DOMINION
Entwicklungs- und Laufzeitplattform zur Umsetzung von serviceorientierten Assistenz- und Automationssystemen

Volker Janz

volker.janz@uni-oldenburg.de

Abstract: DOMINION ermöglicht die in-Vehicle-Umsetzung serviceorientierter Assistenz- und Automationssysteme und stellt hierzu u. a. eine adäquate serviceorientierte Plattform in den Forschungs- und Entwicklungsanlagen des Instituts Verkehrssystemtechnik am Deutsche Zentrum für Luft- und Raumfahrt bereit. Auf Basis des serviceorientierten Ansatzes der Plattform ist eine Integration mit klassischen WebServices möglich, wodurch eine wichtige Grundlage für die Implementierung offener und flexibler Assistenz- und Automationssysteme geschaffen wird. In der vorliegenden Arbeit wird der grundlegende Aufbau der DOMINION-Plattform sowie die Struktur des Datenkerns beschrieben. Im Verlauf der Arbeit werden Beispiele für Services beschrieben und weiterführend auch der Erstellungsprozess eines neuen Services veranschaulicht. Zunächst werden die dafür notwendigen Grundlagen geschaffen und abschließend ein Blick in den praktischen Einsatz von DOMINION gegeben.

1 Grundlagen und Motivation

Im folgenden Kapitel wird eine Übersicht der Grundlagen und Begriffe für das Verständnis der DOMINION-Plattform gegeben. Neben allgemeinen Grundlagen wird der serviceorientierte Ansatz in DOMINION erläutert und eine Motivation abgeleitet, warum ein solches System bei der Entwicklung von Assistenz- und Automationssystemen sinnvoll erscheint.

1.1 Fahrerassistenzsysteme

In seiner allgemein sprachlichen Bedeutung handelt es sich bei einem Fahrerassistenzsystem um ein System durch welches ein Fahrer, also ein Individuum, welches ein Kraftfahrzeug fährt, Beistand bzw. Mithilfe erhält. Ein System wird dabei als die Gesamtheit von Objekten definiert, welche sich in einem ganzheitlichen Zusammenhang befinden und durch die Wechselbeziehungen untereinander gegenüber ihrer Umwelt abzugrenzen sind [Mau09].

In diesem Sinne sind bereits in einem VW T3 von 1983 Fahrerassistenzsysteme vorhanden: Eine automatische Blinkerrückstellung, ein Tachometer, ein elektrischer Starter und ein synchronisiertes Handschaltgetriebe sind technische Systeme, die den Fahrer bei der

Ausübung seiner Fahraufgabe unterstützen [Mau09].

Im Kontext der im späteren Verlauf erläuterten Entwicklungsplattform zur Umsetzung von Fahrerassistenzsystemen ist es jedoch notwendig die Definition solcher Systeme weiter zu präzisieren.

Als Grundlage wird das Prinzip der Arbeitsteilung herangezogen. Bei der seriellen Form der Arbeitsteilung werden verschiedene Aufgaben abwechselnd nacheinander vom Menschen und der Automatik ausgeführt. Bei der parallelen Form werden verschiedene Aufgaben parallel von Mensch und Maschine ausgeführt. In der auch Assistenzfunktion genannten Form der Arbeitsteilung werden die gleichen Aufgaben redundant-parallel von Mensch und Maschine ausgeführt. Hieraus ergibt sich eine Parallelstruktur von Mensch und Maschine. Fahrer und Assistenzsystem erfassen über Sinnesorgane oder Sensoren Informationen aus der Umgebung. Auf Basis der Situationserfassung wirkt das System in geeigneter Form auf das Fahrzeug ein. Im weiteren Verlauf wird unter einem Fahrerassistenzsystem oder auch Assistenz- und Automationssystem (AAS) ein redundant-paralleles System in dem Mensch und Maschine gewisse Aufgaben parallel erledigen verstanden. Wobei die Unterstützung durch das System auf Grundlage der maschinellen Wahrnehmung geschieht und die Möglichkeit des autonomen Eingriffes besteht [Mau09].

Einen aktuellen Entwicklungsschwerpunkt stellen die sog. kooperativen Fahrerassistenzsysteme dar. Hierbei werden in Fahrzeugen befindliche AAS untereinander oder mit infrastrukturbezogenen Systemen verbunden. Die Grundlage stellt die sog. Car-to-X Kommunikation dar. Zum Einsatz kommen hierbei Technologien aus dem Bereich des Mobilfunks der 2. und 3. Generation sowie das bei Computernetzwerken inzwischen weit verbreiteten Wireless Local Area Network (WLAN). Mit Hilfe dieser Kommunikationstechnologien werden Fahrerassistenzsysteme über die sichtbasierte Auswertung des Fahrzeugumfelds hinaus erweitert, so dass auch weit vorausliegende Unfälle oder komplexe Kreuzungssituationen bewältigt werden können [WN05].

1.2 Fahrerassistenzsysteme am Deutschen Zentrum für Luft- und Raumfahrt

Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) ist das Forschungszentrum der Bundesrepublik Deutschland für Luft- und Raumfahrt. Seine umfangreichen Forschungs- und Entwicklungsarbeiten in Luftfahrt, Raumfahrt, Energie und Verkehr sind in nationale und internationale Kooperationen eingebunden. Über die eigene Forschung hinaus ist das DLR als Raumfahrtagentur im Auftrag der Bundesregierung für die Planung und Umsetzung der deutschen Raumfahrtaktivitäten zuständig [fLuRe].

Das Institut für Verkerssystemtechnik am DLR besteht aus Wissenschaftlern aus unterschiedlichen Fachrichtungen welche Forschungs- und Entwicklungsleistungen in den drei Abteilungen Automotive, Bahnsysteme und Verkehrsmanagement erbringen. Damit leisten sie einen Beitrag zur Erhöhung der Sicherheit und Effizienz des straßen- und schienengebundenen Verkehrs. Die in Kapitel 1.1 erläuterten Fahrerassistenzsysteme finden sich in der Abteilung Automotive wieder - Fahrverhalten, Beanspruchung und Unfälle werden untersucht um daraus Anforderungen für Fahrerassistenzsysteme abzuleiten. Unter psy-

chologischen und ergonomischen Gesichtspunkten wird die Fahrerassistenzfunktion vor dem gesamten technologischen Hintergrund des DLR dann so umgesetzt, dass sie den Fähigkeiten und Erwartungen des Fahrers entspricht. Die Umsetzung wird in Fahrversuchen in der Simulation sowie im Realverkehr überprüft. Zur Konzeption und zum Testen der jeweilgen AAS stehen im Institut diverse Systeme zur Verfügung [fLuRe09]:

- `ViewCar`: Das ViewCar ist ein Messfahrzeug zur Analyse der Wahrnehmungsprozesse und des Verhaltens von Fahrern im Straßenverkehr. Es ist mit Sensoren zur Messung und Aufzeichnung der Verkehrsumgebung, der Bedienung des Fahrzeugs und des resultierenden Fahrzeugverhaltens ausgetattet.
- `VR-Labor und HMI-Labor`: Im Virtual-Reality-Labor (VR-Labor) und im Human-Machine-Interface-Labor (HMI-Labor) können neue Fahrerassistenzsysteme und -funktionen schnell und flexibel hinsichtlich Nutzbarkeit und Akzeptanz bewertet werden. Dazu verzichten beide Labore fast vollständig auf reale Hardware: Ein Sitz mit Lenkrad und Pedalerie dient zur Steuerung des virtuellen Fahrzeugs, eine Mittelkonsole mit Touchscreen kann bei Bedarf die Simulation erweitern, der übrige Innenraum des Fahrzeugs ist aber nur virtuell vorhanden.
- `SMPLab`: Das Straightforward Modular Prototyping Laboratory (SMPLab) ist ein modulares Labor zur schnellen, prototypischen Entwicklung insbesondere interaktionsreicher Fahrerassistenz- und Automationssysteme. Ansätze für eine ergonomische Unterstützung des Fahrers können im SMPLab realisiert und getestet werden.
- `Dynamischer Fahrsimulator`: Für die Erprobung von Assistenzfunktionen in einem forgeschrittenen Produktstadium kommt der dynamische Fahrsimulator zum Einsatz. Die realitätsnahe Gestaltung der Simulation ermöglicht eine valide Beurteilung der Funktionen auch in kritischen Situationen und damit einen sicheren Übergang in das Versuchsfahrzeug und in den realen Verkehr.
- `FASCar I`: Das FASCar I ist ein Versuchsfahrzeug zur Erprobung neuartiger, aktiver Assistenzfunktionen mit einem Schwerpunkt auf haptischer Fahrerassistenz. Um herauszufinden, ob der Fahrer richtig auf die Eingriffe eines neuen Assistenzsystems reagiert, sind Fahrten mit dem FASCar I der letzte konsequente Schritt der Entwicklung [fLuRe09].
- `FASCar II`: FASCar II ist ein neues Steer-by-Wire-Fahrzeug - die Lenksäule wurde komplett durch ein Simulationslenkrad ersetzt. Es existiert keine mechanische Kopplung zwischen Lenkrad und Rädern. Das FASCar II wird insbesondere im EU-Projekt HAVEit eingesetzt. [fLuRe10]

1.3 Serviceorientierte Assistenz- und Automationssystemen

Die Abteilung Automotive, welche im Kapitel 1.2 vorgestellt wurde, untersucht seit mehreren Jahren den praktisch genutzten Zugang zur Architekturgestaltung mittels Service-

orientierter Architekturen (SOA). Die hieraus abgeleitete in-Vehicle-Umsetzung serviceorientierter AAS ist zentraler Baustein des HMI-Labors und der anderen Großanlagen der Abteilung Automotive [SHG+10].

Der Begriff SOA stammt ursprünglich aus dem Bereich der Unternehmenssoftware und ist bereits 1996 erstmalig beschrieben worden. So definiert Gartner Research, ein IT-Marktforschungsunternehmen, SOA als Topologie von Schnittstellen, Schnittstellenimplementationen und Schnittstellenaufrufen, die wiederum als Ganzes eine Applikation beschreibt. SOA beschreibt das Verhältnis zwischen Services und Service Consumers, beide sind Software-Module, groß genug, um eine komplette Geschäftsfunktion abzubilden. Forrester Research, wiederum ein Marktforschungsunternehmen, definiert SOA als Art und Weise, wie Anwendungen und Software-Infrastruktur gestaltet, bereitgestellt und verwaltet werden. Gemäß IBM kann SOA sowohl als Architektur als auch als Programmiermodell verstanden werden - SOA bestimmt die Art und Weise, wie man Software baut [Lie07].

SOA erlaubt die Gestaltung von Software-Systemen, die anderen Anwendungen Services zur Verfügung stellen. Die SAP definiert SOA als Enterprise Service Architecture (ESA). ESA ist ein Instrument zur Abbildung betrieblicher Prozesse in die Informationssysteme des Unternehmens. Was alle beschriebenen und alle weiteren Definitionen von SOA gemein haben ist der Ausgangspunkt vom Service als standardisierte Grundkomponente [Lie07].

Der Ursprung der Idee des Services liegt in den Dienstleistungen. Die Verwendung von Services in Organisationen und in der Software folgen einem gemeinsamen Grundsatz, dem Paradigma der Serviceorientierung: Alle Funktionen in einem realen System, seien es Abläufe in Organisationen, Prozesse, Aktivitäten, Funktionen in Softwaresystemen, Applikationen, Teile von Applikationen oder Softwarefunktionen lassen sich als Services darstellen und aus Services aufbauen. [Mas09] Um den Eigenschaften aus verschiedenen Betrachtungswinkeln gerecht zu werden, kann ein Service abstrakt wie folgt definiert werden [SHG+10]:

> „Ein Service repräsentiert eine abgrenzbare und im Systemzusammenhang definierte Leistung, die von einem Anwendungsbaustein erbracht und von anderen Anwendungsbausteinen konsumiert werden kann [SHG+10].“

Weitere wichtige Konzepte innerhalb von Serviceorientierten Architekturen sind Interoperabilität und lose Koppelung [SHG+10]:

- `Interoperabilität`: Einzelne Services einer serviceorientierten Implementierung können durchaus aus unterschiedlichen Kontexten bzw. heterogenen Systemen stammen, sich aber dennoch in einem u. U. neuen Gesamtzusammenhang nutzen lassen.
- `Lose Koppelung`: Ein hoher Grad an Autonomie wird bei Service-Entwicklern bzw. Providern belassen [SHG+10].

Die Instantiierung von SOA-basierten Applikationen basiert auf einer generischen SOA-Plattform, die eine technologische Basis für die Service-Integration bereitstellt. Ein wichtiger Schritt zur Service-Nutzung ist deren Orchestrierung in einem Prozess. Solche Prozesse können z. B. durch die Business Process Execution Language (BPEL) beschrieben werden. BPEL-Container ermöglichen die Prozessausführung - solche Container ermöglichen insbesondere die leichte Nutzung von WebServices. Die Servicebeschreibung erfolgt für WebServices mit der WebService Description Language (WSDL) [SHG+10].

1.4 Motivation

Die Entwicklung und Einführung neuer Assistenz- und Automationssysteme ist ein wichtiger betriebswirtschaftlicher wie technischer Faktor im Automobilbereich. Die Systeme basieren häufig auf verteilten und im Allgemeinen softwarereichen Systemkomponenten, welche mehr und mehr die verschiedenen Handlungsebenen der Fahrzeugführung integriert berücksichtigen [SHG+10].

Um eine einfache Integration von Hard- und Softwarekomponenten zu gewährleisten und eine einheitliche Programmierschnittstelle für verschiedene Laufzeitplattformen, wie z. B. die in Kapitel 1.2 erläuterten Systeme, unter Berücksichtigung der in Kapitel 1.3 beschriebenen serviceorientierung bereitzustellen, wurde am DLR die Entwicklungs- und Laufzeitplattform DOMINION entwickelt [SHG+10].

2 DOMINION

DOMINION ermöglicht die in-Vehicle-Umsetzung serviceorientierter AAS (siehe Kapitel 1.3) und stellt hierzu u. a. eine adäquate SOA-Plattform in den Forschungs- und Entwicklungsanlagen des Instituts DLR/TS bereit. Bei den hierdurch unterstützten Anlagen handelt es sich um unterschiedlich spezialisierte Simulatoren und verschiedene instrumentierte Versuchsfahrzeuge (siehe Kapitel 1.2). Basierend auf der WSDL kommt in DOMINION eine spezielle Vehicle Service Description Language (VSDL) zum Einsatz, welche zur Beschreibung der in-Vehicle Services angeboten wird. Zur Orchestrierung wird die auf in-Vehicle Belange spezialisierte Vehicle Process Execution Language (VPEL) eingesetzt, welche aus dem Business-Analogon BPEL abgeleitet wurde. DOMINION bietet eine Ausführungsumgebung für die VPEL-Spezifikationen, wobei auch WebServices im Rahmen der Orchestrierung mitbetrachtet werden können [SHG+10].

Diese etablierte Form der Services wird über ein spezielles Gateway für in-Vehicle-Systeme zugreifbar. Für echtzeitkritische AAS-Bausteine wird dem Entwickler eine Code-Generierung angeboten, die die Systemintegration erheblich erleichtert. Dieser Code wird aktuell zur Implementierung sicherheitskritischer AAS bzw. AAS-Komponenten genutzt, die aus einer serviceorientierten Perspektive abgeleitet werden [SHG+10].

DOMINION ermöglicht eine schnelle Entwicklung innerhalb der DLR/TS-Labore. Durch die einheitliche DOMINION Laufzeitumgebung können Dienste durchgängig in allen An-

lagen betrieben werden, d. h. sowohl in den Fahrzeugen als auch in den Simulatoren. Abbildung 1 zeigt die DOMINION-Laufzeitumgebung im Kontext der Laborinfrastruktur [SHG+10].

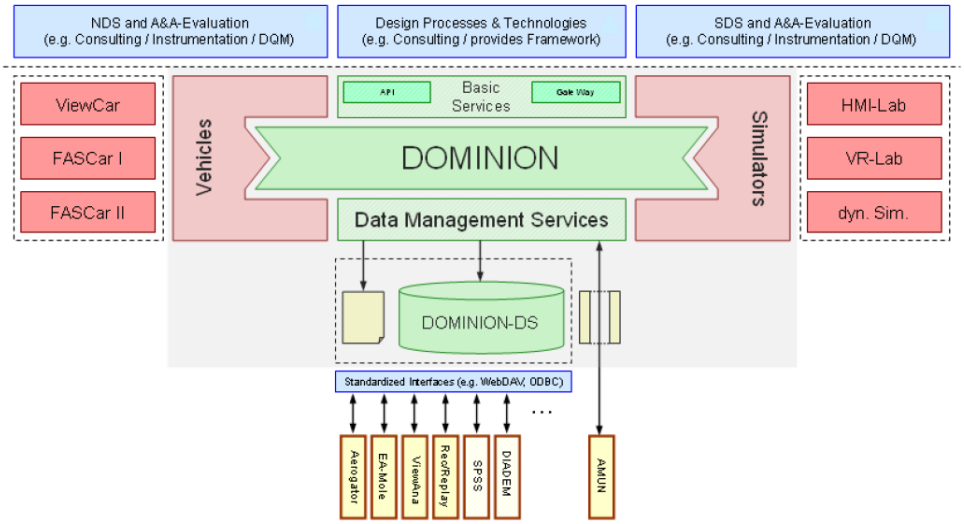

Abbildung 1: DOMINION-Laufzeitumgebung im Kontext der Laborinfrastruktur [SHG+10]

Weiterhin kann DOMINION als Middleware betrachtet werden [BMG+09]. Allgemein dient eine Middleware dazu die Kommunikation und den Datenaustausch zwischen verschiedenen Systemen zu erlauben [SR10]. In diesem Fall sind hiermit die Laboranlagen vom DLR, die AAS sowie Fremdsysteme gemeint.

Die DOMINION-API und das DOMINION-Gateway sind die Hauptinterfaces (also Zusammenstellungen von Basisservices) für die Entwicklung von eng gekoppelten oder lose gekoppelten ASS. Beide Schnittstellen ermöglichen das Deployment der Services für verschiedene Betriebssysteme - momentan werden Microsoft Windows, Unix-Derivate, MacOS, iPhoneOS und Microsoft Windows Mobile unterstützt. Der DOMINION-DataStore repräsentiert die Persistenzschicht für experimentelle Daten und dient als standardisierte Schnittstelle zu einer großen Bandbreite von Analysewerkzeugen wie z. B. DIADEM, EA-Mole oder Ærogator [BMG+09].

2.1 DOMINION-Datenkern

Der DOMINION-Datenkern wird durch eine, auf der Vehicle Service Description Language basierende, XML-Datei repräsentiert. Der mit Hierarchically Arranged and Directed Entity Structure (HADES) bezeichnete Datenkern befindet sich innerhalb der Ordnerstruktur von DOMINION als *Ontology.xml* im Ordner *data*. Die Struktur des Datenkerns

ist domänenorientiert, wie Abbildung 2 zeigt [NH10].

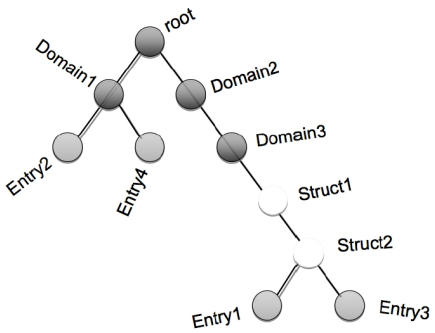

Abbildung 2: Struktur des Datenkerns [NH10]

Das Ausgangselement der XML-Datei ist mit *Dominion* bezeichnet. Innerhalb dieses Hauptelementes befinden sich das Element *Author*, *Version* und *SuperDomain*. Während *Author* und *Version* lediglich Kontaktdaten des Entwicklers sowie Versionshinweise enthalten, sind unter *SuperDomain* die verschiedenen Domains als Elemente enthalten. Eine *Domain* kann die untergeordneten Elemente z. B. zum Anwendungsbereich der Cockpitsensorik zusammenfassen. Das Element *Domain* teilt sich in zwei Bereiche auf: *Data* und *Tasks* [NH10].

Data enthält im Wesentlichen die Elemente *Structure* und *Entry*, wobei *Entry* konkrete Daten, wie Geschwindigkeit oder Lenkradwinkel beschreibt. Durch *Structure* können, ähnlich wie der Datentyp struct bei C++, *Entry*-Elemente zusammengefasst werden [NH10].

Unter *Tasks* sind nun die konkreten Services beschrieben. Ein Service wird im Datenkern als *Application* bezeichnet. Eine Applikation meint im Folgenden somit einen Service innerhalb der DOMINION-Plattform. *Tasks* kann durch *TaskObject*-Elemente weiter unterteilt werden. Innerhalb eines *TaskObject* befinden sich dann die einzelnen Applikation, jeweils repräsentiert durch ein *Application*-Element. Ein wichtiges Attribut von *TaskObject* ist *SampleTime*. *SampleTime* gibt den Zeitintervall an, in dem die Applikationen aufgerufen werden. Eine Applikation wird durch die Elemente *Author*, *Description*, *MetaInformation*, *InputData* und *OutputData* beschrieben. *MetaInformationen* enthält Informationen zur Laufzeitumgebung der Applikation. *InputData* und *OutputData* beziehen sich direkt auf die in *Data* beschriebenen Daten. Es werden die Daten definiert, auf die innerhalb der Applikation zugegriffen werden kann sowie die Daten, die als Ausgabe resultieren [NH10].

2.2 Applikationen bzw. Services in DOMINION

Wie in Kapitel 2.1 beschrieben, werden Services in DOMINION als Applikationen bezeichnet. Eine solche Applikation beteht aus zwei Grundbausteinen: *Application* und *ApplicationController*. Auf Basis des Datenkerns werden durch den Base Application Assembler (BAAL) neue Applikationen generiert. Die für den Entwickler generierten Coderümpfe sind teil der *Application*. Die run-Funktion der *Application* wird in einem bestimmten Zeitintervall durch den *ApplicationController* aufgerufen. Dieser Intervall wird im Datenkern durch das Attribut *SampleTime* festgelegt (siehe Kapitel 2.1). Die Hauptfunktionalität der Applikation wird daher in der run-Funktion der generierten C++ Klasse *D⟨NameDerApplikation⟩.cpp* implementiert. Neben den Coderümpfen wird eine Visual Studio Projektdatei generiert, durch welche die Entwicklung in Microsoft Visual Studio effizient unterstützt wird. Die Applikation kann schließlich für verschiedene Plattformen bereitgestellt werden [NH10].

2.2.1 Erstellen einer neuen Applikation

Der Erstellungsprozess einer neuen Applikationen beginnt immer mit der Beschreibung im Datenkern. Hierzu wird die *data/Ontology.xml* um einen weiteren Application-Eintrag in der entsprechenden Domäne erweitert. Eine solche Beschreibung kann z. B. wie in Abbildung 3 dargestellt aussehen.

```
<Application name="OdysseusConnector">
 <Author>
  <Name>Volker Janz</Name>
  <Email>volker.janz@uni-oldenburg.de</Email>
 </Author>
 <Description>
  Stellt eine UDP Verbindung zu Odysseus her
 </Description>
 <MetaInformation>
  <OperatingSystem>Win32</OperatingSystem>
  <ResearchFacility>all</ResearchFacility>
  <DevelopmentStatus>Development</DevelopmentStatus>
  <HardwareDependencies>*</HardwareDependencies>
  <SoftwareDependencies>*</SoftwareDependencies>
  <VisualStudioBuild>Debug</VisualStudioBuild>
 </MetaInformation>
 <InputData>
  <Input>Environment@Traffic.Car</Input>
 </InputData>
 <OutputData/>
</Application>
```

Abbildung 3: Beschreibung einer Beispielapplikation

Im Anschluss wird die Codegenerierung gestartet. Diese ist durch das Java-Programm BAAL repräsentiert. BAAL befindet sich im Ordner *Development/Tools/DominionJBAAL*. Im Eingabefeld application ist der Name der neu eingefügten Applikation einzutragen.

Nach der Durchführung der Codegenerierung befindet sich im Ordner *Applications* ein neuer Unterordner der das generierte Projekt enthält.

Das Projekt kann mit Hilfe von Microsoft Visual Studio geöffnet und bearbeitet werden. Verändert man nun die Beschreibung der Applikation im Datenkern, wird bei Erstellung des Projektes automatisch BAAL aufgerufen und die Änderungen übernommen. Dabei werden nur die Dateien neu generiert, welche sich in den mit *auto* bezeichneten Unterordnern des Projekts befinden.

Der letzte Schritt stellt dann das Erstellen der Applikation für eine bestimmte Plattform dar. Im Anschluss kann die neue Applikation innerhalb der DOMINION-Laufzeitumgebung eingesetzt werden und ist somit unabhängig von ihrem Einsatzort - sei es ein einfacher Simulator oder ein reales Fahrzeug.

2.2.2 Beispielhafte Applikationen

Der Vorteil eine vorhanden Plattform für die Umsetzung von AAS zu nutzen ist das Vorhandensein von Applikationen und Funktionalitäten die sonst Entwicklungszeit kosten würden. So stellt die DOMINION-Plattform auf Basis ihrer serviceorientierten Architektur bereits viele Applikationen bereit von denen einige wesentliche im Folgenden erläutert werden [SHG+10].

Grafik-Engine NexGenViewer Die Applikation NexGenViewer der DOMINION Plattform ermöglicht die Visualisierung der Simulationswelt. Sie wird bereits in verschiedenen Fahrsimulatoren am DLR eingesetzt [SHG+10]. Die Grafik-Engine basiert auf der Open-Soruce-Grafikbibliothek OpenSceneGraph. Der Vorteil gegenüber dem Einsatz von reinem OpenGL liegt in den vielen Funktionen die bereits durch die Bibliothek bereitgestellt werden - hierdurch werden nötige Einarbeitungs- und Entwicklungszeiten minimiert [SNH+09].

Dominion Environment Detection (DED) DOMINION stellt mit der Applikation DED eine virtuelle Sensorsimulation bereit. Die Applikation generiert Sensordaten aus den Daten der Landschaft, der Fahrroute, den Fahrzeugpositionen und -orientierungen. Diese Daten stammen wiederum aus der Datenbank. Es sind bereits verschiedene Sensoren in der DED-Applikation implementiert, dazu zählen u. a. ein Laser-/Radarscanner, eine Spurerkennung sowie ein Straßeninformationssensor [BMG+09].

DominionNLTwoTrack DominionNLTwoTrack wird genutzt um das Fahrzeug zu simulieren in dem das AAS betrieben wird und ist ein nicht-lineares zweispuriges Fahrzeugmodell und somit in der Lage Fahrzeuggeschwindigkeit und -beschleunigung zu simulieren [BMG+09].

DynamicObjectSimulation Die Applikation DynamicObjectSimulation berechnet die Position und Ausrichtung aller bewegungsfähigen Objekte in der simulierten Welt. Nach-

dem alle sich auf der Straße befindlichen Fahrzeuge erzeugt wurden, wird deren Verhalten simuliert. Prinzipiell ist jedes Verhalten denkbar, von einfachen Spurwechseln bis hin zu Schwarmverhalten. Die Anzahl der Objekte sowie das Verhalten und die Route der einzelnen Objekte ist in einem XML-Skript definiert [BMG+09].

DominionServer Neue Services, welche zur Simulationsumgebung hinzugefügt werden, erstellen zunächst eine Kontrollverbindung zur DominionServer-Applikation. Sie müssen sich identifizieren und um Erlaubnis fragen der Simulationsumgebung beizutreten. Hierdurch verfügt der DominionServer stets über die Information, welche Services innerhalb der Simulationsumgebung zur Verfügung stehen. Darüber hinaus kann er allen anderen Prozessen eine Nachricht über den Beitritt oder das Verlassen eines Services schicken [SHG+10].

DominionServerConsole Die DominionServerConsole-Applikation wird mit der DominionServer-Applikation verbunden. Sie kann dann genutzt werden den Server zu steuern und den Betriebsmodus zu ändern. Die Simulationsumgebung kann die Stadien *running*, *paused* und *stopped* annehmen. Wenn *stopped* initiiert wird, werden alle Services heruntergefahren. Im Status *paused* wird die Simulation solange angehalten bis der Status zurück auf *running* gesetzt wurde [SHG+10].

StickTracerWindow Für den Fall, dass keine realen Eingabemöglichkeiten wie ein Lenkrad oder Pedale existieren, stellt DOMINON die Applikation StickTracerWindow bereit. Diese Applikation ermöglicht die Steuerung des Testfahrzeuges durch die Maus oder die Tastatur. Hierzu ist StickTracerWindow in der Lage beliebig viele Steuerorgane zu emulieren. Neben der Emulierung der Steuerung können haptische Rückmeldungen wie z. B. Vibrationen an reale verbundene Kontrolleinheiten gesendet werden [SHG+10].

Eine weiterer Einsatzbereich für die Applikation ist das Monitoring von real angeschlossenen Geräten. In so einem Fall ist es möglich nicht nur die Position der Steuereinheit anzuzeigen, sondern auch die Vibrationen und Kräfte die hierauf einwirken [SHG+10].

EnvironmentManager Die DOMINION-Applikation EnvironmentManager ermöglicht das Erstellen sowie das Verändern der Simulationsumgebung. Sie spezifiziert den Namen des 3D-Modells welches vom NexGenViewer für die Darstellung der Umgebung genutzt wird (siehe Kapitel 2.2.2). Desweiteren definiert die Applikation den Namen der XML-Datei die von der DynamicObjectSimulation zur Verkehrssimulation genutzt wird (siehe Kapitel 2.2.2). Auch die Steuerung von Ampeln oder anderen unbeweglichen Gegenständen im Szenario wird ermöglicht [SHG+10].

DominionRecord DominionRecord dient zum Aufzeichnen der durch die Simulation generierten Daten. Hierzu werden diese in eine temporär erstellte lokale SQL Datenbank unter Verwendung von SQLite gespeichert. Anschließend besteht die Möglichkeit die gespeicherten Daten in CSV-Dateien zu exportieren und so weiterzuverwenden [SHG+10].

2.3 Integration von WebServices

Der serviceorientierte Ansatz und die Anlehnung an die Web Service Description Language ermöglicht den Einsatz von WebServices in DOMINION [SHG+10]. WebServices sind gemäß der Definition der Gesellschaft für Informatik Softwaresysteme, die automatisiert Daten austauschen und/oder Funktionen auf entfernten Rechnern aufrufen [vKK07]. Der Einsatz dieser Technologie in DOMINION schafft eine wichtige Grundlage für die Implementierung offener und flexibler AAS. So wäre es denkbar auf Verkehrs- oder Wetterinformationen über WebServices zuzugreifen. Diese etablierte Form der Services wird über ein spezielles Gateway für in-Vehicle-Systeme zugreifbar [SHG+10].

3 DOMINION in der Praxis

Die DOMINION-Plattform wurde bereits erfolgreich in verschiedenen Fahrsimulatoren und Testfahrzeugen des DLR eingesetzt. Zudem wurde es in mehreren nationalen und internationalen Projekten für die Durchführung von Experimenten und Klärung von Forschungsfragen genutzt. Darüber hinaus wurden erfolgreich diverse AAS auf Basis der DOMINION-Plattform entwickelt [BMG+09].

3.1 ISi-PADAS Joint Driver-Vehicle-Environment Simulation Platform

Unter der Leitung von Andreas Luedtke vom Oldenburger Institut für Informatik OFFIS e.V. verfolgt das Projekt Integrated Human Modelling and Simulation to support Human Error Risk Analysis of Partially Autonomous Driver Assistance Systems (ISi-PADAS) die Bereitstellung einer innovativen Methodologie um risikobasiertes Design von partiell autonomen Fahrerassistenzsystemen zu unterstützen. Dabei wird die Eleminierung bzw. Abschwächung von Fahrerfehlern durch eine integrierte Fahrer-Fahrzeug-Umgebung fokussiert [IP].

Das in mehrere Teile untergliederte Projekt verfolgt u. a. die Entwicklung einer Simulationsplattform, welche ein ebenfalls im Projekt entwickeltes Fahrermodell, Fahrzeugmodelle und die Fahrumgebung integriert um das Fahrerverhalten detailliert zu simulieren [IP]. DOMINION dient als Basis dieser Joint Driver-Vehicle-Environment Simulation Platform (JDVE) [BMG+09].

3.2 Projektgruppe Stream Cars an der Universität Oldenburg

Die Abteilung Informationssysteme der Universität Oldenburg betreut im Sommersemester 2010 - Wintersemester 2010/2011 die Projektgruppe Stream Cars. In dieser Projektgruppe soll ein Datenstrommanagementsystem (Odysseus) mit der Entwicklungsplattform

DOMINION verbunden werden um die aus Sensorik generierten Informationen zu interpretieren und einem AAS die nötigen Resultate zu übermitteln. Durch DOMINION kann das entstehende System für intelligente und kooperative Fahrzeuganwendungen in einem realen Umfeld umgesetzt werden [IS].

Literatur

[BMG+09] T. Bellet, P. Mayenobe, D. Gruyer, J. C. Bornard, J. Schindler, T. Krehle, J. Alsen und A. Lüdke. Joint-DVE Simulation Platform for phase 1. Dezember 2009.

[fLuRe] Deutsches Zentrum für Luft-und Raumfahrt e.V. Das DLR im Überblick. `http://www.dlr.de`. Aufgerufen am: 19.04.2010.

[fLuRe09] Deutsches Zentrum für Luft-und Raumfahrt e.V. Institutsflyer - Institut für Verkehrssystemtechnik. `http://www.dlr.de/ts/Portaldata/16/Resources/dokumente/service/DLR-TS_Institutsflyer_090420.pdf`, April 2009. Aufgerufen am: 19.04.2010.

[fLuRe10] Deutsches Zentrum für Luft-und Raumfahrt e.V. Institut für Verkehrssystemtechnik - Abteilung Automotive Einbettung / Methodik & Instrumente / Aktivitäten. Vortragsfolien, März 2010.

[IP] ISi-PADAS. The Project. `http://www.isi-padas.eu/content/project`. Aufgerufen am: 21.04.2010.

[IS] Abteilung IS. Abteilung IS: Lehre: Projektgruppen: Stream Cars. `http://www-is.informatik.uni-oldenburg.de/584/`. Aufgerufen am: 21.04.2010.

[Lie07] Daniel Liebhart. *SOA goes real*. Carl Hanser Verlag München Wien, München, 2007.

[Mas09] Dieter Masak. *SOA? Serviceorientierung in Business und Software*. Springer-Verlag Berlin Heidelberg, Heidelberg, 2009.

[Mau09] Markus Maurer. Entwurf und Test von Fahrerassistenzsystemen. In Hermann Winner, Stephan Hakuli und Gabriele Wolf, Hrsg., *Handbuch Fahrerassistenzsysteme*, Seiten 43–54. Vieweg+Teubner / GWV Fachverlage GmbH, Wiesbaden, 2009.

[NH10] Ulf Noyer und Marco Hannibal. Dominion Einführung. Vortragsfolien, März 2010.

[SHG+10] Mark Schröder, Marco Hannibal, Jan Gacnik, Frank Köster, Christian Harms und Tobias Knostmann. Ein Labor zur modellbasierten Gestaltung interaktiver Assistenz und Automation im Automotive-Umfeld. In *AAET 2010*, Seiten 445–462, Februar 2010.

[SNH+09] Julian Schindler, Ulf Noyer, Christian Harms, Frank Flemisch, Aladino Amantini, Dominique Gruyer, Malte Zilinski, Fabio Tango und Frank Köster. Ontology and Basic Version of the Joint DVE Simulation Platform. Mai 2009.

[SR10] Ralph Stair und George Raynolds. *Principles of Information Systems*. Course Technology, Boston, 2010.

[vKK07] Mark von Kopp-Krimpenfort. *Web Service Engineering*. GRIN Verlag, Norderstedt, 2007.

[WN05] H. Wallentowitz und D. Neunzig. Strategien zur Steigerung der Mobilität älterer Menschen. Institut für Kraftfahrwesen RTHW Aachen, Juni 2005.