8. Schlussfolgerung

Dieses Kapitel schließt die Ausarbeitung über die modellgetriebene Automatisierung von Code-Generation, -Integration und -Deployment von autonomen Fahrfunktionen ab. In 8.1 werden Schlüsselaspekte zusammengefasst, in 8.2 die Vorteile und in 8.3 die Grenzen. In 8.4 werden die Lektionen, die im Laufe dieser Ausarbeitung gelernt wurden, beschrieben. Eine Reihe an zukünftigen Forschungsmöglichkeiten und Arbeitsfeldern wird in dem Abschnitt 8.5 in Aussicht gestellt.

8.1 Zusammenfassung:

Die beiden Hauptthemen dieser Ausarbeitung waren die Herstellung der Unterstützung der neueren Version der Roboterautos und das Ermöglichen der CI/CD-Methode per Pipeline.
Für das erste Thema wurden zunächst Analysen durchgeführt und danach die entsprechenden und ausgewählten Anpassungen an den Modellen und dem Post-Processing-Ablauf vorgenommen. Zusätzlich wurden in den Roboterautos Verbesserungen wie eine zuverlässige Energieversorgung der WiFi-Module integriert.

Für die CI/CD-Pipeline wurde die Arduino-CLI in Eclipse integriert und es wurde für die Automatisierung selbst ein konfigurierbares Pipelinesystem entworfen. Bei den Umsetzungen als Eclipse-Plug-In-Projekte wurden verschiedene Workarounds benötigt, aber dennoch konnte ein gut verwendbares und flexibles System implementiert werden. Durch dieses kann der Build-Prozess, also das Erstellen von verwendbarem Arduinocode ausgehend von MUML-Modellen, zuverlässig und automatisch durchgeführt werden. Zusätzlich kann es das Hochladen der generierten Codes auf die entsprechenden AMKs der Roboterautos übernehmen.

8.2 Vorteile:

Alle Vorgänge, die zuvor von der Nutzerschaft durchgeführt werden mussten und dabei nicht nur Zeit kosteten, sondern auch Fehlergelegenheiten bargen, können nun von den Modelltransformationen bis zum Hochladen auf die Roboterautos automatisch durchgeführt werden.
So bestand bei jeder Nutzung der MUML-Werkzeuge Fehlergefahr z.B. beim Festlegen der Einstellungen in den Fenstern der Export-Wizards. Dies erfordert auch Zeit. Nun können für deren Automatisierung Konfigurationen in der „pipelineSettings.yaml“ eingetragen werden, sodass auch das Ausführen von einzelnen MUML-Werkzeugen zuverlässiger und etwas schneller erfolgen kann.
Das Post-Processing war ein Vorgang aus vielen manuellen Schritte, der dadurch eine hohe Menge an Fehlergelegenheiten enthielt und insbesondere für ungeübte Personen viel Zeit erforderte. Durch das Konfigurieren der Post-Processing-Sequenz und dem Starten von ihr kann hierfür die erforderliche Zeit auf Sekunden reduziert werden. Außerdem können so durch die Nutzerschaft keine Fehler mehr hineinrutschen.
Auch beim Hochladen von Code auf den entsprechenden AMKs kann die CI/CD-Pipeline helfen, indem anhand der jeweils angegebenen Dateipfade und Seriennummern zielgerichtet und zuverlässig die verschiedenen Arduinocodes auf die entsprechenden AMKs hochgeladen werden. So werden Verwirrung oder/und unerwartetes Verhalten durch z.B. Fehlauswahlen hinsichtlich der Dateien und der als Ziel angegebenen AMKs vermieden.
Wie man also erkennen kann, stellt es umso mehr eine große Hilfe für z.B. Forscherteams dar, je häufiger diese ausgehend von MUML-Modellen Code generieren lassen und mittels der Roboterautos testen.
Durch verschiedene Post-Processing-Piepelineschritte und Einfügeschritte wie „PostProcessingInsertAtLineIndex“ können die Pipelinekonfigurationen auch auf größere Modelländerungen angepasst werden.

Es kann durch den Pipelineschritt „TerminalCommand“ auch andere Skripte und Programme automatisieren und somit prinzipiell auch für andere Zwecke wie dem Starten von anderen Build-Tools verwendet werden.
Hinsichtlich des Konfigurierens wird durch das Generieren von Beispielen ein leichter und schneller Start in die Nutzung der Pipeline ermöglicht. Zusätzlich sind die verschiedenen Überprüfungen der Pipelineeinstellungen durch das Pipelinesystem hilfreich beim Suchen von Fehlern wie Typenfehlern und vergessenen Parametern. Wenn ein Fehler aufgespürt wird, so wird dieser gemeldet und die Pipeline kann nicht gestartet werden.

Bei den Roboterautos werden durch die nun besser mit Energie versorgten WiFi-Module alle Experimente, die mit einer W-LAN-Verbindung arbeiten, zuverlässiger durchgeführt werden können. Die verbesserte Ausrichtung des hinteren Ultraschallsensors nach rechts sollte auch helfen, bei Überholszenarien das zu überholende Roboterauto bzw. den Partner zuverlässiger abzutasten und so die Erfolgserkennung besser abschätzbar zu machen.

8.3 Grenzen:
Alle im Rahmen dieser Ausarbeitung geschriebenen Eclipse-Plug-In-Projekte wurden mit dem Kontext, dass aus den MUML-Modellen Arduinocode für die beschriebenen Roboterautos generiert werden soll, entwickelt und somit wurden keine Tests für andere Roboterautoplattformen und deren jeweiliger Programmiersprache durchgeführt.
Weiterhin ist es mit dem Pipelinesystem nicht möglich, komplexe Vorgänge mit z.B. unterschiedlichen Ausführungspfaden zu erstellen. Die Post-Processing-Sequenz und die eigentliche Pipelinesequenz werden beide jeweils als eine auszuführende Reihe an Befehlen gesehen, sodass „if-else“-Strukturen mit Pipelineschritten als bedingt auszuführenden Inhalt nicht möglich sind.
Falsche Belegungen der Eingabe-Parameter können in nicht allen Fällen erkannt werden.
„ArduinoCLIUtilizer“ kann momentan auch nicht direkt über die GUI von Eclipse genutzt werden, wenn mehr als ein angestecktes Board registriert wird. So kann also PC-Peripherie, deren Hardware von der Arduino-CLI erkannt wird, das Arbeiten mit ihr empfindlich stören und auch hinsichtlich der internen Programmierung überschrieben werden.

8.4 Gelernte Lektionen:
Das Arbeiten mit Frameworks oder Software, die von anderen Personen programmiert wurden, kann schwieriger und zeitaufwändiger sein als zuvor angenommen. Zusätzlich besteht die Gefahr von Problemen, die nicht gelöst werden können und dann zu Workarounds führen.
Es wurde auch gelernt, dass nicht mehr unterstütze Software viel schneller als erwartet unter Problemen wie nicht mehr vorhandenen oder falsch angegeben Softwarequellen leidet. So gab es beispielsweise bei TEA [TODO: Verweis] Probleme, weil eine wichtige Adresse exklusiv zu neueren Versionen führt und so auch empfohlene Installationsmöglichkeiten nicht mehr zu Eclipse Neon passen.
Auch hat sich gezeigt, dass Elektroniktutorials wie für die Nutzung von den ESP8266-WiFi-Modulen mit AMKs häufig fehlerhaft sind: In einem Großteil wurde der ESP8266 direkt über den Arduino versorgt, aber in der Praxis konnte dies nicht funktionierend nachgestellt werden, weil zumindest unsere Exemplare einen höheren Energiebedarf aufwiesen.

8.5 Zukünftige Arbeiten:
Durch diese Ausarbeitung und deren Resultate werden in verschiedenen Bereichen weitere Forschungsmöglichkeiten oder Anwendungsmöglichkeiten ermöglicht.
Die während dieser Ausarbeitung genutzten Roboterautos stellen beispielsweise durch die hinzugefügte eigene Spannungsversorgung für das WiFi-Modul, das so nun zuverlässiger arbeiten kann, eine zuverlässigere Testplattform dar.

Gleichzeitig können sie für andere Szenarien oder größere Nähe zur Realität weiter verbessert werden, z.B. durch das Nachstellen eines Vorderradantriebes oder weiteren Abstandssensoren. Schrittmotoren können beispielsweise ein zuverlässiges langsames Fahren ermöglichen. Durch die momentanen Motoren erfordert das langsame Fahren Feintuning, damit sie nicht blockieren, und ggf. Änderungen an der Geschwindigkeit des schnellen Roboterautos, damit dieser eine höhere Geschwindigkeit aufweist und nicht von der Fahrbahn abkommt. Stürner hat auch in seiner Ausarbeitung verschiedene Vorschläge festgehalten [Stürner].
Die entwickelten MUML-Plug-In-Projekte enthalten auch viele Punkte, an denen Verbesserungen durchgeführt werden können.
So kann die Integration der ArduinoCLI z.B. durch die Einführung eines Auswahlmenüs, wenn mehr als ein davon erkennbarer Mikrokontroller angeschlossen ist, wesentlich komfortabler zu nutzen werden.
Die Unterstützung von Boards, die nicht über einen Bootloader verfügen oder nicht über USB verbunden werden können, kann auch noch implementiert werden.
An der CI/CD-Pipeline ist das ordentliche Implementieren der Automatisierungen der Kontainer-Transformation und der Generation des Componentencodes ein wichtiger Verbesserungspunkt, weil so der Export-Wizard-Workaround entfernt werden kann und eine eigene Startmethode implementiert werden kann. Das Implementieren von Interaktionsfähigkeiten mit verschiedenen Versionsverwaltungssystemen kann das in der CI/CD-Methode vorgesehene Herunterladen von und Hochladen auf die genutzte Versionsverwaltung ermöglichen. Als durchaus hilfreiche Erweiterung könnte ein verbesserter Kontrollfluss komplexere oder/und mächtigere Abläufe ermöglichen. Das Verbessern des Heraussuchens von Pipelineschritten kann auch so überarbeitet werden, dass es intern das Heraussuchen und Instanziieren von Schritten besser durchführt und dass leichter eigene Klassen für Pipelineschritte hinzugefügt werden können.
Eine ordentliche Integrations der Java-Generics kann die Verwaltung von Variablen und deren Typen verbessern.
Die Erstellung einer Variante, die leicht für andere Automatisierungen verwendet werden kann, wäre durchaus für andere Anwendungszwecke ohne MUML oder veralteter Eclipse-Software sehr hilfreich.
Ein großer Punkt ist die Umsetzung einer Möglichkeit für das Updaten Over-the-Air (OTA). So können auch bei größeren Anzahlen an Roboterautos leichter neue Versionen der zu verwendenden Programmierung verteilt werden und es können auch Versuche hinsichtlich des Updatens ermöglicht werden. Die bereits implementierte Fähigkeit der Arduino-CLI und des Pipelinesystems, „.ino“-Dateien zu „.hex“-Dateien zu kompilieren kann hierbei eine große Hilfe darstellen.
Das automatische Testen von MUML-Modellen kann auch integriert werden.
Auch könnte der Build-Prozess so verbessert werden, dass nach Modelländerungen nur die dadurch anders generierten Dateien neu erstellt und bearbeitet werden.