Requirements and Design Documentation (RDD)

Version 0.2

ESEP – Praktikum – Sommersemester 2017

LANKE

Hartmann	Lennart	1234567	Lennart.Hartmann@haw-hamburg.de
Mendel	Alexander	2188808	Alexander.Mendel@haw-hamburg.de
Eggebrecht	Nils	1234567	Nils.Eggebrecht@haw-hamburg.de
Witte	Karl-Fabian	2246435	Karl-Fabian.Witte@haw-hamburg.de
Veit	Eduard	1234567	Eduard.Veit@haw-hamburg.de

Hamburg, den 1. Juni 2017

Änderungshistorie:

Version	Autor	Datum	Anmerkungen/Änderungen
0.1	K.Witte	04.04.2017	Aus der Vorlage (Version 0.5) von Prof Lehmann doc2tex, um es in Git besser pflegen zu können
0.1.1	K.Witte	11.04.2017	Es wurden Tabellenvorlagen für die Requirements und Use Cases hinzugefügt (Wave und Kite lvl)
0.1.2	A. Mendel	10.05.2017	Komponentendiagramm der Architektur und Beschreibung
0.1.3	A. Mendel	11.05.2017	Use Case Aktivitätsdiagramme eingepflegt

Dieses RDD Template stellt eine grobe Struktur des Dokuments dar. Sie können es (fast) nach Belieben verändern. Der rot-kursive Text soll durch Ihren Text ersetzt werden. Bitte aktualisieren sie bei jeder Meilenstein-Abnahme das Inhaltsverzeichnis.

Inhaltsverzeichnis

1 Teamorganisation

Überlegen sie, welche Regeln sie für die Zusammenarbeit aufstellen wollen und welche Rollen sie im Team verteilen wollen. Dokumentieren sie diese hier zusammen mit weiteren Anmerkungen der Teamorganisation. Listen oder Tabellen sind zum Beispiel ein kompakte und übersichtliche Darstellungsformen für diesen Bereich.

1.1 Verantwortlichkeiten

Bennen sie Verantwortliche innerhalb des Projekts (Projektleiter, Tester, Implementierer, etc.). Auch hier ist eine Listen- oder Tabellendarstellung angebracht.

1.2 Absprachen

Listen sie hier die Absprachen im Team auf, z. B. Jour Fixe, Kommunikation, Respond-Latenz, ...

1.3 Repository-Konzept

Überlegen sie sich, wie sie das Repository und die Ordner organisieren wollen. Welche Regeln wollen sie beim Umgang mit Branches, Auslieferungen, Nachrichten an den Commits usw. im Team einhalten?. Listen sie diese Absprachen hier auf. Überlegen Sie auch, wie die Arbeitsabläufe sein sollen bei der Umsetzung von Arbeitsaufträgen oder bei der Behebung von Fehlern.

2 Projektmanagement

In diesem Kapitel sollten organisatorische Punkte beschrieben und festgelegt werden.

2.1 Prozess

Legen Sie den Prozess fest, nach dem Sie das Projekt umsetzen wollen. Geben Sie ggf. grobe Schritte an, wie Planungsrunden, Sprints, oder ähnliches.

2.2 PSP/Zeitplan/Tracking

Projektstrukturplan, Ressourcenplan, Zeitplan, Abhängigkeiten von Arbeitspaketen, eventueller Zeitverzug, Visualisierung des Projektstandes, etc.

2.3 Qualitätssicherung

Überlegen sie, wie sie Qualität in ihrem Projekt sicher stellen wollen. Listen sie die Maßnahmen hier auf. Beachten sie, dass diese Maßnahmen für die unterschiedlichen Artefakte und Ebenen entsprechend unterschiedlich sein können.

3 Randbedingungen

3.1 Entwicklungsumgebung

Auflistung der Entwicklungsumgebung (Simulator, Hardware, Betriebssystem etc.)

3.2 Werkzeuge

Auflistung der im Projekt verwendeten Werkzeuge inkl. ihrer Versionen.

3.3 Sprachen

Auflistung der Programmiersprachen und Bibliotheken.

4 Requirements und Use Cases

4.1 Systemebene

Die Anforderungen aus der Aufgabenstellung sind nicht vollständig. Die Struktur der nachfolgenden Kapitel soll sie bei der Strukturierung der Analyse unterstützen. Dokumentieren Sie die Ergebnisse der Analysen entsprechend.

4.1.1 Stakeholder

Die Stakeholder dieses Projektes sind in den zwei Kategorien intern und extern unterteilt. In Tabelle **??** sind diese mit ihren Interessen aufgelistet.

Tabelle 1: interne und externe Stakeholder des Projekts

interne Stakeholder	**Interessen**
CEO (Management)	- Gewinn - Rufaufwertung der Firma - effiziente und flexible Arbeiter - Einhaltung des Zeitplans - transparenter Einblick in den Entwicklungsprozess
Developer Team	- motivierende und sinnvolle Arbeit - Gehalt - gutes Teamklima
externe Stakeholder	**Interessen**
Kunde	- Projekt hat geforderte Funktion - Projekt hat gewünscht Qualität - geringe Kosten - schnelles Ergebnis - Regelmäßige Kontrolle des Projekts (Zwischenstand) - Einflussnahme im Projekt, neue Funktionen fordern
Anwender	- einfache Bedienung - hohe Sicherheitsstandards im Betrieb
Wartungsservicekraft	- einfache Wartung - geringe Fehleranfälligkeit - hohe Sicherheitsstandards bei Wartung

4.1.2 Anforderungen

In der Aufgabenstellung sind Anforderungen an das System gestellt. Arbeiten sie diese hier auf und ergänzen sie diese entsprechend der Absprachen mit dem Betreuer. Achten sie auf die entsprechende Atribuierung. Berücksichtigen sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems.

4.1.3 Systemkontext

Use Cases werden aus einer bestimmten Sicht erstellt. Dokumentieren sie diese mittels Kontextdiagramm oder Use Case Diagramm. Die Use Cases und Test Cases müssen zu der hier verwendeten Nomenklatur konsistent sein.

4.1.4 Use Cases

Tabelle 2: KITE LVL USE CASE

Name	**Aussagekräftiger Name, oft Substantiv + Verb (Bsp: Fahrstuhl anfordern)**
Akteur	Initiator
Auslösendes Ereignis	Trigger des Prozesses
Kurzbeschreibung	2-4 Sätze

Dokumentieren sie hier, welche Use Cases sie auf der Systemebene implementieren müssen. Die Test Cases sollen später zu den Use Cases konsistent sein.

Text zu Usecases...

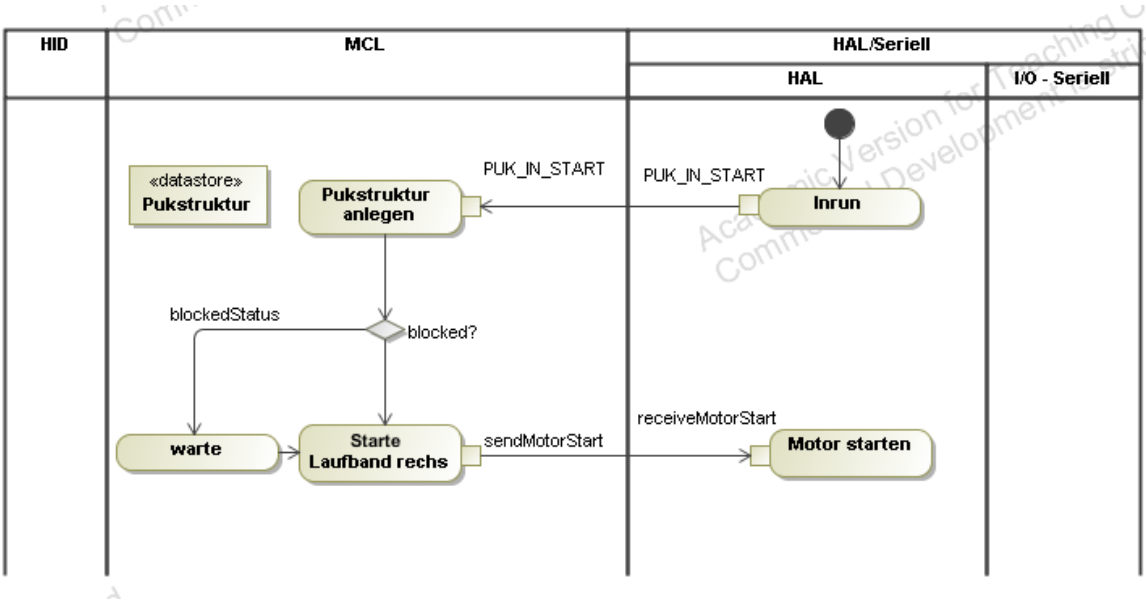

Abbildung 1: Use Case: Puk im Einlauf `muss hier nicht auch Rübergeschrieben werden, falls EinlaufBlocked?`

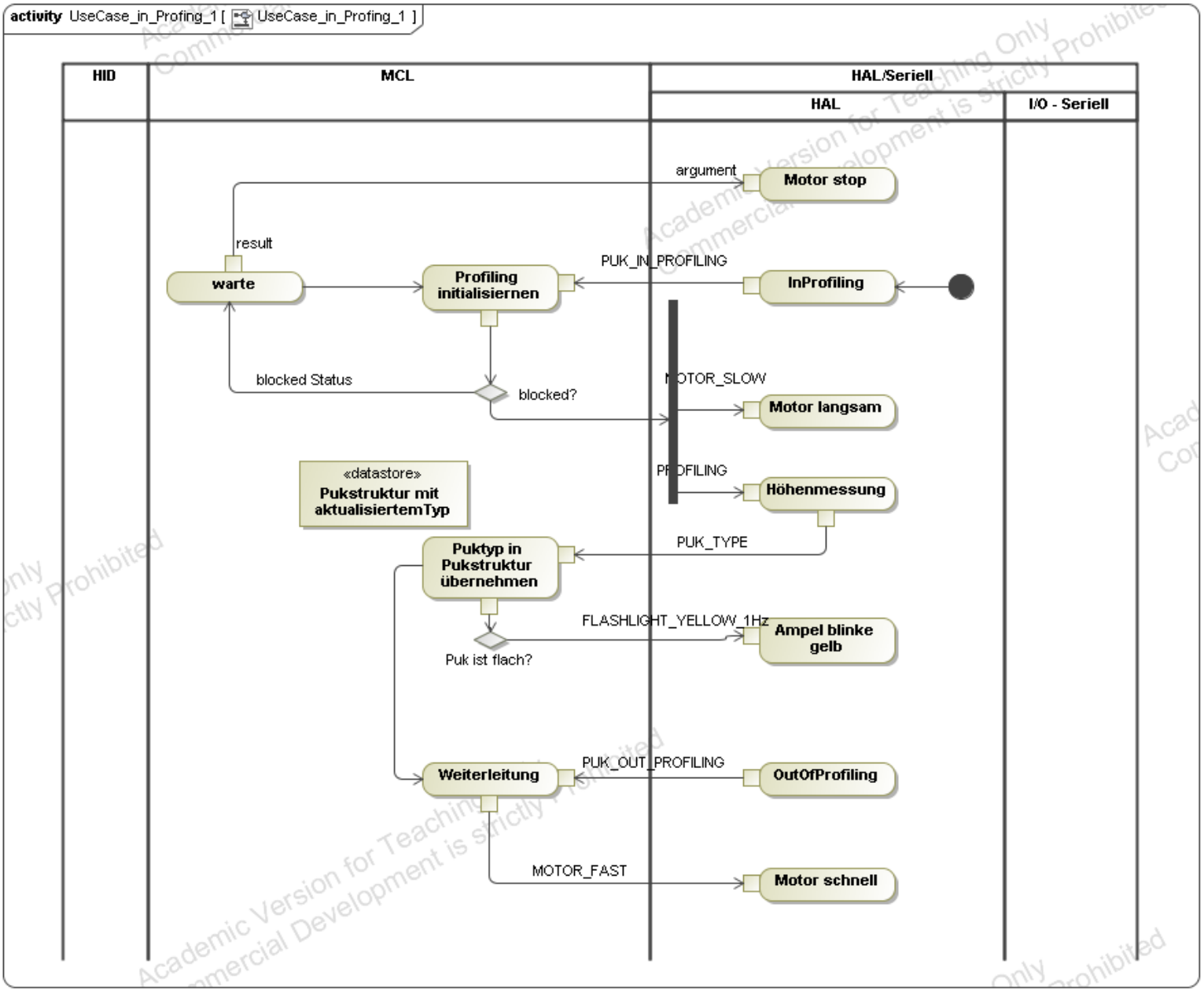

Abbildung 2: Use Case: Puk in Höhenmessung (Profiling)

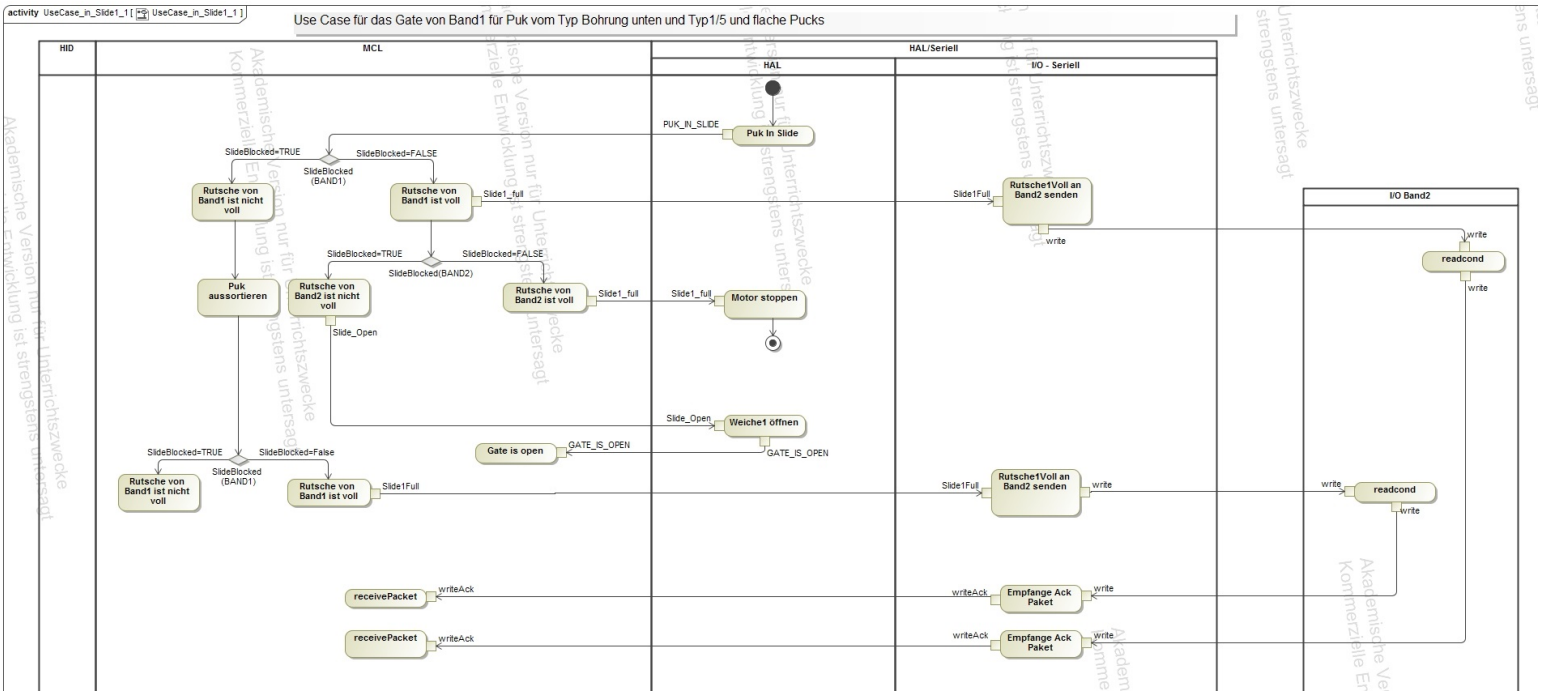

Abbildung 3: Use Case: Puk (Flach/Typ1/Typ5) in Weiche (Band1)

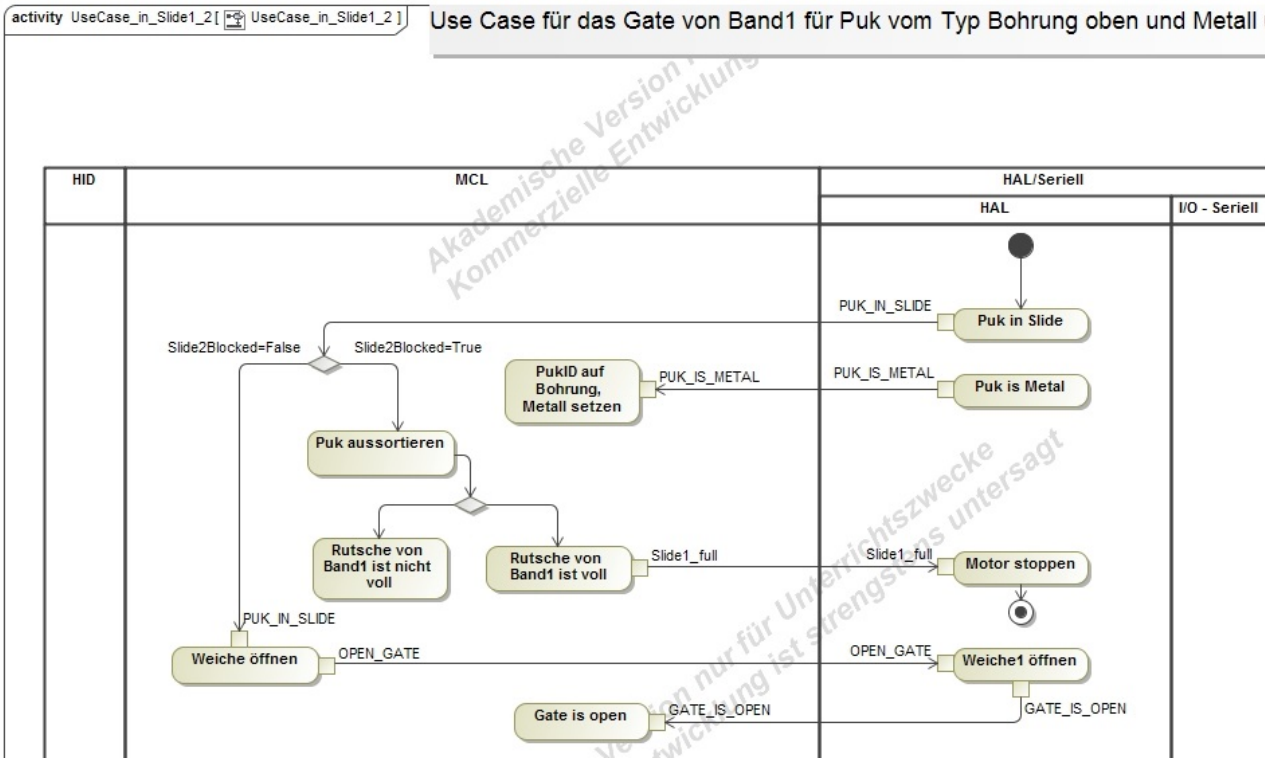

Abbildung 4: Use Case: Puk (Bohrung/Metall/Typ2/Typ4) in Weiche (Band1)

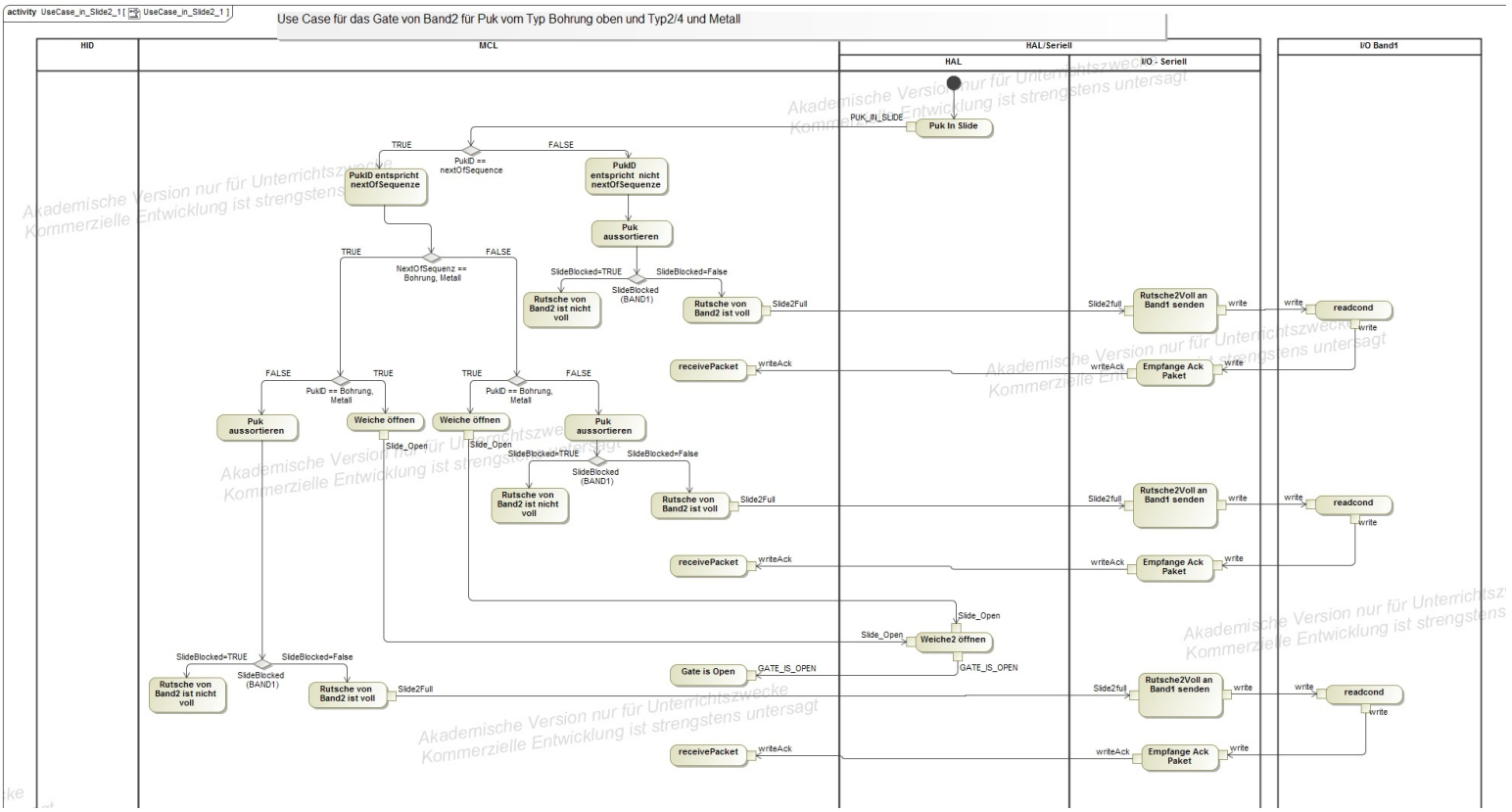

Abbildung 5: Use Case: Puk (Bohrung/Metall/Typ2/Typ4) in Weiche (Band2)

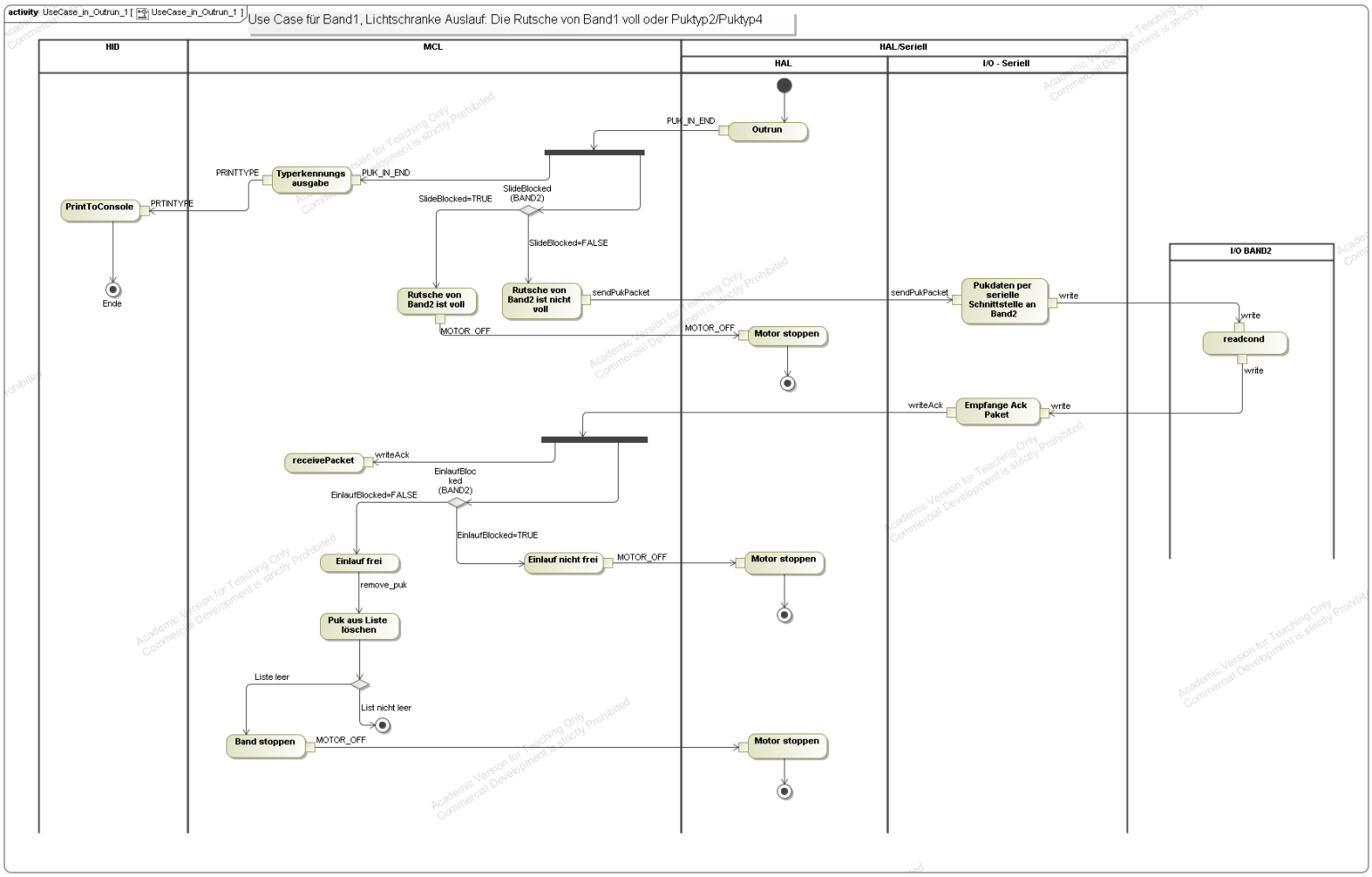

Abbildung 6: Use Case: Rutsche von Band1 voll oder Puk Typ2/Typ4 im Auslauf (Band1)

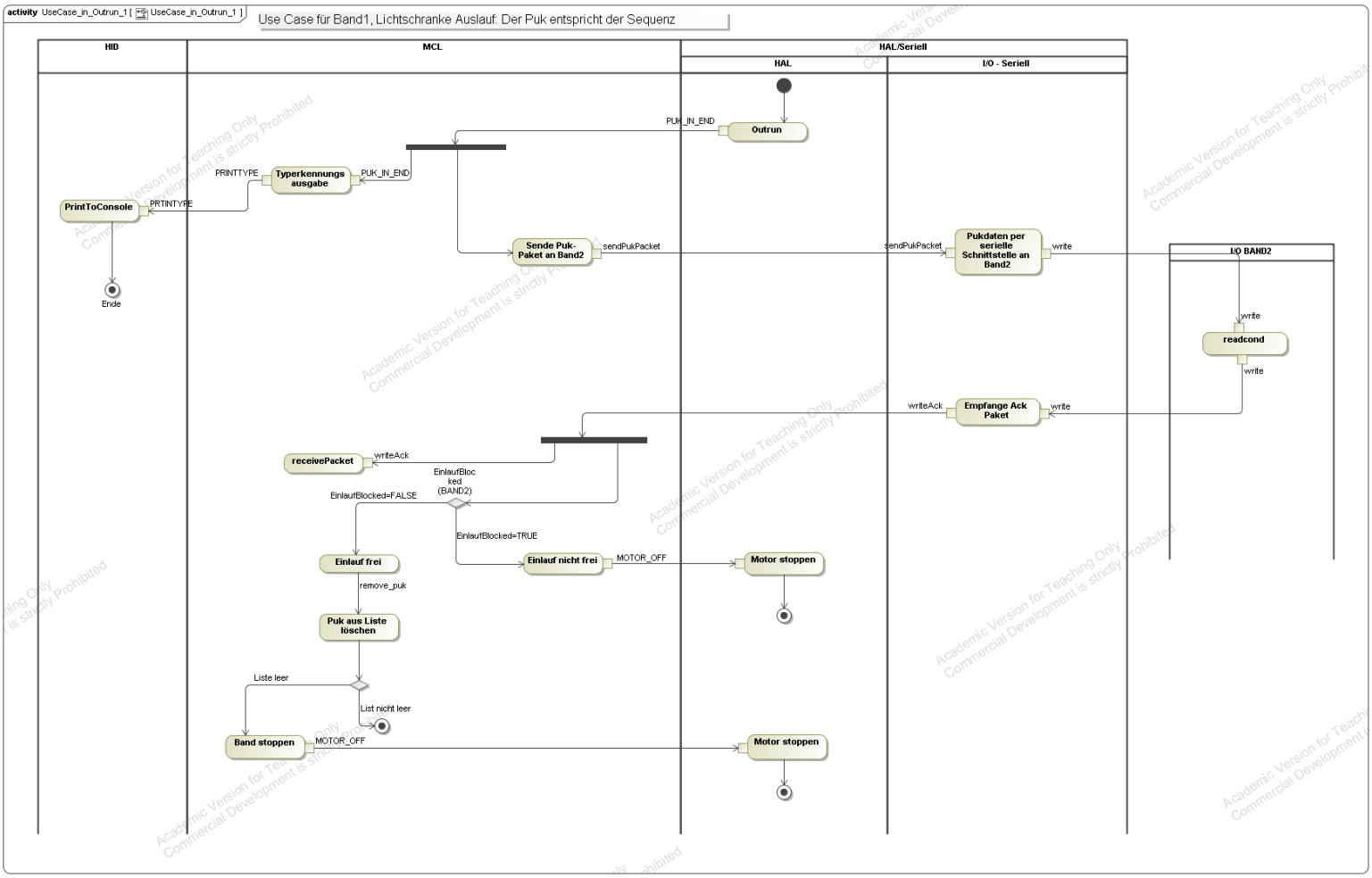

Abbildung 7: Use Case: Puk entspricht der Sequenz (Band1)

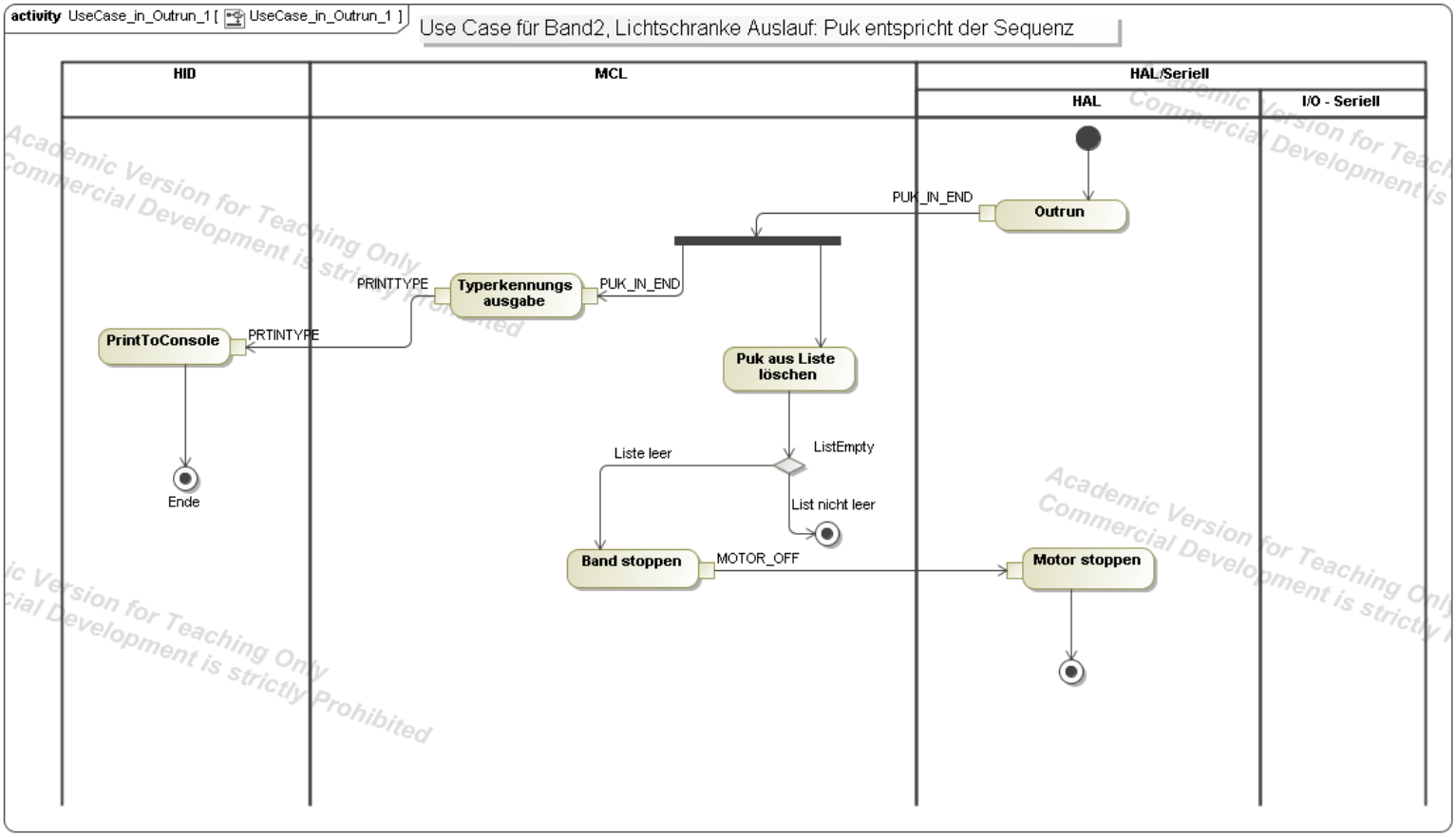

Abbildung 8: Use Case: Puk entspricht der Sequenz (Band2)

4.2 Systemanalyse

Ihr technisches System hat aus Sicht der Software bestimmte Eigenschaften. Was muss man für die Entwicklung der Software in Struktur, Schnittstellen, Verhalten und an Besonderheiten wissen? Wählen sie eine Kapitelstruktur, die am besten zur Dokumentation ihrer Ergebnisse geeignet ist.

4.3 Softwareebene

Tabelle **??** wird mit dem table label im reftag referenziert.

```
\ref{tab:usecase_wave}
allgemein: \ref{table \label}
```

Sie sollen Software für die Steuerung des technischen Systems erstellen. Aus den Anforderungen auf der Systemebene und der Systemanalyse ergeben sich Anforderungen für Ihre Software.
Insbesondere wird sich die Software der beiden Anlagenteile in einigen Punkten unterscheiden.
Dokumentieren sie hier die Anforderungen, die sich speziell für die Software ergeben haben.

Tabelle 3: WAVE LVL USE CASE

<table>
<tr><th>Name</th><th colspan="2">DER USE CASE NAME (Subjekt + Verb)</th></tr>
<tr><td>Akteur
Auslösendes Ereignis</td><td colspan="2">Wer initiiert die Aktion / Aktivität
Trigger-Ereignis</td></tr>
<tr><td>Kurzbeschreibung</td><td colspan="2">Informell</td></tr>
<tr><td>Vorbedingungen</td><td colspan="2">Anforderung an globalen Zustand, Werte von Parametern o.ä.</td></tr>
<tr><td>Essentielle Schritte</td><td colspan="2"><table><tr><th>Intention
der Systemumgebung</th><th>Reaktion des Systems</th></tr><tr><td>Schritt 1: Akteur + Event</td><td>Reaktion 1</td></tr><tr><td>...</td><td>...</td></tr></table></td></tr>
<tr><td>Ausnahmefälle</td><td colspan="2">Später einfügen, Initial nur Normalverhalten</td></tr>
<tr><td>Nachbedingungen</td><td colspan="2">...</td></tr>
<tr><td>Zeitverhalten</td><td colspan="2">Wenn relevant</td></tr>
<tr><td>Verfügbarkeit</td><td colspan="2">So etwas wie erwartete / notwendige MTBF o.ä.</td></tr>
<tr><td>Fragen/Kommentare</td><td colspan="2">Als Merkzetten, Diskussionsgrundlage</td></tr>
</table>

4.3.1 Systemkontex

HAL:
Sensoren: Lichtschranken (4 stk) für Position, Metaldetektor für zweite erkennung, Kontroll Höhenmess, Höhenmesssensor für erste Erkennung, Bedientasten. Weiche offen?, Rutschen Lichschranke (wenn voll)
Aktoren: Motor des Laufbandes, Ampel, LED der Bedienknöpfe. Weiche öffnen

NETZWERK: Kommunikation zwischen GEME BOXEN: serielle Schnittstelle

HDI (Humen Divice Interface): über die Standareingabe und Ausgabe des Computers, (eventuell GUI) um die Maschiene auch nach Quittierung wieder zu starten.

Wie sieht der Kontext Ihrer Software aus? Wie erfolgt die Kommunikation mit Nachbarsystemen? Liste der ein- und ausgehenden Signale/Nachrichten.

4.3.2 Anforderungen

Welche wesentlichen Anforderungen ergeben sich aus den Systemanforderungen für ihre Software? Achten sie auf die entsprechende Atribuierung. Berücksichtigen sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems.

5 Design

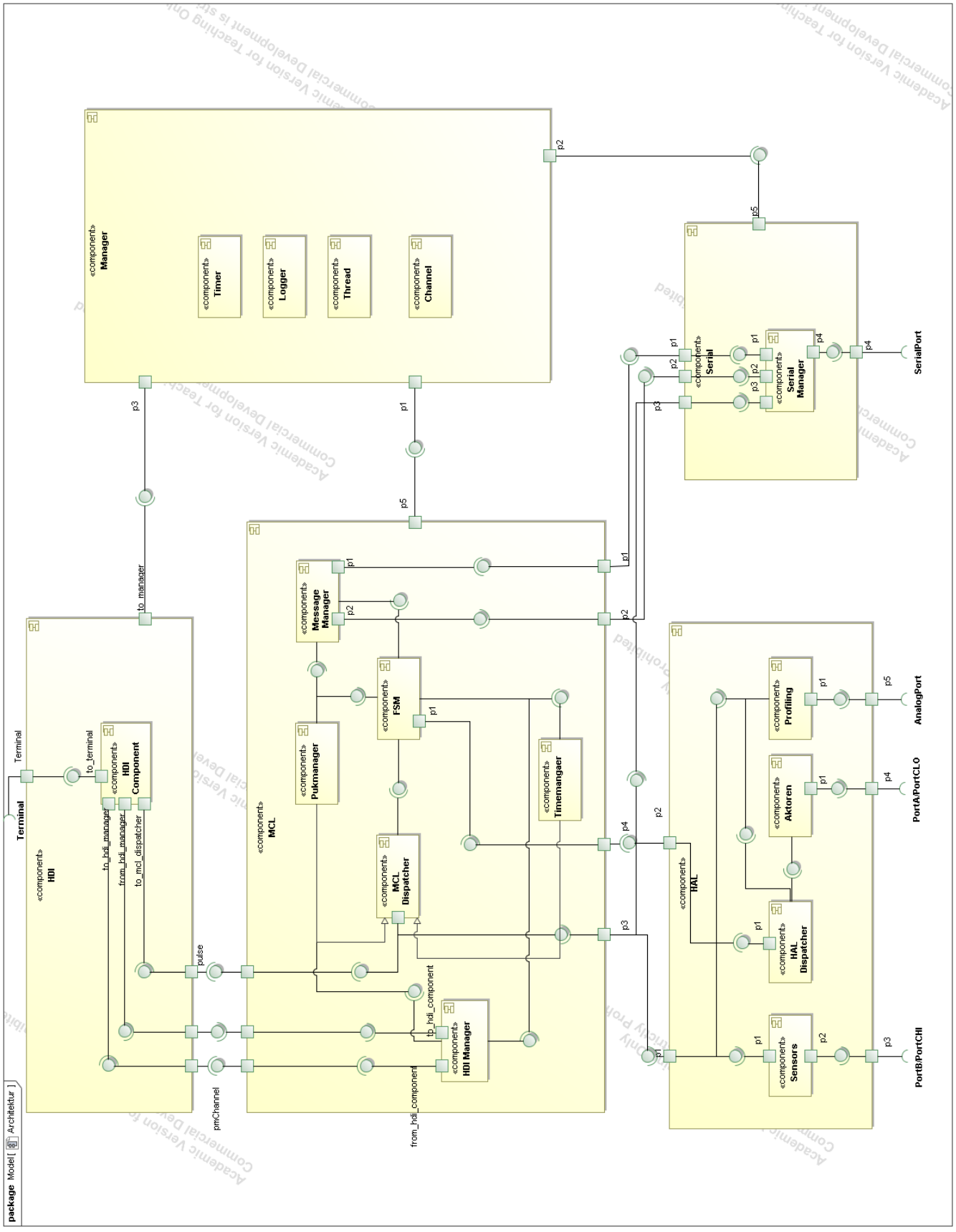

Abbildung 9: Komponentendiagramm der Architektur

Übersicht über die Architektur
Die Architektur ist gemäß Aufgabenbereich in drei Schichten ausgelegt. Die Kommunikation zwischen diesen geschieht über **Pulse Messages**. Hierbei stehen 8 Bit zur Übermittlung von Instruktionen als `enum` zur Verfügung, sowie 32 Bit, welche zum übergeben der Referenz auf zu manipulierende Daten genutzt werden. Ein blockierendes Verhalten wird somit vermieden.

HAL
Die **HAL** (Hardware Abstraction Layer) abstrahiert die Hardware des Förderbandes, indem sie zwischen `enum` und Belegungen der Kontroll-Register der für die Ports A-C genutzten I/O-Karten übersetzt. Die Komponente `Sensors` reagiert auf Digitale Inputs, indem sie bei Updates die geänderten Bits in einer ISR identifiziert und der Kontrollkomponente (**MCL**) deren Semantik in Form eines Signalcodes übermittelt. Sie ist insbesondere für die Erkennung gedrückter Buttons, Metall-Erkennung und Pegelwechseln an Lichtschranken zuständig. Der `Dispatcher` hingegen wandelt von der **MCL** eingehende Kontrollcodes und steuert mit der Komponente `Aktoren` durch Überschreiben der zuständigen Kontrollregister die Aktorik an. Die Komponente *Profiling* wird ebenfalls vom `Dispatcher` angesteuert, nimmt aber in so Fern eine Sonderstellung ein, als sie Autonom durch wiederholte Interaktion mit der analogen I/O-Schnittstelle ein Profil bestimmt und dessen Erkennungscode in als Referenz erhaltene Daten einträgt.

MCL
Diese **MCL** (Master Control Layer) steuert und überwacht alle Abläufe der Anlage. Sie hält und verwaltet die Daten und Timer und regelt mit Pulse Messages alle angrenzenden Schichten. Die Komponente `Dispatcher` verarbeitet alle Eingehenden Pulse Messages, indem sie Transitionen der **FSM** steuert. Sie reagiert sowohl auf Signale anderer Schichten, als auch auf abgelaufene Software-Timer, die über den über einen gemeinsamen `Channel` eingehen.

Die **FSM** (Finite State Machine) hält den aktuellen Betriebszustand der Anlage und bestimmt ob und in welcher Weise auf Eingaben reagiert wird. Die zugehörigen Funktionsaufrufe steuern Timer und Komponenten der eigenen Schicht und schicken Pulse an angrenzende Schichten. Der `TimerManager` erlaubt das Beanspruchen und Stoppen von Timern. Bei Initialisierung kann eine bestimmte Strecke oder Dauer bis zum Interrupt festgelegt werden. Um auf Änderungen der Geschwindigkeit des Förderbandes zu reagieren, können darüber hinaus alle im Modus „Weg“ arbeitenden Timer gleichzeitig neu parametrisiert werden.
Die Komponente `PukManager` verwaltet die auf der Anlage befindlichen Puks als Struktur in einer **Liste**. Sie hat Referenzen auf die als Nächstes in den für Zugriffe relevanten Positionen erwarteten Puks, sodass die Verteilung auf der Anlage mit minimalem Aufwand die abgebildet werden kann. Sie allein ist für Instanziierung und Löschung zuständig. Wir verwenden das *Shared-Memory-Konzept*. Innerhalb einer Anlage werden Referenzen verwendet. Kopiert wird nur beim Transfer auf die zweite Anlage. Die individuelle Puk-Struktur enthält sämtliche zum physikalischen Objekt erfassten Daten. Sie enthält darüber hinaus Referenzen auf zugehörige minimal- und maximal-Timer für die Erwartete Dauer bis zur nächsten erfassbaren Position sowie eine vorläufige Entscheidung über die ggf. zu nutzende Weiche zum Ausleiten.
Der `SerialManager` erlaubt den **FSMs** mehrerer Anlagen zu kommunizieren, um Benachrichtigungen über Fehlerzustände auszutauschen und blockierte Eingänge/Rutschen zu melden. Er ermöglicht ferner beim Transfer von Puks zum angrenzenden Band die korrespondierende Datenstruktur zu übergeben.
Der `HDI-Manager` dient der Ansteuerung der **HDI**. Er soll insbesondere Puk-Strukturen und Messwerte der Kalibrierung für die Anzeige aufbereiten.

HDI
Die Human Device Interface - Layer steuert die QNX-Konsole. Neben der Anzeige von Daten soll sie gegebenenfalls selbstständig die Semantik Tastatureingaben selbständig identifizieren und **MCL** per Pulse benachrichtigen können.

5.1 System Architektur

Erstellung sie eine Architektur für Ihre Software. Geben sie eine kurze Beschreibung Ihrer Architektur mit den dazugehörenden Komponenten und Schnittstellen an. Dokumentieren sie hier wichtige technische Entscheidungen. Welche Pattern werden gegebenenfalls verwendet? Wie erfolgt die interne Kommunikation?

5.2 Datenmodellierung

Bestimmen sie das Datenmodell und dokumentieren sie es hier mit Hilfe von UML Klassendiagrammen unter Beachtung der Designprinzipien. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier ist dann ein Übersichtsbild einzufügen. Geben sie eine kurze textuelle Beschreibung des Datenmodells und deren wichtigsten Klassen und Methoden an.

5.3 Verhaltensmodellierung

Ihre Software muss zur Bearbeitung der Aufgaben ein Verhalten aufweisen. Überlegen sie sich dieses Verhalten auf Basis der Anforderungen und modellieren sie das Verhalten unter Verwendung von Verhaltensdiagrammen. Sie können für die Spezifikation der Prozess-Lenkung entweder Petri-Netze oder hierarchische Automaten verwenden. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier sind dann kommentierte Übersichtsbilder einzufügen.

6 Implementierung

Anmerkung: Nur wichtige Implementierungsdetails sollen hier erklärt werden. Code-Beispiele (snippets) können hier aufgelistet werden, um der Erklärung zu dienen. Anmerkung: Bitte KEINE ganze Programme hierhin kopieren!

7 Testen

Machen sie sich auf Basis ihrer Überlegungen zur Qualitätssicherung Gedanken darüber, wie sie die Erfüllung der Anforderungen möglichst automatisiert im Rahmen von Unit-Test, Komponententest, Integrationstest, Systemtest, Regressionstest und Abnahmetest überprüfen werden.

7.1 Testplan

Definieren sie Zeitpunkte für die jeweiligen Teststufen in ihrer Projektplanung. Dazu können sie die Meilensteine zu Hilfe nehmen.

7.2 Abnahmetest

Leiten sie die Abnahmebedingungen aus den Kunden-Anforderungen her. Dokumentieren sie hier, welche Schritte für die Abnahme erforderlich sind und welches Ergebnis jeweils erwartet wird (Test Cases).

7.3 Testprotokolle und Auswertungen

Hier fügen sie die Test Protokolle bei, auch wenn Fehler bereits beseitigt worden sind, ist es schön zu wissen, welche Fehler einst aufgetaucht sind. Eventuelle Anmerkung zur Fehlerbehandlung kann für weitere Entwicklungen hilfreich sein. Das letzte Testprotokoll ist das Abnahmeprotokoll, das bei der abschließenden Vorführung erstellt wird. Es enthält eine Auflistung der erfolgreich vorgeführten Funktionen des Systems sowie eine Mängelliste mit Erklärungen der Ursachen der Fehlfunktionen und Vorschlägen zur Abhilfe

7.4 Lessons Learned

Führen sie ein Teammeeting durch in dem gesammelt wird, was gut gelaufen war, was schlecht gelaufen war und was man im nächsten Projekt (z.B. im PO) besser machen will. Listen sie für die Aspekte jeweils mindestens drei Punkte auf. Weitere Erfahrungen und Erkenntnisse können hier ebenso kommentiert werden, auch Anregungen für die Weiterentwicklung des Praktikums.

8 Anhang

8.1 Glossar

Eindeutige Begriffserklärungen

8.2 Abkürzungen

Listen sie alle Abkürzungen auf, die sie in diesem Dokument benutzt haben.