1 Diagnostics beyond UDS

Dipl.-Ing. (BA) Christoph Rätz, Vector Informatik GmbH, Stuttgart
Dipl.-Ing. Bernd Gottschalk, Mercedes-Benz AG, Böblingen

Abstract

This lecture provides an overview on the history of vehicle diagnosis as well as a perspective on current trends in this area.

The trends in development of vehicle electronics and mechatronics are being looked at and give an outlook on current developments and their prototypic implementations are given.

Kurzfassung

Dieser Beitrag gibt einen Überblick über die Historie der Fahrzeugdiagnose sowie einen Ausblick auf aktuelle Entwicklungen und zukünftige Trends auf diesem Gebiet.

Dazu werden die Trends in der Fahrzeug-Elektronik- und Mechatronik-Entwicklung beleuchtet, die sich daraus ergebenden Möglichkeiten angesprochen sowie ein Ausblick auf aktuelle Entwicklungen und deren prototypische Umsetzung gegeben.

1 Historie

Diagnoseprotokolle in Kraftfahrzeugen haben eine lange Historie hinter sich:

Angefangen mit Blinkcodes an einer proprietären Diagnosekupplung über proprietäre Protokolle von vernetzten Steuergeräten, ging es weiter mit einem schon einigermaßen standardisierten KWP2000 (ISO 14230-x, veröffentlicht 1999) bis hin zum heutigen Stand der Technik – UDS „Unified Diagnostic Services“. UDS wurde 2006 als ISO-Standard veröffentlicht.

UDS ist in der ISO14229-x für verschiedene Transport-Layer wie CAN, Ethernet, Flexray usw. standardisiert und findet heute breite Verwendung – nicht nur in Kraftfahrzeugen.

Zielrichtung war primär die Diagnose von Fehlfunktionen in mechatronischen, Mikrocontroller-basierten Systemen.

Auch wird UDS ständig erweitert: Dieses Jahr wurde die 3rd Edition veröffentlicht mit wesentlichen Erweiterungen vor allem im Bereich Security. Darüber wurde bereits letztes Jahr auf der Diagnosetagung hier in Dresden berichtet [1].

Wenn auch UDS kontinuierlich weiterentwickelt wird, um neuen Anforderungen gerecht zu werden, basiert es dennoch auf einigen Grundprinzipien bzw. -annahmen:

- Die zur Verfügung stehende Bandbreite zur Kommunikation ist sehr gering
- Aus heutiger Sicht leistungsschwache Mikroprozessoren sollen die Umsetzung ermöglichen
- Mögliche Fehler sind a priori bekannt und können einem DTC „diagnostic trouble code“ zugeordnet werden.

Ein weiteres Standardisierungsprojekt für eine Diagnose-API wird aktuell im ASAM behandelt: SOVD „Service Oriented Vehicle Diagnosis“.

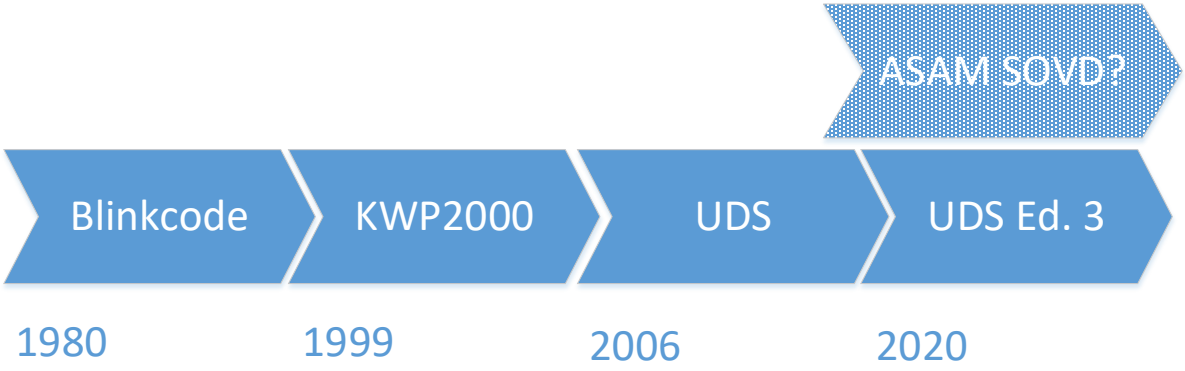

Abbildung 1: Zeitleiste

2 Trends

In der Vergangenheit war das Fahrzeug im Wesentlichen ein in sich abgeschlossenes, weitgehend nicht veränderliches System. Heute wird es kontinuierlich weiterentwickelt, setzt moderne IT Technologien um und ist in Kontakt mit seiner Außenwelt. Mögliche Fehlfunktionen können damit nicht mehr wie eingangs erwähnt bei mechatronischen Systemen a priori benannt werden. Vielmehr sind Methoden wie Software Monitoring und Tracing notwendig.

Die folgende Grafik zeigt einen Überblick über einige der Trends in der Entwicklung von Fahrzeug-Elektronik:

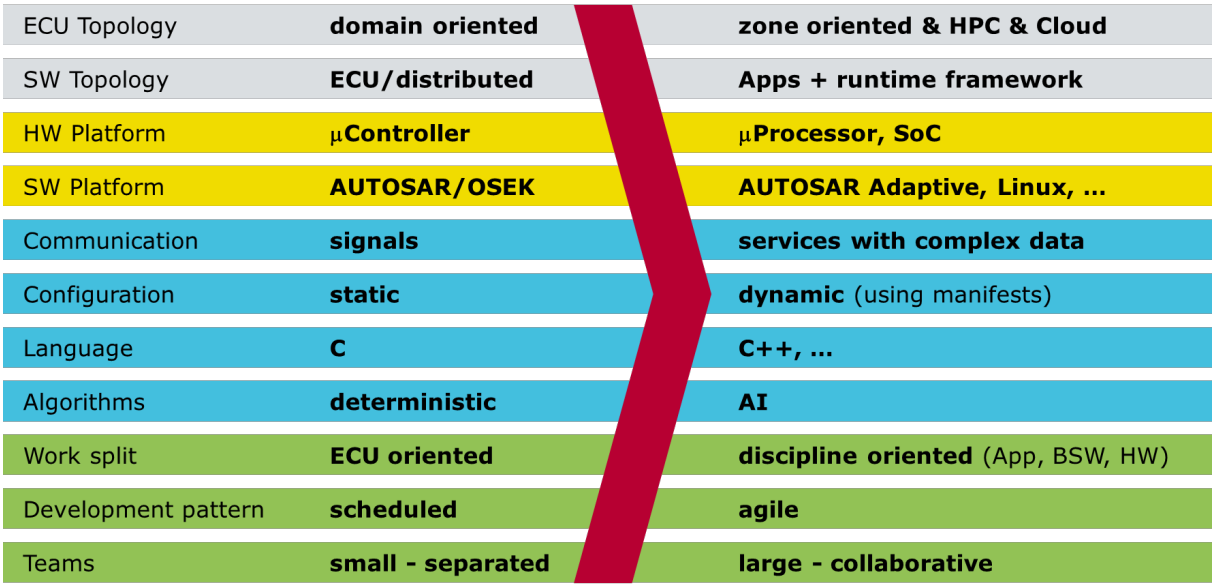

Abbildung 2: Trends in der Fahrzeug-Elektronik

Der Mikrocontroller hat noch lange nicht ausgedient, wird aber zunehmend durch leistungsstarke Mikroprozessor-Systeme ergänzt. Damit kommen auch neue Betriebssysteme, die nicht mehr den klassischen Regeln für Embedded Systeme folgen, zum Einsatz.

So wurde z. B. auch in [3] vorgestellt, dass sich die Diagnose aufgrund mehrerer Faktoren verändern muss und auch kann. Diese sind u. a.:

- Einsatz von Mikroprozessor-Systemen
- Schnellere Release- und Update-Zyklen
- gestiegene Anforderungen wie Datenschutz, Security
- Neue Business-Modelle (z. B. Nachrüstung von Fahrzeug-Funktionen)

Auch die Use Cases für eine Diagnose entwickeln sich recht zügig weiter. Während anfangs nur eine kabelgebundene Diagnose mit einem Werkstatt-Tester im Fokus stand, wurde dieses Gebiet erweitert durch Software-Over-The-Air und Remote-Diagnose.

IT-technische Trends wie die Umsetzung verschiedener Aufgaben in APPs werden auch für die Diagnose gefordert. Dies bedingt eine neue Basis, idealerweise mit IT-Technologien, die dem Stand der Technik entsprechen.

Dadurch ergeben sich auch neue Freiheitsgrade für die Diagnose. Hier seien nur einige exemplarisch genannt:

- Schnelle Bussysteme wie (Gigabit-)Ethernet, Flexray, …
- Security Mechanismen wie TLS, OAuth2, …
- Strukturierte Datenformate wie JSON
- Skalierbare Systeme mit RESTful Ansätzen

Hierfür ist eine sehr breite Toollandschaft verfügbar. Die Technologien gehören zum Stand der Technik sowohl in der Ausbildung als auch in der Industrie. Daher ist die Zahl der Entwickler, die sich mit diesen Technologien auskennen, deutlich höher als die Zahl derjenigen, die z. B. „UDS on CAN“ beherrschen. Es wäre damit sehr viel schneller als früher möglich, eine leistungsfähige Diagnose-Tool-Landschaft aufzubauen. Die Leistung der Entwickler kann damit auf den eigentlich wertschöpfenden Anteil fokussiert werden, anstatt mit einer langwierigen Einarbeitung in spezialisierte Protokolle verzögert zu werden.

Aber wird das alles UDS ersetzen?

Fahrzeuge sind mechatronische Systeme. Für einen Großteil der Aufgaben werden Mikrocontroller-basierte Systeme eingesetzt. Hier hat UDS weiterhin seine Berechtigung. Vielmehr muss also darüber nachgedacht werden, wie sich die beiden Welten kombinieren und integrieren lassen.

Ein Weg kann z. B. der im ASAM Projekt SOVD „Service Oriented Vehicle Diagnosis“ definierte „Classic Diagnostic Adapter“ sein: Er soll die IT-orientierte SOVD Schnittstelle um die „UDS-Mechatronik-Welt“ bereichern. Somit wäre nach außen hin nur noch eine Schnittstelle notwendig.

Auch gibt es verschiedene gesetzlich geforderte Zugänge, die auf UDS basieren. Beispielhaft seien hier genannt:

- Emissions OBD mit der in Entwicklung befindlichen SAE J1979/2
- ePTI „electronic Periodic Technical Inspection“ nach ISO20730 [2]
- End-of-life activation of on-board pyrotechnic devices ISO26021

Ein wichtiger Schritt in Richtung Abbildung auf IT-Technologien ist bereits mit den ISO Standards zum Extended Vehicle gemacht: Mit den Standards ISO20077, 20078 und 20080 werden u. a. Diagnoseaufgaben auf eine http/REST basierte Schnittstelle abgebildet.

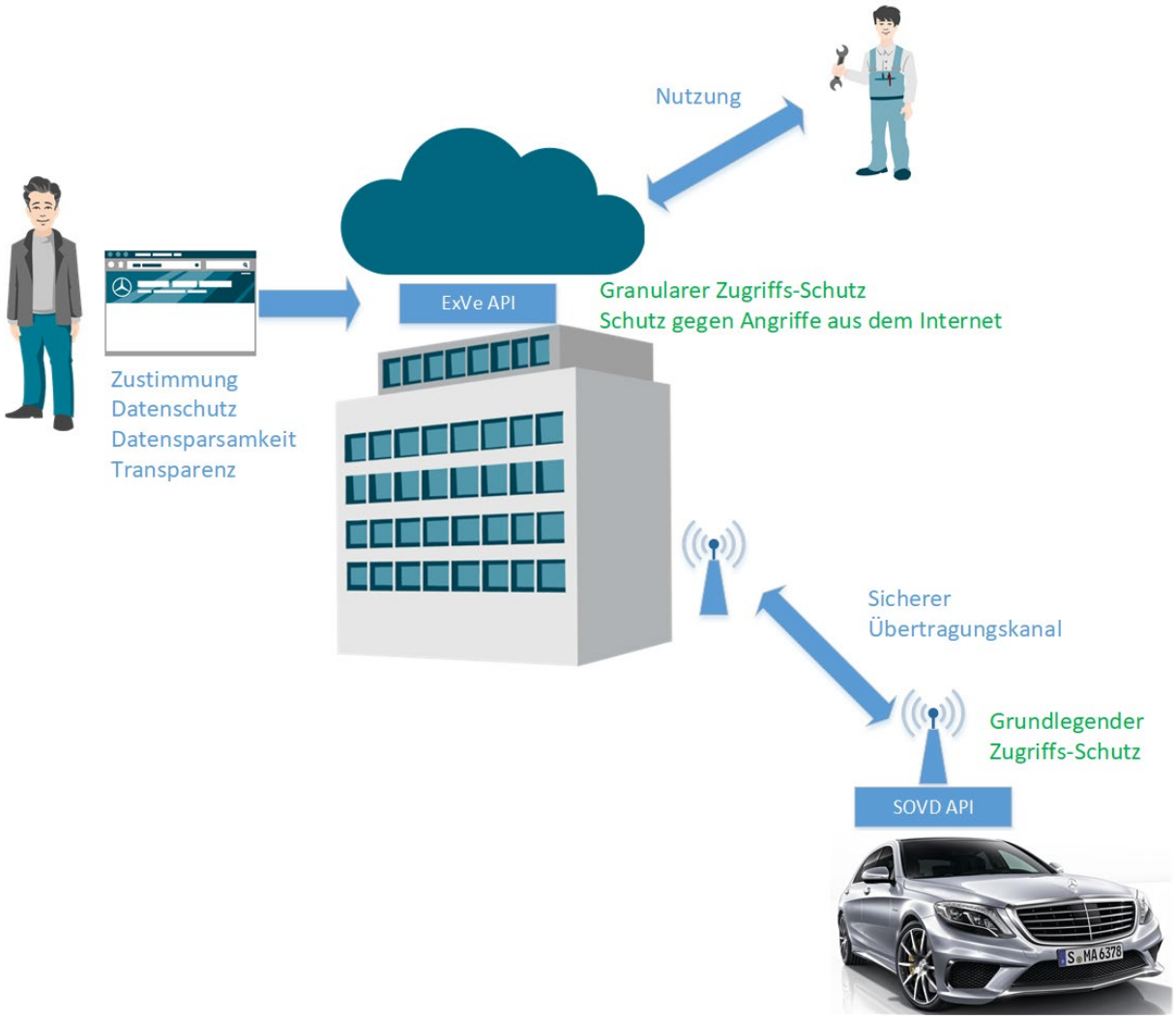

Abbildung 3: Mögliches Zusammenspiel zwischen ASAM SOVD und ISO ExVe

Die beiden Standards passen gut zusammen und ergänzen sich: SOVD ermöglicht den grundsätzlichen Fahrzeugzugang mit grundlegender Security, das ExtendedVehicle ergänzt die feingranulare Zugriffssteuerung mit Kundenzustimmung sowie weit reichenden Security Mechanismen für den Remote Zugriff auf das Fahrzeug.

3 SOVD – Gesamtziele

SOVD, die Service-orientierte Fahrzeugdiagnose, soll die zuverlässige Diagnose von modernen Fahrzeugen mit Hochleistungsrechnern und klassischen Steuergeräten ermöglichen - und das Software-Update solcher Fahrzeuge.

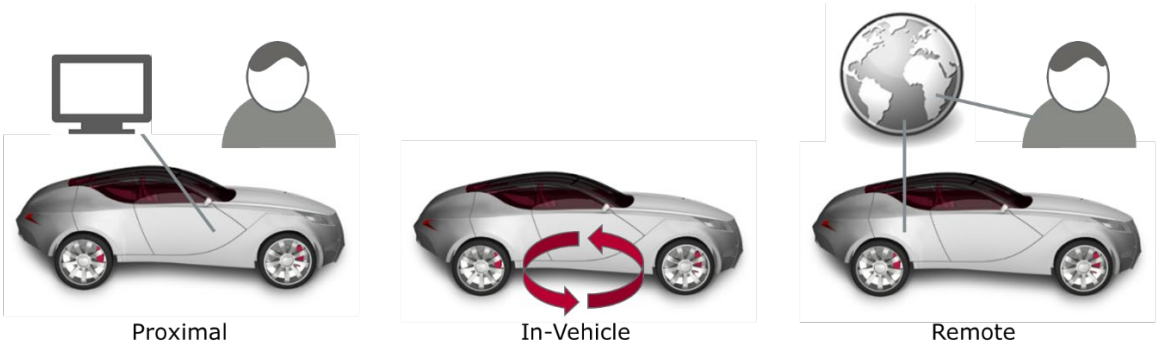

Abbildung 4: SOVD Einsatzszenarien

Ziele

- Ein API für die Diagnose und das Software-Update (fahrzeugübergreifend)
- Ein konsistentes API, das sowohl für neue Systeme als auch für traditionelle Sensor- / Aktor-Systeme genutzt werden kann.
- Einsatzszenarien: proximal (verdrahtet mit dem Auto oder per Nahbereichskommunikation), in-vehicle (mitfahrend im Auto) und remote (aus der Ferne mit dem Auto)
- Ein selbstbeschreibendes API, das – anders als heute - auch eine Diagnose ohne eine externe Beschreibungsdatei ermöglicht. Trotzdem wird man auch künftig irgendeine externe Beschreibung benötigen, um beispielsweise variantenübergreifende Prüfabläufe erstellen zu können.
- Dafür sollen bestehende Technologien ausgewählt und kombiniert werden. Es ist ein ausdrückliches Vermeidungsziel eine neue Technologie zu erfinden.

4 SOVD: native und classic

Innerhalb des Fahrzeugs tragen verschiedenste Softwareanteile und Geräte mit Prozessoren und Controllern zur ganzheitlichen Diagnose bei. Anfragen zu Diagnose und Software-Update werden jeweils an den passenden Bearbeiter weitergeleitet.

Bei solchen Diagnose-Anfragen handelt es sich um Operationen auf einer konkreten Ressource, z.B.

- Lesen von Messwerten und Systemkenngrößen, einzeln oder als Messwertreihe
- Auslesen von Ereignis- und Fehlerspeichern
- Ändern von Parametern
- Starten von speziellen Diagnose-Funktionen
- Eingriff in die normale Steuerungsfunktion von Aktoren und Sensoren

Oder eine Anfrage wird verwendet, um die Selbstbeschreibung („Capability Description“) einer konkreten Ressource (Hardware oder Software-Teil) abzurufen. Die Selbstbeschreibung umfasst alle Diagnoseumfänge, die ggf. abhängig von der Rolle angefragt werden können.

Wird eine Anfrage an einen Hochleistungsrechner gerichtet, so wird die Anfrage direkt und dezentral von seiner Software bearbeitet, sozusagen nativ. Daher spricht man hier auch von native SOVD.

Wird eine Anfrage an ein Steuergerät gerichtet, das die UDS Diagnose unterstützt, so muss die ursprüngliche SOVD Anfrage zunächst übersetzt werden in die UDS Diagnose, die Antwort entsprechend umgekehrt. Es erfolgt also eine Umsetzung von SOVD nach UDS, daher spricht man hier auch von dem Classic Diagnostic Adapter.

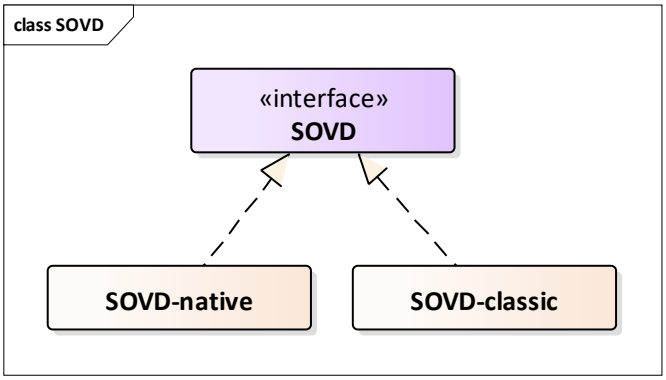

Abbildung 5: SOVD Schnittstelle, native und klassische Umsetzung

Für den Anfragenden sollte es keinen Unterschied machen, ob sich eine Anfrage an einen Hochleistungsrechner oder ein klassisches UDS Steuergerät richtet. Er verwendet die SOVD API und sieht ausschließlich symbolische Werte und Daten.

5 Classic Diagnostic Adapter

Die SOVD-Anfragen an klassische Steuergeräte mit UDS-Diagnose werden an den Classic Diagnostic Adapter geleitet, der die ursprüngliche Anfrage in das UDS-Protokoll übersetzt und an das adressierte Steuergerät versendet. Er überwacht die

Antworten vom Steuergerät und übersetzt sie in eine SOVD Antwort. Außerdem übernimmt er die vom UDS Protokoll geforderten Maßnahmen zur Aufrechterhaltung der Kommunikation und unterschiedlicher Sessions.

Allgemein gesprochen übernimmt der Adapter die aufwendige Extraktion aus dem Bytestrom des UDS Protokolls und die Umrechnung der übertragenen Daten. Dazu benötigt er eine vollständige formale Beschreibung des Diagnoseumfangs jedes Steuergerätes, also die Diagnosebeschreibungsdaten. Diese liegen heute meist im ODX-Format vor.

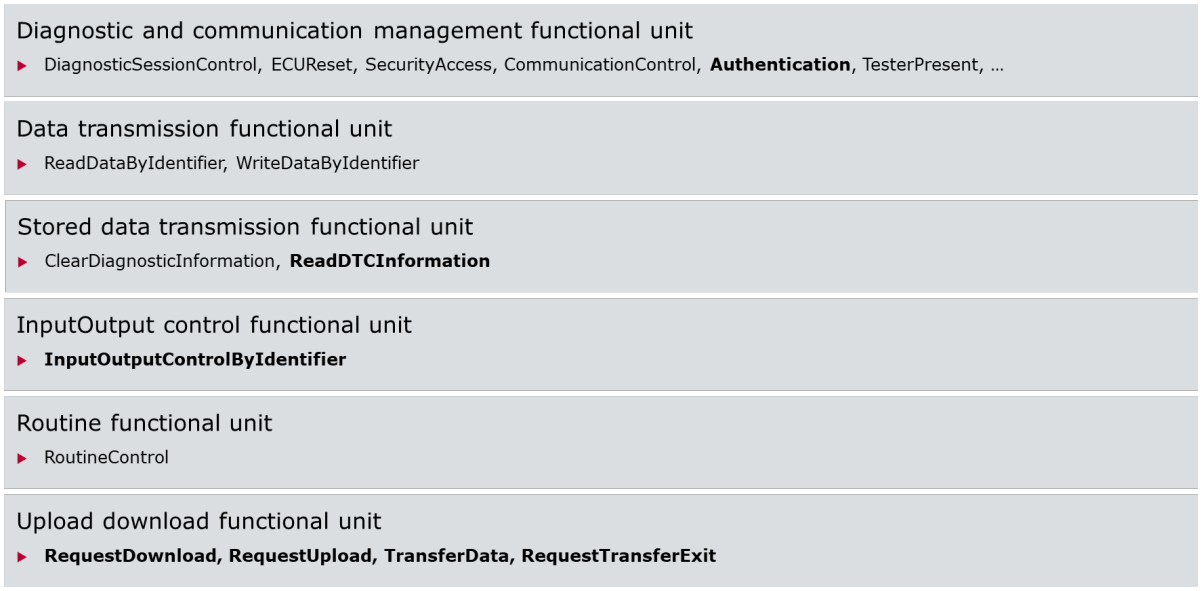

Abbildung 6: Functional Units gemäß UDS Protokoll und zugeordnete Diagnose-Services

Im UDS Standard [5] werden die Protokolldienste verschiedenen Funktionseinheiten (Functional Units) zugeordnet. Die Zusammenstellung verdeutlicht, dass die Funktionalität von UDS weit über das über das einfache Lesen von Datenobjekten hinausgeht.

Ein Classic Diagnostic Adapter bzw. SOVD2UDS Adapter muss alle relevanten Diagnose-Services unterstützen.

Eine besondere Herausforderung bietet das UDS Protokoll, sobald es zustandsabhängig wird, wie z. B. bei DiagnosticSessionControl, Authentication, InputOutputControl, und RoutineControl. Hier ist eine etwas abstraktere Abbildung erforderlich, diese Protokolldienste lassen sich nicht immer sinnvoll 1:1 auf einen SOVD Dienst abbilden.

Die Hauptkomponente eines solchen SOVD2UDS Adapters ist folglich ein vollwertiger und fahrzeugtauglicher UDS-Diagnose-Kernel. Dieser sollte ODX-Daten verarbeiten können und einen Funktionsumfang ähnlich zu dem eines MVCI-Servers besitzen. Allerdings muss der Ressourcenverbrauch im Gegensatz zu einem standardkonformen

MVCI-Server sehr stark optimiert sein, da auch die Ressourcen eines Hochleistungsrechners begrenzt sind und von vielen unterschiedlichen Applikationen beansprucht werden.

Für eine Umsetzung sind folgende Kenngrößen erfolgsentscheidend:

- Speicherbedarf (Primär- und Sekundärspeicher)
- Startzeiten
- Laufzeiten

6 Ein SOVD2UDS Prototyp für ein reales Fahrzeug

In der ASAM Arbeitsgruppe ist heute (zum Zeitpunkt der Erstellung dieses Beitrags) noch keine Entscheidung in Bezug auf die Basis-Technologie für SOVD gefallen. Die Diskussionen dazu starten gerade in der ASAM Arbeitsgruppe und werden in den kommenden Monaten fortgeführt. Die hier vorgestellte prototypische Umsetzung basiert auf http/REST, da die Projektbeteiligten von Mercedes und Vector diese Technologieoption zu Beginn des Projektes als attraktiv und vielversprechend eingeschätzt haben.

6.1 Ziele

Hauptziele der prototypischen Umsetzung waren Antworten auf folgende Fragen:

- Wie ist die Performance der Technologie für diesen Anwendungsfall?
  Wie groß sind insbesondere Start- und Ladezeit, Laufzeit, Speicherverbrauch?
- Welche Vor- und Nachteile bringt der Einsatz der Technologie?
- Wie fühlt sich ein Diagnose-Tester auf Basis der Technologie an?
  Wie groß ist der Aufwand zur Erstellung eines solchen Testers?
- Wie geht man mit den zustandsabhängigen UDS-Services bestmöglich um?
  Was behandelt man im Adapter, was überlässt man dem Aufrufer?
- Welches sind die kritischen Pfade einer serientauglichen Umsetzung?
- Wie geht man mit den Steuergerätebeschreibungsdateien im Fahrzeug bestmöglich um?

6.2 Aufbau und Architektur

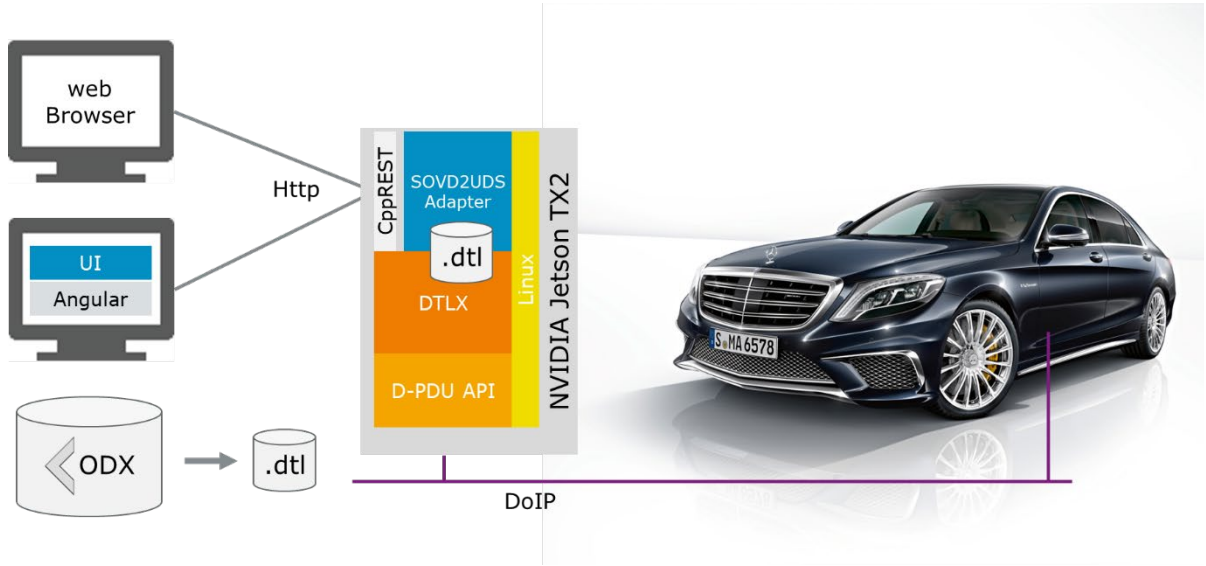

Abbildung 7: Aufbau und Architektur der prototypischen Umsetzung

Die Abbildung zeigt den Aufbau und die Architektur der prototypischen Umsetzung.

Der SOVD2UDS Prototyp der Vector Informatik GmbH läuft auf einem NVIDIA Jetson TX2 Board, welches über DoIP mit einem realen Fahrzeug kommuniziert. Bei dem Jetson TX2 handelt es sich um einen quad-core ARM A57 complex 2 GHz mit 8 GB DDR4. Als Betriebssystem kommt Linux zum Einsatz.

Der SOVD2UDS Prototyp besteht aus folgenden Komponenten:

- Standardisierte D-PDU API [6] für die Kommunikation über CAN oder DoIP
- Fahrzeugtauglicher Diagnose Kernel (DTLX) – Vector Produktkomponente
- SOVD2UDS Adapter
- Frei verfügbare C++ REST Bibliothek

Der Diagnose-Kernel verarbeitet Diagnosebeschreibungsdaten in einem laufzeitoptimierten Binärformat (.dtl), welches aus ODX-Daten generiert wird.

Im Expertenmodus nutzt der Anwender einen normalen Web-Browser (Http), um den SOVD2UDS Prototypen anzusprechen. Alternativ nutzt der Anwender eine komfortable Web-Anwendung, die die übertragenen Inhalte graphisch aufbereitet.

6.3 REST in aller Kürze

- REST wird in der Regel auf http umgesetzt, kann daher in jedem Browser direkt verwendet werden.
- Bei REST steht eine Ressource im Mittelpunkt. Ressourcen sind hierarchisch organisiert.
- Bei einem Aufruf ruft man eine Http-Methode (GET, POST, PUT oder DELETE) für eine bestimmte Ressource auf.
- Ein REST API soll ohne Vorkenntnisse genutzt werden können - außer der initialen URL. Der Aufruf der initialen URL liefert eine Liste von Links.
- REST ist zustandslos, eine Anfrage enthält alle Informationen, die für den Server bzw. Client notwendig sind, um die Nachricht zu verstehen.

6.4 REST in der Praxis

Der folgende Screenshot zeigt einen Web-Browser, der die vom Body-Steuergerät abgefragte aktive Diagnose-Information darstellt.

In der Adressleiste sieht man die Ressource, auf die lesend zugegriffen wird. Wird eine Ressource ohne explizite REST-Methode angefragt, führt der Browser implizit ein GET aus.

Der SOVD2UDS Adapter sucht den passende UDS-Service aus und sendet ihn an das Steuergerät. Im Fenster wird der Body der Antwort vom SOVD2UDS Adapter dargestellt, hier im formatierten JSON-Format. Alle einzelnen Werte werden bereits interpretiert geliefert.

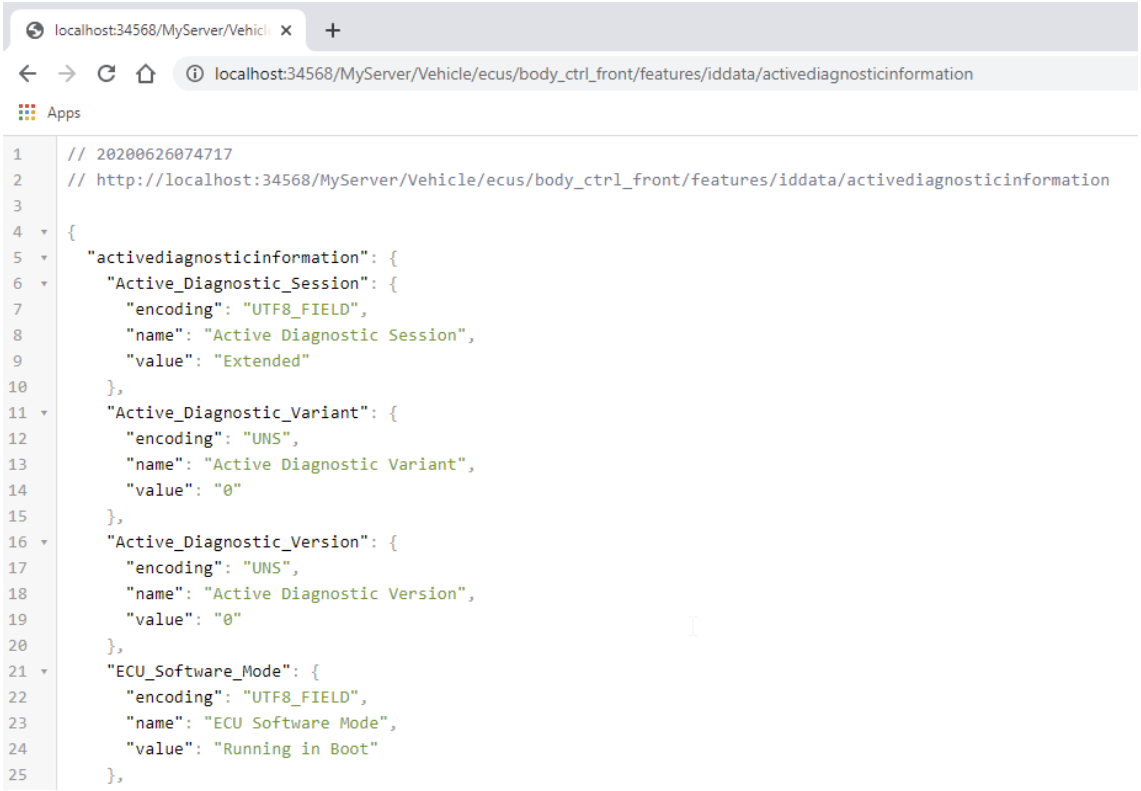

Abbildung 8: Active Diagnostic Information, abgefragt vom SOVD2UDS Adapter

Der folgende Screenshot zeigt einen Web-Browser, der die vom Türsteuergerät unterstützen identifizierenden Diagnose-Dienste auflistet. Diese Liste wird aus den Beschreibungsdaten erzeugt (und nicht vom Steuergerät abgefragt).

Der Anwender kann sich mittels der in der Antwort übertragenen Ressourcen durch den Diagnoseumfang eines Steuergerätes durchhangeln - hier sogar einfach per Hyperlink.

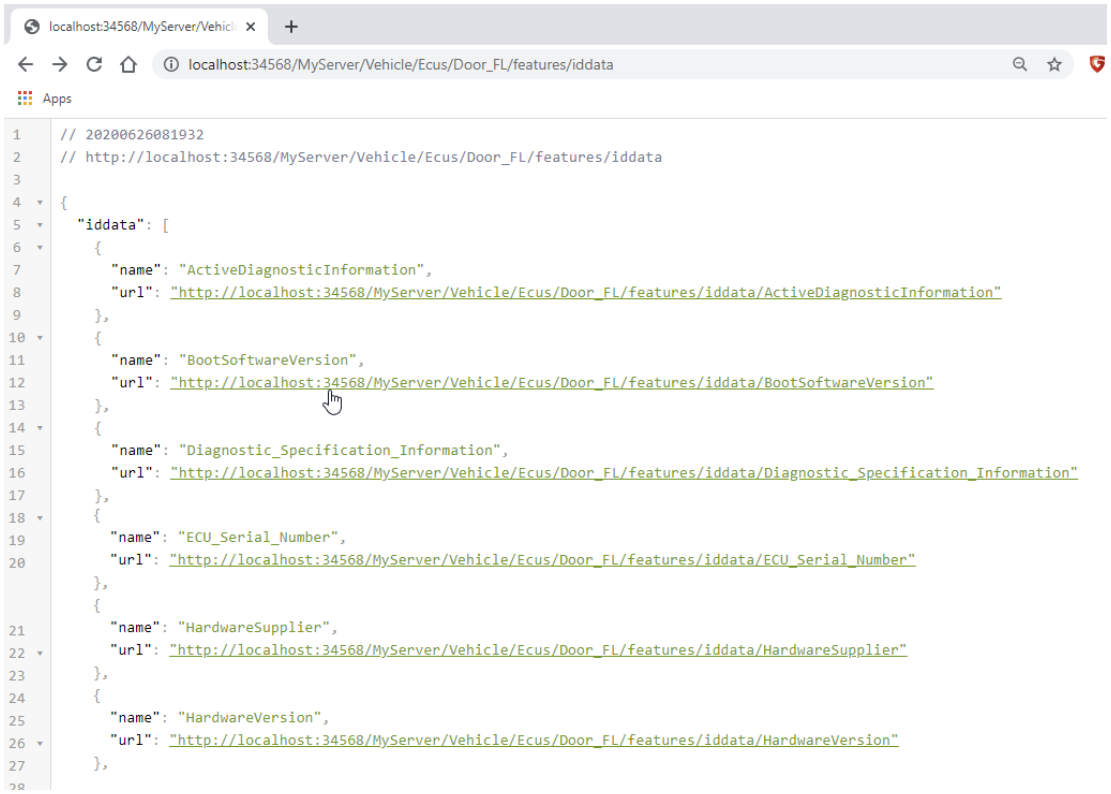

Abbildung 9: Identifikationsdienste, aus der Datenbeschreibung erzeugt

Der folgende Screenshot zeigt eine Web-Anwendung, die die oben skizzierte REST-API verwendet und die übertragenen Inhalte des Fehlerspeichers graphisch aufbereitet.

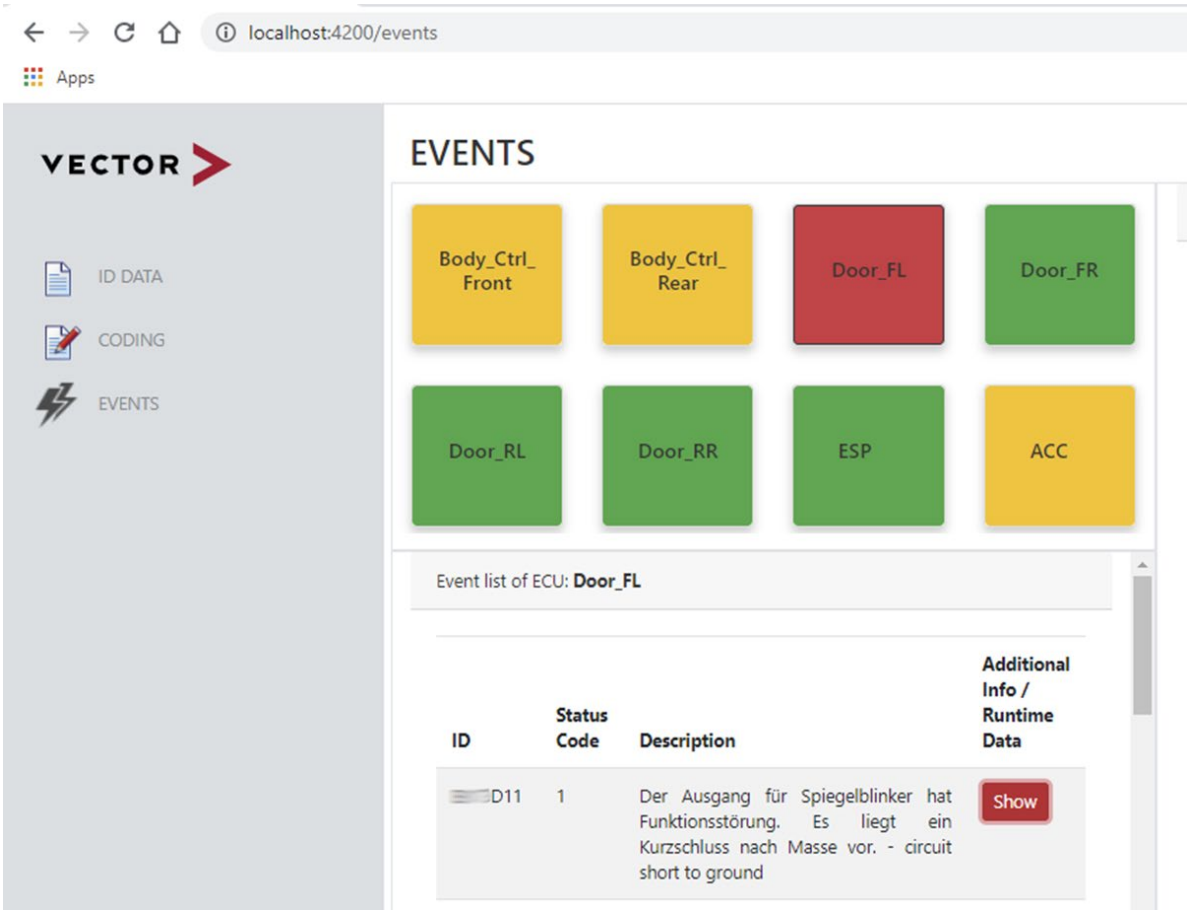

Abbildung 10: Fehlerspeicherinhalt, hier einfach graphisch aufbereitet

6.5 Kennzahlen

Wie oben ausgeführt ist die Performance entscheidend für die Serientauglichkeit. Die erreichten Kennzahlen erfüllen die Erwartungen voll bzw. übertreffen sie sogar.

Speicherverbrauch

Fahrzeug	PDX-Datei (MB)	Laufzeit-Datei (MB)	RAM (MB)
Komplette Baureihe	242	16	110

Abbildung 11: Speicherverbrauch

Lade- und Startzeit

Fahrzeug	Startzeit (ms) *
Komplette Baureihe	400

Abbildung 12: Lade- und Startzeit

* Die Startzeit umfasst nicht das Öffnen der enthaltenen Logical Links und die Variantenerkennung. Diese Zeit wird wesentlich von der die Kommunikationszeit bzw. den Timeouts der nicht verbauten Steuergeräte bestimmt.

Laufzeit

ReadActiveDiagnoseInformation	Laufzeit (ms)
Antwortzeit des Steuergerätes	5
UDS und (ODX-) Interpretation	2
REST-Adapter	1
http (End2End)	8
gesamt	**16**

Abbildung 13: Laufzeit (Referenz: ReadActiveDiagnoseInformation)

6.6 Technologiebewertung http/REST für die Diagnose

Insgesamt sind die Erfahrungen mit http/REST sehr positiv. Folgende Punkte sind hier erwähnenswert:

Vorteile:

- Über die Ressourcen-Struktur lässt sich die Hierarchie im Fahrzeug (sowohl physikalisch als auch logisch) elegant abbilden.
- Insgesamt ist wenig „Protokoll“ sichtbar, der Fokus liegt stark auf den Inhalten. So sind die Daten direkt verwertbar und anwendertauglich.
- Die Selbstbeschreibung über HATEOAS (Hypermedia as the Engine of Application State) bietet die Möglichkeit zu navigieren und durch die Inhalte zu surfen. Die übertragenen Inhalte sind dynamisch und können leicht an unterschiedliche Diagnoseinhalte oder Rechte und Rollen angepasst werden.
- OAuth2 eignet sich gut um Ressourcen-spezifische Zugriffsregeln umzusetzen (für Http, nicht nur für REST).
- Insgesamt gibt es viele frei verfügbare Werkzeuge, die elegante Lösungen mit wenig Aufwand ermöglichen.
- Http/Tcp basierte Technik ist gut geeignet für eine App2App Kommunikation.
- Der Fokus auf physikalische Werte erleichtert die Entwicklung spezialisierter Diagnose-Anwendungen.

Nachteile:

- Ein anderes Kommunikationsparadigma als Request/Response (z. B. on event) lässt sich mit REST über Http nicht direkt abbilden.
- Im Vergleich zu einer nativen UDS Anbindung ist die Laufzeit schlechter, erfüllt aber die Anforderungen für die angedachten Anwendungsfälle
- Http bedingt einen Overhead, der speziell für die Übertragung einzelner, kurzer UDS Dienste erheblich ist.
- Zustandsbehaftete Umfänge des UDS-Protokolls wie z. B. DiagnosticSessionControl, InputOutputControl, und RoutineControl lassen sich abbilden, müssen aber mittels Hilfs-Konstrukten ausgedrückt werden.

6.7 Fazit

Neue Fahrzeugarchitekturen und Hochleistungsrechner im Fahrzeug erfordern auch eine neue Diagnose. Diese neue Diagnose ergänzt die klassische Diagnose, es ersetzt sie nicht.

Genau wie die kontinuierlich weiterentwickelten Systeme muss sich auch die Diagnose kontinuierlich weiterentwickeln. Es drängt sich auch in der Diagnose der Einsatz moderner IT-Technologien auf.

Mercedes und Vector haben in einem gemeinsamen Projekt die Technologie http/REST prototypisch für die Diagnose umgesetzt und evaluiert. Die gesteckten Projektziele wurden erreicht, alle oben gestellten Fragen beantwortet.

Die verwendete Technologie http/REST ist fahrzeugtauglich. Auf der einen Seite bringt sie viele neue Möglichkeiten, auf der anderen Seite hält sich der Ressourcenverbrauch / Overhead in Grenzen.

Jede verwendete Technologie benötigt Laufzeit und Speicherbedarf zusätzlich zum nativen UDS-Diagnose-Kernel, insofern ist auch die Laufzeit und Speicherverbrauch des eingesetzten des UDS-Diagnose-Kernels erfolgsentscheidend. Der für den Prototyp eingesetzte Diagnose-Kernel ist fahrzeugtauglich und benötigt vergleichsweise sehr wenig Speicher bei sehr guten Laufzeiten.

Jeder installierte Web-Browser (ohne Plugins) kann bereits als rudimentärer Diagnose-Tester genutzt werden. Einige Plugins bieten zusätzlichen Nutzen und verbreitern die Einsatzmöglichkeiten des Browsers. Es war einfach und wenig aufwendig eine komfortable graphische Benutzeroberfläche prototypisch zu erstellen.

Literatur

[1] W. Knape, „Neue Security Konzepte in der Diagnose“, “, Diagnose in mechatronischen Fahrzeugsystemen XIII, TUDpress, 2019, S. 202-212…

[2] B. Gottschalk, „Ein neuer Diagnose-Standard: ISO 20730 – electronic Periodic Technical Inspection“, Diagnose in mechatronischen Fahrzeugsystemen XIII, TUDpress, 2019, S. 108-118

[3] S. Stimmler, M. Zoll, „Impact of future Vehicle Architectures on Diagnostic Processes and Tools”, Vector Congress, Stuttgart – November, 20th, 2018

[4] Dr. G. Heling, “Setting the Direction of Automotive EE”, Vector Congress North America, Novi - October 8th, 2019

[5] ISO 14229-1:2020 Road vehicles – Unified diagnostic services UDS) – Application layer

[6] ISO 22901-2:2017 Road vehicles – Modular vehicle communication interface (MVCI) – Diagnostic protocol data unit (D-PDU API)