8 Evaluation

Dieses Kapitel befasst sich mit dem Evaluieren der erreichten Umsetzungen, die im Laufe dieser Ausarbeitung erbracht wurden.

Dies ist wie folgt aufgeteilt:
In 8.1 werden der Studienaufbau und sein Taxonomieansatz beschrieben.
In 8.2 werden Qualitätsbewertungen zu den im Rahmen dieser Ausarbeitung erstellten Eclipse-Plug-In-Projekte durchgeführt.
In 8.3 werden die Ergebnisse zusammengefasst.
In 8.4 findet eine Diskussion zu diesen statt.
In 8.5 wird auf mögliche Gefahren für die Validität eingegangen.

8.1 Studienaufbau:

Zu den im Laufe dieser Ausarbeitungen vorgenommenen Umsetzungen wird die Codequalität ausgewertet. Zu dem Pipelinesystem wurde untersucht, wie gut es die Eigenschaften von CI/CD-Pipelines erfüllt.
Bei diesen Themen bzw. deren Bewertungen wird jeweils eine Taxonomie angewandt.
Mit der Ausarbeitung von Usman et al. [„Taxonomies in software engineering: A Systematic mapping study and a revised taxonomy development method“] als Bezugspunkt sind die Taxonomien in 8.2 [TODO: Verweis] wie folgt aufgebaut:

- Absicht:
  Bei der jeweiligen Umsetzungen soll geprüft werden, wie gut die jeweils allgemein akzeptierten Aspekte, Attribute, Faktoren oder Funktionen, die in dem Forschungsfeld oder in der Branche üblicherweise als wichtig erachtet werden, erfüllt werden.
- Beschreibungsgrundlagen bzw. Terminologie:
  Es werden die Begriffe bzw. es wird die Terminologie, die in dem Forschungsfeld oder in der Branche der jeweiligen Umsetzung üblicherweise verwendet werden, übernommen.
- Klassifizierungsprozedur:
  Es wird geprüft, wie weit die Umsetzung die jeweiligen Eigenschaften oder/und Fähigkeiten, die in dem Forschungsfeld oder in der Branche üblicherweise als wichtig erachtet werden, erfüllt. Der jeweilige Anteil wird in einer Bewertungsskala von 0 bis 3 klassifiziert bzw. eingestuft.
  Diese wird als Erfüllungsstufe bezeichnet und ist wie folgt definiert:
  - 3: Die Umsetzung erfüllt alle beschriebenen Eigenschaften oder/und besitzt alle beschriebenen Fähigkeiten.
  - 2: Die Umsetzung erfüllt die beschriebenen Eigenschaften zu einem hohen Anteil oder/und besitzt einen hohen Anteil der beschriebenen Fähigkeiten.
  - 1: Die Umsetzung erfüllt die beschriebenen Eigenschaften zu einem geringen Anteil oder/und besitzt einen geringen Anteil der beschriebenen Fähigkeiten
  - 0: Die Umsetzung erfüllt keine der beschriebenen Eigenschaften oder/und besitzt keine der beschriebenen Fähigkeiten.

8.2 Qualitätsbewertungen:

In diesem Abschnitt wird die Qualität des Codes der Eclipse-Plug-In-Projekte „ArduinoCLIUtilizer“, „MUMLACGPPA“ und „PipelineExecution“ nach den in Kapitel 3.4 [TODO: Verweis] beschriebenen Kriterien bewertet.
Danach wird beurteilt, wie weit die CI/CD-Pipeline die in Kapitel 3.5 beschriebenen Eigenschaften und Fähigkeiten erfüllt.

8.2.1 Codequalität:

Funktionale Eignung:

- Funktionale Vollständigkeit: Die Modelle des Overtaking-Cars-Projektes werden automatisch entsprechend exportiert, transformiert, nachbearbeitet und hochgeladen. Das OTA-Hochladen bzw. -Updaten hingegen funktioniert nicht, weil die drei Ansätze nicht umgesetzt werden konnten (Siehe TODO: Nachtragen). Insgesamt funktioniert also mit dem automatischen Arbeitsablauf von den Modellen ein Großteil der Funktionen und nur ein kleiner Teil nicht. **Erfüllungsstufe: 2**.

- Funktionale Korrektheit: Manuellen Überprüfungen zufolge produzieren die Nachbearbeitungsschritte (PostProcessing*) korrekten Code. Dieser lässt sich mit der ArduinoCLI oder der ArduinoIDE kompilieren und auf die Boards übertragen. Aufgrund von Beschränkungen seitens des Plug-In-Framework von Eclipse können von den Schritten DialogMessage und SelectableTextWindow keine Fenster erzeugt werden. Deren Nachrichten werden improvisiert auf der Konsole der Eclipse-Workbench-Instanz, von der aus die Plug-Ins gestartet wurden, angezeigt. Dies ist im Vergleich zu den anderen funktionierenden Funktionen und Abläufen nur ein kleiner nicht funktionierender Teil der Gesamtmenge. Erfüllungsstufe: 2.

- Funktionale Angemessenheit: Die Änderungen und Eclipse-Plug-Ins zusammen können fast alle bisher genannten Anforderungen erfüllen. Die Nachbearbeitungsschritte könnten nach größeren Modelländerungen nicht mehr zum generierten Code passen. Dies ist jedoch praktisch gesehen normal und wird deshalb nur als ein kleines Problem gesehen. Aufgrund von Eclipse können von den Schritten PopupMessage und SelectableTextWindow keine Fenster erzeugt werden. Deren Nachrichten werden improvisiert auf der Konsole der Workbench-Instanz, von der aus die Plug-Ins gestartet wurden, angezeigt. Dies stellt jedoch bei der Anwendung nur ein kleines Problem dar. Insgesamt sind die Änderungen und Eclipse-Plug-Ins zusammen bis auf die beiden Problemstellen komplett für das Arbeiten mit MUML, den automatisierten Abläufen, den Roboterautos und dem Anwendungsszenario geeignet. Erfüllungsstufe: 2

Kompatibilität:

- Koexistenz: Weder die ArduinoCLI noch die ArduinoIDE haben einen negativen Einfluss auf das System. Die im Rahmen dieser Ausarbeitung erstellten oder modifizierten Projekte haben keinen negativen Einfluss auf andere Plug-Ins oder Software. Es ist also die Koexistenz mit anderer Hardware vollständig gegeben. Erfüllungsstufe: 3
- Interoperabilität: Von dem Code, der im Rahmen dieser Masterarbeit geschrieben wurde, wird das ArduinoCLIUtilizer-Eclipse-Plug-In gut von den Eclipse-Plug-Ins MUMLACGPPA und (indirekt) PipelineExecutor verwendet. Das MUMLACGPPA-Eclipse-Plug-In wird ebenfalls gut von dem PipelineExecutor-Plug-In verwendet. Aufrufe von anderen Eclipse-Plug-ins sind auch möglich. Es konnten jedoch keine Möglichkeiten gefunden werden, Eclipse-Plug-Ins von außen zu starten. Alle Eclipse-Plug-Ins, die im Rahmen dieser Ausarbeitung erstellt wurden, sind miteinander und mit Eclipse interoperabel, jedoch nicht mit anderer Software, was eine geringe Interoperabilität bedeutet. Erfüllungsstufe: 1

Verwendbarkeit:

- Eignungserkennbarkeit: Es ist hauptsächlich durch das Lesen der Ausarbeitung und der Readme-Dateien erkennbar, dass die erstellten Eclipse-Plug-Ins für das Generieren von hochladbarem kompiliertem Code für AMKs und dem Installieren von diesem auf arduinobasierte Roboterautos ausgehend von MUML-Modellen konzipiert sind. Zusätzlich sind die Resultate mancher Post-Processing-Abläufe nur für das Anwendungsszenario oder ausreichend ähnlichen Projekten korrekt. Teile der Pipeline und ein Großteil des ArduinoCLIUtilizer-Plug-Ins können aber auch außerhalb davon genutzt werden. Einerseits würden andere Personen diese Plug-Ins wahrscheinlich wegen des Contexts der MDSE mit MUML für die Roboterautos der Universität Stuttgart wahrscheinlich verwerfen oder nicht finden, obwohl sie diese möglicherweise brauchen oder nutzen könnten. Andererseits macht die Begrenzung auf Eclipse Neon das Arbeitsfeld dieser Ausarbeitung eine Nische: „Eclipse C++ Tools for Arduino 2.0“ und die oben vorgestellten Automatisierungsmöglichkeiten wie z.B. EASE und TEASE (siehe [TODO: Verweis]) würden unter neueren Eclipse-Editionen wahrscheinlich funktionieren und die im Rahmen dieser Ausarbeitung erstellten Plug-Ins obsolet machen. Personen, die mit MUML und AMKs oder ähnlichem arbeiten, würden jedoch wahrscheinlich leicht bei einer Suche diese Ausarbeitung sowie ihre Ergebnisse finden und deren Eignung einschätzen können, was für eine hohe Eignungserkennbarkeit in dem relevanten Personenkreis spricht. Erfüllungsstufe: 2
- Lernbarkeit: Die Kontextmenüeinträge tauchen nur dann auf, wenn auf die jeweils betreffenden Dateien geklickt wird. Bei .zip-Dateien, die keine Bibliothek für die ArduinoCLI enthalten, bricht diese den Vorgang ab. Generierbare Beispiele und Standardeinstellungen helfen beim schnellen Nutzen. Die Namen der Klassen und Variablen wurden so gewählt, dass sie zu dem jeweiligen Verwendungszweck passen oder diesen kurz beschreiben. Es finden auch Erklärungen hinsichtlich den geplanten Schritten mit Fenstern statt. In vielen Fällen kann die programmierte Software Fehler selbstständig erkennen und zeigt beim Abbruch eine Erklärung, was z.B. fehlerhaft eingetragen ist, an. Man kann also ihre Nutzung leicht erlernen. Erfüllungsstufe: 3
- Bedienbarkeit: Dieselben Faktoren wie direkt oben unter Lernbarkeit gelten auch für die Bedienbarkeit. Das Wechseln zu der die Plug-Ins startenden Workbench-Instanz kann als Bedienungshürde vernachlässigt werden. Die Bedienbarkeit ist also komplett gegeben. Erfüllungsstufe: 3
- Nutzerfehlerschutz: Die Kontextmenüeinträge tauchen nur dann auf, wenn auf die jeweils betreffenden Dateien geklickt wird. Bei .zip-Dateien, die keine Bibliothek für die ArduinoCLI enthalten, bricht diese den Vorgang ab. Es konnten bei der Pipeline viele Sicherheitsvorkehrungen konzipiert und umgesetzt werden. Es werden der Datenfluss und und die Typenkorrektheit über die Variablen überprüft. Folglich kann also z.B. in einem Upload-Schritt keinen Ordnerpfad für den Port-Parameter verwenden, ohne bereits vor dem Starten der Pipeline auf den Fehler hingewiesen zu werden. Zusätzlich wird die Pipeline nicht gestartet, falls Fehler gefunden wurden. Jedoch können z.B. am Anfang oder bei direkt ausgefüllten Parametern falsche Werte bzw. Einstellungen eingetragen werden. Bei solchen Fällen würden in der Regel die entsprechenden Abläufe scheitern und so die Pipeline beenden, ohne einen Output zu liefern. Gegen falsche oder schädigende Pfade (z.B. solche, die in ein anderes Projekt hineinzeigen) existieren keine Sicherheitsvorkehrungen seitens der

pipelinebasierten Automatisierung. Jedoch müssten diese für Pfade außerhalb des Projekts als absolute Pfade angegeben werden und relative Pfade würden nur zu Orten innerhalb des Projekts aufgelöst werden. Schäden können also praktisch nur entstehen, wenn z.B. der Pipelineschritt DeleteFolder böswillig genutzt wird, oder wenn leichtsinnig mit den Pfaden gearbeitet wird. Fehlkonfigurierte Roboterautos sollten wegen ihrer geringen Größe und Antriebsstärke keine oder keine nennenswerten Schäden verursachen können. In Anbetracht der jeweiligen Bedingungen für das Auftreten von Schäden sowie deren entsprechenden Auswirkungen kann man also bei diesen Plug-Ins nur von wenigen Problemen bzw. Zwischenfällen ausgehen. Gegen Böswilligkeit oder absoluter Achtlosigkeit kann sowieso fast nichts unternommen werden. Erfüllungsstufe: 2

- Benutzerinterface-Ästhetik: Es erfolgen keine Arbeitsschritte mit der Kommandozeile oder anderen Programmen, sondern innerhalb von Eclipse. Die Aufrufe erfolgen jeweils per Auswählen des entsprechenden Eintrags in dem Kontextmenü und in der Liste der Exportmöglichkeiten. Das Konfigurieren der Pipeline erfolgt textuell in einer Konfigurationsdatei, ist aber gut lesbar und schreibbar. Infolge von Beschränkungen seitens der Eclipse-IDE können von Exportprozessen aus keine Fenster angezeigt werden. Als Improvisation werden diese Meldungen in der Konsole der Workbench-Instanz, von der aus die Plug-Ins gestartet wurden, ausgegeben. Dies ist für Entwickler mit wenig Erfahrung beim Umgang mit der Eclipse-IDE unintuitiv und gewöhnungsbedürftig. Die Konsolenausgaben werden jedoch nur als ein kleines Problem bei der Nutzung gesehen, weil der jeweils geplante Inhalt in diesen dennoch gut lesbar ist, und bei der Nutzung überwiegen die gut bedienbaren anderen Teile. Erfüllungsstufe: 2
- Zugänglichkeit: Auch diejenigen, die wenig Erfahrung mit arduinobasierten Mikrocontrollern haben, können damit gut arbeiten, weil sehr viel von den internen Abläufen übernommen wird oder Informationen leicht aufgerufen werden können. Je nach Änderung an den Modellen und den notwendigen Anpassungen der Nachbearbeitungsschritte wird jedoch grundlegendes Programmierverständnis notwendig sein, um die Nachbearbeitungsschritte anzupassen. Infolge von Beschränkungen seitens der Eclipse-IDE können von Exportprozessen aus keine Fenster angezeigt werden. Als Improvisation werden diese Meldungen in der Konsole der Workbench-Instanz, von der aus die Plug-Ins gestartet wurden, ausgegeben. Dies ist für Entwickler mit wenig Erfahrung beim Umgang mit der Eclipse-IDE unintuitiv und gewöhnungsbedürftig. Wie bei der Benutzerinterface-Ästhetik werden die Konsolenausgaben nur als ein kleines Problem bei der Nutzung gesehen, weil der jeweils geplante Inhalt in diesen dennoch gut lesbar ist. Bei der Nutzung überwiegen die gut bedienbaren anderen Teile. Erfüllungsstufe: 2

Zuverlässigkeit:

- Ausgereiftheit: Die Plug-Ins liefern bei der normalen Nutzung korrekte Ergebnisse, wie Tests und manuelle Vergleiche gezeigt haben. Erfüllungsstufe: 3
- Verfügbarkeit: Ein Schnellstart ist durch das Erstellen der Konfigurationsdateien mit den Standard-Einstellungen möglich. Das Starten erfolgt durch das Auswählen der jeweiligen Menüeinträge. Lediglich das Starten einer neuen Workbench-Instanz mit den Erweiterungen für die MUML-Toolsuite ist als Vorbereitung erforderlich (siehe [TODO: Verweis auf eine

ausführliche Erklärung in Grundlagen oder verwandte Arbeiten]), aber dieser Moment ist vernachlässigbar im Vergleich zu der gesamten Arbeitszeit. Erfüllungsstufe: 3

- Fehlertoleranz: Die Pipeline ist von der Eclipse-IDE und der MUML-Toolsuite abhängig und würde ohne diesen nicht mehr funktionieren. Jedoch sind dies in Anbetracht der Ziele dieser Ausarbeitung die gegeben Rahmenbedingungen. Es funktioniert nicht, wenn wichtige Software, wie z.B. die Arduino-CLI, fehlt. Ein Großteil der Schritte benötigt andere Eclipse-Plug-Ins oder die Arduino-CLI und hängt von deren Durchführung und Korrektheit ab. Bei Durchführungsabbrüchen oder bei als unerwünscht erkannte Rückmeldungen beendet die Pipeline ihre Ausführung und meldet die Probleme. Für die normale Nutzung sollte dies ausreichen, aber bei voraussichtlich seltenen Fehlern, die in keiner Weise gemeldet werden, können die für diese Ausarbeitung geschriebenen Plug-Ins nichts ausrichten. Erfüllungsstufe: 2
- Wiederherstellbarkeit: Bei standardmäßiger Nutzung werden die zwei Ordner „generated-files“ und „deployable-files“ generiert. Es werden keine MUML-Modelle usw. gelöscht oder geändert. Folglich würden im schlimmsten Fall lediglich die Zwischenergebnisse geschädigt werden oder verloren gehen, aber ein Neustart der Pipeline würde dies leicht beheben. In weniger schlimmen Fällen kann die Pipeline zwar nicht direkt mit dem gescheiterten Schritt fortfahren, aber die resultierenden Probleme können leicht behoben werden. Sollte beispielsweise nach der Generierung der in diesem Zustand unvollständigen Codedateien das Post-Processing unterbrochen werden, so kann u.a. durch das Starten der Post-Processing-Sequenz das Post-Processing für den kompilierbaren Code neu gestartet werden. Insgesamt wäre also eine Pipeline-Fehlfunktion nicht für die Projektarbeit verhängnisvoll, sondern die Wiederherstellung der generierten Dateien würde leicht und schnell erfolgen. Erfüllungsstufe: 2

Wartbarkeit:

- Modularität: Die Abhängigkeiten zwischen den Modulen und Klassen bestehen nur soweit, wie es für die Durchführung der Aufgaben erforderlich ist. Erfüllungsstufe: 3
- Wiederverwendbarkeit: Abgesehen von dem Plug-In-Projekt ProjectFolderPathStoragePlugIn, das nur den Pfad des Projektordners speichert, gilt folgendes:
  - Das Projekt ArduinoCLIUtilizer kann alleine verwendet werden oder sein Inhalt kann von anderen Projekten aufgerufen werden.
  - Aus dem Plug-In-Projekt MUMLACGPPA kann prinzipiell mit relativ wenigen Handgriffen der Code für das Anwendungszenario entfernt werden (z.B. durch das Kopieren der nutzbaren Schritte und dem Entfernen der Pakete mit „mumlpostprocessingandarduinocli“ im Namen). So kann dieses als Basis für Pipelines für andere Anwendungen umfunktioniert werden. Alternativ kann in dem Konstruktor für den PipelineSettingsReader ein anderes PipelineStepDictionary, das die Nutzung von anderen Schritten beinhaltet, eingetragen werden.
  - PipeLineExecutionAsExport und ggf. die anderen Klassen aus de.ust.pipelineexecution.ui.exports können mit ein paar Änderungen wie dem Entfernen

der Improvisationen für z.B. ContainerTransformation per doExecuteContainerTransformationPart (…) für andere Anwendungszwecke wiederverwendet werden.

Die Änderungen für das Wiederverwenden sollten also voraussichtlich relativ leicht und zügig durchgeführt werden können. Erfüllungsstufe: 2

- Analysierbarkeit: Anhand der UML-Diagramme (siehe [TODO: Verweis]), den Readmes und den Beschreibungen aus „Integration der ArduinoCLI“ und „Ergänzungsmaterial für die Integration der ArduinoCLI“ kann relativ gut abgeschätzt werden, wie die beabsichtigten Änderungen weitere Änderungen an den anderen Modulen und Komponenten verursachen würden. Jedoch fehlen weitere Diagramme, die alles bis in das kleinste Detail erklären. Dies sollte jedoch schätzungsweise nur eine relativ geringe Verzögerung beim Verstehen des Codes verursachen. Insgesamt sind die Hürden für das Analysieren also gering. Erfüllungsstufe: Erfüllungsstufe: 2
- Testbarkeit: Es wurden für die Funktionalität des Pipelinesystems verschiedene JUnit-Tests erstellt, die fast alle Aspekte davon abdecken. Nach Änderungen am System sollte also das Nutzen dieser automatischen Tests mögliche Fehler schnell und relativ genau aufzeigen. Bei den Pipelineschritten sind jedoch manuelle Tests auf deren Auswirkungen erforderlich. Die Modularität sollte den Aufwand hierfür jedoch gering halten, also wird die diese Schwäche nicht als ein großer Faktor eingeschätzt.
  Es ist soweit kein Weg für das Simulieren von Arduino-Code für die verwendeten Roboterautos vorhanden oder bekannt, aber Yannick Liszkowski [Conceptualization and Implementation of a Digital Twin for Autonomous Driving] arbeitet an einem digitalen Zwilling für Tests und Simulationen. Jedoch zeigen sich beim Kompilieren per Compile früh Generations- oder Post-Processing-Fehler im Code, aber danach sind physische Tests auf das Verhalten erforderlich. Dieses hängt hauptsächlich von den MUML-Modellen und den Werkzeugen ab, die wiederum nicht Teil dieser Ausarbeitung sind. Seitens der im Rahmen dieser Ausarbeitung vorgenommenen Anpassungen oder erstellten Erweiterungen muss in diesem Aspekt also „ArduinoCLIUtilizer“ nach Änderungen manuell überprüft werden, wobei sein relativ einfacher Aufbau nur zu einem geringen Testaufwand führen sollte. Insgesamt ist die Testbarkeit also relativ gut oder leicht.
  Erfüllungsstufe: 2

Portabilität:

- Adaptierbarkeit: Alles ist als Java-Sourcecode gegeben. Die Code-Teile für die Aufrufbarkeit auf der Eclipse-GUI sind von Eclipse abhängig, können aber prinzipiell leicht ersetzt werden. Die funktionalen Code-Teile sind abgesehen von denen, die mit Teilen der MUML-Toolsuite arbeiten, nur gering von der Eclipse-Umgebung abhängig. Somit könnten diese sehr leicht auf eine andere Umgebung oder einen anderen Nutzungskontext angepasst werden. Die Portierbarkeit der Code-Teile, die mit Teilen der MUML-Toolsuite arbeiten, hängt von der MUML-Toolsuite ab, aber dies betrifft nur einen kleinen Teil der Features. Insgesamt ist ein Großteil auch ohne ihr (die MUML-Toolsuite) adaptierbar. Erfüllbarkeit: 2
- Installierbarkeit: Für die Installation werden die Eclipse-Plug-In-Projekte in den Workspace der startenden Workbench-Instanz importiert. Die Deinstallation erfolgt durch das Entfernen

der Plug-In-Projekte aus den Workspace oder auch einfach durch das Schließen des Workspaces. Erfüllungsstufe: 3

- Ersetzbarkeit: Dies ist wegen mangelnder Erfahrung und fehlenden bekannten Vergleichsmöglichkeiten unklar und somit wurde hierzu keine Bewertung vorgenommen.

8.2.2 Qualität der CI/CD-Pipeline:

Geschwindigkeit: Bei manuellen Tests, deren Zeitmessung mit dem ersten Klick auf „Export…“ starteten, hat sich gezeigt, dass die Durchführung der Pipeline mit den standardmäßigen Pipelineeinstellungen eine Ausführungszeit von ca. 38 Sekunden aufweißt und somit deutlich schneller erfolgt als das Ausführen der entsprechenden Arbeitsschritte von Hand, von denen in der selben Zeit nur die Arbeitsschritte für die Container-Transformation, die Generation des Kontainer-Codes und des Komponenten-Codes erfolgen können. Das manuelle Durchführen des Post-Processings erfordert wahrscheinlich auch nach Übung minddestens eine Minute. [TODO: hier evtl. noch eine Messung durchführen.] Erfüllungsstufe: 3

Konsistenz: Der einzige variable Faktor außerhalb der Pipelineeinstellungen und MUML-Modelle können die Portnamen/Portnummern für die angeschlossenen AMKs sein. Aber diese können von LookupBoardBySerialNumber ausgegeben werden und das Speichern in Variablen sowie das Nutzen von deren Werten in Upload-Schritten kann diese Probleme mit den Ports kompensieren. Manuelle Überprüfungen haben gezeigt, dass in manchen Dateien wie z.B. unter fastCarDriverECU in ECU_Identifier.h die Reihenfolge der Definitionen der ComponentInstances-Identifier variiert, aber die anderen generierten Dateien aus demselben Pipelinedurchlauf sind auf die entsprechenden Unterschiede eingestellt und die folgenden Arbeitsschritte werden davon nicht beeinflusst. Somit können die Reihenfolgenunterschiede ignoriert werden. Erfüllungsstufe: 3

Enge Versionskontrolle: Aufrufe von Git sind über AutoGitCommitAllAndPushCommand oder TerminalCommand möglich. Andere Systeme für die Versionsverwaltung erfordern jedoch das Schreiben eines Skriptes, das das Hochladen durchführt und per TerminalCommand gestartet wird. Die Versionskontrolle ist also praktisch gesehen nur geringfügig erfüllt, aber es bestehen hierfür verschiedene Systeme mit ihren eigenen Sicherheitsvorkehrungen, die nicht im Rahmen dieser Ausarbeitung umgesetzt werden können. Erfüllungsstufe: 1

Automatisierung: Es ist hauptsächlich nur das Konfigurieren der Schritte notwendig, aber dies ist normal. Aufgrund der Improvisierung mit dem Export tauchen je nach Pipeline-Schritten noch die Auswahl-Fenster auf. Dieses Problem bzw. diese Improvisation ist jedoch im Vergleich zu den vorhandenen automatisierten Abläufen nur ein geringes Problem. Erfüllungsstufe: 2

Integrierte Rückmeldungsschleifen: Feedback muss in Form der Schritte OnlyContinueIfFulfilledElseAbort, PopupWindowMessage oder SelectableTextWindow erfolgen. OnlyContinueIfFulfilledElseAbort dient zum Abbrechen. Es kann also außer dem Melden von Exceptions nicht selbstständig Rückmeldungen durchführen. In Anbetracht des manuellen Eintragens wird dieser Aspekt nicht als komplett, sondern als größtenteils erfüllt angesehen. Erfüllungsstufe: 2

8.3 Ergebnisse:
In diesem Abschnitt werden die verschiedenen Aufgabenfelder der Zielsetzung dieser Ausarbeitung und ihre Ergebnisse beschrieben.

8.3.1: Volle Unterstützung von neuer oder geänderter Hardware, Software und Bibliotheken:
Das Anpassen der Modelle, des Post-Processing-Ablaufes und das Ersetzen internen Bibliothek für die Kommunikation zwischen den Arduinos wurde durchgeführt der resultierende Code kompiliert und kann auf die AMKs der Modellautos entsprechend hochgeladen werden. Labortests haben gezeigt, dass abgesehen von der Häufigkeit der Kommunikation zwischen den Roboterautos das Verhalten der Roboterautos den Erwartungen entspricht.
Der interne Workaround der seriellen Kommunikation trotz der in den Modellen angegebenen I2C-Methode ist eine notwendige unschöne Stelle.

8.3.2: Integration der Arduino-CLI:
Alles, was zuvor über die Arduino-IDE manuell durchgeführt wurde, kann mittels der Integration der Arduino-CLI innerhalb von Eclipse durchgeführt werden. Alle Fähigkeiten werden in dem Kontextmenü angezeigt, wobei je nach Typ , also „.zip“-, „.ino“- oder „.hex“-Datei, nur die jeweiligen anwendbaren Aktionen aufgelistet werden, wodurch die Nutzbarkeit leicht fallen sollte. Wenn es von dem Pipelinesystem verwendet wird, erfüllt es seine Aufgaben auch gut.

8.3.3: Die CI/CD-Pipeline:
Das Pipelinesystem hat fast alle Anforderungen gut oder vollständig erfüllt, weil in fast allen Aspekten die Erfüllungsstufe 2 oder 3 erreicht wird. Die einzige Ausnahme ist die Versionskontrolle, zu der verschiedene Versionskontrollsysteme und mögliche Sicherheitsvorkehrungen existieren. Somit sollte zu diesem Aspekt sowieso entsprechend zu dem gewählten System ein passendes Skript geschrieben und verwendet werden. Die vollständige Automatisierung kann von dem Pipelinesystem jedoch sehr gut durchgeführt werden, weil alles von den MUML-Werkzeugen bis zu dem Hochladen abgedeckt ist und es wurden für eine bessere Flexibilität der Pipeline z.B. für das Post-Processing weitere Schritte konzipiert. Es können sogar über „TerminalCommand“ andere Programme aufgerufen werden.
Der nachbereitete bzw. nutzungsbereite Code ist auch korrekt, wie Tests und Vergleich gezeigt haben.
Jedoch ist die Nutzung als Export ist ungewohnt und die Ausgabe von Nachrichten und Textfenstern auf der Konsole der startenden Workbench-Instanz eine gewöhnungsbedürftiger Workaround.

8.3.4: Allgemeine Code-Qualität:
Bei den Qualitätskriterien erreichen die erstellten Eclipse-Plug-Ins „ArduinoCLIUtilizer“, „MUMLACGPPA“ und „PipelineExecution“ immer die Erfüllungsstufen 2 oder 3, weißen also eine gute Qualität auf.

8.4: Diskussion:
Das Ziel dieser Ausarbeitung sind das Herstellen der Unterstützung der aktuellen Generation der verwendeten Roboterautos sowie die Automatisierung des Build-Prozesses, der in seiner manuellen Ausführung viel Zeit beansprucht hat und durch das hohe Fehlerrisiko leicht scheitern konnte.
Für das Unterstützen der aktuellen Generation der verwendeten Arduino-Roboterautos wurden die notwendigen Änderungen an den MUML-Modellen und dem Post-Processing-Ablauf erforscht und durchgeführt. Zusätzlich wurden kleine Verbesserungen wie eine verbesserte Kommunikation innerhalb der Roboterautos zwischen den jeweiligen AMKs und eine verbesserte Ausrichtung des hinteren Abstandsensors durchgeführt.
Die Automatisierung des Code-Generations-Prozesses und die Ermöglichung von CI/CD wurde erforscht und umgesetzt. Der Integrationsteil von CI/CD ist als vielseitige konfigurierbare Pipeline entworfen und implementiert, die den Prozess anhand einer Konfigurationsdatei automatisch und flexibel durchführt. Der Deploymentteil ist per USB-Kabel möglich.

So ist durch diese erneut erweiterte MUML-Tool-Suite für die Code-Generation nur noch das Bereitstellen aller Modelle und Konfigurationsmöglichkeiten erforderlich.

Zusätzlich kann das Pipelinesystem prinzipiell leicht für andere Zwecke angepasst werden und so auch andere Prozesse automatisieren.

Automatiserte und verschiedene manuelle Tests haben gezeigt, dass die im Rahmen dieser Ausarbeitung durchgeführten Implementation korrekt arbeiten.

8.5: Gefahren für die Gültigkeit:
In diesem Abschnitt werden die verschiedenen Gefahren für die Gültigkeit dieser Ausarbeitung und ihrer Ergebnisse beschrieben.
Bei dem Anwendungsszenario ist dieselbe Unsicherheit, die Stürner in seiner Gefahrenanalyse erwähnt hat [TODO] der Fall. Gleichzeitig wurde die Entwurf der CI/CD-Pipeline sehr daran orientiert, wodurch es also ggf. auch nicht genau zu einen Anwendungsfall aus der realen Welt passen könnte.
Zusätzlich können die selben Unsicherheiten hinsichtlich der Gültigkeit der erstellten Software, die in Stürners Ausarbeitung erwähnt werden, auch hier auftreten, weil teilweise darauf aufgebaut wird. So wurde die dynamische Validität des Codes, z.B. eine detaillierte Überprüfung, ob das modellierte Verhalten korrekt durchgeführt wird, nicht geprüft und im Rahmen dieser Ausarbeitung konnten nur Beobachtungen von außen durchgeführt werden.
Der Autor dieser Ausarbeitung hat die Auswertungen alleine betrieben und die Bewertungen könnten subjektiv beeinflusst sein. Zusätzlich liegt bei ihm hinsichtlich des Bewertens Erfahrungsmangel vor.