1.: Einführung:
[(((Kontext:)))]
Die modellgetriebene Softwareentwicklung (MDSE) ist ein vielseitiger Ansatz mit verschiedenen Nutzungsmöglichkeiten und Vorteilen bei der Entwicklung von verschiedenen Systemen [Object Management Group (OMG). MDA Guide rev. 2.0. Tech. rep. Object Management Group (OMG), 2014 (cit. on pp. 1, 9, 11)]. So kann man die häufig zur Planung und Kommunikation genutzten formalen Modelle beispielsweise mit den entsprechenden Werkzeugen frühzeitig als komplette Systeme simulieren und verifizieren [Object Management Group (OMG). MDA Guide rev. 2.0. Tech. rep. Object Management Group (OMG), 2014 (cit. on pp. 1, 9, 11)]. Durch die Transformation zu Code wird Programmierzeit eingespart und das Resultat entspricht direkt der Planung [https://www.sciencedirect.com/topics/computer-science/model-driven-software-development]. Ein Anwendungsziel sind verteilte cyberphysische Systeme (auf English cyber pyhsical systems, CPS).
In diesen werden nicht nur informatische Systeme und Software sondern auch mechanische Komponenten miteinander verbunden. So werden Sensoren verwendet um Daten zu sammeln, Computersysteme werten diese aus und senden für eine Reaktion Befehle an die Aktoren, die wiederum diese ausführen und so auf die physische Welt Einfluss ausüben [https://wirtschaftslexikon.gabler.de/definition/cyberphysische-systeme-54077].
Im Fall von Autos sammeln verschiedene Sensoren Daten über die Umgebung und über das Auto selbst, die Boardelektronik wertet diese aus und je nach Resultat dieser Auswertungen werden z.B. Warnsignale an die Insassen gesendet.
Die Entwicklung von solchen Systemen kann durch verschiedene Entwicklungswerkzeuge erleichtert werden. Eines davon ist MechatronicUML. Es ist eine für cyberphysische Systeme entworfene Tool-Suite und kann CPSse sowohl hinsichtlich ihrer Strukturen als auch hinsichtlich ihrer Verhaltensweisen modellieren, indem sie zustandsbasierte Teilsysteme und ihren Nachrichtenaustausch koordiniert [DPP+16 bzw. The MechatronicUML Design Method: Process and Language for PlatformIndependent Modeling, Technical Report tr-ri-16-352. 2016]. Zusätzlich kann sie bei ausreichenden Modelldaten Code generieren [Stü22].

1.1: Problemstellung: %[((Problem)))]
Stürner hat im Rahmen seiner Arbeit "Generating Code for Distributed Deployments of Cyber-Physical Systems Using the MechatronicUML" [Stü22] MUML analysiert und erweitert, um Code für arduinobasierte Roboterautos auf Basis von plattformunabhängigen Verhaltensmodellen generieren zu lassen. Tests haben jedoch gezeigt, dass der resultierende Arbeitsprozess auch außerhalb des Startens der Abläufe bzw. Plug-Ins und dem jeweiligen Auswählen der entsprechenden erforderlichen Einstellungen manuelle Handlungen außerhalb der grafischen Oberfläche erfordert. Dieser Prozess kann gegenwärtig nicht automatisch durchgeführt werden. Die Entwicklungspraxis CI/CD kann so auch nicht durchgeführt werden.
[(((Dieser Teil musste nur nach oben, d.h. hierher, verschoben werden. Irgendwie war er in den Lösungsteil bzw. zu weit runter geraten. --->))))]
"CI/CD (Continuous Integration/Continuous Delivery) sind bewährte DevOps-Methoden zur Automatisierung in der Anwendungsentwicklung." [Hata] Alle Phasen der Entwicklung werden automatisiert und dadurch werden u.a. Probleme bei der Code-Integration vermieden [Hata]. Schließlich kostet jeder Ablauf der Arbeitsschritte nicht nur Zeit und bereitet Aufwand, sondern stellt auch Gelegenheiten für Fehler dar (siehe [Fow20]). Es können beispielsweise beim Starten der Code-Generation Missgeschicke beim Einstellen der Konfigurationen passieren, wie Tests von Stürners Arbeit gezeigt haben. Zusätzlich können bei dem Entwicklungspersonal die Entwicklungsumgebungen variieren, was Integrationsprobleme verschlimmert. [Hata]. Eine CI/CD-Pipeline ist eine Methode, eine automatisierte Durchführung dieser Schritte umzusetzen [Hatb].

Die aktuelle Generation der verwendeten Arduino-Roboterautos wird in Folge einer neuen Hardware und deren Biblithek [Reib][Reia] auch nicht vollständig unterstützt.
[(((<--- Ende verschobener Teil)))]
[(((Aus dem Lösungsansatz eingeschoben --->)))]

1.2 Zielsetzung: % Objective/Ziel

Deshalb wurde im Rahmen dieser Ausarbeitung die Automatisierung des Code-Generations-Prozesses und die Ermöglichung von CI/CD erforscht und umgesetzt. Der Integrationsteil von CI/CD ist als vielseitige konfigurierbare Pipeline implementiert, die den Prozess anhand einer Konfigurationsdatei automatisch und flexibel durchführt. Hierbei erfolgt ein automatisiertes Aufrufen der MUML-Werkzeuge auf die MUML-Modelle und beispielsweise das Kompilieren und Hochladen erfolgen auf Basis der Arduino-CLI, einer Kommandozeilensoftware für das Arbeiten mit Arduinocode. Der Deploymentteil ist per USB-Kabel möglich.
[(((<--- Aus dem Lösungsansatz eingeschoben)))]
Für das Unterstützen der aktuellen Generation der verwendeten Arduino-Roboterautos wurden die notwendigen Änderungen erforscht und durchgeführt. Zusätzlich wurde das Post-Processing ebenfalls angepasst.
[(((Aus dem Lösungsansatz eingeschoben --->)))]
Im Rahmen dieser Ausarbeitung wurde die volle Unterstützung der momentan an der Uni Stuttgart genutzten Roboterautos hergestellt, indem die Modelle und Post-Processing-Abläufe aktualisiert wurden.
[(((<--- Aus dem Lösungsansatz eingeschoben)))]
[(((Entfernen: Zusätzlich müssen die Mikrocontroller der Roboterautos für jede neue Programmierung per USB verbunden werden.)))]
[(((Aus dem Lösungsansatz eingeschoben --->)))]
Als ein geplantes Resultat dieser erneut erweiterten MUML-Tool-Suite wird für die Code-Generation nur noch das Bereitstellen aller Modelle und Konfigurationsmöglichkeiten erforderlich werden.
[(((<--- Aus dem Lösungsansatz eingeschoben)))]

1.3 Lösungsansatz: %[(((Method/Methode)))]

Wie bereits angeschnitten wurde, verfolgte diese Ausarbeitung eine Reihe an Zielen, um die oben genannten Schwächen zu beheben.
So wurde für das Unterstützen der momentanen Roboterautotechnik zunächst untersucht, wie MUML entsprechend erweitert werden konnte, um anschließend die vielversprechendste Umsetzung zu wählen. [(((… Nach oben verschoben)))]
Für das Modellieren der zu implementierenden Pipelineschritte wurden die genauen Abläufe und Aufrufe in den manuellen Schritten erforscht. Für die Basis der Pipeline wurden bestehende Automatisierungsmöglichkeiten für Eclipse Neon getestet und miteinander verglichen. Die als am besten bewertete Möglichkeit, also das Implementieren eines eigenen Pipelinesystems als Eclipse-Plug-In, wird als Basis für die Implementierung der Pipeline verwendet. [(((Entfernen: Sollte kein Kandidat die Tests erfolgreich bestehen können, so wird die Implementierung von Grund auf als Eclipse-Plug-In erfolgen.)))]
[(((… Nach oben verschoben)))] Für das Herstellen der Unterstützung der momentan an der Uni Stuttgart genutzten Roboterautos wurden Vergleiche durchgeführt werden, um die Unterschiede zwischen der Version von Roboterautos, die Stürner konzipiert hat [Stü22], und der aktuell verwendeten Version (Stand 9. Mai 2024) zu finden und aufzulisten. Diese Informationen wiederum wurden für das Auswerten der notwendigen Anpassungen von Stürners Modellen für die

Roboterautos [Stü22] verwendet. Für die Stellen, an denen mehrere Änderungsansätze gefunden wurden, werden die jeweiligen Aspekte miteinander verglichen und dann für die Implementierung der passendste Ansatz ausgewählt.
[(((OTA weggelassen)))]
Um die Qualität der Umsetzungen zu prüfen, wurden für die Themen jeweils eine Taxonomie aufgestellt und angewandt. Deren Kriterien sind die allgemein anerkannten Eigenschaften oder/und Funktionen zu dem jeweiligen Analyseziel.
[(((Gegenchecken: Die Fallstudie, die in Stürners Arbeit durchgeführt wurde, oder genauer gesagt, dessen Anwendungsszenario [Stürner22] wird in dieser Ausarbeitung ebenfalls zu Demonstrationszwecken verwendet und angepasst werden.)))]

[(((Ab hier praktisch neu)))]
%[(((Result/Resultat)))]
1.4: Ergebnis:
Die Anpassungen an die aktuelle Version der Roboterautos wurden erfolgreich durchgeführt. Die Automatisierung per Pipeline wurde auch erfolgreich umgesetzt und erfüllt ihre Anforderungen, d.h. außgehend von dem MUML-Modellen wird zuverlässig und innerhalb einer Minute direkt verwendbarer Arduino-Code erstellt und auch zielgerichtet auf die verschiedenen angeschlossenen Arduino-Mikrokontroller (kurz: AMKs) verteilt. Das Pipelinesystem ist auch relativ robust gegen Nutzungsfehler und bietet eine große Flexibilität. Das Konfigurieren ihrer Schritte und Einstellungen fällt sowohl beim Erlernen als auch beim Nutzen leicht. Durch das Befolgen von Qualitätsrichtlinien nach ISO-25010 [TODO: Quelle] kann sie auch später leicht und zuverlässig modifiziert werden.
Von der CI/CD-Methode konnte auch alles bis auf das Testen und das Interagieren mit einer Versionsverwaltung ermöglicht werden.

%[((Conclusion/Schlussfolgerung))]
Zusammengefasst ist das Ziel dieser Ausarbeitung größtenteils erfüllt worden: Die oben genannten Schwächen wurden behoben, die volle Unterstützung der momentan an der Uni Stuttgart genutzten Roboterautos hergestellt, der Code-Generations-Prozess automatisiert und die Entwicklungspraxis CI/CD wurde größtenteils ermöglicht.

%[((Contribution/Beitrag))]
Als ein Resultat oder Beitrag dieser so erneut erweiterten MUML-Tool-Suite erfordern die Codegeneration und das automatische Hochladen auf die verschiedenen AMKs per USB nur noch das Bereitstellen aller Modelle und Konfigurationen.