Nachher versuchten wir die Elemente zu bestimmen, welche im Context Mapper nicht abgebildet werden können. Diese Elemente werden als Kommentare übersetzt. Als letztes haben wir noch hart kodierte Werte im Context Mapper Model bestimmt, die besonders beim Bounded Context sinnvoll waren. Die definierten Mapping-Heuristiken haben wir mit Beispielen ausgearbeitet und getestet. Über

3 `https://github.com/ddd-crew/bounded-context-canvas`

mehrere Iterationen haben wir die Mappings verbessert und allenfalls zusammen mit unserem Betreuer und Stefan Kapferer angepasst. Wenn Annahmen getroffen werden mussten, stützten wir uns immer auf den meist gebrauchten Verwendungszweck.
Besonders wichtig war, dass wir keine Logik zu den Modellen hinzufügen und lediglich eine Konvertierung machen. Die Konvertierung soll möglichst verständlich und einfach nachvollziehbar für die Nutzer sein. Ebenfalls achteten wir darauf, wie sinnvoll der Context Mapper Output für weitere Arbeiten mit dem Context Mapper Tool ist.
Im Anhang F.6 sind die detailierten Mapping-Tabellen zu finden. Im User Guide F sind Beispiele zu den Mappings und detailierte Eklärungen, wie die Templates verwendet werden sollen, definiert.

3. Umsetzung

Die folgenden zwei Abschnitte geben einen kurzen Einblick in die Software Architektur und Design mit einem Context Diagramm, und erläutern wie die Mapping von Miro zu CML umgesetzt wurden. Die detaillierten Informationen zur Software Architektur und Design sind im Anhang C zu finden, sowie die eingesetzten Technologien.

3.1. Einblick Software Architektur und Design

Die Abbildung 3.1 zeigt ein Context Diagramm, sowie der Datenfluss während einer Boardkonvertierung. Der gelbe Pfeil stellt den Benutzerinput dar. Der User wählt über die Webapplikation sein Miro Board und den Board Type. Das gewählte Board wird über die HTTP-basierte restful-API von Miro abgefragt. Der blaue Pfeil symbolisiert die API Abfrage. Die Boardelemente von Miro, mit dem grünen Pfeil dargestellt, werden in miro2cml in Plain Old Java Objects umgewandelt, konvertiert und auf ein Context Mapper Model gemappt. Der graue Spinner stellt die Konvertierung dar. Nach der Konvertierung kann mithilfe des CML Serializer das Model in ein CML File umgewandelt werden. Das CML File, rot dargestellt in der Abbildung, wird danach im Browser des Benutzers als Preview in Form von Plain Text und zum Download angeboten.

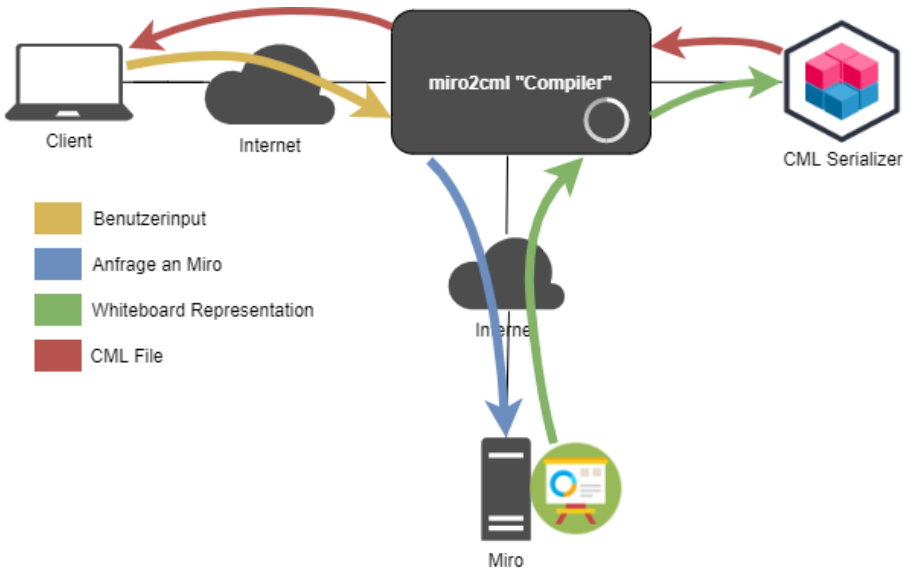


Abbildung 3.1.: Context Diagramm

3.2. Umsetzung der Mappings

In Miro gibt es fünf verschiedene Arten von Widgets: Stickers, Shapes, Texts, Lines und Cards. Je nach Art des Widgets können unterschiedliche Werte über die HTTP-basierte restful-API ausgelesen werden. Alle Widgets haben eine eindeutige Identifikationsnummer, Informationen zum Styling, sowie die Position auf dem Board. Bei den Lines kann ein Start- und Endwidget vorhanden sein. Bei den Stickers, Shapes, Texts und Cards kann der Texts ausgelesen werden. [6]
Je nach Eigenschaften der Miro Templates werden die Elemente über ihre Position, Farbe und anhand des Textinhaltes ausgelesen und gemappt. Beispiel zu den Mappings und der Verwendung der Boards sind im Anhang F.

3.2.1. Umsetzung User Story Map Framework

Bei dem User Story Map Framework Template werden hauptsächlich Cards verwendet. Ein Regex überprüft, ob die Cards dem Format F.1 einer User Story entsprechen. Die erkannten User Stories werden danach in eine CML User Story umgewandelt.

3.2.2. Umsetzung Event Storming

Beim Event Storming werden Stickers und Lines verwendet. Die einzelnen Elemente werden über ihre Farbe, die Position und den Lines zwischen den Stickers identifiziert. Bei diesem Mapping ist die Position von den Miro Widgets entscheidend. Um ein sinnvolles Mapping zu gewährleisten, muss der Nutzer diverse Vorgaben einhalten. Die Vorgaben sind im Anhang F.4.1 dokumentiert. Beispielsweise müssen die Lines exakt auf den Sticker platziert werden, damit die Verbindungen über Start- und Endwidgets ausgelesen werden können.

3.2.3. Umsetzung The Bounded Context Canvas

Beim Bounded Context Canvas werden Texts und Shapes genuzt. Es werden drei unterschiedlichen Ansätzen fürs Mapping verwendet. Zum einen das Identifizieren der Shapes über die Farbe. Als zweites die Erkennung der Texts mithilfe von Regexs über die Titel (vorgegeben vom Template) der Felder. Als letztes das Ermitteln von Texts anhand ihrer Position in Relation zu den anderen Elementen.

4. Ergebnisse & Diskussion

4.1. Ergebnisse

Es konnte gezeigt werden, dass die semantische Lücke zwischen den lose formatierten Whitboards und streng formulierten Context Mapper geschlossen werden konnte. Die User Story Map, The Bounded Context Canvas und das Event Storming können vom "Compiler"konvertiert werden, wenn die Templates gemäss den Anleitungen ausgefüllt sind.
In der Abbildung 4.1 ist das Whiteboard von einem Event Storming Beispiel zu sehen. Aus unserer Sicht das anspruchvollste Template, weil es nur sehr wenige Formatierungsvorgaben hat. Mit dem Input aus Abbildung 4.1 generiert der "Compiler"den CML Code der im Listing 4.1 zu sehen ist. Die Abbildung und das Listing illustieren, inwieweit sich die Miro-Darstellung von der mit dem Context Mapper unterscheidet und stellt die Herausforderung des "Compiler"miro2cml dar.

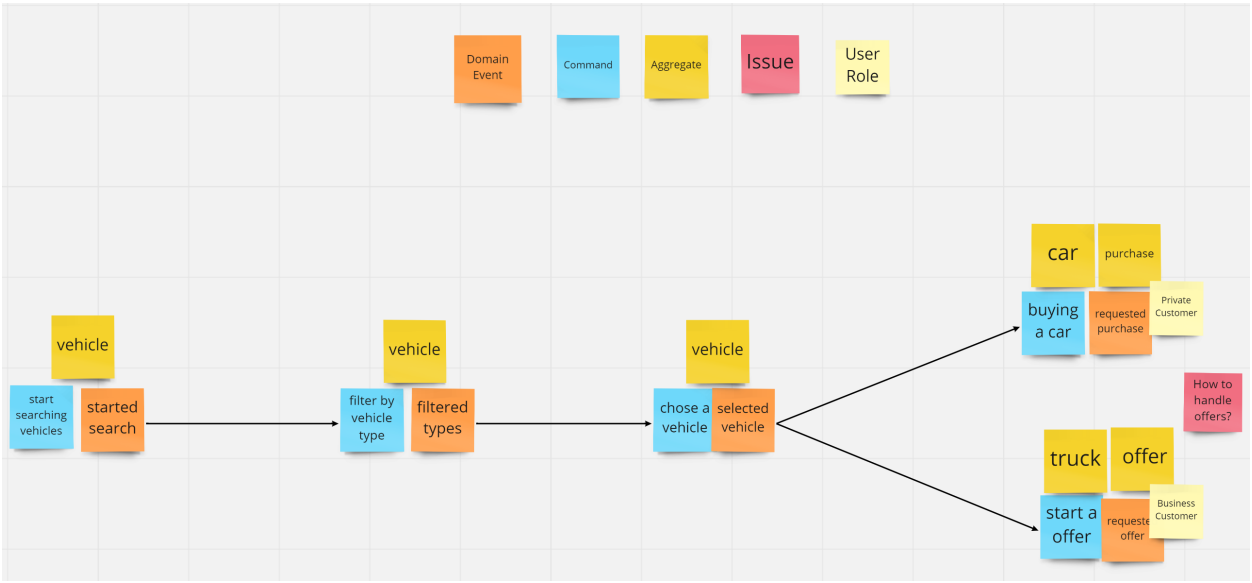


Abbildung 4.1.: Example of Event Storming Board

```
/* Issue: How to handle offers */
BoundedContext EventStormingBoundedContext {

 Application {
        CommandEvent start_searching_vehicles
        CommandEvent filter_by_vehicle_types
        CommandEvent choose_a_filter
        CommandEvent start_a_offe
        CommandEvent buying_a_car
        Flow {

        command start_searching_vehicles emits event started_search
        event started_search triggers command filter_by_vehicle_types

        command filter_by_vehicle_types emits event filtered_types
        event filtered_types triggers command choose_a_filter

        command choose_a_filter emits event selcted_vehicle
        event selcted_vehicle triggers
         command buying_a_car x start_a_offer

        command buying_a_car [triggered by "Private Customer"]
         emits event requested_purchase
        command start_a_offer [triggered by "Business Customer"]
         emits event requested_offer
  }
 }

 Aggregate vehicle {
  DomainEvent started_search
  DomainEvent filtered_types
  DomainEvent selcted_vehicle
  }
 Aggregate car {
  DomainEvent requested_purchase
  }
 Aggregate truck {
  DomainEvent requested_offer
 }
Aggregate purchase
Aggregate offer
}
```

Listing 4.1: Example CML Output for Event Storming

Die Abbildung 4.3 zeigt das User Interface nach der Konvertierung eines Boards. Im Miro Input Bereich kann der Benutzer den Boardtype wählen oder EducatedGuessed für die automatische Boarderkennung. Die Filterfunktion erlaubt eine schnelle Suche der Boards. Der Context Mapper Output zeigt ein Preview des CML-Files und bietet die Funktion das CML- und Logfile können herunterzuladen. Detaillierte Informationen zu der Konvertierung sind jeweils im Logfile enthalten. Die wichtigsten Informationen der Konvertierung sind im ersten Abschnitt des Context Mapper Outputs zusammengefasst. Falls kleine Fehler aufgetreten sind während der Konvertierung ist dieser Bereich orangen eingefärbt, wenn es keine Fehler hat grün. Falls das Board nicht richtig erkennt wurde oder Konvertiert werden konnte, ist der Bereich rot eingefärbt.
Zusätzlich zu der Konvertierung bietet die Webapplikation Tutorials zu allen drei Templates an. Damit die Benutzer nachvollziehen können, wie die Konvertierungen stattfinden, sind die Mapping-Heuristiken ebenfalls über die Webapplikation zugreifbar. In der Abbildung 4.2 ist ein Ausschnitt vom Bereich Supported Templates zu sehen.

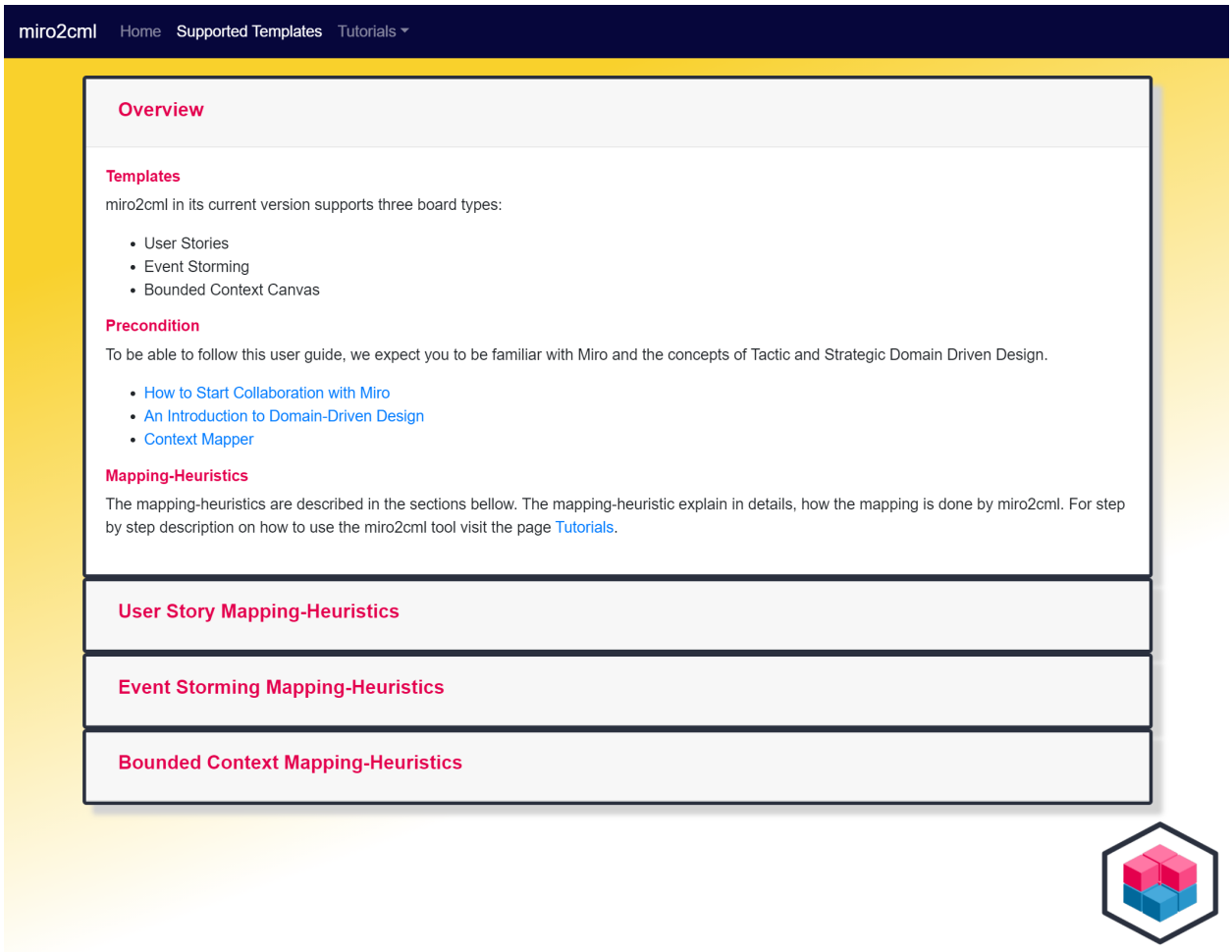


Abbildung 4.2.: User Interface Supported Templates

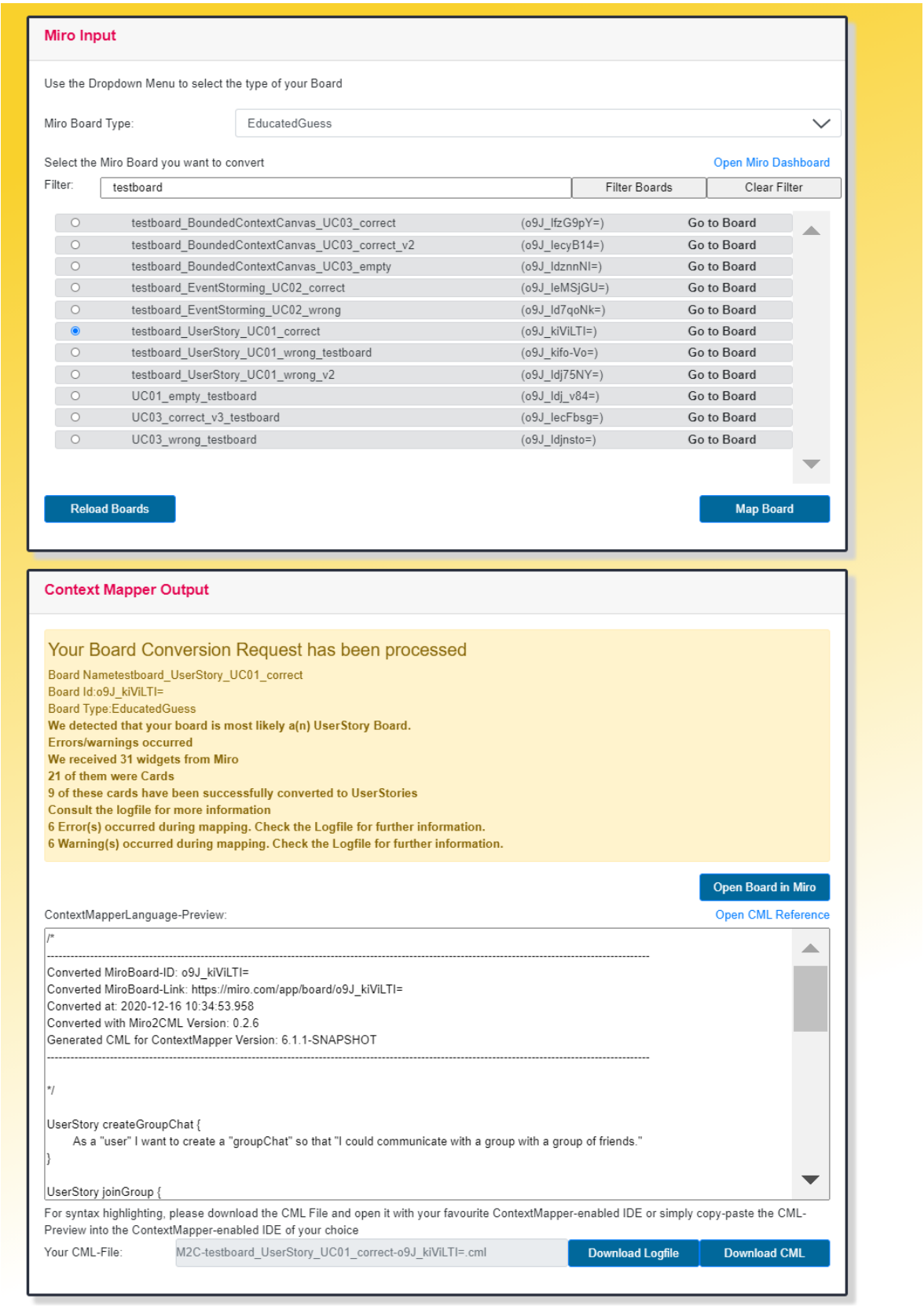


Abbildung 4.3.: User Interface nach Board Konvertierung

Wir haben drei der vier Use Cases umgesetzt, der UC04 transfer DDD Model to CML A.3.5 wurde aus Zeitgründen weggelassen. Zusätzlich zu den Use Cases wurden Landingzones A.5 bestimmt. Im Use Case 01 tranfer StoryMap to UserStory A.3.2 wurde die Landingzone Target erreicht, es können mehrere User Story Map Templates auf einem Miro Board enthalten sein und gleichzeitig konvertiert werden. Der UC02 transfer Event Storming Diagramm to CML A.3.3 und UC03 transfer Bounded Context Canvas to CML A.3.4 wurde die Landingzone Minimal erreicht. Eine vollständige Konvertierung wird nur bei einem Template pro Board sichergestellt. Die Erreichung der Nichtfunktionalen Anforderungen sind im Anhang E.4 zu finden.

4.2. Diskussion

Das Überwinden der semantischen Lücke zwischen Miro und dem Context Mapper war nur möglich mithilfe von Templates und Regeln, wie die Templates befüllt werden sollen. Dies zeigt die erste Schwäche des "Compilers" auf. Wegen den Templates sind die Freiheiten in der Modellierung, die sonst bei Whiteboards gegeben sind, deutlich eingeschränkt. Durch die Templates und Regeln schränkt sich die miro2cml Applikation im Bereich der Benutzbarkeit und Erlernbarkeit ein.
Ebenfalls gibt es Einschränkungen im Bezug auf die Robustheit, weil das Mapping teils über Textinhalte oder die Position der Elemente stattfindet. Dem Nutzer können bei diesen Inputs sehr schnell Fehler unterlaufen, welche von unserer Seite her nur bedingt auffangbar sind. Diese Schwäche wird durch konstruktive Rückmeldungen an den Benutzer abgefedert.
Des weiteren gibt es einzelne Werte, die aufgrund von Annahmen festgelegt wurden, wie beispielsweise der Feature Type des Bounded Context. Das CML-File kann zwar vom Nutzer nach der Konvertierung einfach angepasst werden, es ist jedoch ein Mehraufwand. In einer zukünftigen Version könnten man die Annahmen umgehen indem im Inputformular zusätzliche Optionen gesetzt werden können.
Die Mapping-Heuristiken wurden zusammen mit Prof. Dr. Olaf Zimmermann und Stefan Kapferer, Mitarbeiter am Institute for Software und Entwickler vom Context Mapper, ausgearbeitet. Die Heuristiken wurden mit Beispielen überprüft. Weil die gesamte Anzahl der Beispiele unter 20 ist und hauptsächlich von den gleichen zwei Personen entworfen wurden, die den "Compiler" entwickelt haben, sind die Überprüfungen der Mappings nur bedingt aussagekräftig. Für die Überprüfung der Mapping-Heuristiken wären Beispiele von Projekt unabhängigen Nutzer aus allen drei Zielgruppen geeigneter gewesen. Ebenfalls sollten die Bespiele einen grösseren Umfang und mehr Variationen haben.
Die nicht funktionale Anforderung der Benutzerfreundlichkeit zeigte sich als grosse Herausforderung. Besonders die Darstellung der Boards vom Benutzer war schwierig umzusetzen. Da ein Nutzer meistens mehr als 20 Boards hat, wird es in einer Liste sehr schnell unübersichtlich und somit ist die Benutzerfreundlichkeit nicht gegeben. Wir wollten dies mit einer vorgängigen Überprüfung der Boards umgehen und sie nach Boartypen sortieren. Dies war jedoch nicht möglich, aufgrund von Limitationen mit der Miro-Schnittstelle. Aus diesem Grund implementierten wir eine List mit einer Filterfunktion, damit den Benutzern eine Hilfestellung bei der Suche der Boards angeboten wird. Ein weiterer Punkt bei der Benutz-

barkeit war, dass der User innerhalb von möglichst wenig Klicks sein konvertiertes CML-File hat. Deshalb hatten wir uns entschieden, den Input und Output auf der gleichen Seite darzustellen. Ebenfalls entschieden wir uns dafür, dass CML-File nicht nur zum Download anzubieten, sondern auch direkt im Browser als Vorschau darzustellen. Die Filterfunktion, sowie die CML-Vorschau sind in der Abbildung 4.3 zu erkennen.

Literaturverzeichnis

[1] HTTPS://MIRO.COM/: *About Miro | Meet the team | Our mission*. https://miro.com/about/. Version: 10.12.2020

[2] KAPFERER, Stefan ; ZIMMERMANN, Olaf: Domain-specific Language and Tools for Strategic Domain-driven Design, Context Mapping and Bounded Context Modeling. In: *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*, SCITEPRESS - Science and Technology Publications, 25.02.2020 - 27.02.2020. – ISBN 978–989–758–400–8, S. 299–306

[3] CONTEXT MAPPER: *User Requirements*. https://contextmapper.org/docs/user-requirements/. Version: 08.12.2020

[4] EVENTSTORMING: *EventStorming*. https://www.eventstorming.com/. Version: 31.03.2020

[5] EVANS, Eric: *Domain-driven design: Tackling complexity in the heart of software*. Boston : Addison-Wesley, 2004. – ISBN 0132181274

[6] MIRO DEVELOPER PLATFORM: *Introduction*. https://developers.miro.com/reference#widget-types. Version: 12.12.2020

[7] *Miroverse | Miro Community Templates Gallery*. https://miro.com/miroverse/. Version: 09.12.2020

[8] *ISO/IEC 25010:2011(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. Version: 09.12.2020

[9] MIRO DEVELOPER PLATFORM: *Introduction*. https://developers.miro.com/reference. Version: 09.12.2020

[10] *OAuth 2.0 — OAuth*. https://oauth.net/2/. Version: 09.12.2020

[11] MIRO DEVELOPER PLATFORM: *Introduction*. https://developers.miro.com/reference/#fields. Version: 09.12.2020

[12] MIRO DEVELOPER PLATFORM: *Introduction*. https://developers.miro.com/reference/#get-board-widgets-1. Version: 17.12.2020

[13] Context Mapper: *CML Reference - Introduction*. https://contextmapper.org/docs/language-reference/. Version: 08.12.2020

[14] *Nick Tune's Strategic Domain-Driven Design Template | Miroverse*. https://miro.com/miroverse/category/strategy-and-planning/strategic-domain-driven-design-template/. Version: 09.12.2020

[15] Context Mapper: *Language Semantics*. https://contextmapper.org/docs/language-model/. Version: 08.12.2020

[16] Agile Alliance: *What are User Stories?* https://www.agilealliance.org/glossary/user-stories/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'user*20stories))~searchTerm~'~sort~false~sortDirection~'asc~page~1). Version: 2015

[17] GitHub: *ddd-crew/bounded-context-canvas*. https://github.com/ddd-crew/bounded-context-canvas. Version: 09.12.2020

[18] GitHub: *ddd-crew/bounded-context-canvas*. https://github.com/ddd-crew/bounded-context-canvas/blob/master/resources/bounded-context-canvas-4v-blank.jpeg. Version: 09.12.2020

[19] GitHub: *ContextMapper/context-mapper-examples*. https://github.com/ContextMapper/context-mapper-examples/tree/master/src/main/cml/lakeside-mutual. Version: 09.12.2020

[20] GitHub: *ContextMapper/context-mapper-examples*. https://github.com/ContextMapper/context-mapper-examples/tree/master/src/main/cml/insurance-example. Version: 09.12.2020

[21] Renzel, Klaus and Keller, Wolfgang: *Client/Server Architectures for Business Information Systems: A Pattern Language*

[22] @hgraca: *Documenting Software Architecture*. https://herbertograca.com/2019/08/12/documenting-software-architecture/. Version: 2019

[23] Schatten, Alexander ; Biffl, Stefan ; Demolsky, Markus ; Gostischa-Franta, Erik ; Östreicher, Thomas ; Winkler, Dietmar: *Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Heidelberg, Neckar : Spektrum Akademischer Verlag, 2010. http://dx.doi.org/10.1007/978-3-8274-2487-7. http://dx.doi.org/10.1007/978-3-8274-2487-7. – ISBN 9783827424877

[24] *The C4 model for visualising software architecture*. https://c4model.com/#ComponentDiagram. Version: 17.11.2020

[25] *org.contextmapper.dsl.contextMappingDSL (context-mapper-dsl 6.2.0 API).* `https://www.javadoc.io/doc/org.contextmapper/context-mapper-dsl/latest/org/contextmapper/dsl/contextMappingDSL/package-summary.html`. Version: 17.12.2020

[26] Miro Developer Platform: *Introduction.* `https://developers.miro.com/reference/#ratelimiting`. Version: 17.12.2020

Anhang

A. Anforderungsspezifikationen

A.1. Einführung

A.1.1. Zweck

Das vorliegende Dokument definiert die Anforderungsspezifikationen von *miro2cml*.

A.1.2. Gültigkeitsbereich

Das Dokument beschränkt sich auf die Projektdauer der Studienarbeit im HS2020/21.

A.1.3. Version

Version 1.2, 6. November 2020

A.1.4. Übersicht

- Allgemeine Beschreibung
- Use Cases
- Personas
- Nichtfunktionale Anforderungen
- Beschreibung der Schnittstellen zu Umsystemen

A.2. Allgemeine Beschreibung

A.2.1. Produkt Funktion

Das Produkt *miro2cml* soll ein Converter von Miro-Templates in CML sein. Die Mapping-Heuristiken zwischen den JSON-Daten, die aus dem Miro-Board ausgelesen werden, und dem CML Resultat, werden in einem separaten Dokument aufgeführt. Das Produkt soll von Software Architekten und Requirements Engineers, die noch keine Erfahrungen mit dem Context Mapper haben innerhalb von 5-10 Minuten erlernbar sein. Auch sollte es robust genug sein, um im Berateralltag einsetzbar zu sein, sowie Anwendung in Design- und Analyse-Workshops finden können.

A.2.2. Benutzer Charakteristiken

miro2cml richtet sich besonders an Software Architekten und Requirements Engineers, die mit Miro arbeiten und ihre Entwürfe und Modelle in den Context Mapper konvertieren möchten. Mehr dazu ist im Abschnitt Personas zu finden.

A.2.3. Projektspezifische Einschränkungen

Die Applikation *miro2cml* ist von Miro und vom Context Mapper abhängig. Besonders durch die Miro-API sind Limitation vorgegeben auf die kein Einfluss genommen werden kann.

A.3. Use Cases

A.3.1. Use Case Diagramm

In der Abbildung ist das Use Case Diagramm zu sehen. Alle vier Use Cases sind abhängig von Miro und dem Context Mapper CML.

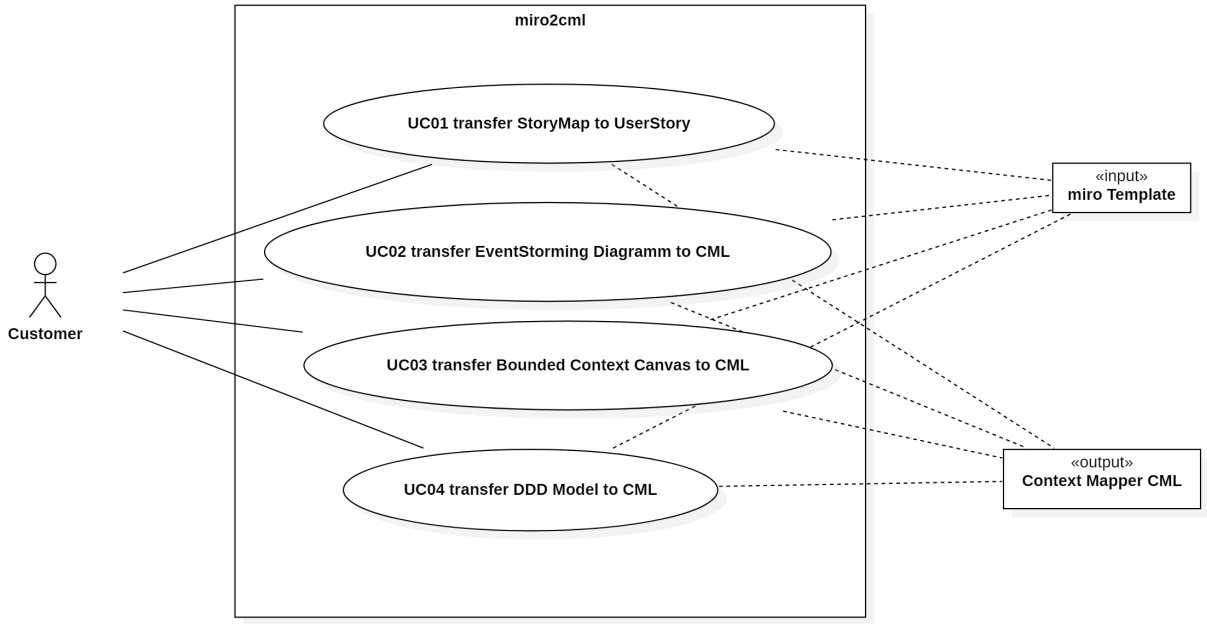


Abbildung A.1.: Use Case Diagramm

A.3.2. UC01 transfer StoryMap to UserStory

Ein Benutzer hat seine User Stories in Miro im Template Story Map eingetragen. Er will nun die User Stories in CML konvertieren, damit er im Context Mapper Domains und Subdomains erstellen kann. *miro2cml* konvertiert das ausgefüllte Template in entsprechende

User Stories in CML. Für eine vollständige Konvertierung müssen die User Stories mit dem folgendem Schema ausgefüllt werden:

```
As a <actor> I want to <action>
 a <object> so that <goal>.
```

Als Ausbaufunktion werden zusätzlich Referenzen und Attribute unterstützt mit den folgenden Schemata:

```
As a <actor> I want to <action> a
<object> for a <reference> so that <goal>.
As a <actor> I want to <action> a
<object> with its <attribute, attribute> so that <goal>.
```

A.3.3. UC02 transfer Event Storming Diagramm to CML

Ein User hat in einem Design Workshop zusammen mit seinem Team ein Event Storming durchgeführt die Ergebnisse sind in einem Miro Board festgehalten. Der User möchte das Board nach CML konvertieren um von den Architektur Refactorings des Context Mappers profitieren zu können. Das Event Storming Template existiert noch nicht, wird aber vom Institute For Software Rapperswil zur Verfügung gestellt.

A.3.4. UC03 transfer Bounded Context Canvas to CML

Ein Benutzer arbeitet mit dem Miro Strategic Domain-Driven Design(DDD) Template von Nick Tune[1]. Wenn der User mit der Modellierung fertig ist, möchte er die Modelle in den Context Mapper konvertieren. Der User legt besonderen Wert darauf, dass das Bounded Context Canvas[2] in CML konvertiert wird. Im Context Mapper möchte er weitere Konzepte des DDD anwenden, wie beispielsweise Service Cutting.

A.3.5. UC04 transfer DDD Model to CML

Ein User erarbeitete sich ein grafisches DDD Modell mithilfe von verschiedenen Widgets in Miro. Das Modell enthält eine ContextMap mit mehreren Bounded Context. Die Beziehungen zwischen den Bounded Contexts sind ebenfalls dargestellt. Er möchte seine Grafik in CML ausdrücken, damit er es mit seinen Mitarbeiter diskutieren und weiter ausarbeiten kann. Zu diesem UseCase wird zu einem späteren Zeitpunkt ein entsprechendes Miro-Template definiert in Miroverse[3]. Miroverse ist ein Markplatz um eigene Miro-Templates zu erstellen[7].

1 https://miro.com/miroverse/category/mapping-and-diagramming/strategic-domain-driven-design-template/
2 https://github.com/ddd-crew/bounded-context-canvas
3 https://miro.com/miroverse/

A.4. Personas

A.4.1. Requirements Analyst Ralf

Ralf ist seit fünf Jahren als Requirements Analyst tätig. Er ist besonders gut im Umgang mit den Kunden und dessen Erwartungen. Weil die Kunden häufig online Sitzungen bevorzugen, definiert er die Requirements in Miro denn so kann das Miro-Board den Kunden geteilt werden. Der Software Architekt möchte jedoch immer die Anforderungen in textueller Form von Ralf. Er ist gezwungen die grafischen Darstellung von Hand in ein Word-Dokument zu konvertieren. Er ist auf der Suche nach einer gemeinsamen Schnittstelle für den Software Architekten und ihn, die ihm weniger Zeit kostet. Ralf ist technisch nicht sehr versiert, deshalb benötigt er ein Tool das einfach zu verstehen und simpel in der Handhabung ist. Ralf bevorzugt Webtools, weil sie meist nur eine kurze Einarbeitungszeit benötigen.

A.4.2. Software Architect Hannah

Hannah ist Software Architektin mit jahrelanger Erfahrung. Sie wird häufig zu Analyse Workshops eingeladen, da sie durch ihre Erfahrung viele wertvolle Inputs geben kann. Besonders häufig werden Techniken des Domain Driven Design angewendet wie beispielsweise Event Storming. Sie findet es jedoch schade, dass die Grafiken, die in den Workshops entstehen nicht direkt weiterverarbeiten kann.

A.4.3. Domain Driven Design Expert Bob

Bob ist Experte im Domain Driven Design. Er ist massgebend bei der Entwicklung der Konzepte von DDD beteiligt. Er hat in Miro bereits eigene Templates definiert, mit denen Techniken von DDD umgesetzt werden können. Er ist daran interessiert, dass DDD weiter verbreitet wird in der Kombination mit Miro. Er möchte zusätzlich zu den Grafiken eine schriftliche Dokumentation verwenden in Form von CML-Dateien. Idealerweise kann er seine Darstellungen automatisiert in CML umwandeln lassen. Bob präferiert im Allgemeinen Command Line Tools, da er nicht gerne mit Web-UI arbeitet.

A.5. Weitere Anforderungen

Qualitätsmerkmale

Die Zusammenstellung der Anforderungen der Qualitätsmerkmale basiert auf dem Produktqualitätsmodel aus ISO/IEC 25010:2011 [8]

Funktionale Stabilität

- Funktionale Vollständigkeit: UseCases UC01-UC3 von diesem Dokument werden umgesetzt. Bei genügend Kapazität soll zusätzlich UC04 umgesetzt werden. Für alle Use-

Cases wurden zusätzlich drei Landingzones definiert, wie sie in der Abbildung zu sehen sind.

Landing Zone	Anforderung an Miro Board
Minimal Goal	Die Konvertierung vom Miro Board nach CML wird nur unterstützt, wenn auf dem Board nur ein Template vorhanden ist.
Target	Bei der Konvertierung können mehrere gleiche Templates auf einem Miro Board in ein CML File konvertiert werden.
Outstanding	Es können mehrere Miro Boards gleichzeitig zu einem CML File konvertiert werden. Die Boards müssen jedoch die gleichen Templates verwenden.

- Funktionale Korrektheit: Die Software soll für korrekt formatierte Miro-Boards (genaue Formastspezifikation werden im Usermanual definiert) als Input einen semantisch und syntaktisch korrekten CML-Output generieren. Die Korrektheit wird anhand mindestens einem Beispielboard pro unterstütztem Board-Type überprüft.

Performance

Es soll eine möglichst kurze Verarbeitungszeit für die Miro-Board zu CML Konvertierungen angestrebt werden. Eine Konvertierung für ein Board mit maximal 50 Miro-Widgets soll maximal 5 Minuten benötigen.

Benutzerfreundlichkeit

- Nutzerfehlerschutz: Der Nutzer wird bei Erkennung von falsch formatiertem Input innerhalb von maximal 30 Sekunden nach Start des Konvertierungsvorgangs gewarnt.
- Bedienbarkeit: Das UserInterface soll simpel und übersichtlich gehalten werden um dem Nutzer eine intuitive und einfache Bedienung zu ermöglichen, sodass sich Nutzer (ensprechend Personas) in jedem Teil der Benutzeroberfläche in maximal 3 Minuten zurecht finden.
- Erlernbarkeit: Die Software soll ohne Vorkenntnisse in CML innerhalb von maximal 10 Minuten in ihren Grundfunktionen erlernbar sein, sodass bereits mindestens das erste Board konvertiert werden kann.

Sicherheit

Es sollen keine Miro-Useraccountdaten (auch Access-Token) oder Inhalte von Miro-Boards über unverschlüsselte Kanäle transportiert werden.

Portabilität

Die Software soll auf den Betriebssystemen Ubuntu 20.04 und Windows 10 einsetzbar sein (gegebenenfalls unter Anwendung von weiteren Softwarekomponenten (zum Beispiel Webbrowser)).

Modularität

Die Software soll modular aufgebaut werden und in einzelne logische Komponente aufgetrennt sein, sodass die Software einfach und mit minimalem Aufwand auf neue CML-Konzepte und Notationen erweitert werden kann. Zur Erreichung dieses Qualitätsmerkmals werden verschiedener möglichen Änderungsszenarien definiert und eine maximale Bearbeitungszeit beziehungsweise Umsetzungszeit in Entwicklungsstunden angegeben.

Szenario	**maximale Bearbeitungszeit[h]**
Es gibt eine kleine Veränderung eines existierenden Board-Templates	8
Es soll ein neuer Bordtyp, welcher eine ähnliche Komplexität wie ein bereits implementierter Boardtyp hat, unterstützt werden. Wobei beide Boardtypen das gleiche CML-Feature abbilden.	20
Es soll die Unterstützung für ein neues CML Feature implementiert werden	40
Es soll eine neue Zielsprache, welche ähnliche Grundvoraussetzungen (im Bezug Verwendbare Codeteile und Schnittstellen) mitbringt, generiert werden	40
Es soll eine neue Datenquelle, welche ähnliche Grundvoraussetzungen wie Miro (Restful-HTTP Schnittstelle mit ähnlicher Komplexität, ähnliches Datenschema) mitbringt, verwendet werden.	40

Die Codebasis soll entsprechend dieser Änderungsszenarien strukturiert sein, sodass die genannten Zielzeiten realistisch umsetzbar sind. Zur Sicherstellung dieses Qualitätsmerkmals wird der Ansatz des **Codereviews während des Projektverlaufs** herangezogen. So werden Gegenseite Code-Reviews fortlaufend durchgeführt und ein externes Codereview von Stefan Kapferer eingeholt. Dafür wird zu Ende der Semesterwoche 9 die dann vorhandene Code-Basis an Stefan Kapferer übergeben. Das Feedback dieses Code-Reviews wird dann umgesetzt. Je nach Bedarf kann dann danach eine weitere Feedback-Umsetzung Iteration eingeleitet werden.

A.5.1. NFR Spezifikation

Einzelne Qualitätsmerkmale sollen anhand des Quality Attribute Scenario Templates weiter spezifiziert werden.

Scenario für miro2cml, NFR 1		
Scenario(s)		Inputvarianz
Business Goals(s)		erhöhe Usability und somit die Zufriedenheit der User indem die Applikation Robust auf unerwarteten Input reagiert.
Relevant Quality Attributes(s)		Inputvarianz, Applikationsstabilität
Scenario Components	Stimulus	Convertion Prozess gestartet für ein Miro-Board dessen Inhalt nicht dem erwartetem Format entspricht.
	Stimulus Source	Enduser
	Environment	Zur Laufzeit, normale Betriebsbedingungen, keine erhöhte Last.
	Artifact	Alle Komponenten und Schnittstellen auf den verschiedenen logischen Layern, welche benötigt werden zur Verarbeitung des Miro-Boards
	Response	User wird über nicht konvertierbaren Input informiert und bekommen die Möglichkeit Korrekturen vorzunehmen. Nicht direkt betroffene Teile des Boards werden trotzdem konvertiert.
	Response Measure	User wird über Inputvarianz informiert.
Questions		Wie schnell kann Inputvarianz festgestellt werden?
Issues		Responsetime von Miro-API ist nicht garantiert.

Scenario für miro2cml, NFR 2		
Scenario(s) Business Goals(s) Relevant Quality Attributes(s)		Erstnutzung des Services Nutzerzufriedenheit durch einfaches erlernen der Nutzung und rasches erzielen von Erfolgen. Erlernbarkeit , (Bedienbarkeit)
Scenario Components	Stimulus	Nutzer navigiert durch das Programm mit der Intention ein einfaches Miro-Board zu CML konvertieren zu lassen.
	Stimulus Source	Neuer Enduser (Persona: Requirements Analyst Ralf)
	Environment	Zur Laufzeit, normale Betriebsbedingungen, keine erhöhte Last.
	Artifact	Alle Komponenten und Schnittstellen auf den verschiedenen logischen Layern, welche benötigt werden zur Verarbeitung des Miro-Boards
	Response	User erlernt den Umgang mit dem Service und konvertiert sein erstes Miro-Board zu CML
	Response Measure	User durchläuft innerhalb von 10 Minuten die Applikation und hat am Schluss sein Miro-Board in CML umgewandelt.
Questions Issues		Zielerreichung ist abhängig von technischer Durchlaufzeit

Scenario für miro2cml, NFR 3		
Scenario(s)		Nutzung des Services
Business Goals(s)		Nutzerzufriedenheit durch einfache und intuitive Handhabung des Services.
Relevant Quality Attributes(s)		Bedienbarkeit
Scenario Components	Stimulus	Nutzer navigiert über das grafische Webinterface durch das Programm mit der Intention ein einfaches Miro-Board zu CML konvertieren zu lassen.
	Stimulus Source	Enduser (Persona: Requirements Analyst Ralf)
	Environment	Zur Laufzeit, normale Betriebsbedingungen, keine erhöhte Last.
	Artifact	Alle Komponenten und Schnittstellen auf den verschiedenen logischen Layern, welche benötigt werden zur Verarbeitung des Miro-Boards
	Response	Der User erkennt ohne Vorkenntnisse auf jedem Teil der Bedienoberfläche sowohl dessen Funktion als auch die von User erforderte Interaktion und kann diese durchführen.
	Response Measure	Der User kann für jeden Teil der Bedienoberfläche innerhalb von 3 Minuten die notwendigen Schritte erkennen und durchführen um den Verlauf der Programmnutzung voranschreiten zu lassen.
Questions		Wie verhält es sich mit den Unterschieden zwischen Personas?
Issues		

Scenario für miro2cml, NFR 4		
Scenario(s)		Durchlaufzeit der Nutzung des Services
Business Goals(s)		schnellere Durchlaufzeiten beziehungsweise kürzere Wartezeiten für den User und somit erhöhte Nutzerzufriedenheit
Relevant Quality Attributes(s)		Performance
Scenario Components	Stimulus	Convertion Prozess gestartet für ein Miro-Board dessen Inhalt korrekt formatiert ist.
	Stimulus Source	Enduser
	Environment	Zur Laufzeit, normale Betriebsbedingungen, keine erhöhte Last.
	Artifact	ein einfaches Miro Board, welches maximal 50 Widgets beinhaltet, Alle Komponenten und Schnittstellen auf den verschiedenen logischen Layern, welche benötigt werden zur Verarbeitung des Miro-Boards.
	Response	User erhält zum Input passenden CML Output.
	Response Measure	User erhält innerhalb von 5 Minuten oder weniger nach Start des Konvertierungsvorgangs die Möglichkeit den CML Output herunterzuladen.
Questions		Für UC1: keine, für andere UseCases: Wie Rechenintensiv wird Mapping?
Issues		Responsetime von Miro-API ist nicht garantiert.

Scenario für miro2cml, NFR 5		
Scenario(s)		Nutzung des Services auf unterschiedlichen Geräten
Business Goals(s)		Erweiterung der potentiellen Nutzerbasis durch Erschliessung unterschiedliche Betriebssystemnutzergruppen, Steigerung der Nutzerzufriedenheit durch Ermöglichung der Service Nutzung auf unterschiedlichen Systemen.
Relevant Quality Attributes(s)		Portabilität
Scenario Components	Stimulus	Nutzer nutzt den Service auf mehreren Geräten mit unterschiedlichen Betriebssystemen.
	Stimulus Source	Enduser
	Environment	Zur Laufzeit, normale Betriebsbedingungen, keine erhöhte Last.
	Artifact	Alle Komponenten und Schnittstellen auf den verschiedenen logischen Layern, welche benötigt werden zur Verarbeitung des Miro-Boards
	Response	Der User kann ohne viel Aufwand den Service in unterschiedlichen Umgebungen einsetzen.
	Response Measure	Der Initialisierungsaufwand für die gängigen Betriebssysteme unterscheidet sich um nicht mehr als 2 Minuten
Questions Issues		

A.6. Schnittstellen

A.6.1. Miro

miro2cml arbeitet mit JSON-Daten die mittels RESTfull-HTTP Schnittstelle von Miro[9] abgefragt werden.

Authorization

Miro Apps arbeiten mit OAuth 2.0[10]. Um API-Requests abfragen zu können, muss zuerst ein Access Token generiert werden mithilfe eines "Authorize"Links. Das Access Token kann über ein Query-Parameter oder über ein Authorization Header mitgeschickt werden. In der Abbildung A.2 ist das Sequenzdiagramm des OAuth 2.0 Flow zu sehen.

Features

Die API Calls können in sieben verschiedene Scopes eingeteilt werden, die unterschiedliche Berechtigungen und Möglichkeiten haben. *miro2cml* fokussiert sich auf den Scope boards:read. Bei diesem Scope können diverse Elemente vom Board ausgelesen werden, dies erfolgt über den lesenden Zugriff auf das Board. Jedes Objekt hat eine unique Id und ein Feld type. Miro behaltet sich vor, dass Format des Identifier ändern zu können, aber er wird

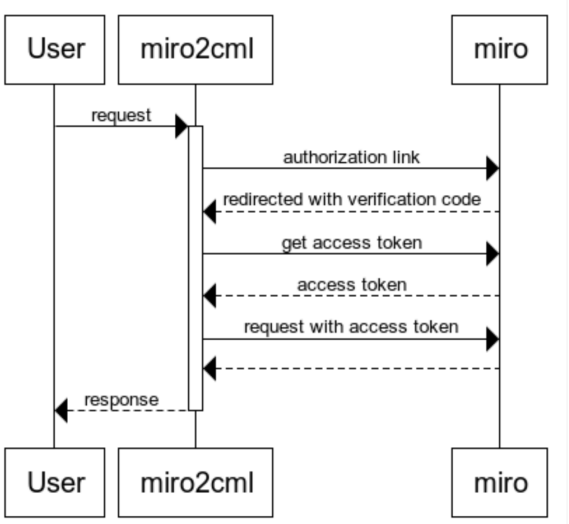


Abbildung A.2.: Sequenzdiagramm Authorization

nie grösser als 128 characters sein. Wenn ein Objekt einen spezifischen Typ hat, werden die dazugehörigen Felder mitgeliefert. Grundsätzlich gibt es für jedes Objekt ein Standard- und ein Mini-Format. Das Standard-Format wird verwendet, wenn ein einzelnes Objekt abgefragt wird und das Mini-Format, wenn ein Set von Items zurückgegeben wird.[11] Falls es bei Funktionen Abweichungen von diesem Format gibt, ist es in der Dokumentation explizit erwähnt.

Limitationen

Eine Applikation hat pro Minute und pro User 10000 Credits für API Calls. Die API Calls sind in vier Levels mit unterschiedlichen Kosten aufgeteilt, wie in der Tabelle zu sehen ist. Wenn die Anzahl Request überschritten wird, gibt es ein 429 Too many request Fehler. In jedem Response sind im HTTP Header die Felder X-RateLimit-Limit, X-RateLimit-Remaining und X-RateLimit-Reset, um den aktuellen Status der API Calls zu überprüfen.

Weight	Cost of one API Call
Level 1	50 Credists
Level 2	100 Credits
Level 3	500 Credits
Level 4	2000 Credits

Wichtige API-Calls

Die Funktion **List All Widgets** wird voraussichtlich für das Auslesen der Boards benötigt. Sie gibt die ersten 1000 Widgets von einem Board zurück. Sie benötigt als Path Parameter die Board-Id und optional den Query-Parameter widgetType, um die Widgets über den Typ zu filtern. Sie hat ein Rate Limiting von Level 3 und ist im Scope boards:read. Die Funktion

List All Widgets ist auf die ersten 1000 Elemente des Boards limitiert, daher findet keine Pagination statt. Ein Miro-Board kann aber durchaus aus mehr als 1000 Elementen bestehen, was zum Problem führt das grosse Boards nicht ausgelesen werden können [12].

A.6.2. Context Mapper

Die Applikation miro2cml hat die Zielsprache Context Mapping (CML). CML unterstützt das Modellieren von Bounded Context und dessen Beziehungen mit taktischen und strategischen DDD Patterns. Strategischen Patterns, die von der CML unterstützt werden:

- Context Map
- Bounded Context
- Subdomain (Core, Supporting, Generic)
- Domain Vision Statement
- Partnership (P)
- Shared Kernel (SK)
- Customer/Supplier (C/S)
- Open Host Service (OHS)
- Published Language (PL)
- Conformist (CF)
- Anticorruption Layer (ACL)
- Responsibility Layers
- Knowledge Level

Taktische DDD Patterns, die von der CML unterstützt werden:

- Module
- Aggregate (and Aggregate Root)
- Entity
- Service
- Value Object
- Domain Event

- Repository

Zusätzliche Features

- User Requirements: User Stories und Use Cases
- Imports: um andere CML files zu importieren

Die genaue Syntax und die semantischen Regeln der Patters sind auf der Website des Context Mappers[4] zu finden.[13]

4 https://contextmapper.org/docs/home/

B. Domainanalyse

B.1. Einführung

Das miro2cml Projekt bewegt sich in, beziehungsweise vereinigt zwei Domänen, die Miro-Domäne und die des Context Mappers. Diesen Sachverhalt gilt es zu analysieren. Die Context Mapper-Domäne ist bereits in der Onlinedokumentation des ContextMappers[1] beschrieben, infolge dessen liegt der Fokus hier auf der Miro Domäne und der Überschneidung der beiden Domänen.

B.1.1. Zweck und Ziel

Ziel der Domainanalyse ist es die Probleme der Domäne zu verstehen und sowohl Anforderungen und Möglichkeiten für Mapping identifizieren. Weiter sollen Aktoren, Aktivitäten, Regeln und des Dämonenvokabulars erkannt werden.

B.1.2. Gültigkeitsbereich

Die Gültigkeit dieser Domainanalyse beschränkt sich auf die Laufzeit der Studienarbeit während des Herbstsemesters 2020.

B.1.3. Version

Version 1.1, 4. November 2020

B.1.4. Beschreibung der Domänen

Die Miro-Domäne beschreibt das Online Tool Miro, ein Online-Whiteboard, mit dem kollaborativ gearbeitet werden kann. Ein möglicher Verwendungszweck dieses Tools liegt im Software-Engineering beziehungsweise in der Softwarearchitektur. In der nachfolgenden Analyse beschränken wir uns auf diesen Verwendungszweck. Miro ermöglicht das gemeinsame Arbeiten an Modellen, der Ausarbeitung von UseCases und so weiter. Der Fokus von Miro liegt auf der gemeinsamen Nutzung zur Erstellung von Modellen und anderen Komponenten. Die ContextMapper-Domäne beschreibt das Tool ContextMapper inklusive der dahinterliegenden ContextMapping-Language (kurz CML). Der ContextMapper ermöglicht es Domain-Driven Design Patterns zu modelieren. Diese Modelle werden mit der DSL(Domain-specific

1 `https://contextmapper.org/docs/language-model/`

Language) CML beschrieben. Der Fokus vom ContextMapper liegt in der Analyse und Weiterverarbeitung der Modelle.

B.1.5. Überschneidung der Domänen

miro2cml soll eine Einweg-Brücke zwischen den beiden Domänen, ausgehend von der Miro-Domäne und endend in der ContextMapper-Domäne, darstellen. Als gemeinsame Schnittmenge haben Sie das darzustellende Konzept, wobei die beiden Tools dieses Konzept unterschiedlich darstellen und einen unterschiedlichen Fokus mitbringen.

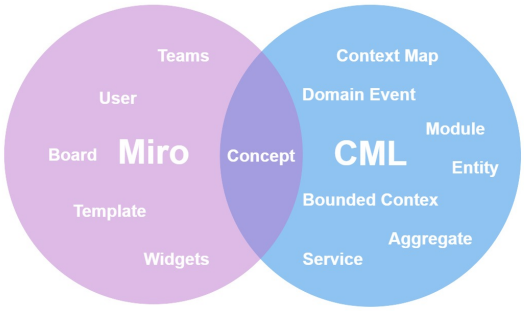


Abbildung B.1.: Venn Diagramm der Domänen

B.2. Domainmodel

B.2.1. Domain Diagramm

Es werden die wichtigsten Konzepte genauer betrachtet. Dabei wird erst die Miro-Domäne ausgehend von den Miro-User-Accounts betrachtet. Danach wird kurz die ContextMapper-Domäne angeschaut und dann die Schnittmenge der beiden Domänen, die Konzepte, analysiert.

B.2.2. Miro User Account

Der User Account bildet die Berechtigungen des Users in der Miro-Domäne ab. Ein User Account kann sich in mehreren Miro-Teams befinden und darüber zugriff auf bestimmte Boards erhalten oder auch direkten Zugriff auf Boards haben.

B.2.3. Miro Team

Miro-Teams sind Gruppierungen von Useraccounts und vereinfachen die Zusammenarbeit, da Rechte an den Boards kollektiv für ein ganzes Team definiert werden können.

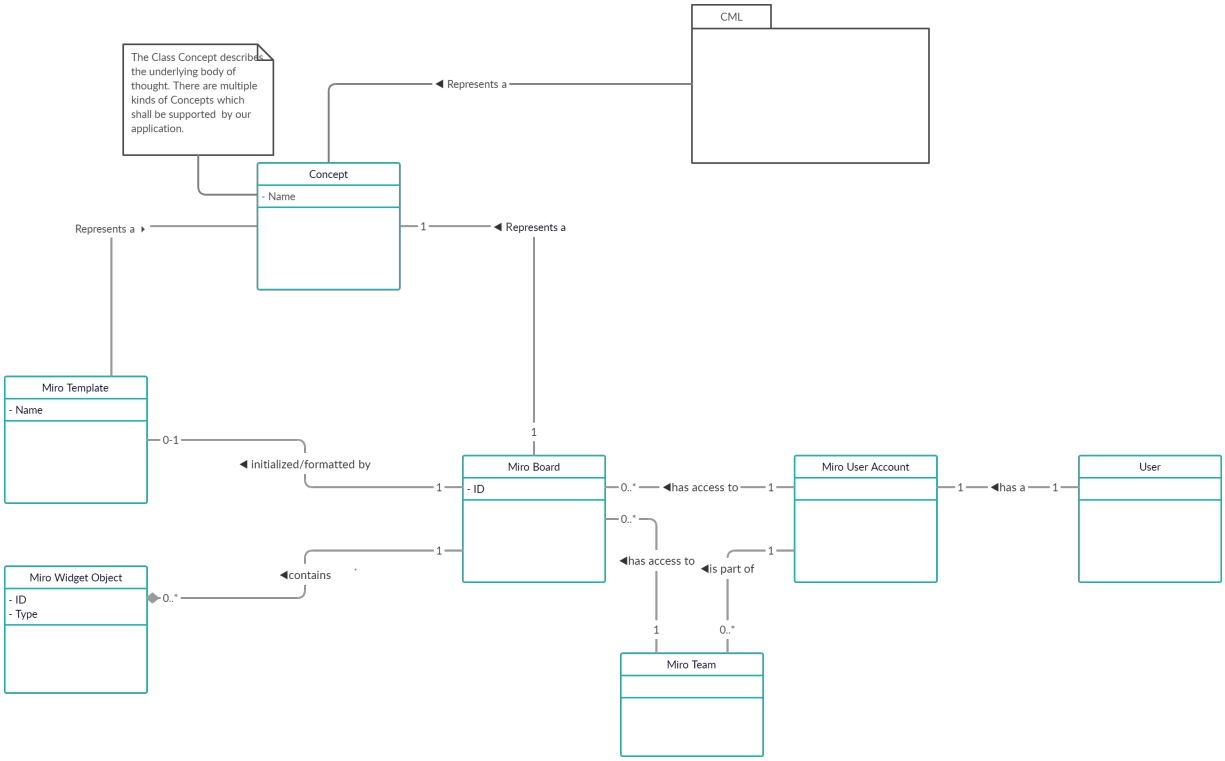


Abbildung B.2.: Domainmodell des miro2cml Projekts

B.2.4. Miro Board

Das Miro Board kann als online Version eines herkömmlichen Flipcharts gesehen werden. Es beinhaltet eine Sammlung von Miro Widget Objects, welche den Inhalt eines Boards ausmachen. Achtung: Lesender Zugriff $\neq$ Schreibender Zugriff, Miro kennt unterschiedliche Berechtigungen auf Boards für Lese- und Schreibzugriff.

B.2.5. Miro Widget Object

Miro Widget Objekte, werden benutzt um das Miro Board mit Inhalten zu befüllen. Es gibt unterschiedliche Typen von Widget Objekten, je nach Typ wird ein anderes Symbol dargestellt. Je nach Typ hat ein Objekt unterschiedliche Attribute. Die meisten Objekte haben allerdings jeweils ein Attribut für Position, Größenangaben, Text und Style. Die wichtigsten Typen (und ihre wichtigsten Eigenschaften):

- **Sticker:** Ein Sticker, stellt eine rechteckige Post-it Notiz dar.
- **Shape:** Shapes sind geometrische Formen, wie zum Beispiel Rechtecke, Dreiecke und Kreise.
- **Text:** Text-Widgets, werden dafür benötigt, um Text auch alleine, ohne die Verwendung eines anderen Widgets darstellen zu können

- **Line:** Das Line-Widget stellt eine Linie zwischen zwei anderen Widgets dar. Speziell ist hier, dass auf die Attribute für Postition, Grösse und Text verzichtet wurde. Dafür müssen zwei Widget-Ids eingetragen werden, um den Start und Endpunkt der Linie zu definieren.
- **Card:** Card-Widgets können dazu benutzt werden, einen Titel und eine Beschreibung darzustellen, diese einem Datum zuzuordnen und einer Person zuzuweisen. Dies könnte zum Beispiel dazu benutzt werden um Arbeitspakete zu definieren und zu verteilen.

B.2.6. Miro Template

Zur Nutzung der Boards können sogenannte Templates herangezogen werden. Templates sind wie eine Art Bauplan für die Anordnung der Inhalte auf einem Board und können genutzt werden um ein festest und einheitliches Format für ein Board zu definieren. Bei der Erstellung eines Boards wird der Nutzer gefragt ob er ein Template zur Initialisierung verwenden möchte. Auch später können weitere Templates auf ein Board importiert werden. Dadurch das Templates frei definiert werden können, gibt es auch potentiell Templates welche die selben Konzepte abbilden die auch in CML abgebildet werden können. Die Eigenschaft eines Templates, dass sie genutzt werden können um ein bestimmtes Format zu definieren, ist für uns in unserem Projekt besonders wichtig, da dies eine Chance fürs Mapping darstellt. Wenn für die Nutzung eines Boards ein Template verwendet wurde und sich der Nutzer an das vom Template vorgeschlagene Format hält, ergibt sich dadurch eine Struktur der Widgets anhand der man ein Mapping durchführen kann. Ein wichtiger Unterschied zwischen Miro-Templates und Templates wie sie in anderen technischen Tools besteht darin, dass während Templates meist ein eher starres Konstrukt bilden, sprich wen es definiert wurde kann danach davon nicht abgewichen werden, so sind Miro-Templates strukturell nicht bindend. Wen ein Template auf ein Board importiert wird, wird im Grunde lediglich eine Ansammlung von Widgets auf das Board kopiert. Es gibt zwei grundsätzliche Arten von Miro-Templates:

- **Normative Templates:** Normative Templates erstellen eine Art Formular, dass der Nutzer ausfüllen kann. Ein prominentes Beispiel dafür wäre das Bounded Context Canvas Template von Nick Tune. [14] Das Format ist hier sehr strikt vorgegeben und die Inhalte sind dadurch (wenn der Nutzer sich an die Vorlage hält) immer am gleichen Ort. Dies ist für das Mapping natürlich besonders interessant, da so die Position eines Widgets zur Identifikation des abgebildeten Sachverhalts genutzt werden kann.
- **Vorschlagende Templates:** Vorschlagende Templates sind offener vom Format her und stellen dem Nutzer einer Starter-Kit von Vorlagen für einzelne Elemente zur Verfügung. Die Anordnung der Elemente ist dabei dem Nutzer Überlassen. Zwei Beispiele von Konzepten die sich gut über ein Starter-Kit Template abbilden liessen wären Event-Storming und Context-Map. Bei vorschlagenden Templates fällt der Faktor der Position als möglicher Mapping-Indikator aus oder muss zumindest mit etwas mehr Logik versehen werden. Hier findet sich jedoch eine Chance fürs Mapping in der Eigenschaft das die Vorlagenobjekten die jeweils einen bestimmten Sachverhalt abbilden bestimmte Attribute haben und somit die darauf basierenden Widget die gleichen Attribute

verwenden. Dadurch ist Rückschluss auf den durch das konkrete Widget abgebildete Sachverhalt möglich.

B.2.7. CML

Der ContextMapper bildet die Zieldomäne ab. In CML können auch die Concepts abgebildet werden.

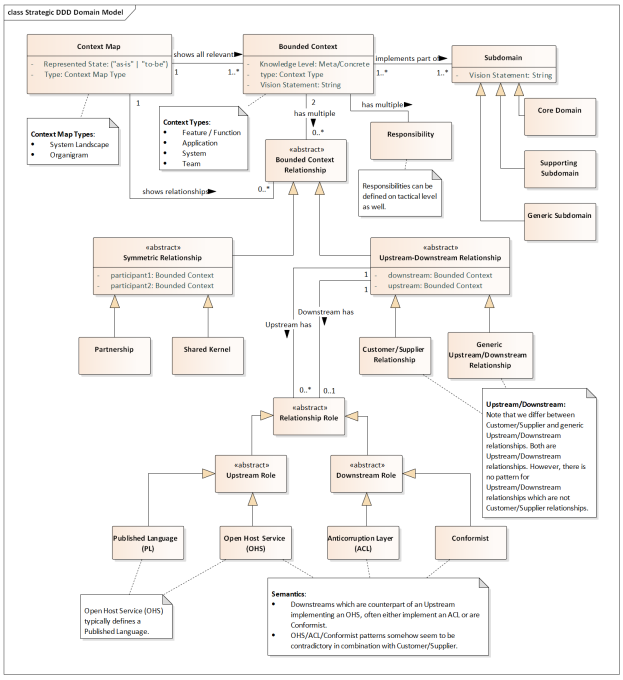


Abbildung B.3.: Strategisches DDD Domain Model der CML[15]

B.2.8. Concept

Die Domänen-Klasse Concept kommt in beiden Domänen vor und stellt dass von Miro in CML zu transferierende Gedankengut dar. Diese Domänen-Klasse stellt das Bindeglied zwischen den beiden Domänen dar und ist daher von fundamentaler Wichtigkeit um den Inhalt eines Miro-Boards in CML zu transferieren. Die Concept Domänen-Klasse ist eine Bündelung verschiedener Artefakte mit den gemeinsamen Eigenschaften, dass Sie für Softwareengineering/Software-Architektur nützlich sind und sowohl visuell in Miro als auch mit der CML beschrieben werden können. Unsere Usecases stehen in direkte Relation zu diesen Konzepten. Durch die offene und abstrakte Definition des Begriffs Konzept, ist es möglich das in Zukunft durch Erweiterungen seitens Miro oder am ContextMapper weitere Konzepte entstehen können. Wie wollen hier aber unseren Fokus auf die vier Konzepte User Stories, Bounded Context Canvas, Event Storming und Context Map legen. Den es sind genau diese vier Konzepte, deren Transformation von Miro zu CML, durch unsere UseCases

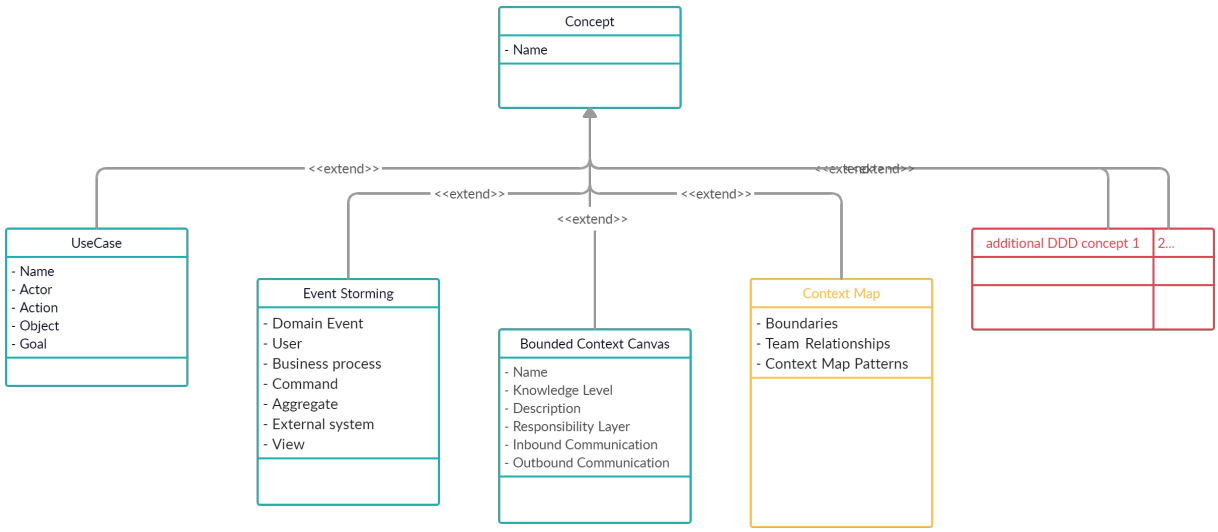


Abbildung B.4.: Aufschlüsselung der Concepts

abgedeckt werden.
Im Diagramm sind die Konzepte und somit auch die UseCases, welche von unserem Prototypen am Ende des Projekts unterstützt werden sollen türkis eingefärbt. Die optionalen Konzepte sind gelb und die out-of-scope Konzepte/UseCases rot eingefärbt. Die einzelnen konkreten Konzepte werden hier nochmal kurz aufgeschlüsselt und dann im separaten Dokument für die Analyse der für die Mapping-Heuristiken für die Konzepte nochmals genauer erörtert.

User Stories

User Stories [16] werden im Agilen Software-Development dazu genutzt um die zu erledigende Arbeit in funktionale Inkremente aufzuteilen. Von jeder User Story wird dabei grundsätzlich erwartet, dass sobald sie implementiert wurde, sich der Wert des Gesamtprodukts steigert. Dabei soll der Umfang einer Story so klein wie möglich sein. User Stories werden meist in der Form **As a... I want to... so that...** angegeben. Diese Grundstruktur wir auch Connextra Format genannt. Diese Grundstruktur kann man weiter aufschlüsseln indem man die Absichtserklärung in eine Aktion und ein Objekt auftrennt. Dadurch ergibt sich das folgende Format:

As a <actor> I want to <action> a <object> so that <goal>.

Weiter kann die Struktur noch so angepasst werden dass zusätzlich Referenzen und Attribute unterstützt werden, dadurch ergeben sich die folgenden Schemata:

As a <actor> I want to <action> a <object>
for a <reference> so that <goal>.
As a <actor> I want to <action> a
<object> with its <attribute, attribute> so that <goal>.

Bounded Context Canvas

Das Bounded Context Canvas wurde von der DDD-Crew erstellt[17]. Es ist ein Miro Template, in welchem alle wichtigen Merkmale von einem Bounded Context erfasst werden. In der Abbildung B.5 ist ein leeres Beispiel Template zu sehen[18].

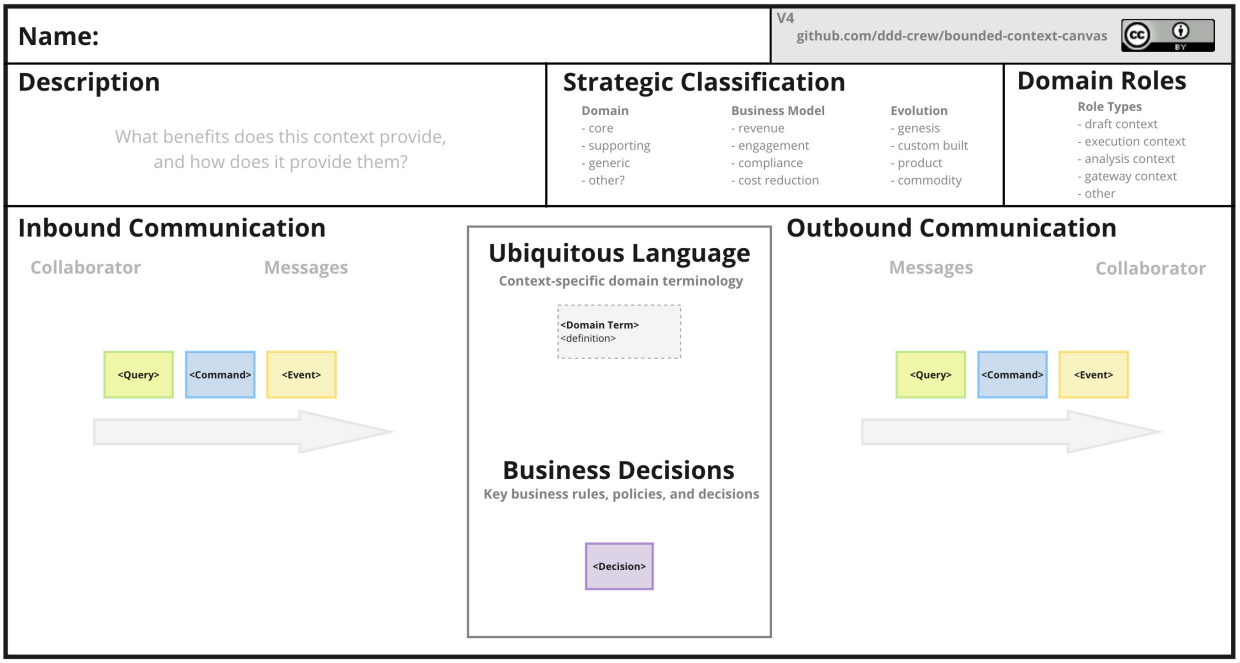


Abbildung B.5.: Bounded Context Canvas [2]

Event Storming

Unter Event Storming versteht man eine Form von Workshop, bei dem man in einer Gruppe eine Komplexe Business Domäne untersucht[4]. Das Resultat von einem solchen Workshop ist häufig eine Landkarte von StickyNotes, die einen Geschäftprozess darstellen, wie in der Abbildung B.6 zu erkennen ist [19].

Context Map

In einer Context Map wird ein Netz von Bounded Context und deren Beziehungen grafisch dargestellt. Die Beziehungen der Bounded Context werden mithilfe von gängigen Abkürzungen beschriftet. In der Abbildung B.7 ist ein Beispiel von einer Context Map dargestellt[20].

1 https://github.com/ddd-crew/bounded-context-canvas/blob/master/resources/bounded-context-canvas-4v-blank.jpeg
2 https://github.com/ContextMapper/context-mapper-examples/tree/master/src/main/cml/lakeside-mutual

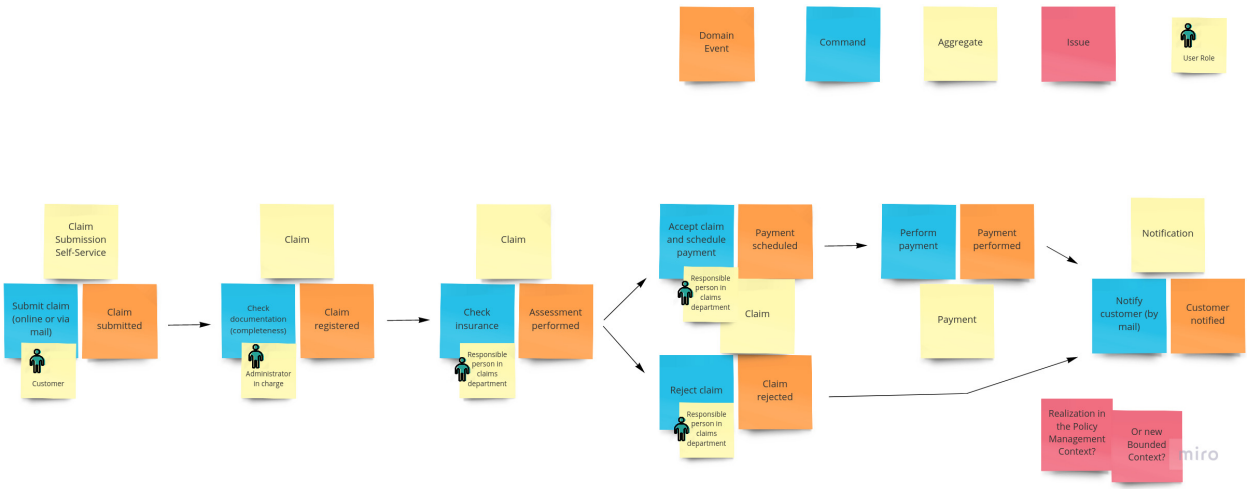


Abbildung B.6.: Beispiel Event Storming Result[3]

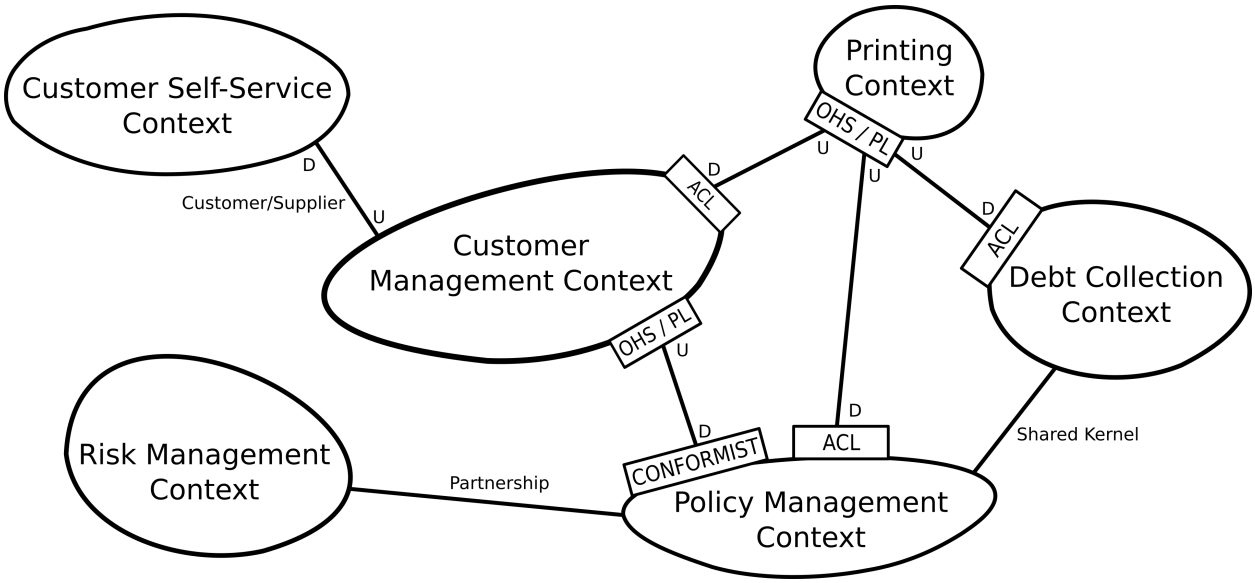


Abbildung B.7.: Beispiel Context Map[4]

3 https://github.com/ContextMapper/context-mapper-examples/tree/master/src/main/cml/insurance-example

C. Softwarearchitektur und Design

C.1. Einführung

C.1.1. Version

Version 2, 17. Dezember 2020

C.1.2. Zweck

Das vorliegende Dokument beschreibt die Softwarearchitektur und das Design der Applikation miro2cml.

C.1.3. Gültigkeitsbereich

Das Dokument beschränkt sich auf das Projekt miro2cml und die Dauer der Studienarbeit im HS2020/21.

C.1.4. Referenzen

Die Referenzen sind als Fussnoten angegeben.

C.1.5. Übersicht

- Architektonische Ziele und Einschränkungen
- Logische Architektur
- Externe Schnittstellen
- Deploymentdiagramm
- Grössen und Leistungen

C.2. Architektonische Ziele, Entscheide und Einschränkungen

In diesem Kapitel wird beschrieben, welche architektonischen Ziele und Entscheidungen anhand der definierten nicht funktionalen Anforderungen und der Aufgabenstellung getroffen

wurden und welche Einschränkungen gelten.

C.2.1. Applikationsform

Ein wichtiger architektonischer Entscheid, stellt die Wahl der Applikationsform dar. Um diesen Entscheid zu fällen haben wir die unterschiedlichen gängigen Applikationsformen analysiert und die Vor- und Nachteile mit den nicht funktionalen Anforderungen der Personas abgeglichen. Die Tabelle zeigt die Vor- und Nachteil der verschiedenen Applikationsformen auf.

Variante	**Vorteile**	**Nachteile**
Command Line Interface	Resource usage: verteilt Portabel einfache Umsetzung UI: einheitlich potentiell Batchable (sehen aber keinen wirklichen Use Case dafür)	UX: schlecht Installation benötigt
Natives GUI	Resource usage: verteilt Portabel(zum Beispiel mit JAVA FX) UX: ok UI: mögliche Abweichungen zwischen unterschiedlichen Plattformen	Installation benötigt erfordert Verwendung von einem GUI Framework (neue Technologie)
Web-UI	Portabel UI: einheitlich UX: sehr gut (miro ist bereits im web-browser, neuer Tab für Converter öffnen fühlt sich natürlich an) keine Installation benötigt	Resource usage: zentralisiert (kann mit mehreren Instanzen und Loadbalancer verteilt werden, sodass kein Bottle-Neck entsteht.) Presentation Layer benötigt zusätzliche Technologie Presentation Layer benötigt zusätzliche Schnittstelle Deployment/Accessibility Concerns

Wir haben uns entschieden eine Web Applikation zu erstellen, weil dies die Anforderungen der Portabilität und Benutzerfreundlichkeit am besten erfüllt. Bei der Benutzerfreundlichkeit ist wichtig, dass der Benutzer nach 10 Minuten bereits das erste Board konvertiert hat. Diese Anforderungen wurde im NFR 2 (A.5.1) spezifiziert. Bei der Web Applikation ist keine Installation notwendig, so kann Zeit gespart werden, was es uns erleichtern sollte dieser Anforderung gerecht zu werden. Auch ist die erwünschte gute User Experience, nur mittels eines Grafischen UIs zu erreichen. Eine Web Applikation kann man problemlos auf den gewünschten Betriebssystemen nutzen und somit werden wir der Anforderung Portabilität (A.5) gerecht.

C.2.2. Distribution Pattern

Wir haben uns für das Pattern, Distributed Presentation entschieden. Durch diese Aufteilung befindet sich auf dem Tier 1 Client nur die Presentation. Der Dialog Control befindet sich im Tier 2, dem Server[21]. Mit dieser Aufteilung werden wir den Anforderungen Modularität (A.5) und Security (A.5) gerecht. Mit diesem Pattern erhalten wir einen modularen Aufbau. Dabei verwenden wir das MVC, also Model-View-Controller Pattern. Im Vergleich zum Remote User Interface erhalten wir eine höhere Sicherheit, da nur die View im Client gerendert wird. Wir müssen jedoch weiterhin darauf achten, dass die Datenübertragung immer über https geschieht.

C.2.3. Technologiewahl

Die Implementationssprache Java ist durch den Projektauftrag vorgegeben. Das meist genutzte Framework für Java ist Spring. Da ein Teammitglied bereits Erfahrungen mit Spring in einem Projekt sammeln konnte und Spring im Unterricht im Modul Application Architecture Teil des Curriculum ist, haben wir uns für dieses Framework entschieden. Beim Frontend/Presentation Layer suchten wir eine Technologie die sich sowohl für die Nutzung in Web- als auch Standalone-Umgebung funktioniert. Thymeleaf bietet genau das und ist zusätzlich häufig genutzt für HTML5 JVM Webentwicklungen. Bei der Technologiewahl wurde zusätzlich auch darauf geachtet, dass gute Dokumentationen / Tutorials vorhanden sind. Aus diesen Gründen haben wir uns dann für Spring Boot und Thymeleaf entschieden.

C.2.4. Datenbank

Wir haben uns dazu entschieden unsere Applikation ohne Datenbank zu entwerfen und somit von den herkömmlichen Two-Tier(mit integrierter Datenbank)/Three-Tier Architekturen abzuweichen. Diesen Entscheid haben wir gefällt weil sich durch die funktionalen und nicht-funktionalen Anforderungen keine Notwendigkeit für eine Datenbank ergibt und durch das weglassen der Datenbank Komponente sich die Gesamtkomplexität des Systems senken lässt. Eine Datenbank wäre unter anderem dann sinnvoll, wen ein Nutzer sich einem Service gegenüber Authentifizieren muss und daher die entsprechenden Authentifizierungsdaten gespeichert werden müssen oder die Applikation Daten langfristig zwischenspeichern muss.Zur

Nutzung unseres Services ist zwar durchaus eine Authentifizierung notwendig, jedoch nicht unserem Service sondern der Plattform Miro gegenüber. Diese Authentifizierung wird jedoch direkt über eine von Miro zur Verfügung gestellten Schnittstelle erledigt.

Entscheidungsbegründung mit Y-Template [22]:
In the context of the database *facing* performance(turnaround time) *we decided for* not using a database *and neglected* any kind of database *to achieve* a minimal time for first-time use and a small entry barrier *acceting downside* that we need to save the authorization with Miro via the session and that we can not store data of the users.

C.2.5. Datenschutz

Ein weiteres wichtiges Ziel ist das Schützen der Nutzerdaten und Konten. Da die Daten über eine Restful-HTTP Schnittstelle ausgetauscht werden und auch Access-Token über diese ausgetauscht werden, muss zwingend darauf geachtet werden, dass die Kommunikation stets über HTTPS verläuft. Ansonsten werden werden die Daten, beziehungsweise noch gravierender die Access Token, in Klartext übermittelt. Sollte es einem Angreifer gelingen, ein Access Token abfangen können, würde dies bedeuten dass der Angreifer sämtliche Daten der Boards extrahieren kann, auf die der Nutzer lesenden Zugriff hat.

C.3. Logische Architektur

Dieser Abschnitt beschreibt die Logische Architektur. Angefangen wird mit einem Überblick mithilfe eines Schichtendiagramms. Danach wird mit einem Component Diagramm auf die Business Layer Architektur eingegangen, welches als Überleitung zum Package Diagramm dient. Im Package Diagramm ist eine detaillierte Beschreibung der Architektur und des Designs.

C.3.1. Schichtendiagramm

In der Abbildung C.1 ist das Schichtendiagramm zu sehen. Das Schichtendiagramm dient der Übersicht und enthält die wichtigsten semantischen Informationen [23]. der Abbildung ist das Layer Diagramm zu sehen. Das Pattern, Distributed Presentation, dass im vorherigen Abschnitt erläutert wurde, ist ebenfalls zu erkennen. Die View im Web-UI(orange), welche auf dem Client gerendert wird, wird per HTTPS von unserer Applikation(blau) ausgeliefert. Die Applikation heisst miro2cml, wie in der Abbildung zu sehen ist und kommuniziert über HTTPS mit Miro und dem Nutzer, daher wird die Applikation folgend Applikationsserver genannt. Über die Obhut des Dialaog Controls verfügt jedoch der Applikationsserver. Dadurch erstreckt sich der Applikationsserver nicht nur über die Business Logik sondern auch über Teile des Presentation Layers. Der Applikationsserver enthält auch Funktionalität, welche dem Data Access Layer zuzuordnen ist, denn die Kommunikation mit der Datenquelle Miro ist direkt in den Applikationsserver integriert.

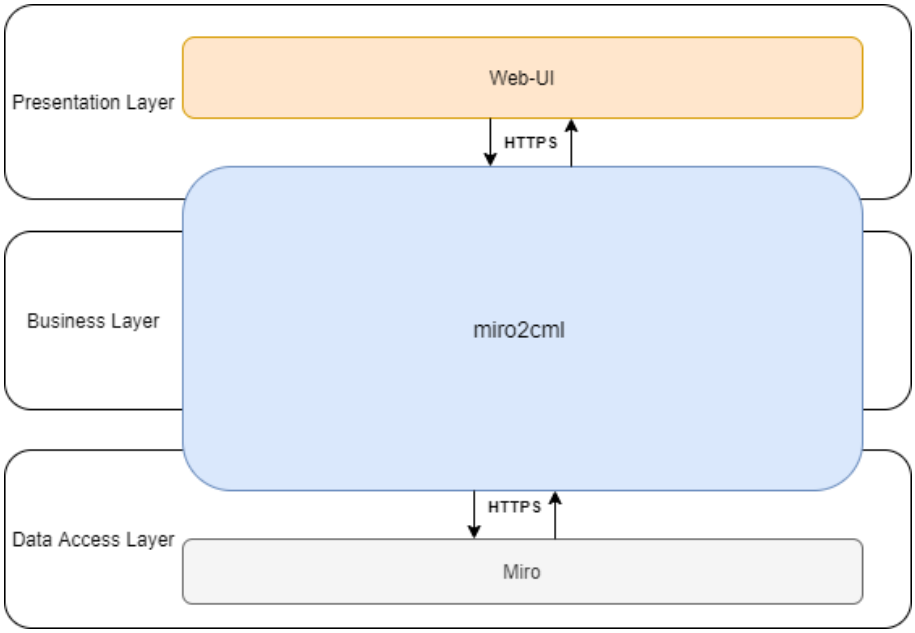


Abbildung C.1.: Schichtendiagramm

C.3.2. Component Diagramm

In der Abbildung C.2 ist das Component Diagramm nach C4[1] dargestellt. Das Component Diagramm zeigt die Beziehungen zwischen den Komponenten auf [24]. Der miro2cml Teil aus dem Schichtendiagramm C.1 ist innerhalb des schwarz umrandeten Kasten abgebildet. Auf die einzelnen Komponenten wird im Abschnitt Package Diagramm genauer eingegangen.

1 https://c4model.com/#ComponentDiagram

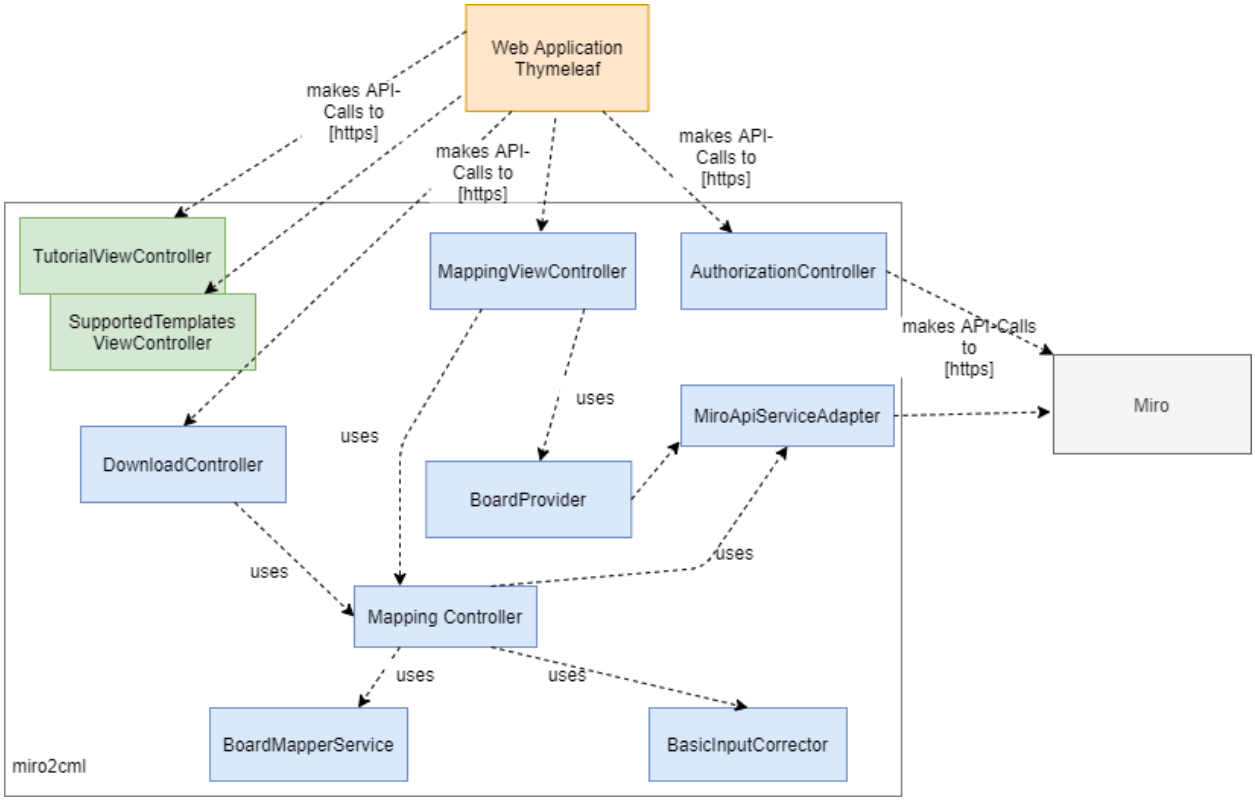


Abbildung C.2.: Component Diagramm

C.3.3. Package Diagramm

In der Abbildung C.3 ist das Package Diagramm der miro2cml Applikation zu sehen. Ziel dieses Package Diagramms ist es die Strukturierung der Codebasis aufzuzeigen. Die folgenden Sektionen befassen sich mit der Funktionalität und der zugrunde liegender Logik der einzelnen Packages.

C.3.4. Ressource Package

In diesem Package befinden sich die Files vom Frontend, die Views. Im Template Package sind die Thymeleaf Files abgelegt und die Files fürs Styling und weitere statische Ressourcen befinden sich im static ressource Package.

C.3.5. Presentation Package

Dem Presentation Package sind die fünf Controller zuzuordnen, die für den Dialog Control zuständig sind. Daneben gibt es noch das Sub-Package Model, hier werden die Modelklassen vom Präsentationlayer definiert. Das Sub-Package Utility bündelt die Funktionalität des Session Handlings. Alle fünf Controller greifen auf die Packages utility und model zu.
Besondere Beachtung im Presentation Package muss der Miro-AuthentifizierungsContoller

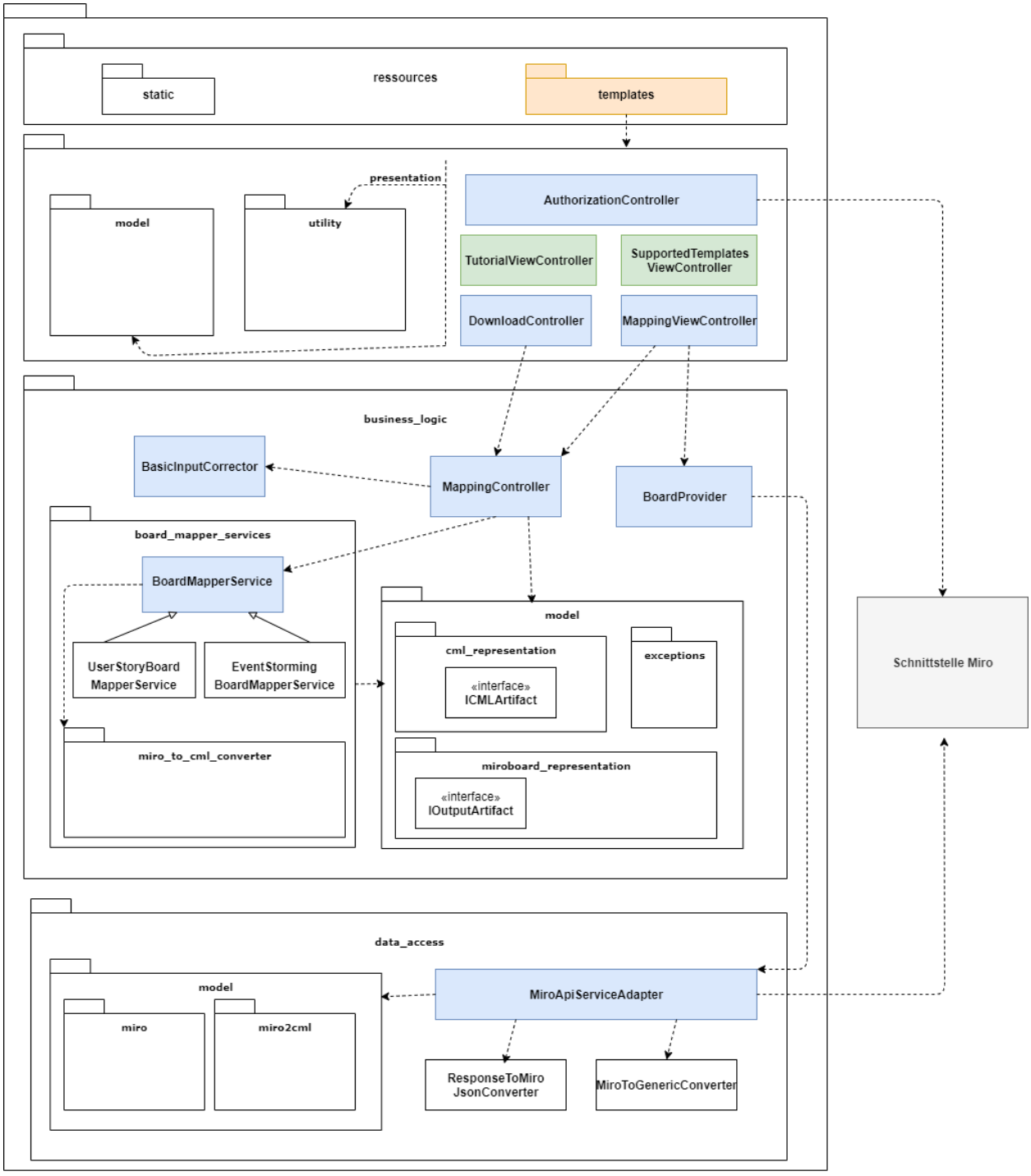


Abbildung C.3.: Package Diagramm

zugeteilt werden. Diese Klasse handelt den kompletten Authentifizierungsprozess ab, dafür sind sowohl Userinteraktion als auch Kommunikation mit Miro, welche eigentlich dem Data Access Layer zuzuordnen wäre, notwendig. Hier haben wir uns bewusst dazu entschieden, das Trennungsprinzip der Layer zu verletzen. Diese Entscheidung, haben wir getroffen um die Komplexität des Systems zu minimieren in dem wir eine aufwendige Propagierung der Methodenaufrufe/Daten zwischen den Layern vermeiden.

C.3.6. Business Logic Package

Im Business Logic Package befindet sich die Programmablauflogik, die vom Presentation Layer aufgerufen wird. Der BoardRepresentationProvider stellt dem Presentation Layer eine Möglichkeit zur Verfügung, die möglichen InputBoards anzufordern. Der MappingController enthält die Programmablauflogik zur Akquirierung der Inputdaten, ruft den BasicInputCorrector auf, kümmert sich um das Aufrufen des richtigen Mapping Services und die Weitergabe der Mapping-Resultat an den Web-Controller aus dem Präsentation Layer. Teil dieser Resultate sind auch die Logs, die während dem Transformieren und Abrufen der Daten entstehen. Die Transformation der vom Data Access Layer erhaltenen Daten geschieht anhand des Strategy Patterns, dabei wird die Konvertierung der Miro Objekten in CML-Objekte basierend auf dem BoardTyp durchgeführt. Die einzelnen Services, welche im Package Board_Mapper _Services definiert sind, beinhalten keine Logik in Bezug auf die Board Typen. Die Selektion der Mapping Strategie geschieht bereits im Mapping Controller. Weiter gibt es ein Sub-Package Model, welches die notwendigen Modellklassen für die Konvertierung in sich vereinigt. Wichtig sind hier zwei Interfaces. Das Interface IOutputArtifact wird genutzt um zu signalisieren, dass es sich bei der implementierenden Klasse um ein Resultat handelt, welches dem Nutzer über die Webschnittstelle zur Verfügung gestellt werden kann. Die Klasse CmlModel ist eine Implementierung dieses Interfaces, welche zur Erfüllung dieser Anforderung die CML-Library des Context-Mappers verwendet. Das Interface ICMLArtifact definiert die Anforderungen, an die Implementationen für CML-Artifakte. Bei den Mapping Services und CML-Artifakte werden nicht alle Klassen aufgeführt um das Diagramm übersichtlicher zu gestalten.

C.3.7. Data Access Package

Der MiroApiServiceAdapter bildet die externe Schnittstelle zur Miro API, sie kümmert sich sowohl um das versenden des GET-Requests an die Miro API als auch die Verarbeitung der Response. Intern verwendet diese Klasse die beiden Hilfsklassen ResponseToMiroConverter und MiroToGenericConverter. Die Klasse ResponseToMiroConverter wird für das Parsen des JSON Objekts, welches sich im InputStream befindet verwendet. Dabei wenden wir Library Jackson für die Transformation des JSON Objekts in unsere Miro-Modellkassen an. Da die Miro-Modellklassen sehr nahe am JSON Format sind, ist die Weiterverarbeitung der Daten umständlich. Diese Problematik wird durch die Klasse MiroToMiro2cmlConverter gelöst, welche Methoden zur Verfügung stellt um diese Miro-Modellklassen in sinnvoller typisiertere Klassen, den miro2cml-widget Modelklassen zu überführen. Eine genauere Beschreibung

dieses Packages findet im Zuge der Beschreibung der Externen Schnittstelle statt.

C.3.8. Patterns

Durch die Verwendung von Spring Boot, kommen die Patterns Inversion of Control und Dependency Injection zum Einsatz. Des weiteren wird im Presentation Layer das Pattern Model-View-Controller verwendet. Für die Konvertierung von dem MiroObjekten zu CML-Objekten wird das StrategyPattern eingesetzt. Dabei wird eine Selektion der auszuführenden Mapping Strategie im MappingController getätigt. Es gibt eine Mapping Strategie pro unterstütztem BoardTyp. Die einzelnen Mapping Strategien beinhalten keine Logik, welche vom selektieren BoardTyp abhängig ist.

C.4. Externe Schnittstellen

Dieses Kapitel widmet sich den externen Schnittstellen des miro2cml Projekts. Namentlich sind dies die Schnittelle zu Miro, die Miro-API und die Schnittelle zum Context Mapper, der Context Mapper Serializer. Dabei soll für diese Abhängigkeiten ein Überblick geschaffen werden, wie wir in unserem Projekt diese externen Schnittstellen nutzen und auch eine Analyse stattfinden, was für das Austauschen der Schnittstelle notwendig wäre.

C.4.1. Miro

Miro ist die Externe Komponente welche den Input für unsere Applikation zur Verfügung stellt. Dementsprechend sind die dafür Relevanten Komponenten hauptsächlich im Data Access Layer gebündelt. Eine Ausnahme bildet der Authentifizierungs-OAuth Flow, welcher aufgrund der notwendigen Nutzerinteraktion im Presentation Layer implementiert wurde. Miro bietet eine Umfangreiche Restful-HTTP Schnittstelle an [9]. Diese Schnittstelle liefert die Daten in JSON Form aus, also in der Javascript Object Notation. Diese eignet sich zwar für die Übermittlung, ist jedoch für uns für die weitere Verwendung in unserer Applikation leider ungeeignet. Daher ist es Notwendig die empfangenen Daten in POJOs, also PlainOld-JavaObjects zu transformieren. Unsere Data Access Layer Funktionalität basiert auf einer Hauptklasse, zwei Hilfsklassen und einigen Modellklassen. Die Modellklassen sind dabei in zwei Kategorien aufgeteilt, die Kategorie Miro, welche von der Struktur direkt abhängig von der Form der JSON-Daten die von der Miro-Schnittstelle geliefert werden und die Kategorie Generisch, welche eine normalisierte Darstellung der Objekte ist. Die Miro Modellklassen sind im Sub-Package model des DataAccessLayers gebündelt während die generischen Klassen im Hauptpackage model untergebracht sind. Das folgende Diagramm beschreibt die Struktur des DataAccessLayers:

Dabei ist der MiroApiServiceAdapter die Klasse, welche vom Business Layer aufgerufen wird um Daten von Miro abzufragen. Der MiroApiServiceAdapter bietet zwei Methoden an, getMiroBoards() und getBoardWidgets(). Wird eine der beiden Methoden aufgerufen

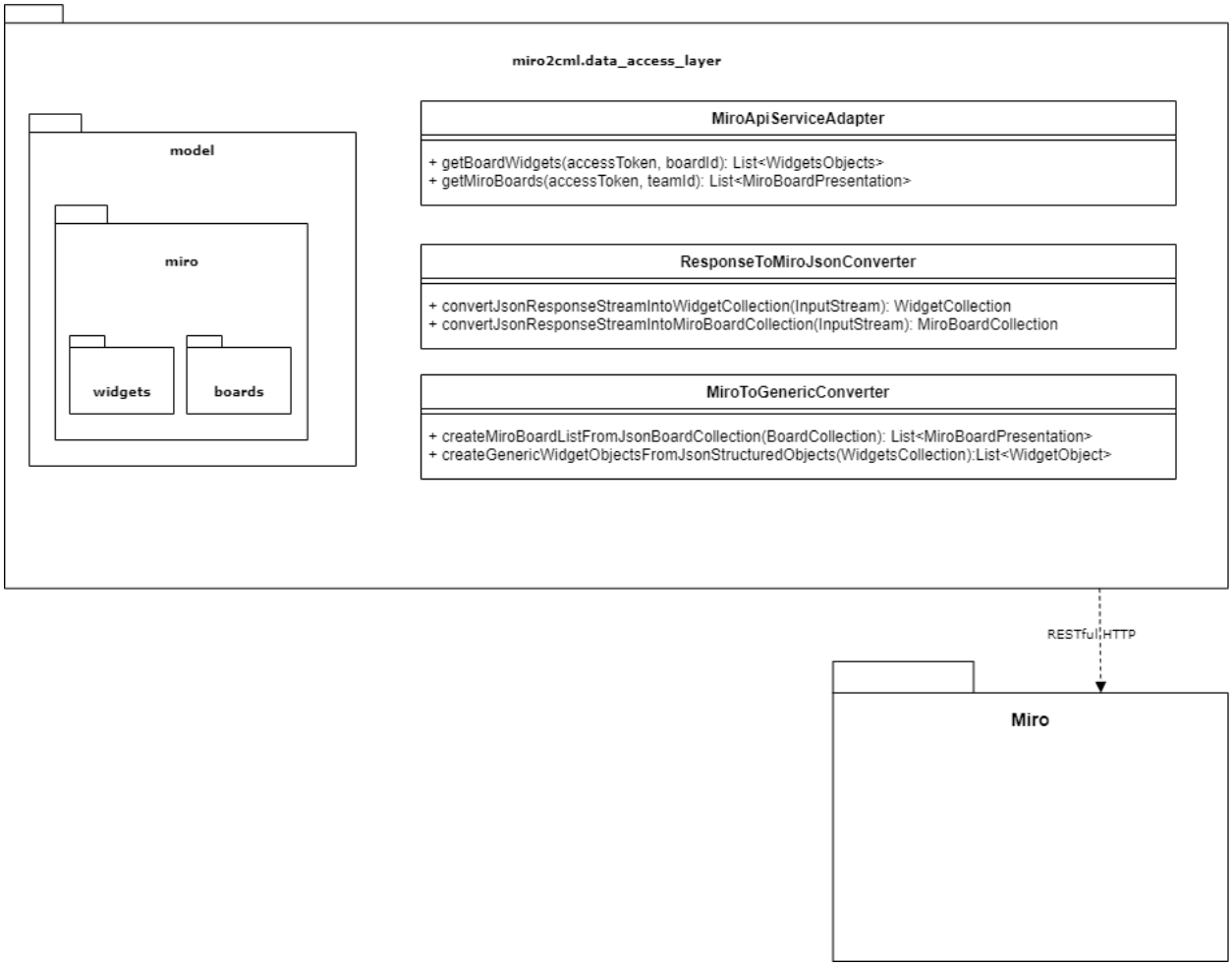


Abbildung C.4.: Struktur des Data Access Layers

so wird ein entsprechender Request an die MiroApi geschickt. Die Antwort wird dann an den ResponseToJSONConverter weitergeleitet, welche mithilfe der Jackson Libary und den ”JSON”-Modelklassen im model.miro Package die erste Transformation der Daten vornimmt. Hier ist anzumerken, dass es sich bei den ”JSON”-Modellklassen syntaktisch natürlich nicht um JSON-Klassen handelt sondern um POJO-Klassen. Jedoch ähneln sie strukturell noch sehr stark an einem JSON-Objekt. Theoretisch wäre es zwar möglich die Daten in dieser Form so der Businesslogik zu übergeben, allerdings wäre die Handhabung der Objekte durch die stark an JSON orientieren Objektstruktur eher unbequem, weswegen wir uns dazu entschieden diese Objekte in eine Form zu bringen, welche einfacher zu handhaben ist. Dafür ruft der MiroApiServiceAdapter den MiroToGenericConverter auf, welcher die erhaltenen ”JSON”-Objekte in wahre Pojos umwandelt. Diese Abkapslung der ursprünglichen Form der Inputdaten bedeutet zwar etwas mehr Programmier-Aufwand und auch mehr Rechenaufwand im Dataaccesslayer, erhöht aber die Lesbarkeit des Codebasis in den anderen Ebenen der Software und ist auch zeitgleich Grundvoraussetzung für die Austauschbarkeit der Datenquelle. Soll eine neue Datenquelle, zum Beispiel ein andere kollaborative Platt-

form, ebenfalls einer Restfull-HTTP Schnittstelle unterstützt werden, so müssen die Daten lediglich in diese generische Form gebracht werden. Es benötigt aber keine Anpassung an den darüber liegenden Ebenen. Das folgende Sequenzdiagramm visualisiert den Ablauf eines getBoardWidgets() Calls:

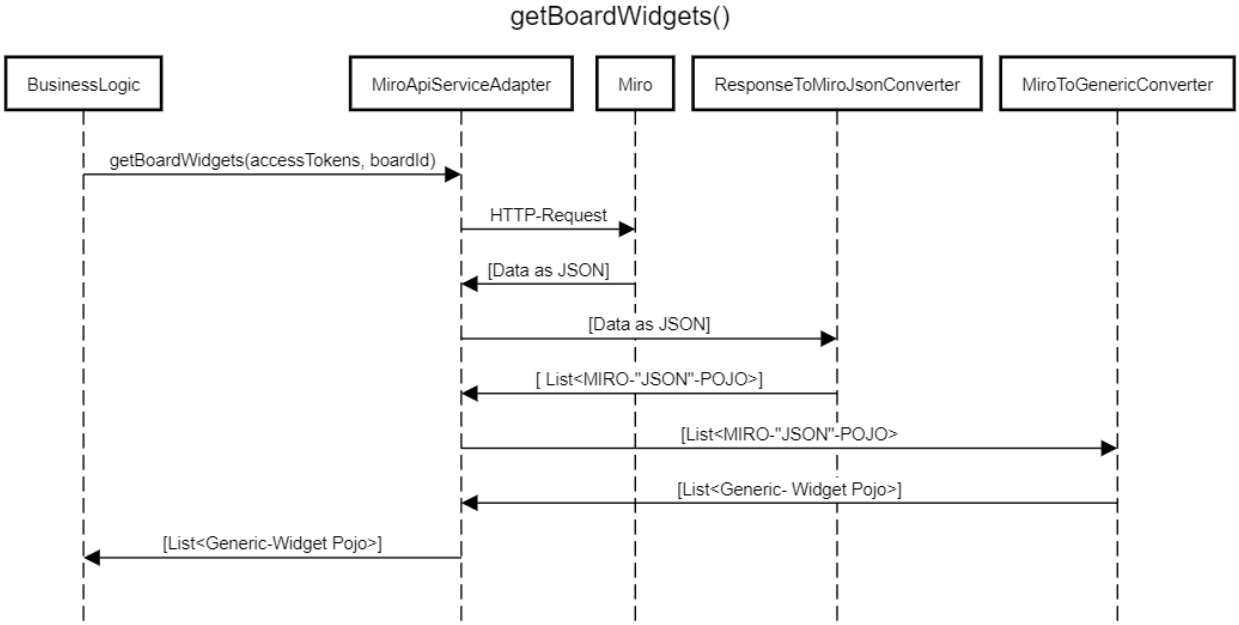


Abbildung C.5.: Sequenzdiagramm für getBoardWidgets()

Anforderungen an Substitutionskomponenten und allenfalls notwendige Anpassungen

Um der nicht funktionalen Anforderung der Modularität gerecht zu werden haben wir analysiert, was den Notwendig wäre um die externe Abhängigkeit Miro zu substituieren. Damit ein Austauschen der Miro-Komponente, durch eine andere Datenquelle möglich wird, muss die alternative Datenquelle die Folgenden Kriterien erfüllen:

- Bieter eine Schnittstelle an, die konsumiert werden kann um Inputdaten für Mapping abzuholen.
- Bieter eine Schnittstelle an, um durch mehreren Datensätze zu navigieren (Selektion der Inputdaten).
- Nutzt AccessToken für Nutzeridentifikation bietet einen OAuth2 Flow für die Nutzerauthentisierung an.
- Datenstruktur lässt sich einfach auf Board - Widget Struktur mappen.

Wen diese Grundbedingungen gegeben sind, müssten die folgenden Anpassungen an der Codebasis vorgenommen werden um die Datenquelle auszutauschen:

- Ersetze Modellklassen in Data Access Layer
- Ersetze MiroApiServiceAdapter durch ServiceAdapter für andere Datenquelle.
- Ersetze OAuth2 Flow in Miro-Authentifizierungsklasse (Presentation Package)
- Ersetze ResponseToMiroConverter durch eine Klasse, welche Mapping der Inputdaten auf neue Modellklassen ermöglicht
- Ersetze MiroToGenericConverter durch eine Klasse, welche die neuen Modelklassen auf die globalen Modelklassen mappt.

Soll Miro nicht einfach nur ersetzt sondern durch eine andere Datenquelle ergänzt werden, so sind die den oben genannten Änderungen nicht durch Ersetzen, sondern durch Ergänzen zu erreichen und zusätzlich auch die folgenden Änderungen durchzuführen:

- Implementation einer Datenquellenselektion (in Presentation Layer und Business Logik Layer)
- Anpassen des Authentifizierungsprozesses, sodass nur Authentifizierung bei der entsprechend selektierten Datenquelle notwendig ist.

C.4.2. Context Mapper

Die Zielsprache unser Applikation ist die Context Moddeling Language (CML). Diese Zielsprache können wir mit der Context Mapper Library generieren. Hier soll nun ein Überblick geschaffen werden, wo diese Library ihren Einsatz findet, welche Bedingungen erfüllt sein müssen und welche Restriktionen zu beachten sind. Die Nutzung der Context Mapper Library geschieht im Business Logik Layer, dort wird das eigene CML-Model in ein cml File konvertiert. Die Konvertierung kann nur stattfinden, wenn der Input semantisch korrekt ist. Ein weitere wichtiger Punkt ist, dass bei der Library in der ContextMap Kommentare nicht an beliebigen Stellen sein dürfen. Momentan sind Kommentare am Anfang des Files unterstützt. Dadurch, dass die Library nur korrekte Inputs verarbeiten kann, müssen wir das CMLModell vor der Konvertierung auf Korrektheit überprüfen. Der Vorteil ist jedoch, dass durch die Verwendung der CML Library das Output File korrektes CML enthält. Die CML Library ist auf Javadoc genau spezifiziert. [25] Die Schnittstelle befindet sich im Package cml_model. In cml_model sind die genutzten CML Elemente als Klassen dargestellt. Jede CMLArtifact Modellklasse implementiert das Interface ICmlArtifact und stellt somit die Methode EObject() zur Verfügung. Diese Funktion, dient dazu aus unseren eigenen Modellklassen das CML-Library Gegenstück zu generieren. Diese E-Objekte können dann dem CML-Model hinzugefügt werden. Damit ein Austauschen der CML-Komponente, durch eine andere Ziel-DSL möglich wird, muss die alternative Ziel-DSL die folgenden Kriterien erfüllen:

- Bieter eine Schnittstelle an die zur Serialisierung verwendet werden kann.
- Hat gleiche grundlegende Semantik wie CML

Wen diese Grundbedingungen gegeben sind, müssten die folgenden Anpassungen an der Codebasis vorgenommen werden um die Datenquelle auszutauschen:

- Ersetze CML-Model durch Ziel-DSL Modell (Business Service Package)
- Ersetze Ersetze Interface ICMLArtifact durch Interface, welche die Methoden definiert die für das Anhängen des Elements an das Model der Ziel-DSL notwendig sind.
- Passe Modellklassen in Business Service Package so an, sodass sie das neue Interface implementiert wird.
- Passe Mapping Services an, sodass diese dass Ziel-DSL Modell zusammensetzen.

Soll CML nicht einfach nur ersetzt sondern durch eine andere Ziel-DSL ergänzt werden, so sind die den oben genannten Änderungen nicht durch Ersetzen, sondern durch Ergänzen zu erreichen und zusätzlich auch die folgenden Änderungen durchzuführen:

- Implementation einer Output-DSL Selektion (in Presentation Layer und Business Logik Layer)

C.5. Deploymentdiagramm

Durch die Verwendung des Springboot Frameworks liegt ein Deployment als ausführbare FAT-JAR nahe. Das Projektteam entschied sich dazu, dieses JAR-File in ein Docker Image zu verpacken und als Docker Container auszuliefern. Dies resultiert im Deployment Diagramm das in der Abbildung C.6 dargestellt ist.

C.6. Grössen und Leistungen

Diese Sektion befasst sich mit den Kenngrössen und Leistungen des Prototyps und seiner Komponente. Dabei wird auch auf die damit verbundenen Limitationen und auch auf mögliche Lösungen eingegangen. Durch unsere Entscheidung keine Datenbank zu führen, haben wir keine eigene Logische Komponente im Data Access Layer. Dies und unser Designentscheid, Thymleaf unter Anwendung des Distributed Presentation Patterns, zu verwenden führt dazu, dass unsere Software eine grosse logische Komponente, also einen Monolithen bildet. Durch die Strukturierung als Monolith ist es nicht möglich Sub-Komponente auf verschiedene Plattformen zu verteilen, was Einschränkungen in der Skalierung mit sich bringt. Entsprechend sind die Möglichkeiten der vertikalen Skalierung vergleichsweise schneller ausgereizt. Um diese Problematik zu kompensieren wird die Software als Docker-Image ausgeliefert(oder alternativ auch als WAR-File), wodurch die Auslieferung auf viele Systeme parallel erfolgen kann. Dadurch wird es einfach horizontal zu skalieren, zum Beispiel unter Anwendung von Docker-Swarm oder Kubernetes Cluster. Hier ist die Monolithen Struktur wiederum von Vorteil da die einzelnen Instanzen in sich abgeschlossene Systeme bilden. Jede Instanz, kann gleichzeitig mehrere Nutzer bedienen. Allerdings, gibt es einige Limitationen

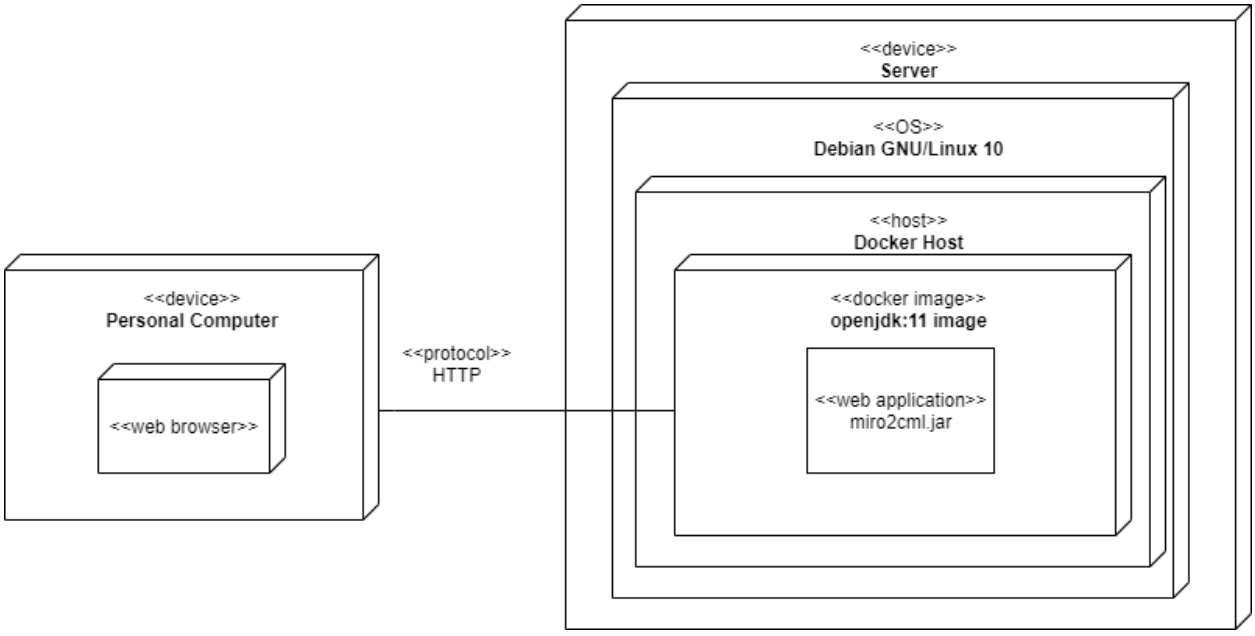


Abbildung C.6.: Package Diagramm

der Miro API welche die Funktionalität einschränken. Dadurch ergibt es sich zum Beispiel das nur die ersten 1000 Widgets eines Boards über die API auslesbar sind und die API ein Rate-Limit verfügt. Dies hat zur Folge, das pro Minute und Nutzer nicht mehr als 20 Boards gemappt werden können. [26]

D. Qualitätssicherung

D.1. Einführung

D.1.1. Version

Version 1.1, 16. Dezember 2020

D.1.2. Gültigkeitsbereich

Das Dokument beschränkt sich auf die Projektdauer der Studienarbeit im HS2020/21.

D.1.3. Referenzen

Die Referenzen sind als Fussnoten angegeben.

D.1.4. Übersicht

Das Dokument beschreibt die Qualitätsmassnahmen, die Sicherung der Geschichte und die Codestatistiken. Am Ende des Dokuments ist beschrieben, wie die Continous Integration(CI) und Continous Delivery(CD) aufgebaut ist.

D.2. Qualitätsmassnahmen

D.2.1. Coding Guidelines

Um die Code Style Guidelines durchzusetzten wird das Tool sonarlint[1] eingesetzt.

D.2.2. Definitons of Done

Die folgende Definition of Done wurde für die Arbeitspakete festgelegt:

- Keine Fehler oder Warnungen
- Pipeline erfolgreich durchlaufen
  - Build ohne Fehler
  - Unit Tests: alle Tests sind erfolgreich, sinnvolle Testcoverage

1 `https://www.sonarlint.org/`

- Nichtfunktionale Anforderungen erfüllt
- Code Review Prozess durchlaufen
- In Develop Branch gemerged
- Dazugehöriges Issue ist abgeschlossen
- Falls nötig: Dokumentation angepasst oder neues Arbeitspaket für Dokumentationsänderungen angelegt

D.2.3. Bug Monitoring

Gefundene Bugs werden im GitLab als Issue erfasst und mit dem Tag *incident* markiert.

D.2.4. Stunden-Erfassung auf den Arbeitspaketen

Die Stundenerfassung mit Toggl[2] ist mit GitLab verknüpft und die Zeit wird pro Arbeitspaket im Toggl erfasst. Somit kann im Nachhinein nach vollzogen werden, wieviel Zeit für die einzelnen Arbeitspakete verwendet wurde.

D.2.5. Code-Reviews

Code-Reviews wurden bei jedem Pull-Request durch das andere Teammitglied durchgeführt. Im Code-Review wurde besonders auf die gelernten Clean Code Techniken geachtet. Zusätzlich machte Stefan Kapferer am 25.11 ein Code Review mit dem Fokus auf Modularität und Abhängigkeiten. Sein Feedback wurde in Form von einem Refactoring umgesetzt.

D.2.6. Dokumentation-Reviews

Die Dokumente werden gegenseitig gegengelesen. Die einzelnen Software Dokumente werden ebenfalls dem Betreuer für die inhaltliche und sprachliche Korrektur zum Review abgegeben. Die Verbesserungsvorschläge werden jeweils in einer neuen Dokumentversion umgesetzt. Für einzelne Dokumente wird ein sprachliches Review von externen Personen durchgeführt.

D.2.7. Unit Tests (Microtesting und Integrations Tests)

Das Java Backend wird mit JUnit[3] getestet. Der Test Fokus ist in der Business Logik. Insbesondere die Klassen, die für das Mapping zuständig sind, sollen eine hohe Testabdeckung aufweisen, weil sie den Kern der Applikation abbilden.
Im Thymeleaf Frontend wird mit Thymeleaf Testing[4] getestet. Es zeigte sich im Verlauf des Projekts, dass das Testen im Frontend mühsam ist, da häufige Anpassungen im Frontend

2 https://toggl.com/
3 https://junit.org/junit5/
4 https://github.com/thymeleaf/thymeleaf-testing

gemacht werden mussten und immer wieder die Tests aufs Neue angepasst werden mussten. Der Ertrag der Tests war nur sehr gering, weil das Frontend manuell beinahe schneller getestet ist in diesem kleinen Umfang. Wir schränkten aus diesem Grund die Tests ein.

D.2.8. Systemtests

Die Systemtest werden manuell durchgeführt auf der Beta Version. Die Testspezifikationen- und Protokolle sind im Anhang E zu finden.

D.2.9. Perfomance- und Usabilitytest

Die Performance- und Usabilitytest werden ebenfalls manuell durchgeführt auf der Beta Version. Sie dienen zur Überprüfung der Nichtfunktionalen Anforderungen. Die Testspezfikationen- und Protokolle sind im Anhang E.4 zu finden.

D.3. Sicherung der Geschichte

Die Dokumente werden jeweils mit einem Datum und der Version versehen. Bei kleinen Änderungen wird die Version um 0.1 erhöht, bei grossen Änderung um eins. Alle Dokumente sind im GitLab Repository abgelegt und die genauen Änderungen können in den GitLab-Commits eingesehen werden.

Das Code-Repository ist ebenfalls im GitLab. Die Beta Version wird mit einem Tag gekennzeichnet für die Sicherung. Das Code-Repository wird als zip File gespeichert für die Abgabe.

D.4. Codestatistik

D.4.1. Verwendete Programmiersprachen

In der Abbildung D.1 sind die Prozentanteil der verwendeten Sprachen zu sehen.

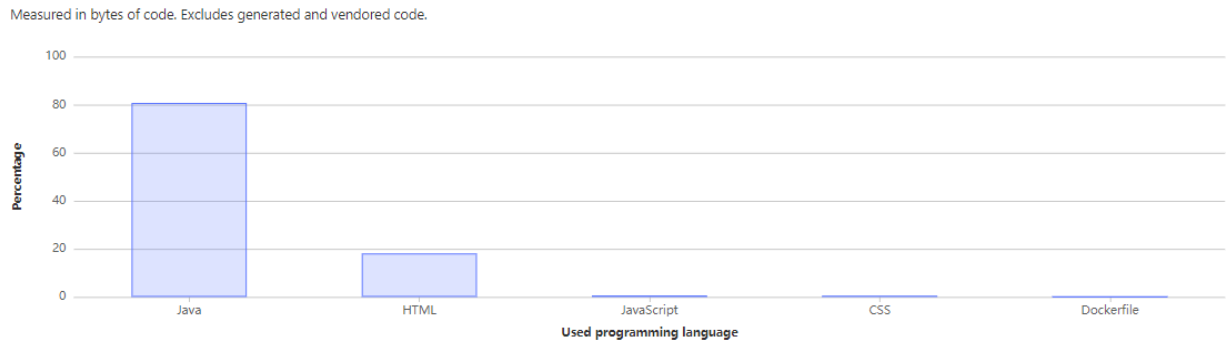


Abbildung D.1.: Verwendete Sprachen in Prozent

D.4.2. Code Statistik von SonarQube

In der Abbildung D.2 ist die Auswertung von SonarQube[5] zu sehen. Die Anzahl Smells hätte noch verringert werden können mit weiteren Refactorings, diese konnten jedoch aus Zeitgründen nicht durchgeführt werden. Beim Security Hotspot wird die Main Methode rot angezeigt, was aus unserer Sicht ignoriert werden kann. Die Duplikationen finden hauptsächlich in den Widgetklassen statt. Widgets sind die Element die von Miro ausgelesen werden. Die Elemente enthalten jeweils die gleichen Attribute wie zum Beispiel das Styling. Wir haben überlegt, dies mit einer Vererbung zu umgehen, haben uns dagegen entschieden, weil wir die Widgets als Datenklassen verwenden und wir es als übersichtlicher empfinden. Die Code Coverage liegt bei 0. Wir sind uns nicht sicher, ob ein Fehler unterlaufen ist bei der Messung der Coverage. Die Anzahl Unittests mit 59 kann aus unserer Sicht sicherlich noch verbessert werden. Weil wir den Hauptfokus beim Testen auf die Businesslogik gelegt haben und viele Datenklassen verwenden, haben wir die Code Coverage nicht hoch erwartet und kein Ziel für die Coverage gesetzt. Wir haben das Testen von Datenklassen beziehungsweise Testen von Getter und Settern als nicht sinnvoll empfunden und nutzten die dadurch gewonnene Zeit für System- und Usabilitytests. Ein weiterer Punkt, der uns das Testen erschwert hat, waren die Testdaten. Es war sehr aufwändig ein Miro Board zu simulieren mit den Test Daten, weil die Widgets sehr viele Elemente insbesondere im Styling enthalten.

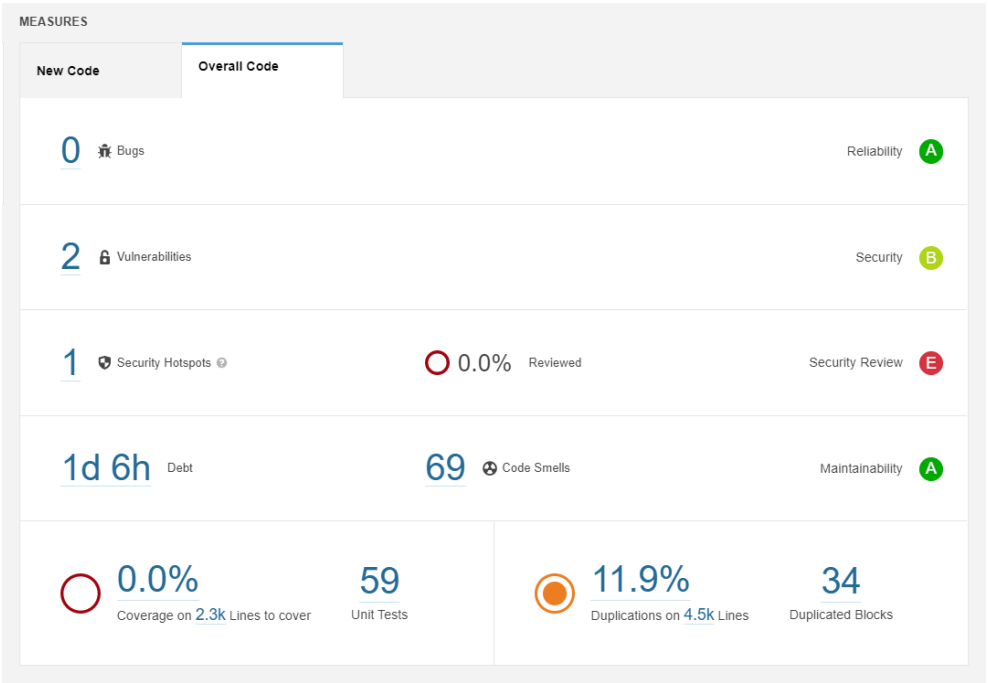


Abbildung D.2.: Auswertung SonarQube

5 https://docs.sonarqube.org/latest/

D.4.3. Lines of Code

In der Abbildung D.3 sind die Anzahl Zeilen pro Programmiersprache aufgelistet. Die Grafik wurde mit dem Plugin Statistic[6] erstellt.

Extension	Count	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG	Lines CODE
bat *(BAT files)*	1x	2kB	2kB	2kB	2kB	89	89	89	89	68
css *(CSS files)*	1x	1kB	1kB	1kB	1kB	110	110	110	110	0
gradle *(GRADLE files)*	1x	0kB	0kB	0kB	0kB	1	1	1	1	1
html *(HTML files)*	17x	62kB	0kB	7kB	3kB	1095	8	118	64	1062
java *(Java classes)*	87x	278kB	0kB	20kB	3kB	6789	5	341	78	5527
js *(JS files)*	1x	2kB	2kB	2kB	2kB	73	73	73	73	63
json *(JSON files)*	1x	2kB	2kB	2kB	2kB	69	69	69	69	69
properties *(Java properties*	8x	1kB	0kB	0kB	0kB	73	1	32	9	40
thtest *(THTEST files)*	3x	3kB	0kB	1kB	1kB	79	16	41	26	72
Total:	**120x**	**354kB**	**9kB**	**39kB**	**16kB**	**8378**	**372**	**874**	**519**	

Abbildung D.3.: Auswertung Lines of Code

D.5. CI/CD

CI/CD stehf für continuous integration und entweder continuous delivery oder continuous deployment, in unserem Fall für continuous deployment. Wir nutzen CI/CD um garantieren zu können, dass jeder unserer commits getestet ist und korrekt builded. Wen diese Zwei bedingungen Erfüllt sind und der Commit auf einen Branch gemergt wird, welcher deployt werden soll, dann übernimmt CD. Dieser Prozess wird oft auch CI/CD Pipeline genannt, weil der Sourcecode eine Pipline von Instruktionen durchläuft. Um zu beschreiben wie unsere CI/CD Pipeline funktioniert, müssen wir erst einmal einige zugrundeliegende Konzepte erklären.

GitLab

Gitlab[7] ist ein Sourcecode Versionierung und Verwaltung-Tool. Es ist im Grunde ein Git-Server mit vielen zusätzlichen Funktionalitäten. Wir haben uns dazu entschieden, die GitLab Instanz des Instituts für Software zu verwenden. [8]

Docker

Docker[9] bietet die Möglichkeit Images zu definieren, diese Images dienen als Applikations-"Schablonen". Aus diesen Docker Images können Docker Container gebuilded werden (Ein Container ist also eine Instanzierung eines Images), welche dann auf jedem Gerät lauffähig ist,

6 https://plugins.jetbrains.com/plugin/4509-statistic
7 https://about.gitlab.com/
8 https://gitlab.dev.ifs.hsr.ch/
9 https://www.docker.com/

auf welchem Docker installiert ist. Oder anderst ausgedrückt, Docker isoliert eine Applikation von ihrer Umgebung und löst so das ït works on my machine" Probblem. [10]

Gitlab CI/CD

Die GitLab CI/CD ist eine der zusätzlichen Features die GitLab von einem generischen Git-Server abheben. GitLab CI/CD bietet die Möglichkeit, Instruktionsfolgen zu definieren, die sowohl manuell als auch automatisch ausgeführt werden können. Bei der automatischen Auslösung sind sowohl zeitliche als auch trigger-on-commit Verfahren möglich. Die Instruktionsfolgen sind in Stages unterteilt. Jede Stage definiert eine Instruktionsfolge und eine Umgebung, in der die Instruktionen ausgeführt werden sollen. Zur Definition der Umgebung können Docker Images verwendet werden. Die eigentliche Ausführung der Instruktionen wird auf GitLab Runnern durchgeführt. Resultate solcher Stages können als Artefakte deklariert werden. Solche Artifakte können sowohl über das GitLab Web-UI heruntergeladen als auch von anderen Stages wiederverwendet werden. Weiter können einzelne Stages so definiert werden, dass diese nur für Commits auf bestimmten Refs, also Branches/Tags möglich sind.

Gitlab Runner

Die Runner können sowohl auf der gleichen Host-Maschine wie die GitLab Instanz als auch auf einer komplett anderen Host-Maschine laufen. Die CI/CD Instruktionen rufen einen Runner für die eigentliche code-execution auf. Dabei werden die Instruktionen und die Umgebungsdefinition einer Stage an einen Runner übergeben, dieser lädt zuerst die Umgebung und führt dann die Instruktionen aus.

GitLab Container Registry

Die GitLab Container Registry, sollte eigentlich eher GitLab Image Registry heissen, den in dieser Regestry können Docker Images hinterlegt, also registriert werden. Ist ein Image erst einmal in der Registry hinterlegt, kann das Image über die Standard Docker Befehle ge-pullt werden. Um den Datenschutz zu gewährleisten ist dafür zwar eine Authentifizierung gegenüber der Registry nötig, diese ist durch die in die CI/CD eingebauten Authorisierungs-Token jedoch einfach umzusetzen. Aber auch von außerhalb der CI/CD Umgebung kann auf diese Registry problemlos, wen auch mit etwas mehr Konfigurationsaufwand zugegriffen werden.

D.5.1. Unsere CI/CD Pipeline

Unsere Pipeline basiert auf vier Stages: test, build, prepare-deploy und deploy. Diese vier Stages werden nun kurz erläutert.

10 `https://www.docker.com/`

Test

Die Test Stage, kümmert sich darum die Integrität der Codebasis zu überprüfen, dabei werden die vordefinierten UnitTests ausgeführt. Das ganze geschieht über das laden einer Gradle Umgebung und das ausführen des Gradle Tasks **test**. Die dabei entstehende Test Zusammenfassung haben wir als Artifakt deklariert. GitLab erkennt die auf dem jUnitTest-Framework basierenden Testresultate und parst die Zusammenfassung, sodass diese bequem im Web-UI betrachtet werden kann.

Build

Die Build Stage, kümmert sich, wie der Name schon sagt um das Builden der Applikation. Dabei wird wieder zuerst mithilfe von Docker eine Gradle Umgebung geladen. Sobald die Umgebung geladen ist, wird der Gradle Tasks **bootJar** ausgeführt. Dadurch builded Gradle eine FatJar. Diese Fatjar wird als Artifakt gespeichert.

Prepare Deploy

Die Prepare Deploy Stage, bereitet das Ausliefern der Applikation vor. Dafür wird als erstes die in der Build Stage gebuildete FatJar bezogen und eine Verbindung zur GitLab Container Registry aufgebaut. Als nächstes wird das Docker Image gebuilded und in der Registry hinterlegt. Diese Stage wird nur für die drei Hauptbranches **develop, test und master** ausgeführt. Zusätzlich haben wir uns einen Tag **test-deploy** eingeführt. Ist ein Commit mit dem test-deploy Tag getaggt, ist dieser Commit ebenso für das automatische Deployment qualifiziert. Dies ermöglichte es uns, während der Entwicklung mit minimalen Zusatzaufwand unseren Code auch ausserhalb der localhost Umgebung manuell testen zu können.

Deploy

Und zu guter Letzt wird in der Deploy Stage die Applikation in Form eines Docker Images bequem aus der GitLab Container Registry bezogen. Dabei wird diese Stage nur, für vier Git-Refs, für die auch schon prepare-deploy ausgeführt wurde, durchgeführt.

E. Testprotokoll

E.1. Einführung

E.1.1. Version

Version 1.1, 16. Dezember 2020

E.1.2. Zweck

Spezifiziert die System-, Performance- und Usabilitytests von miro2cml.

E.1.3. Gültigkeitsbereich

Das Dokument beschränkt sich auf die Dauer der Studienarbeit im HS2020/21.

E.1.4. Referenzen

Die Referenzen sind als Fussnoten angegeben.

E.2. Grundlage für die Erstellung der Tests

Für die Grundlage der Erstellung der Tests dient das Dokument Anforderungsspezifikationen. Sowie zusätzlich die Sitzungsprotokolle mit den definierten Landing Zones.

E.3. Systemtest

Die Systemtests sind manuell durchgeführt worden auf dem *develop* Branch (lokal).

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
T01	User kann sich bei Miro authentisieren über miro2cml	Die Benutzerdaten werden über Session gespeichert.	Saskia	16.12	erfolgreich

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
T02	User kann seine Boards und den Boardtyp für die Konvertierung wählen	Dem Inputformular entsprechend findet die Konvertierung mit dem gewählten Board und Boardtyp statt.	Saskia	16.12	erfolgreich
T03	User kann sein korrektes InputBoard für UserStories konvertieren	Anhand des Inputsboards wird ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich
T04	User kann sein korrektes InputBoard für EventStorming konvertieren	Anhand des Inputsboards wird ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich
T05	User kann sein korrektes InputBoard für BoundedContextCanvas konvertieren	Anhand des Inputsboards wird ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich
T06	User kann sein korrektes InputBoard für UserStories konvertieren mit Eduguessed	Anhand des Inputsboards wird die richtige Konvertierung gewählt und ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich
T07	User kann sein korrektes InputBoard für EventStorming konvertieren mit Eduguessed	Anhand des Inputsboards wird die richtige Konvertierung gewählt und ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich
T08	User kann sein korrektes InputBoard für BoundedContextCanvas konvertieren mit Eduguessed	Anhand des Inputsboards wird die richtige Konvertierung gewählt und ein korrekter CML Output generiert.	Saskia	16.12	erfolgreich

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
T09	User kann sein teils korrektes InputBoard für UserStories konvertieren und erhält ein konstruktives Feedback	Der korrekte Teil des Boards wird vollständig konvertiert, für den Teil der falsch ist, gibt es eine konstruktive Rückmeldung an den Benutzer.	Saskia	16.12	erfolgreich, im LogFile genau Angaben wo der Fehler liegt
T10	User kann sein teils korrektes InputBoard für EventStorming konvertieren und erhält ein konstruktives Feedback	Der korrekte Teil des Boards wird vollständig konvertiert, für den Teil der falsch ist, gibt es eine konstruktive Rückmeldung an den Benutzer.	Saskia	16.12	teilweise, bei einem semantisch falschen Inputboard wirft der Serializer einen Fehler deshalb konstruktives Feedback beschränkt
T11	User kann sein teils korrektes InputBoard für BoundedContext-Canvas konvertieren und erhält ein konstruktives Feedback	Der korrekte Teil des Boards wird vollständig konvertiert, für den Teil der falsch ist, gibt es eine konstruktive Rückmeldung an den Benutzer.	Saskia	16.12	erfolgreich, im LogFile genaue Angaben wo der Fehler liegt
T12	User erhält eine sinnvolle Rückmeldung, wenn er ein Board konvertieren möchte, dass nicht dem Input entspricht.	Es wird eine sinnvolle Rückmeldung an den Benutzer zurück gegeben.	Saskia	16.12	teils erfolgreich, erhält Meldung, dass das Board nicht dem Input entspricht, die Mappingfehler geben einen Hinweis darauf, wo das Problem liegen könnte
T13	User kann mehrere UserStories Templates auf einem Board haben.	Es werden alle Templates vollständig übersetzt.	Saskia	16.12	erfolgreich, Landingzone target erreicht

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
T14	User kann mehrere EventStorming Templates auf einem Board haben.	Es werden alle Templates vollständig übersetzt.	Saskia	16.12	funktioniert nicht, Landingzone minimal erreicht
T15	User kann mehrere BoundedContext Canvas Templates auf einem Board haben.	Es werden alle Templates vollständig übersetzt.	Saskia	16.12	nur ein Template wird übersetzt, Landingzone minimal erreicht
T16	Pro Template: User kann mehrere Boards zu einem CML File konvertieren	Es werden alle Boards vollständig in CML konvertiert.	Saskia	16.12	wurden nicht implementiert (Landingzone maximal)

E.4. Usability- und Performancetests

Usabilitytest wurden in erster Linie mit dem Betreuer Olaf Zimmermann und mit dem Context Mapper Experten Stefan Kapferer durchgeführt. Wöchentlich wurde der aktuelle Projektstand als Demo vorgeführt oder als Link zum Testen abgegeben. Das Feedback, dass wir von ihnen erhalten haben, wurde direkt in den Arbeitspakten als Issues aufgenommen und dementsprechend umgesetzt. Die in der Tabelle aufgeführten Tests dienen zur Überprüfung, ob die nicht funktionalen Anforderungen erfüllt worden sind. Die Systemtests sind manuell durchgeführt worden auf dem *develop* Branch (lokal). Für die Zeitmessung wurden im Browser die Entwicklungswerkzeuge verwendet.

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
NFR01	Nutzerfehlerschutz, Inputvarianz	Der Nutzer wird bei der Erkennung von falsch formatiertem Input gewarnt. Der fehlerhafte Teil des Inputs soll ignoriert werden, der Rest des Inputs soll jedoch trotzdem gemappt werden.	Saskia	16.12	mit den definierten Testboard immer der Fall
NFR02	Erlernbarkeit	User durchläuft innerhalb von 10 Minuten die Applikation und hat am Schluss sein Miro-Board in CML umgewandelt.	Saskia	16.12	erfolgreich

Nr	Beschreibung	erwartetes Verhalten	Pers	Dat.	Result
NFR03	Bedienbarkeit	Der User kann für jeden Teil der innerhalb von 3 Minuten die notwendigen Schritte erkennen und durchführen um den Verlauf der Programmnutzung voranschreiten zu lassen.	Saskia	16.12	erfolgreich
NFR04	Performance	User erhält innerhalb von 5 Minuten oder weniger nach Start des Konvertierungsvorgangs die Möglichkeit den CML Output herunterzuladen.	Saskia	16.12	mit den definierten Testboard immer der Fall
NFR05	Portabilität	Der Initialisierungsaufwand für die gängigen Betriebssysteme unterscheidet sich um nicht mehr als 2 Minuten	Saskia	16.12	erfolgreich, weil Browseranwendung bei allen System gleich lange (getestet mit Windows und Ubuntu)

NFR02 und NFR03 wurde zur Überprüfung mit zwei Mitstudenten durchgeführt. Die Mitstudenten kannten Context Mapper und Miro bereits aus der Vorlesung. NFR02 dauerte 5-8 Minuten. NFR03 dauerte 2-3 Minuten.

F. User Guide

F.1. Introduction

Miro2cml is a web application that converts MiroBoards into Context Mapper Language (CML). Miro2cml works with MiroTemplates and corresponding mapping heuristics, which allow an appropriate translation to CML.

F.1.1. Precondition

To be able to follow this user guide, we expect you to be familiar with Miro and the concepts of Tactic and Strategic Domain Driven Design.
Links:

- How to Start Collaboration with Miro
- An Introduction to Domain-Driven Design
- Context Mapper

F.1.2. Supported Templates

Three templates are supported by miro2cml, they are listed below:

- User Stories
- Event Storming
- Bounded Context Canvas

F.1.3. Overview

- How to get started
- Tutorials: Each chapter is dedicated to each template and contains four sections:
    1. Step: Create the MiroBoard
    2. Step: Convert the Board
    3. Step: How to use the CML
    4. Frequently asked Questions
- Mapping Tables

F.2. How to get started

The first thing you need to do is authenticate yourself with Miro. Afterwards your boards will appear in the gray list. Choose your Board Type and select your MiroBoard you like to convert, as shown in the figure F.1.

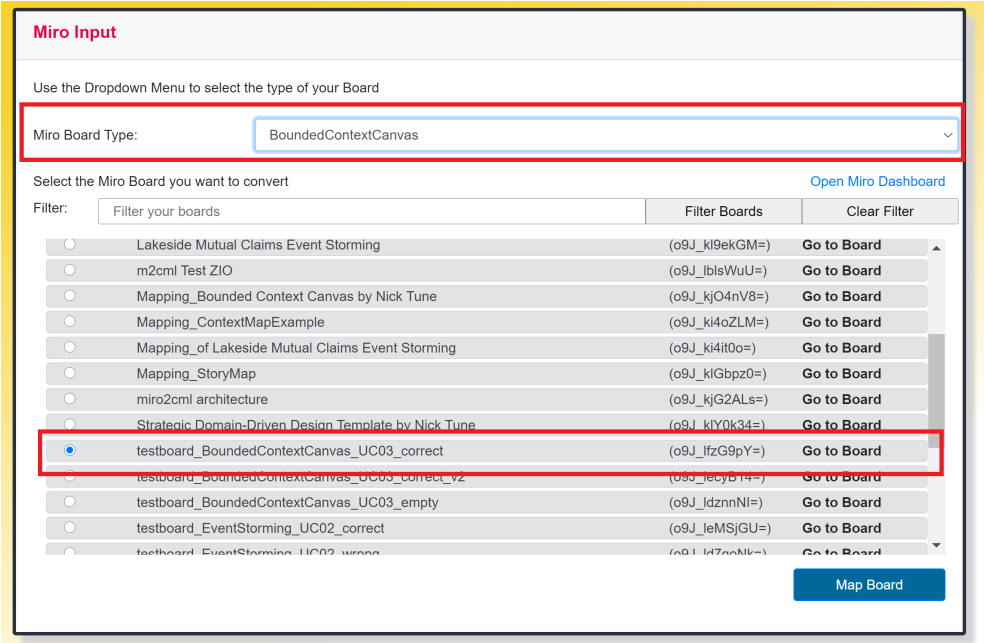


Figure F.1.: Board selection form

With the Button Map Board you can convert your board. When the conversion is finished a new field appears: Context Mapper Ouput. If the first section is green, the input board was perfect. In the figure F.2 you can see such an ouput. If the first section is orange there are some small errors. If it is red, the board could not be converted. In the log file as well as in the colored section are the information what went wrong. The preview shows the CML output or the log File, if something went wrong. The files can be downloaded via the buttonn Download Logfile and Download CML.

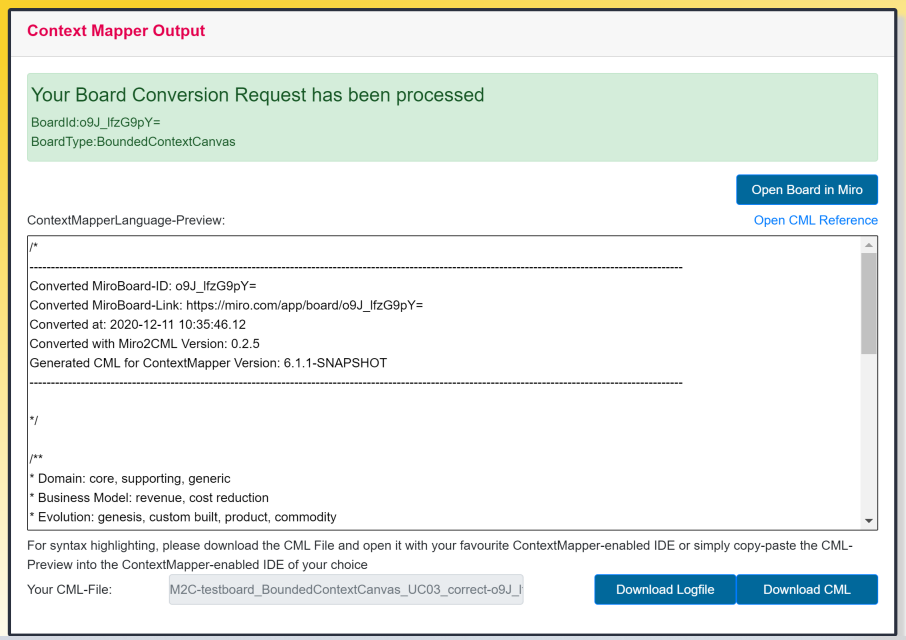


Figure F.2.: Example Context Mapper Output

F.2.1. Board Type: EducatedGuessed

There is a possibility to set the board type to EducatedGuessed. The converter then decides which template was used and maps the board accordingly.

F.3. User Stories

F.3.1. Step One: Create the MiroBoard

First you select in Miro the template User Story Map Framework. The template can be found in the section "Chose a Template" in Miro. The preview of the template is. The preview of the template is shown in the figure F.3.

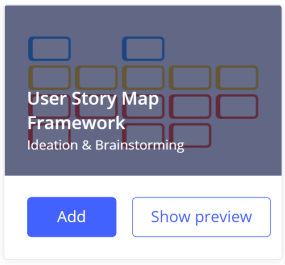
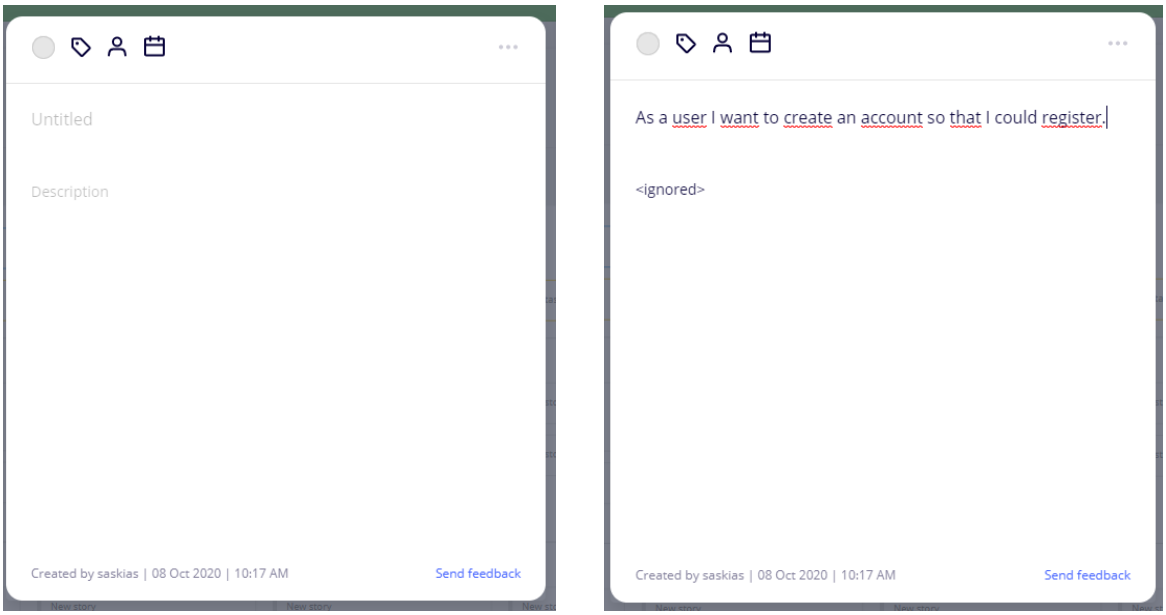


Figure F.3.: Template: User Story Map Framework

Then you can fill in the User Stories. The miro2cml converter notes only the title of the Cards, you could use the rest of the Cards like you want. In the picture is an empty and a filled out Card to see.



(a) Empty Card (b) Filled out Card

Figure F.4.: Example for Cards

Make sure that you use one of these formats for the User Stories:

- As a <actor> I want to <verb> a <entity> so that <benefit>.
- As an <actor> I want to <verb> an <entity> so that <benefit>.
- As a <actor> I want to <verb> an <entity> so that <benefit>.
- As an <actor> I want to <verb> a <entity> so that <benefit>.

Listing F.1: Possible formats for User Stories

Objects	Allowed Characters	Java Regex
< actor >	upper and lowercase letters, numbers	[A-Za-z0-9]+
< verb >	optional leading lowercase letters followed by upper and lowercase letters, numbers	[a-z]?[a-zA-Z0-9]+
< entity >	upper and lowercase letters, numbers	[a-zA-Z0-9]+
< benefit >	upper and lowercase letters, numbers, point, comma, colon space, apostrophe	[A-Z,. ':a-z0-9]+

The yellow and blue Cards are not converted, in the figure F.5 red bordered. They are used as Tasks and Activities in the Template and therefore are not relevant for the mapping.

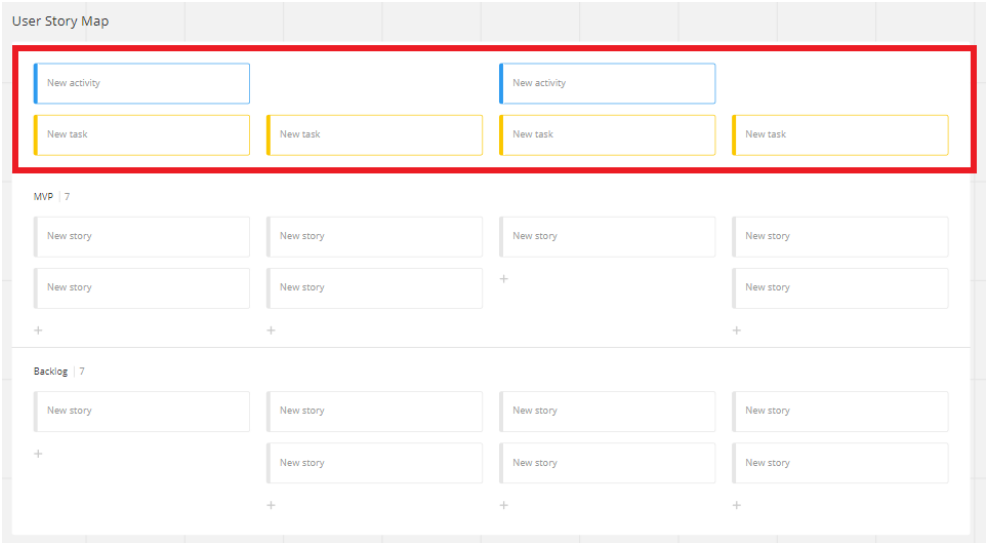


Figure F.5.: Empty User Story Map Template

F.3.2. Step Two: Convert the Board

After you have created your board, you can convert it with the miro2cml Tool. Choose your board and the Template: User Stories. For the example in the figure one the following output will be generated:

```
UserStory createaccount {
    As a "User"
      I want to create an "account"
      so that "I could register".
}
```

Listing F.2: Example CML Output for User Stories

The name of the User Story is composed of the verb and the entity. The rest is the same as your input. Take a look at the subsection F.6.1 mapping table for further informations.

F.3.3. Step Three: Use the CML ouput

The CML File with your User Stories is a good start for design work with the Context Mapper Tool. Follow the Link for a step by step tutorial for further design work. You could skip the first step because you already have your Requirements defined.
If you're new to the Context Mapper here a Link to a guide.

F.3.4. Frequently Asked Questions (FAQ)

Which inputs are converted?

The User Story Map contains only Cards. Therefore just Cards will be converted. All other Miro elements are ignored. The Cards which are yellow and blue are ignored because they illustrate the Tasks and Activites in the User Story Map. The title of the Cards are relevant for the mapping. The description and the other fields of the Card will be ignored.

Why are my Cards not converted?

- The Cards are yellow or blue.
- They have not the correct text format. Check if you use the correct format like introduced in the Listing F.1.
- The User Stories are not in the title defined of the Miro Cards.

How is the title of the User Story generated?

The title is the verb concatenated with the entity of the User Story.

Why are some numbers added to the title of the User Story?

The Context Mapper expect unique titles for the User Stories. If there are exists equal titles the miro2cml converter add an 1 at the end of the name to make sure that the conversion produce a correct CML File.

Why do some verbs have quotations marks at the beginning and at the end?

The Context Mapper knows the CRUD verbs (create, read, update, delete). Therefore they do not need quotation marks. All other verbs are between the quotations marks to mark them as a string.

F.4. Event Storming

F.4.1. Step One: Create the Board

The mapping rules for the Event Storming are based on the Event Storming Cheatsheet form the DDD-Community. For the Event Storming is no Miro Template available. The origin of the template idea came from a sample event storming from Lakeside Mutual. So we start with creating a new MiroBoard. First we have to define the Stickers colors for the Domain Event, Command, Aggregate, Issue and User Role,like you see in the picture. The miro2cml start the mapping with the five defined Stickers, it is important that these Stickers have different colors and are written correctly. (The Cheatsheet also describes the Policy and System Stickers, which are not supported in the miro2cml Converter. You are allowed

to use those Stickers but they will be ignored in the convert process. Please use for these Stickers different colors, than the already defined Stickers.

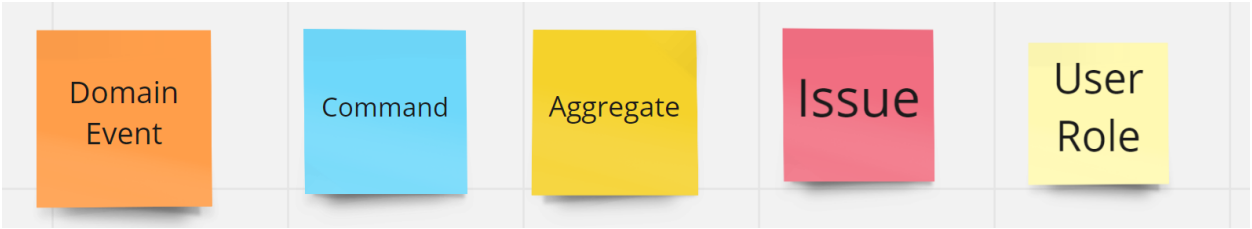


Figure F.6.: Required Stickers for Event Storming Template

Then you can start with creating your Event Storming Diagram. We have created some extra rules, which are described in the next list, to make sure, that the Board conversions results in a meaningful CML ouput.
Group the different phases of your event as described together:

- one Domain Event and one Command
- one or more Aggregates
- optional one User Role
- optional one Command that triggers the Domain Event (incomming arrow)
- optional the Command triggers one or two other Domain Event (outgoing arrow)

In the figure F.7 you see an example of an Event Storming Board. You see some of the different formats for the phases. The Stickers that belong together have to be positioned near each other.The Issue Stickers are used as comments. They will also be converted by miro2cml.

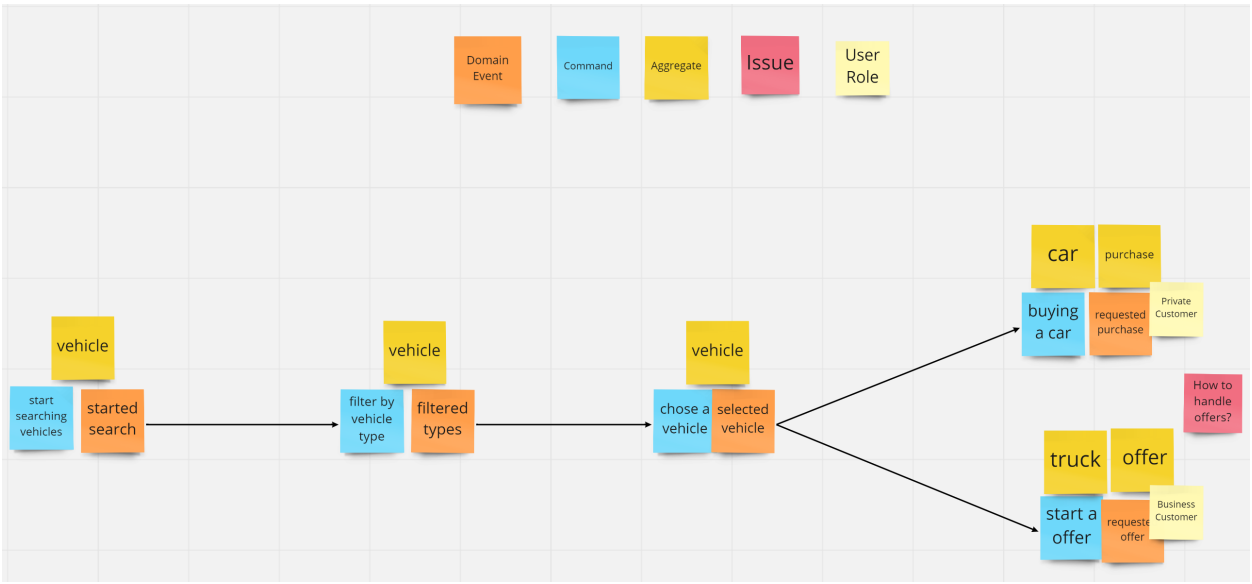


Figure F.7.: Example of Event Storming Board

In order that the convert could recognize the arrows, the arrows must start at the Domain Event Sticker and end at the Command Sticker.

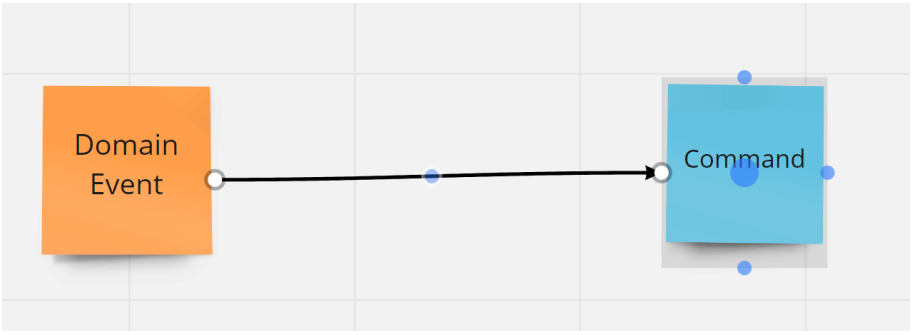


Figure F.8.: How to use the arrows correctly

F.4.2. Step Two: Convert the Board

After you have created your board, you can convert it with the miro2cml Tool. Choose your board and the Template: Event Storming.
The example returns the CML Code below. The Issues Stickers are in the comment at the beginning of the file. In the Flow section are your Domain Events, Commands and arrows represented. The Aggregates are also defined. The first (most left) aggregate of a phase contains the corresponding Domain Events. The Commands are defined in the application.

```
/* Issue: How to handle offers */
BoundedContext EventStormingBoundedContext {

 Application {
        CommandEvent start_searching_vehicles
        CommandEvent filter_by_vehicle_types
        CommandEvent choose_a_filter
        CommandEvent start_a_offe
        CommandEvent buying_a_car
        Flow {
        command start_searching_vehicles emits event started_search
        event started_search triggers command filter_by_vehicle_types

        command filter_by_vehicle_types emits event filtered_types
        event filtered_types triggers command choose_a_filter

        command choose_a_filter emits event selcted_vehicle
        event selcted_vehicle triggers
                command buying_a_car x start_a_offer

        command buying_a_car [triggered by "Private Customer"]
         emits event requested_purchase
        command start_a_offer [triggered by "Business Customer"]
         emits event requested_offer
  }
 }

 Aggregate vehicle {
  DomainEvent started_search
  DomainEvent filtered_types
  DomainEvent selcted_vehicle
  }
 Aggregate car {
  DomainEvent requested_purchase
  }
 Aggregate truck {
  DomainEvent requested_offer
 }
Aggregate purchase
Aggregate offer
}
```

Listing F.3: Example CML Output for Event Storming

F.4.3. Step Three: Use the CML ouput

After you have created the CML ouput you could make more specifications in the CML model. Here is a Link to a Event Storming tutorial from the context mapper tool.

F.4.4. Frequently Asked Questions (FAQ)

Why are my Stickers not converted?

- The color of the Sticker is not defined as a Event Storming element.
- They have not the correct formatted. Check if you use the correct format like introduced in the steps one to three. The Stickers from the same step should be near each other.
- The semantic model is not correct.

What does a serializer error mean?

Something is wrong with your event storming model. Take a look at our tutorial to make sure, that you put in a correct model. Each step must contain a Command, Domain Event and an Aggregate. If the model is correct, make sure that the Sticker are recognized. Take a look at the logfile for further information's about the conversion. The Stickers from the same step should be near each other. If they are right connected you will find a log in the logfile with group found.

Why is my board not recognized?

The Event Storming Template expect you to define the colors for the Commands, Aggregates, Domain Events, User Role and Issue with Stickers and the key words. Make sure that all Stickers have a different color. If it still does not work, then verify if the key words are correct written.

How to determine which Aggregate contains the Domain Events?

Each Group which contain a Domain Event and a Command. Expect minimal one Aggregate. If there are more than one Aggregates the leftest one contains the Domain Events.

F.5. Bounded Context Canvas

F.5.1. Step One: Create the Board

First you have to select the Template The Bounded Context Canvas by Nick Tune. It is available in Miroverse. You can access Miroverse from the Choose a templates area in the section Categories . The button to access Miroverse is red colored in the figure F.9.

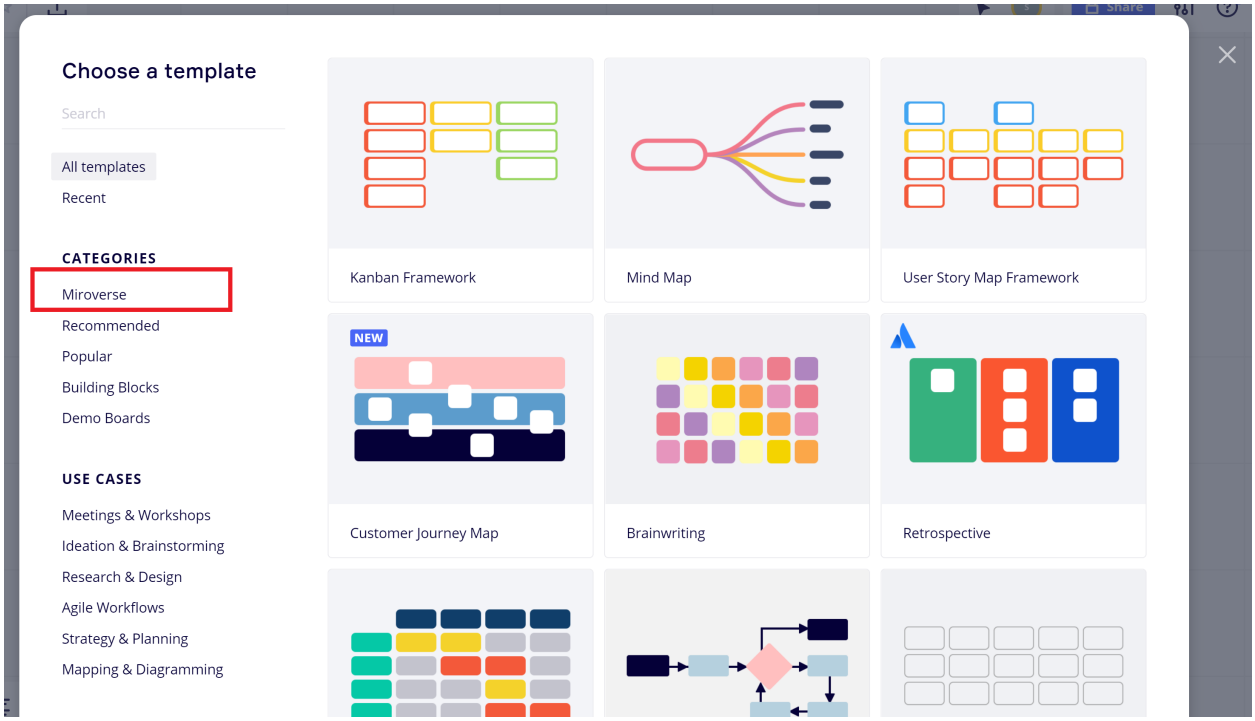


Figure F.9.: Miro Templates access to Miroverse

The easiest way to find the template in Miroverse is to use the search field, as can be seen in the figure F.10.

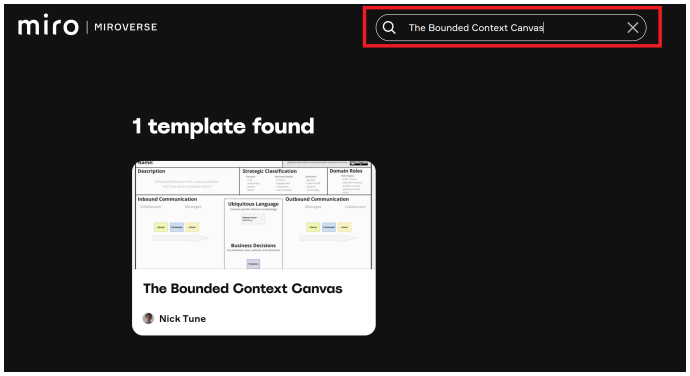


Figure F.10.: Miroverse: Tempalte The Bounded Context Canvas

Then you fill in your specifications about the Bounded Context, which you want to model. For further information about the template take a look at the corresponding instructions. It is important that you do not change the headings in the Template, because they are used for the conversion. Also take care that you leave the following hyphen - , how they are, in the field Strategic Classification and in the field User Role. In the picture F.11 is a correct filled out template illustrated.

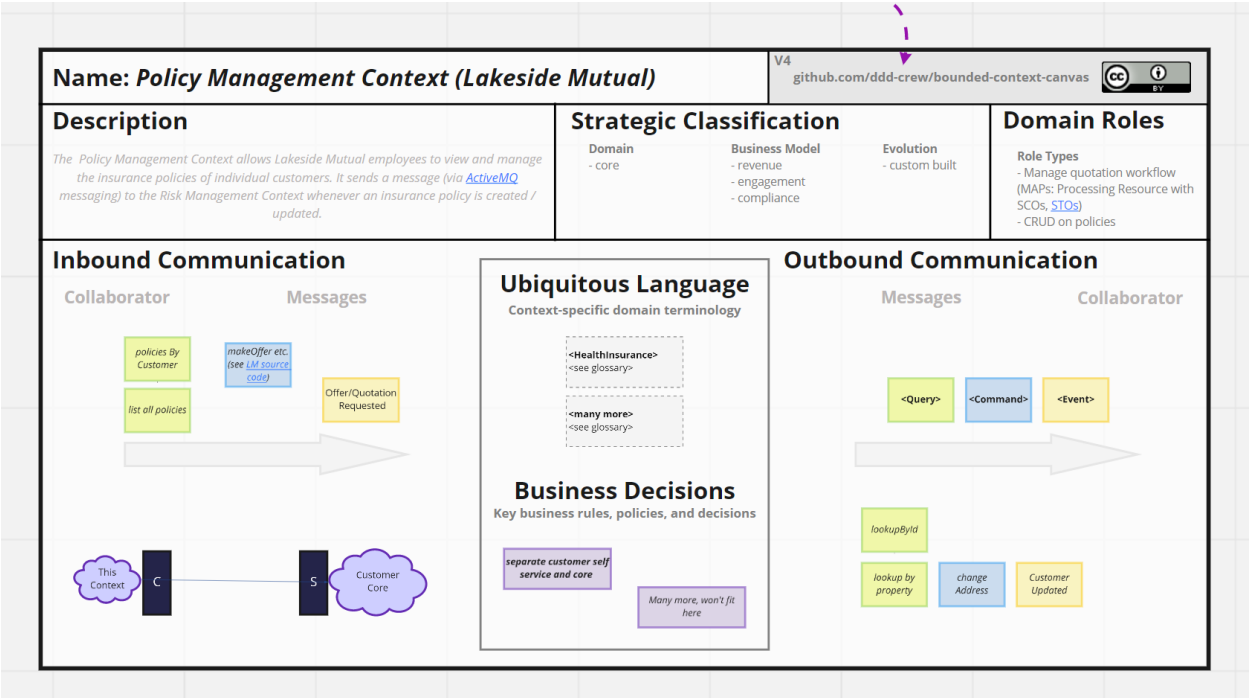


Figure F.11.: The Bounded Context Canvas Example

F.5.2. Step Two: Convert the Board

After you have created your board, you can convert it with the miro2cml Tool. Choose your board and the Template: Bounded Context Canvas. All three fields in the Strategic Classification, the Ubiquitous Language, the Business Decisions and the Outbound Communications will be converted in to a comment. The Name is converted in the name of the Bounded Context. The description is represented in the domain Vision Statement. The Role Types are interpreted as responsibilities. The type and the knowledge level are defined by the converter. The Knowledgelevel is CONCRETE and the type is FEATURE. In the Bounded Context is a aggregate which contains the Inbound Communications as Domain Events and Entities which have the Command and Query functions.
The example from the picture will produce this output:

```
/**
* Domain: core
* Business Model: revenue, engagement, compliance
* Evolution: custom built
* Ubiquitous Language: <HealthInsurance> <see glossary>,
      <many more> <see glossary>
* Business Descisions: separate customer self service and core  ,
       Many more, won't fit here
* Outbound Communications: lookup by property,
       change Address, Customer Updated, lookupById
*/
BoundedContext Policy_Management_Context_Lakeside_Mutual {
      domainVisionStatement "The  Policy Management
            Context allows Lakeside Mutual employees to view
            and managethe insurance policies of
            individual customers.
            It sends a message (via http://activemq.apache.org
            /ActiveMQ messaging) to the Risk Management
            Context whenever an insurance policy is
            created / updated."
      type FEATURE
      responsibilities "- Manage quotation workflow
            (MAPs: Processing Resource with SCOs,
            https://microservice-api-patterns.org/
                  patterns/responsibility/
                  operationResponsibilities/
                  StateTransitionOperationSTOs)" ,
            "- CRUD on policies"
      knowledgeLevel CONCRETE
Aggregate Policy_Management_Context_Lakeside_Mutual_Aggregate {
      DomainEvent OfferQuotation__Requested
      Entity Policy_Management_Context_Lakeside_Mutual_Queries {
            def String policies_By_Customer;
            def String list_all_policies;
      }
      Entity Policy_Management_Context_Lakeside_Mutual_Commands {
            def String makeOffer_etc_see_LM_source_code;
      }
}
}
```

Listing F.4: Example CML Output for Bounded Context Canvas

F.5.3. Step Three: Use the CML ouput

After you have created the CML ouput you can make more specifications in the Bounded Context. Here is a Link to the context mapper specifications.

F.5.4. Frequently Asked Questions (FAQ)

Why are my fields not recognized?

The fields are identified with text search by the titles. You should not replace the titles with your own ideas. The fields Strategic Classification and Role Types contains listings that begin with a hyphen. With the hyphens the mapping can distinguish the contents for this reason the layout of the fields should not be adjusted. For further informations take a look at the subsection F.6.3 mapping table.

Why are my inbound and outbound communications elements not recognized?

The fields Event, Command and Query must have the same color as in the template defined. The easiest way to use this elements is to copy the given fields and adjust the content

Why is the field Description not found?

The text field bellow the heading Description must be between the Description heading and the Inbound Communication heading. It also must be left of the Field Domain.

F.6. Mapping Tables

F.6.1. User Story Map: Mapping Table

– As a <actor> I want to <verb> a <entity> so that <benefit>.
– As an <actor> I want to <verb> an <entity> so that <benefit>.
– As a <actor> I want to <verb> an <entity> so that <benefit>.
– As an <actor> I want to <verb> a <entity> so that <benefit>.

Listing F.5: Supported User Story Formats

Objects	Allowed Characters	Java Regex
< actor >	upper and lowercase letters, numbers	[A-Za-z0-9]+
< verb >	optional leading lowercase letters followed by upper and lowercase letters, numbers	[a-z]?[a-zA-Z0-9]+
< entity >	upper and lowercase letters, numbers	[a-zA-Z0-9]+
< benefit >	upper and lowercase letters, numbers, point, comma, colon space, apostrophe	[A-Z,. ':a-z0-9]+

Card Title	Card Color (Hex)	Context Mapper Output

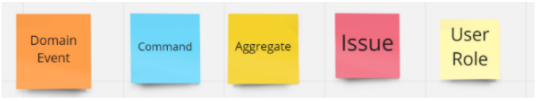
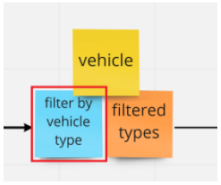
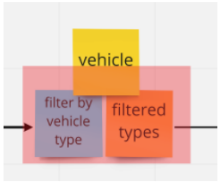
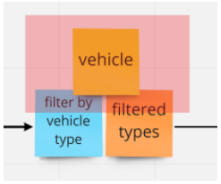
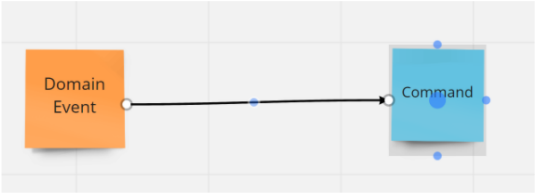
As an User I want to create a new album so that I am able to collect my songs.	red (#FF5733)	`UserStory createnewalbum {`
		`As a "User"`
		`I want to create a "new album"`
		`so that "I am able to collect my songs".`
		`}`

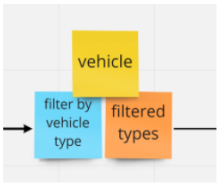
As a Customer I want to buy a song so that I am able to add it to my list.	green (#0FAF51)	`UserStory buysong {`
		`As a "Customer"`
		`I want to "buy" an "song"`
		`so that "I am able to add it to my list".`
		`}`

As an User I want to create a new album so that I am able to collect my songs.	yellow (#fbc800)	no output, because the card is yellow
As an User I want to create a new album so that I am able to collect my songs.	blue (#2d9bf0)	no output, because the card is blue

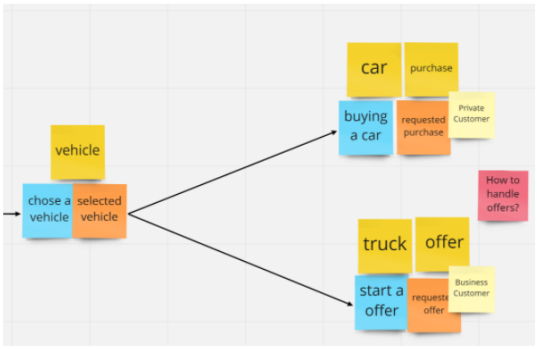
Card One: As a Customer I want to buy a song so that I am able to add it to my list.
Card Two: As a Singer I want to buy a song so that I could use its lyrics.
	both green (#0FAF51)	`UserStory buysong {`
		`As a "Customer"`
		`I want to "buy" an "song"`
		`so that "I am able to add it to my list".`
		`}`
		
		`UserStory buysong1 {`
		`As a "Singer"`
		`I want to "buy" an "song"`
		`so that "I could use its lyrics".`
		`}`

F.6.2. Event Storming: Mapping Table

Miro Input Elements	Context Mapper Specification/Output
	set the colors: Command -> blue, DomainEvent -> orange, Aggregate -> yellow, User Role -> light yellow, Issue -> red
	The Commands are the start point for the mapping. For the commands are corresponding Domain Events, Aggregates and User Stories searched.
	The Mapper searches for one Domain Event in the red marked area depending on the command.
	The Mapper searches for Aggregates in the red marked area depending on the command.
	The Mapper searches for one User Role in the red marked area depending on the command.
	A Domain Event could trigger a Command. This is represented with an arrow. The arrow must start at the Domain Event Sticker and end at the Command Sticker as shown.




```
application {
 CommandEvent filter_by_vehicle_type
 Flow {
    command filter_by_vehicle_type emits event filtered_types
    event filtered_types triggers command [outgoing arrow]
 }
 Aggregate vehicle{
    DomainEvent filtered_types
 }
}
```




```
/* [How to handle offers?] */
application {
 CommandEvent chose_a_vehicle
 CommandEvent buying_a_car
 CommandEvent start_a_offer
 Flow {
    command choose_a_filter emits event selected_vehicle
    event selected_vehicle triggers
        command buying_a_car x start_a_offer

    command buying_a_car [triggered by "Private Customer"]
        emits event requested_purchase
    command start_a_offer [triggered by "Business Customer"]
        emits event requested_offer
 }
  Aggregate vehicle{
     DomainEvent selected_vehicle
  }
  Aggregate car{
      DomainEvent requested_purchase
  }
  Aggregate purchase
  Aggregate truck{
      DomainEvent requested_offer
  }
  Aggregate offer
}
```

F.6.3. Bounded Context Canvas: Mapping Table

Bounded Context Canvas Element	Explanation	Context Mapper Output
Field Name: Name: Policy Management Context	The name field is converted to the name of the bounded context.	`BoundedContext Policy_Management_Context{...}`
Field Description: The Policy Management Context allows Lakeside Mutual employees to view and manage the insurance policies of individual customers.	The description of the Bounded Context is interpreted as the domain vision statement in the Bounded Context model in CML. The description field must be located between the title Description and the title Inbound Communication and must be left of the field Domain, so that the Converter recognizes it.	`domainVisionStatement "The Policy Management Context allows Lakeside Mutual employees to view and manage the insurance policies of individual customers."`
Strategic Classification: Domain - core	The input from the field domain is converted to a comment, because there is no element that matches the domain in the context mapper.	`/* Domain: core`
Strategic Classification: Business Model - revenue - engagement -compliance	The input from the field Business Model is converted to a comment, because there is no element that matches the Business Model in the context mapper.	`/* Business Model: revenue, engagement, compliance`
Strategic Classification: Evolution - custom built	The input from the field Evolution is converted to a comment, because there is no element that matches Evolution in the context mapper.	`/* Evolution: custom built`
Domain Roles: Role Types - Manage quotation workflow - CRUD on policies	The Role Types are converted to responsibilities in CML.	`responsibilities = "- Manage quotation workflow" "- CRUD on policies"`
Inbound Communications: Event (orange boxes) Offer/Quotation Requested	The Inbound Communication Events are converted to DomainEvents in CML.	`DomainEvent OfferQuotation_Requested`
Inbound Communications: Command (blue boxes) makeOffer	The Inbound Communication Commands are converted to operations/method inside an the Entity *BoundedContextName_Commands*. The return value is defined as String based on the assumption that this is the most common return value. The return value can be adjusted later by hand.	`Entity BoundedContextName_Commands{ def String makeOffer }`

Inbound Communications: Query (orange boxes) policies By Customer	The Inbound Communication Queries are converted to operations/method inside an the Entity *BoundedContextName_Queries*. The return value is defined as String based on the assumption that this is the most common return value. The return value can be adjusted later by hand.	`Entity BoundedContextName_Queries{`
		`def String policies_By_customer`
		`}`

Business Decisions: purple boxes seperate customer self service and core	The input from the Business Decisions boxes are converted to a comment, because there are no elements that matches Business Decisions in the Context Mapper.	`/* Business Decisions:` `separate customer self service and core`
Ubiquitous Language: light grey boxes <HealthInsurance>	The input from the Ubiquitous Language boxes are converted to a comment, because there are no elements that matches Ubiquitous Language in the Context Mapper.	`/* Ubiquitous Language: <HealthInsurance>`
Outbound Communications: Event, Command, Query Query: LookUpById, Event: Customer Updated, Command: Change address	The outbound communications are converted to a comment, because they represent events and operations from an other Bounded Context.	`/* Outbound Communication:` `LookUpById, Customer Updated, Change address`
no input	The type of the Bounded Context is implicitly set to FEATURE because this is the most common use case.	`type FEATURE`
no input	The knowledge level is implicitly set to concrete because this is the most common use case.	`knowledgeLevel CONCRETE`
all other elements: like Collaborator Types	These elements are ignored during conversion, because it was really difficult to identify these Types and their connections and it is not the main functionality of the template.	no CML output