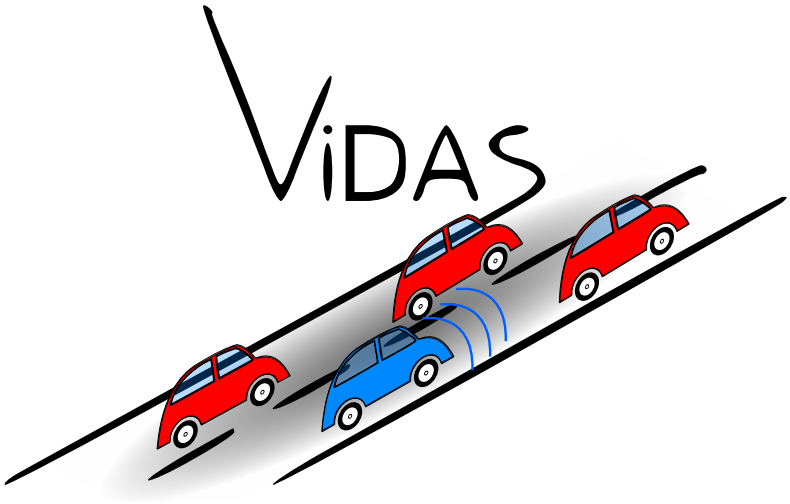

Projektgruppe **ViDAs**
Carl von Ossietzky Universität Oldenburg
Department für Informatik
2009-2010

Anpassung der Aufgabenstellung nach dem ersten Review

Version 1.0
21. April 2010

Jianyu Bao, Peter Battram, Alex Enkelmann, Andreas Gabel,
Jens Heyen, Thilo Koepke, Christoph Lässche, Sven Sieverding

1 Anpassungen

Die im Folgenden genannten Anpassungen der Aufgabenstellung werden vorgenommen, um die Anregungen aus dem ersten Review in das Projekt einfließen zu lassen und das Projekt innerhalb des vorgesehenen Zeitrahmens abschließen zu können. Abbildung 1.1 zeigt die ursprünglich geplante Aufgabenstellung, Abbildung 1.2 stellt die Anpassungen grafisch dar.

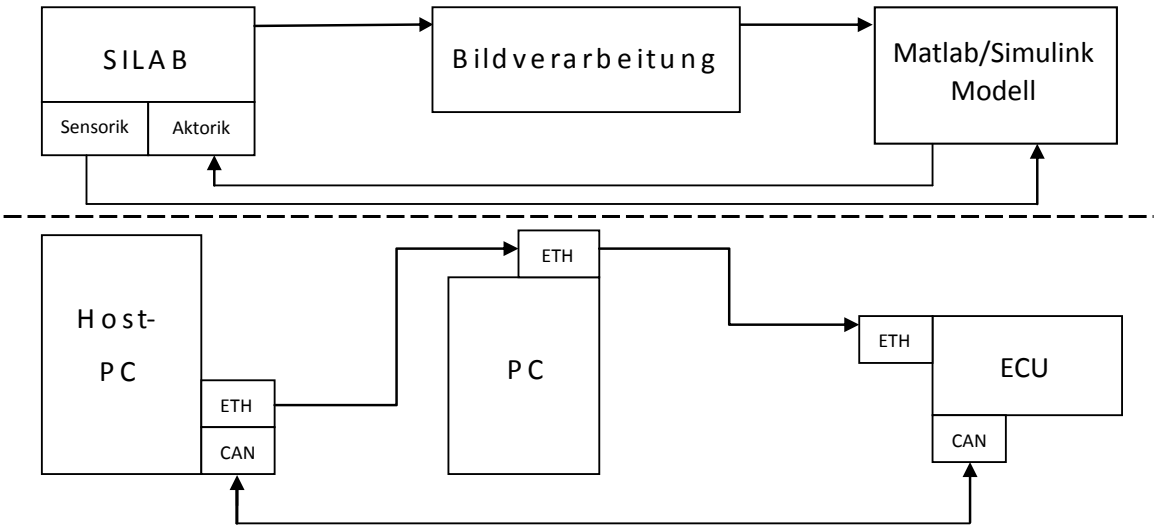

Abbildung 1.1: Ursprünglich geplante Aufgabenstellung

1.1 Bildverarbeitung

Die Informationen von Position und Art von Verkehrsschildern sowie von Fahrbahnmarkierungen sollten durch eine Bildverarbeitungskomponente aus Kamerabildern gewonnen werden. Die Informationen sollten über eine extra definierte Schnittstelle an das Matlab/Simulink-Modell gesendet werden.
Auf den Einsatz einer Kamera sowie der Bildverarbeitung wird im Weiteren verzichtet. Stattdessen wird eine neue Komponente die benötigten Informationen direkt von SILAB abfragen und über die bereits definierte Bildverarbeitungsschnittstelle an das Matlab/Simulink-Modell senden.
Durch den modularen Aufbau und der vorhandenen Schnittstelle kann zu einem späteren Zeitpunkt ein Bildverarbeitungsmodul eingefügt werden.

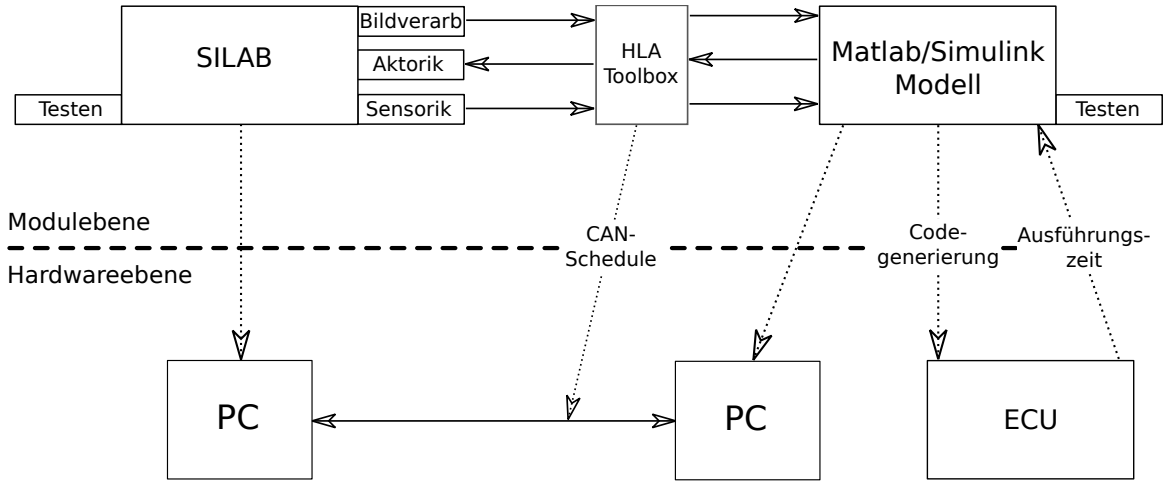

Abbildung 1.2: Anpassung der Aufgabenstellung

Dieses Vorgehen bietet zudem den Vorteil, dass diese Komponente auch Informationen über Verkehrsschilder ermitteln kann, die von Fremdfahrzeugen verdeckt werden.

1.2 Hardware

Das Matlab/Simulink-Modell sollte auf einer ECU ausgeführt werden, für die ein auf einem FPGA implementiertes System-on-Chip verwendet werden sollte. Für die Kommunikation zwischen SILAB und dem Matlab/Simulink-Modell war ein CAN-Bus vorgesehen. Auf dieses Vorgehen wird im weiteren Projektverlauf verzichtet.

Stattdessen wird das Matlab/Simulink-Modell auf einem PC ausgeführt und die Kommunikation während der Entwicklungsphase des Prototyps über die *High-Level-Architecture* realisiert. Somit entfällt der Einsatz des Matlab/Simulink-Modells auf der ECU. Die Nutzung der High-Level-Architecture vereinfacht das Testen, da das Verhalten des Matlab/Simulink-Modells zur Laufzeit in Matlab/Simulink beobachtet werden kann, und so das Verhalten im Fehlerfall leichter nachvollzogen werden kann und Fehler schneller behoben werden können . Nach dem ersten Prototyp wird dann das Matlab/Simulink-Modell als Modul in SILAB eingebunden, um eine bessere Performanz zu erreichen. Der Nachrichtenaustausch erfolgt hierbei über eine interne schnittstelle, der Nachrichtenaustausch aber gemäß eines CAN-Schedules durchgeführt, um den späteren Einsatz eines CAN-Busses zu ermöglichen.

Um die Betrachtung von eingebetteter Hardware nicht vollkommen außen vor zu lassen, wird statt der Portierung des vollständigen Matlab/Simulink-Modells nur ein Submodul auf die ECU portiert.
Anhand der Portierung eines Submoduls soll die prinzipielle Machbarkeit und das Vorgehen eines solchen Vorgangs gezeigt werden. Zu diesem Zweck wird aus einem Submodul des Matlab/Simulink-Modells mittels des Realtime-Workshop-Embedded-Coder Programm-Code erzeugt und um eine Anbindung an die Zielhardware erweitert. Die Ausführung erfolgt dann

auf einer ECU, die durch einen FPGA mit implementierten System-on-Chip realisiert wird.

Für die ECU wird zudem eine Umgebung entwickelt, mit der die Ausführungszeit einzelner Submodule bestimmt werden kann. Diese Zeiten werden dann im Matlab/Simulink-Modell berücksichtigt und somit wird dem Modell ein realeres Zeitverhalten aufgezwungen.

1.3 Testen

Der Bereich des Testens umfasst bisher Modultests, Integrationstests sowie Systemtests. Um diese Tests strukturiert durchführen zu können, wurden bereits in der Anforderungsdefinition im Abschnitt 5 (Testfälle und -szenarien) Testfälle aus den Anwendungsfällen (Abschnitt 3.3) abgeleitet und dokumentiert. Zur Verwaltung dieser Testfälle und -szenarien sowie deren Resultate wird eine Testdatenbank verwendet.

Dieses Vorgehen wird nun noch um die folgende Punkte erweitert. Zunächst wird eine Sicherheitsanalyse der gesammelten Anforderungen anhand des Code of Practice durchgeführt, um diese systematisch auf Korrektheit und Vollständigkeit zu überprüfen. Diese Sicherheitsanalyse beinhaltet eine FMEA (Failure Mode and Effect Analysis (link todojens)), die der Abschätzung von Risiken und Vorbeugung von Fehlern dient.

Zudem wird die Ausführung der Testfälle und -szenarien teil-automatisiert. Dies ermöglicht die Durchführung und Auswertung mehrerer Testläufe verschiedener Testfälle/-szenarien mit geringerem Aufwand. Hierzu werden spezielle Schnittstellen benötigt, die von eine eigenen Gruppe entwickelt werden.