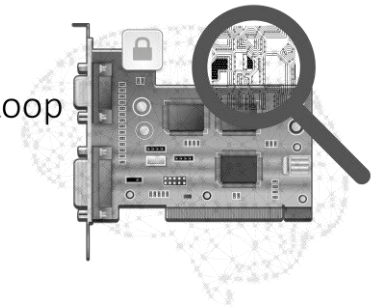
Security
Hardware-in-the-Loop
Observing-AI
Risk-Awareness
Testing



Security-Centered HiL-Platform Offering Risk-aware Testing

Gefördert von:

Bundesministerium
für Bildung
und Forschung

Partnerspezifischer Abschlussbericht des Verbundprojekts SHORT

Teilvorhaben: Innovatives HiL-Testsystems zur Gewährleistung
der Security von Automotive-Technologien
und Einbettung in einen sicheren Entwicklungsprozess

13.04.2023

Dokument Informationen	
Typ	Abschlussbericht
Berichtszeitraum	01.04.2019 bis 30.09.2022
Zuwendungsempfänger	iSyst, intelligente Systeme GmbH
Förderkennzeichen	16KIS0965K
Datum	30.09.2022
Vorhabensbezeichnung	Security-Centered HiL-Platform Offering Risk-aware Testing
Teilvorhaben	Innovatives HiL-Testsystems zur Gewährleistung der Security von Automotive-Technologien und Einbettung in einen sicheren Entwicklungsprozess
Laufzeit des Vorhabens	01.04.2019 bis 30.09.2023

Kontaktinformationen		
Name	Position	Telefon / E-Mail
Michael Frommberger	Projektleiter	+49 911 37665 – 114 michael.frommberger@isyst.de
Michael Jellen	Teamleiter und technische Umsetzung	+49 911 37665 – 116 michael.jellen@isyst.de

Short-Konsortium	
Partner **Koordinator**	iSyst Intelligente Systeme GmbH (iSyst) Hugo-Junkers-Straße 9 90411 Nürnberg Dr. Michael Frommberger +49 911 37665 114 Michael.Frommberger@isyst.de Michael Jellen +49 911 37665 116 Michael.Jellen@isyst.de
Partner	Technische Hochschule Deggendorf (THD) Dieter-Görlitz-Platz 1 94469 Deggendorf Prof. Dr. Martin Schramm +49 991 3615 578 Martin.Schramm@th-deg.de
Partner	Luxoft GmbH Parkring 57-59 85748 Garching bei München Dr.-Ing. Nicolas Gay N.Gay@luxoft.com Hatice Yenidede Hatice.Genc@dxc.com

INHALT

1 Thema und Zielsetzung von SHORT

Im Folgenden wird eine kurze Übersicht über das Projektthema gegeben.

1.1 Aufgabenstellung

Die Vernetzung von eingebetteten Systemen ist eine der wichtigsten Voraussetzungen für autonomes Fahren, Car2X-Kommunikation, Car2Car-Kommunikation, Industrie 4.0 und das Internet of Things. Vor allem in der Automobilbranche werden Kilometer an Kabeln verbaut, um die Vernetzung der fahrzeuginternen Kommunikation zu gewährleisten. Obwohl der Trend von vielen einzelnen im Fahrzeug verbauten Systemen wieder zurück geht, hin zu abgekapselten Systemen für sicherheitskritische Themen, bleiben viele Schnittstellen dabei angreifbar. Der hohe Vernetzungsgrad erhöht die Bedrohung durch Angreifer, da er eine breite Angriffsfläche bietet. Das Problem wird noch verstärkt, wenn die Fahrzeuge mit der Außenwelt kommunizieren, wo verschiedene Hersteller verschiedenste Technologien einbinden.
Die Verwendung von Zertifikaten und Signaturen sowie Verschlüsselungen der Übertragungen wie sie im AUTOSAR Release 4.3 ausführlich definiert sind, sollen die hohen Sicherheitsanforderungen unterstützen. Um diese gewinnbringenden Technologien welche Rund um das Thema der vernetzten Systeme stehen aber auch sicher einsetzen zu können, ist es erforderlich, schon während der Entwicklung auf eine deutlich bessere und effizientere Absicherung als in früheren Entwicklungsprozessen zu setzen. Um dies zu gewährleisten ist hier eine ganzheitliche Betrachtung der Themen Safety und Security notwendig.

Im Projekt SHORT wurde der spezielle Anwendungsfall des teleoperierten Fahrens näher betrachtet. Beim teleoperierten Fahren werden Fahrzeuge ferngesteuert, d.h. ohne Beteiligung eines im Fahrzeug anwesenden Fahrers. Zustandsdaten und Telemetriedaten werden per Funk, Mobilfunk oder andere Funktechnologien an den Operator übertragen. Die Angriffsflächen einer abgesicherten Kommunikation liegen hier also nicht nur innerhalb des Fahrzeugs oder nahe am Fahrzeug, sondern können auch innerhalb des gesamten Funknetzwerks zum Tragen kommen.
Neben der Übertragungsqualität stellen auch die Vertrauenswürdigkeit und Integrität der Daten eine hohe Anforderung an die Systeme. Bei der Entwicklung dieser Systeme müssen neben der Einhaltung der Normen und Methoden zur Absicherung auch die funktionalen Tests mit in das Safety und Security Konzept einbezogen werden. Ein besonderer Augenmerk im Projekt SHORT lag daher auf dem HiL-System für den funktionalen Test. Das HiL-System ermöglicht eine Nachbildung der Umgebung für ein Steuergerät, wobei durch die Implementierung entsprechender Algorithmen und Protokolle in das Testsystem realitätsnahe Szenen nachgebildet werden sollen.

Die wesentliche Hauptaufgabe des Teilprojekts der Firma iSyst war es, zum übergeordneten Gesamtprojekt beizutragen, sowie die Projektkoordination zu übernehmen.
Die Arbeiten innerhalb dieses Teilvorhabens befassten sich mit folgenden Punkten:

- Erarbeitung eines Konzeptes für die Verbindung von Security-Tests mit dem funktionalen Test von eingebetteten Systemen unter Berücksichtigung der Möglichkeiten von HIL-Testsystemen
- Entwicklung automatisierbarer Testmethoden für den Security-Test eingebetteter Systeme
- Entwicklung von Methoden der automatisierten Bewertung des Verhaltens des Device under Tests (DUT) bei der Ausführung von Security-Tests (automatisierte Bestimmung von Passed/Failed)
- Entwicklung eines umfassend Entwicklungs- und Testprozesses bei dem die Anforderungen an die Security und die Safety (funktionale Sicherheit) gleichermaßen berücksichtigt sind und die Testbarkeit aller Aspekte gewährleistet wird
- Implementierung von Testmethoden und Testtools für den automatisierten Security-Test von eingebetteten Systemen unter Berücksichtigung von Risikobewertungen
- Implementierung automatisierter Methoden zur Generierung von Testfällen und Testvektoren (Testdatenraum für einen Testparameter)
- Integration von bestehenden Datenbanken für Schwachstellen (z.B. Common Vulnerabilities and Exposures (CVE – deutsch: Bekannte Schwachstellen und Anfälligkeiten) und KI-Systeme für die automatisierte Erzeugung von Testfällen und Testvektoren
- Implementierung und Erprobung der entwickelten Tools und Methoden an einen praktischen Anwendungsfall in Form eines Demonstrators

1.2 Voraussetzungen

Angestrebtes Ziel von SHORT ist es durch die Entwicklung eines HiL-Testsystems bereits in der Entwicklungsphase sowohl die funktionale Sicherheit (Safety) als auch Security zu gewährleisten. Wobei im Teilvorhaben der iSyst intelligente Systeme GmbH die Konzeption und Entwicklung der HiL-Lösungen im Fokus stand.

1.3 Planung und Ablauf des Vorhabens

SHORT gliedert sich in sechs Arbeitspakete (AP 1 bis AP 6) und umfasst, mit der gewährten halbjährigen Verlängerung, eine Laufzeit von 42 Monaten (01.04.2019 bis 30.09.2022). Der Zusammenhang der Arbeitspakete ist schematisch in Abbildung 1 dargestellt.

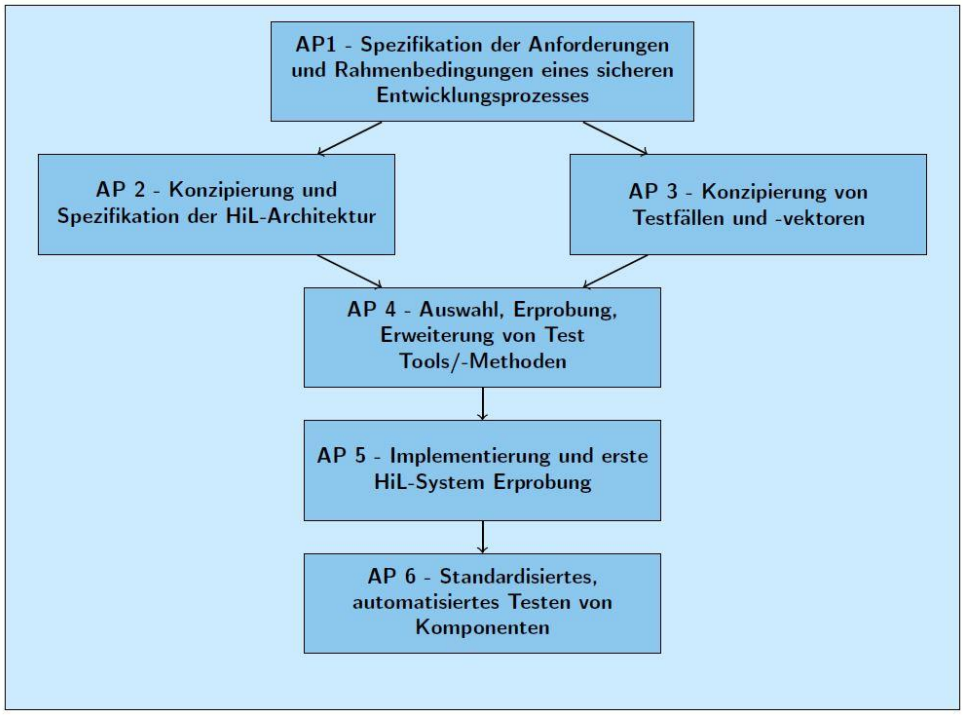


Abbildung 1: Übersicht der Arbeitspakete

In Abbildung 2 ist die zeitliche Planung des Projektes in Bezug auf Arbeitspakete (AP) und Tasks (T) dargestellt.

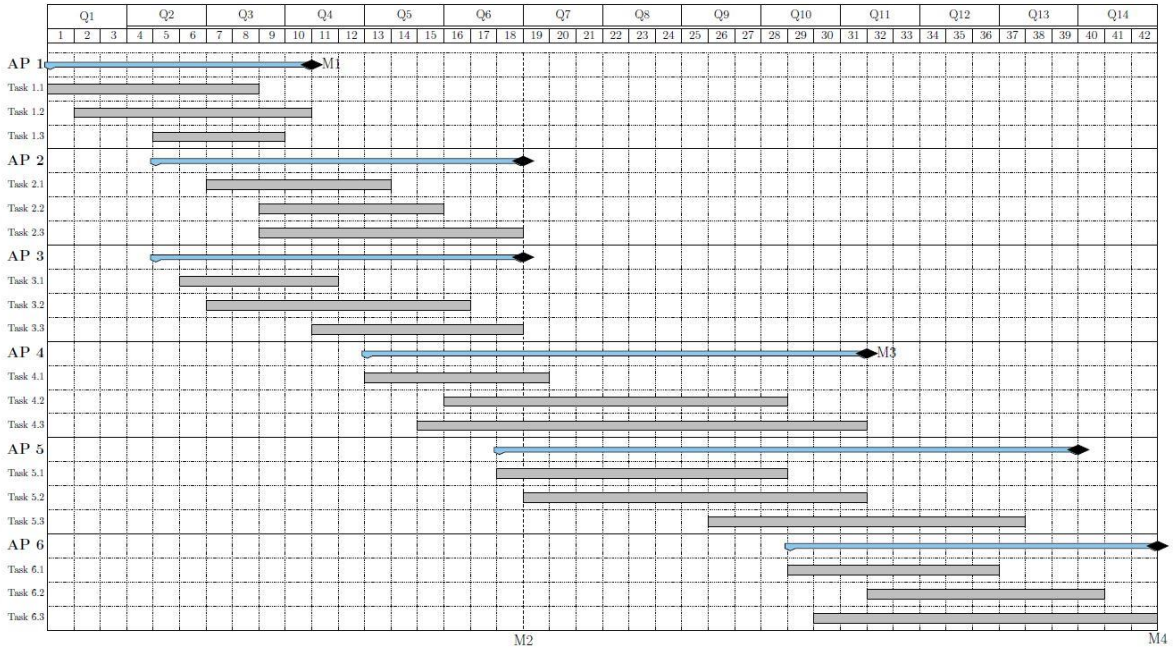


Abbildung 2: Gantt-Chart - Darstellung der Arbeitspakete und Meilensteine

Während der Projektlaufzeit von SHORT kam es durch die COVID-19-Pandemie zu einigen Herausforderungen bzgl. der vorgesehenen zeitlichen Projektplanung.

So haben sich die Abarbeitung bzw. der Start einiger Arbeitspakete zeitlich verzögert und spätere Tasks nach hinten verschoben. Durch die Coronamaßnamen bei der TH Deggendorf und den Industriepartnern konnten zwar die generellen Arbeiten am Projekt fortgesetzt werden, allerdings deutlich weniger effektiv als ohne diese Einschränkungen, woraus sich eine Verzögerung von ca. einem halben Jahr ergab. Dies konnte durch genaue Betrachtung der einzelnen Arbeitspakete und Tasks von der Fa. iSyst ermittelt, bzw. bestätigt werden. Deshalb wurde Anfang 2022 der Antrag auf kostenneutrale Verlängerung um sechs Monate für das Projekt SHORT beim Projektträger gestellt. Der Änderungsbescheid hinsichtlich Genehmigung der Verlängerung um ein halbes Jahr ist am 04.03.2022 eingegangen.

1.4 Wissenschaftlicher und technischer Stand

Der funktionale Test im automotive Bereich beschäftigt sich vorwiegend mit der Kommunikation, Sensorik und Aktorik. Diese Bereiche können durch ein HiL-Testsystem auch mit harten Echtzeitanforderungen abgetestet werden. An diesem technischen Stand soll angeknüpft werden, um Security-Tests zur Validierung kryptografischer Verfahren im Inter- und Intra-Fahrzeug-Kontext zu etablieren.

1.5 Angabe von Fachliteratur, Informations- und Dokumentationsdiensten

- Als Standardwerk für Softwaretests bei der Fa. iSyst wird das Buch „Basiswissen Softwaretest“ [1] genutzt.
- Eine weitere Quelle für aktuelle Softwaretestmethoden und Tools ist das ISTQB (International Software Testing Board) [2]
- ”modTF - ein modulares Framework zur Testautomatisierung – Durchgehende Traceability von den Anforderungen zu den Testresultaten”, Kristian Trenkel, Embedded Software Engineering Kongress 2013, Sindelfingen, ISBN: 978-8343-2408-5, S. 358-367
- ”Konzept und Umsetzung einer modularen, portierbaren Middleware für den automatisierten Test eingebetteter Systeme”, Kristian Trenkel, Universitätsverlag Chemnitz, 2016, ISBN: 978-3-944640-70-9
- ”Absicherung digitaler Sensorschnittstellen in sicherheitskritischen Anwendungen”, Kristian Trenkel, 5. Landshuter Symposium Mikrosystemtechnik, Landshut, ISBN: 978-3-9812696-9-7, S. 21-29
- ”Einsatz von Debuggern im Hardware-in-the-Loop-Test”, Kristian Trenkel, Embedded Software Engineering Kongress 2016, Sindelfingen, ISBN: 978-3-8343-2504-4, S. 478-484
- ”Modulare Testsysteme am Entwicklerarbeitsplatz”, Kristian Trenkel, Embedded Design 1/2016, S. 34-36

- ”Herausforderungen beim Test vernetzter Fahrzeuge”, Kristian Trenkel, ASQF Testing Day Franken 2017
- ”Absicherung von Testsystemen – das kalibrierte Hardware-in-the-Loop-Testsystem”, Kristian Trenkel, Embedded Software Engineering Kongress 2017, Sindelfingen, ISBN: 978-3-8343-3426-8, S. 432-440

1.6 Wesentliche Ergebnisse

Eines der Hauptergebnisse war die Integration eines Fuzzers mit KI-Anteilen, in ein HiL-Testsystem. Dabei ergeben sich für das HiL-Testen neue Möglichkeiten aber auch Herausforderungen, welche noch zu untersuchen sind.
Die Art und Weise des Testens mit einem KI-gestützten Tool unterscheidet sich von konservativen Methoden schon bei der Spezifikation von Testfällen.
Die folgenden Punkte können als Ergebnis aus dem SHORT-Projekt für die Fa. iSyst belegt werden:

- Bedrohungsanalyse eines zu testenden Objekts mit Hilfe von Softwaretools wie z.B. Microsofts® Threat Modeling Tool in den Softwareentwicklungsprozess mit einzubeziehen
- Analyse und Einbettung der IoT Protokolle ZMQ und MQTT in eine HiL-Testarchitektur
- Evaluierung neuer Schnittstellen und neuer Softwaretools wie dem „BooFuzz“-Fuzzer der TH-Deggendorf
- Integration der neuen Testtools – speziell die Integration des Fuzzers mit der cW-GAN-GP[1] Architektur
- Erlernen des Umgangs und der Erfahrungssammlung mit KI-unterstützten Werkzeugen

1.7 Zusammenarbeit mit anderen Stellen

Außerhalb des Konsortiums wurde mit keiner anderen Stelle zusammengearbeitet. Innerhalb des Konsortiums wurden die Erfahrungen aus Industrie (Fa. Luxoft und Fa. iSyst) und Forschung (TH-Deggendorf) zusammengeführt. Neueste wissenschaftliche Erkenntnisse wurden von der Technischen Hochschule Deggendorf eingebracht. Die Zusammenarbeit mit den Projektpartnern erfolgte in mehreren persönlichen Treffen, sowie alle 14 Tage per Microsoft Teams.

[1] Siehe Abschlussbericht der TH-Deggendorf

2 Eingehende Darstellung des Projekts SHORT

Zusammenfassung der in den Arbeitspaketen erzielten Ergebnisse und Fortschritte

2.1 Erzielte Ergebnisse

Im ersten Arbeitspaket wurden Anforderungen erarbeitet, wie das HiL-Testsystem in den Entwicklungsprozess der funktionalen und sicherheitsrelevanten Tests zu integrieren ist.

2.1.1 AP 1 – Spezifikation der Anforderungen und Rahmenbedingungen eines sicheren Entwicklungsprozesses

Im Rahmen des Projekt Kick-Offs wurden die Möglichkeiten zur Verwendung eines Systems des Projektpartners Luxoft (ehemals Fa. Objective) als Beispiel für ein Device Under Test (i.w. DUT) besprochen. Es wurde das „Teleop-System“ mit der zentralen Komponente des „Teleop-Modules“ (siehe Abbildung 3) ausgewählt.
Dabei ist das Teleop-System für die Fernsteuerung und -Bedienung von Fahrzeugen und Baumaschinen über WLAN oder Mobilfunk gedacht. Es vereinigt damit alle Aufgabenstellungen des Projektes SHORT in einem System.

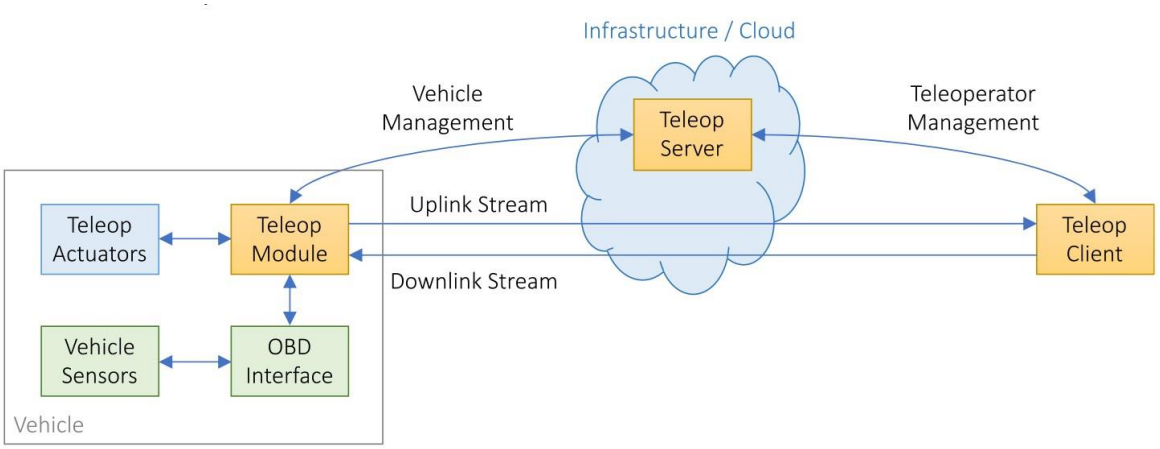


Abbildung 3: Konzept des TeleOp-Systems

Das TeleOp-Module stellt dabei die Schnittstelle zum Fahrzeug dar, welches ferngesteuert werden soll. Während die eigentliche Fernsteuerung über den Client erfolgt. Hierzu wird dem Operator ein HMI über einen node.js-basierten Webserver zur Verfügung gestellt. Das Management der Kommunikation zwischen Module und Client erfolgt über den Server. Alle drei Umgebungen, also Client, Server und Module, werden in einem Docker-Container ausgeführt. Die Kommunikation zwischen den einzelnen Umgebungen kann der Abbildung 4 entnommen werden.

Server, Module und Client kommunizieren über das Machine-to-Machine-Protokoll „Message Queuing Telemetry Transport“ (MQTT). Der dafür benötigte MQTT-Broker ist auf dem Server installiert. Der Nachrichtenaustausch zwischen den Umgebungen findet über das „Publishen“ und „Subscriben“ auf bestimmte „Topics“ statt, wobei für die Serialisierung Google Protobuf verwendet wird.
Weiterhin werden im TeleOp-System Protokolle zum Videostreaming (TCP, UDP, RTP) und die ZMQ-Bibliothek zum Streaming von Sensordaten verwendet.

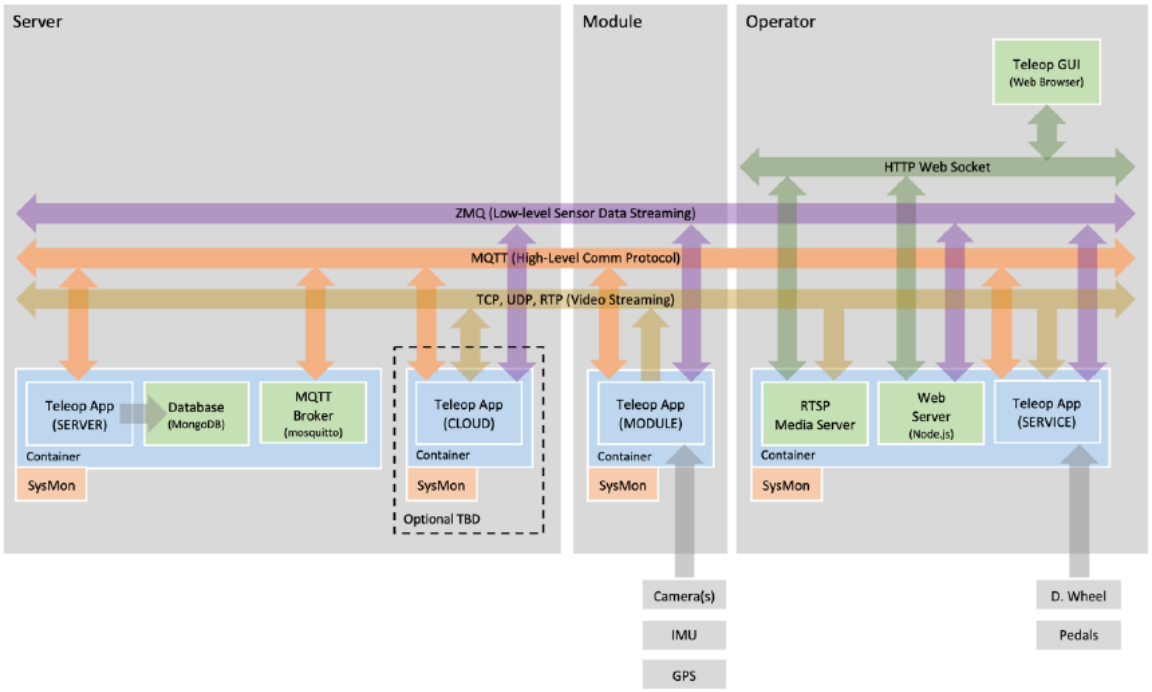


Abbildung 4: Übersicht, TeleOp-Kommunikation

Das im vorigen Abschnitt erwähnte DUT stellt die Implementierung eines teleoperierten Fahrzeugs über ein Cloud-Netzwerk dar. Das effiziente Testen eines solchen Systems unter Berücksichtigung aller potenziellen funktionalen Sicherheits- und IT-Sicherheitsrisiken in der HIL-Umgebung ist der Hauptfokus dieses Projekts. Dieser Abschnitt wird in AP3 weiter ausgearbeitet, bis dahin sind die primären Anforderungen, die in diesem Zusammenhang berücksichtigt werden, wie folgt:

- Funktions- und Verhaltensintegrität der Komponenten und ihrer Interaktion untereinander auf Systemebene.
- Sicherer Telebetrieb unter schwierigen Netzbedingungen. (Schlechte Netzkonnektivität, Kommunikationslatenz, Paketverluste usw.)
- Cybersicherheit des teleoperierten Systems und Datenschutz.

Um die Anforderungen für ein geeignetes Testsystem zu entwickeln, kann das DUT – Modell an geeigneten Stellen „freigeschnitten“ werden. Unter Freischneiden versteht man hier, das Betrachten des Systems an seinen Grenzen und schneidet in Gedanken

die Außenwelt ab. Hierdurch ergeben sich die Schnittstellen, welche für das HiL-Testsystem von Relevanz sind.

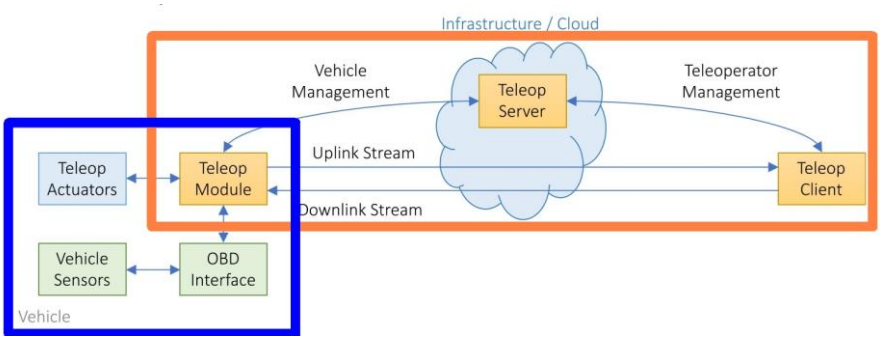


Abbildung 5: Freischneiden des DUT

In Abbildung 5 stellt das orangefarbene Rechteck das Teleop-System dar. Das blaue Rechteck markiert dabei Bauteile, welche im Fahrzeug verbaut werden. Es ergeben sich dadurch die Grenzen des Systems, wie sie in Abbildung 6 dargestellt sind.

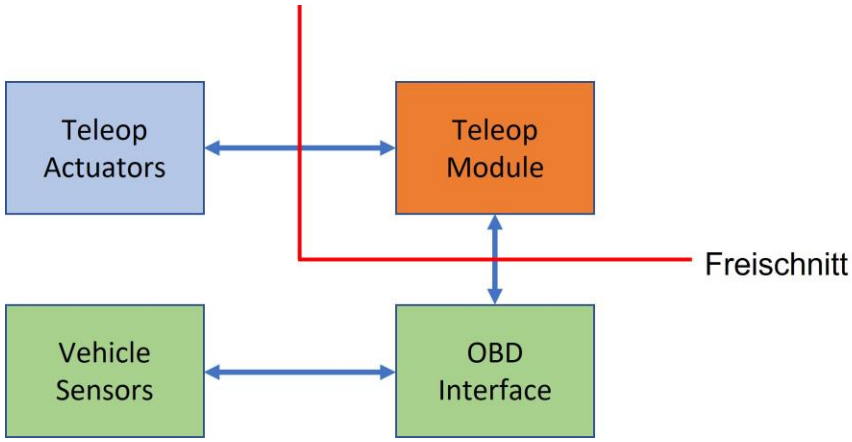


Abbildung 6: Teleop-Module Freischnitt

Dies bedeutet für das HiL-Testsystem auf der einen Seite, dass es das TeleOp-Module mit der Simulation von Aktuatoren, wie auch mit einer Simulation eines OBD-Interfaces bedienen muss. Auf der anderen Seite muss das TeleOp-Module durch eine Netzwerkschnittstelle / Funkschnittstelle bedient werden. Die Spezifikation der Anforderungen soll im folgenden Absatz erläutert werden.

Als DUT bzw. System Under Test (SUT) steht das gesamte TeleOp-System des Projektpartners Luxoft oder Teile davon (Teilsystemtests separiert nach Server, Client, Module) im Zentrum des geplanten HiL-Systems.
Die Basis für das HiL-System bieten die Systeme der Fa. iSyst, welche im Rahmen des Projekts entsprechend um Komponenten zum Testen der IT-Security erweitert werden sollen.
So muss zum einen im Fall von Teilsystemtests die Simulation des Netzwerks in die Umgebungssimulation auf dem Echtzeitrechner integriert werden. Zum anderen werden

Schnittstellen benötigt, um die ausgewählten IT-Security-Testmethoden bzw. Testtools an das HiL-System anzubinden bzw. zu integrieren und die spezifizierten IT-Security-Testfälle mithilfe des Steuerrechners automatisiert ablaufen zu lassen.
Im Hinblick auf den Einsatz von Künstlicher Intelligenz (KI) müssen die System- bzw. Fehlermeldungen des DUT von dem IT-Security-Testtool ausgewertet und zum Anlernen der ausgewählten KI-Methodik bzw. der KI-Algorithmen verwendet werden können. Damit soll es möglich sein, die Testfälle und Testvektoren bzgl. der IT-Security entsprechend zu optimieren.
Zusätzlich ist das HIL-Testsystem in Anbetracht des Gesamtsystemtests (TeleOp-System, Netzwerkkommunikation, Sensoren und Aktoren) in der Lage, die Werte der Sensoren und Aktoren zu simulieren, um die automatisierten Gesamtsystemtests durchzuführen.
Die im SUT verwendeten Protokolle MQTT, UDP (Videostreaming) und ZMQ (Sensor Data Streaming) müssen dazu von der Umgebungssimulation des HiL nachgebildet werden können (siehe Abbildung 7). Außerdem müssen die Testkomponenten in der Lage sein, Pakete dieser Protokolle zu erzeugen und zu manipulieren.

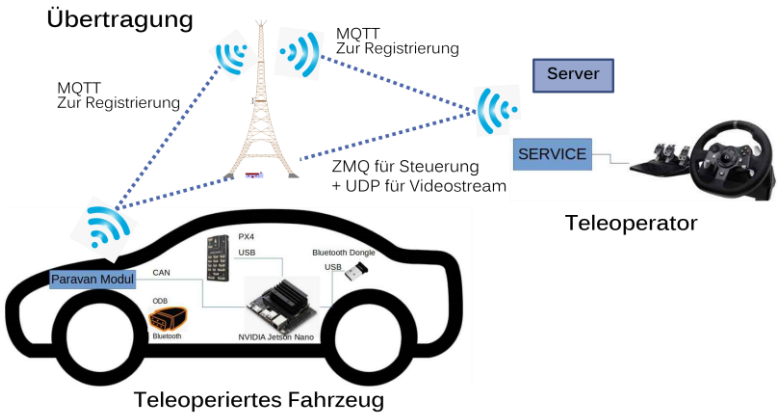


Abbildung 7: Übersicht Protokolle und Schnittstellen

Wenn man von der Abbildung 7 einen Schritt weiter geht, und die Testbereiche für die Funkstrecken mit definiert, ergibt sich folgendes Bild:

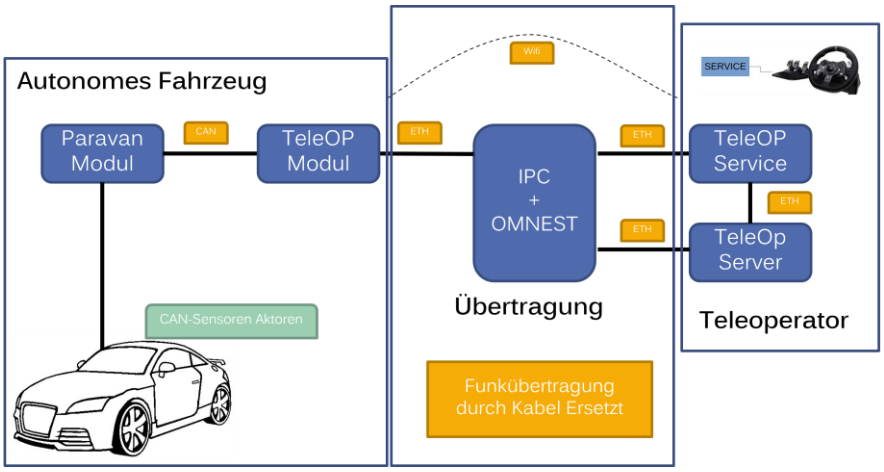


Abbildung 8: Testbereiche Teleop-System

Auf der linken Seite in Abbildung 8 ist das Teleop-Module zu sehen. Als Systemgrenzen steht auf der einen Seite die Telekommunikation mit dem Teleop-Server und auf der anderen Seite das PARAVAN-Drive-by-Wire-Modul [3]. Das PARAVAN-Modul sollte die Ansteuerung eines realen Fahrzeugs übernehmen und per CAN einfach mit dem Teleop-Module verbunden werden können. Bis zum Projektschluss konnte dies jedoch nicht von der Fa. Luxoft umgesetzt werden und der Demonstrator wurde für die Integration mit einem RC-Car weiterentwickelt.
Im mittleren Bereich der Abbildung 8 liegt die Funk- bzw. Übertragungsstrecke. Diese soll mit dem IPC als sogenannter Man-In-The-Middle umgesetzt werden. Die Systemgrenzen hier liegen rein an den Schnittstellen der Protokollverbindungen, (ZMQ / MQTT / UDP).
Die rechte Seite zeigt die Module Teleop-service (Client) und Teleop-server. Die Systemgrenzen dieser Module liegen auf der einen Seite auf der Kommunikation zum Teleop-Module und auf der anderen Seite muss die Schnittstelle des Teleoperators (Mensch) bzw. des Teleop-Clients (automatisierte Ansteuerung) bedient werden.

2.1.2 AP2 – Konzipierung und Spezifikation der HiL-Architektur

Ein HiL-Testsystem wird immer speziell auf den Prüfling zugeschnitten. Dabei ist entscheidend, welche Schnittstellen der Prüfling nach außen hin offenlegt, sowie welche Schnittstellen mit abzutesten sind.
Das folgende Bild gibt einen Eindruck eines im Alltag der iSyst GmbH konzipierten HiLs.

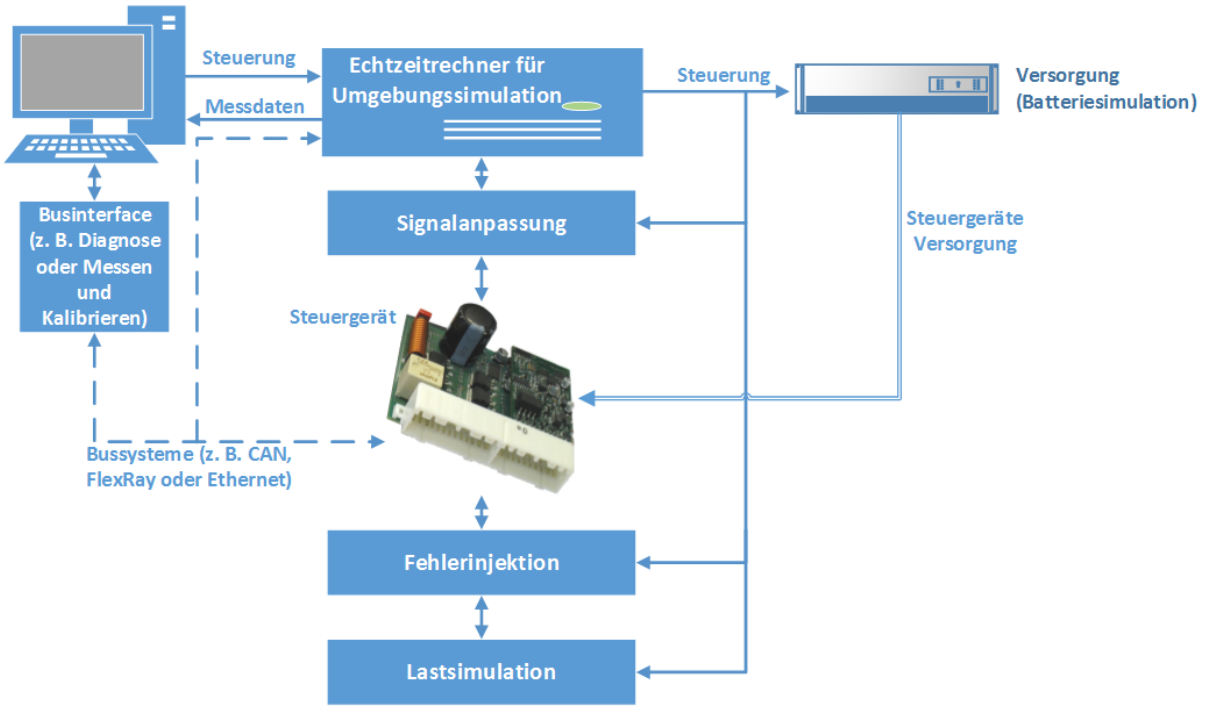


Abbildung 9: Genereller HiL-Testaufbau

Im Zentrum steht das zu testende Objekt (DUT). Ein HiL-Testsystem muss dieses DUT entsprechend in eine Testumgebung betten und es an allen Schnittstellen mit den erwarteten Daten beaufschlagen.
Im SHORT-Projekt besteht das Device under Test (DUT) allerdings aus mehreren Komponenten, welche zusammen getestet werden sollen. Hieraus ergibt sich ein etwas komplexerer Testaufbau, welcher im weiteren Verlauf näher definiert werden soll. Hierzu wurden in AP1 die Schnittstellen von allen Projektpartnern analysiert und definiert.

Auf Basis der Bewertung vorhandener Security-Testverfahren auf die Tauglichkeit für das neue HiL-System durch die THD konnte sich am Ende das Fuzzing als geeignetes Testverfahren für die genannten Schnittstellen durchsetzen.

Was ist Fuzzing oder Fuzz-Testing: (Zitat): „Fuzz-Testing (oder auch "*Fuzzing*") ist eine dynamische Testmethode, die zum Auffinden von Bugs und Sicherheitsproblemen in Software eingesetzt wird. Bei einem Fuzz-Test wird ein Programm mit ungültigen, unerwarteten oder zufälligen Eingaben ausgeführt, mit dem Ziel, die Anwendung zum Absturz zu bringen.“ [4]
Eine detaillierterer Erläuterung wird im Abschlussbericht der TH-Deggendorf wiedergegeben. Dabei wird das Fuzz-Testing von der TH-Deggendorf auch um einen KI-Anteil erweitert. Im weiteren Verlauf dieses Berichts wird an manchen Stellen nur von dem „Fuzzer“ oder BooFuzz berichtet. Hier ist das Tooling der TH-Deggendorf gemeint.

Die Integration eines Fuzzers in das HiL-Testsystem wurde von der iSyst untersucht, mit dem Ergebnis, dass ein Middleware basiertes Echtzeitsystem die Anforderungen erfüllen sollte.
Durch die offenen Schnittstellen innerhalb des Echtzeit-Linux-Systems und der genutzten Middleware (siehe unten) stellt die Integration nur geringe Aufwände dar. Es muss eine Schnittstellenbeschreibung für den Fuzzer als Anforderungsdokument erstellt werden. Aus diesem Dokument kann dann eine Applikation abgeleitet und implementiert werden.

Vorteile einer Middleware im Gegensatz zu proprietärer Software:

- Erhöhte Flexibilität durch eine Vielzahl von Schnittstellen und fertigen Funktionen
- Leichtere und schnellere Anwendungsentwicklung
- Modularität und mehr wiederverwendbarer Code
- Vereinfachte Wartung und Modernisierung bestehender Installationen
- Unabhängigkeit von Hardware- und Software-Lieferanten
- Längere Nutzungsdauer und Nachhaltigkeit

Aufbau und Verwendung der eingesetzten Middleware:
In Abbildung 10 ist der grundsätzliche Aufbau eines Rechnersystems mit einer Middleware dargestellt. Im Prinzip stellt die Middleware eine offene Verbindung zwischen den Applikationen (hier in Grün dargestellt) und der unteren Schicht, den Treibern, dar.

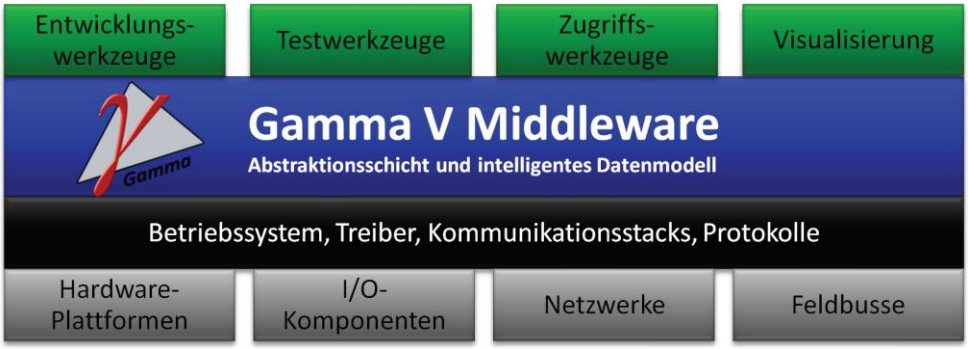


Abbildung 10: Aufbau eines Systems mit der Middleware GammaV

Ein Fuzzingtool, wie von der TH-Deggendorf ausgewählt, würde in den obigen Bereich der Testwerkzeuge fallen. Für den HiL-Aufbau und die Automatisierung heißt das, es muss eine Schnittstelle zwischen der Middleware GammaV und dem Fuzzingtool geschaffen werden. Hier kann durch eine einfache C-Implementierung in ein PlugIn die Verbindung erfolgen.
Ein weiterer Vorteil der Middleware wird klar, wenn man sich die verschiedenen Testumgebungen ansieht. Bei der TH-Deggendorf liegt der Fokus auf den Ethernet-Verbindungen und den Protokollen, die darauf laufen. Bei der iSyst GmbH liegt wiederum der Fokus auf dem Gesamtsystem. Die Middleware läuft auf beiden Systemen gleichermaßen. Einmal auf dem Intel NUC – MiniPC und einmal auf dem Industrie-PC (IPC).

Für den Test wichtig sind die Prozessvariablen, welche zwischen dem Steuerrechner und dem IPC bzw. NUC ausgetauscht werden. In Abbildung 11 ist die Interprozesssteuerung dargestellt. Während eines HiL-Testlaufs, werden auf beiden Rechnerinstanzen je ein GammaV-Service gestartet. Dieser ermöglicht den Austausch der Prozessvariablen. Dabei läuft das Echtzeitsystem mit einem Zyklus von 1ms (harte Echtzeit) und der Steuerrechner mit seiner eigenen von Windows getakteten Zykluszeit.

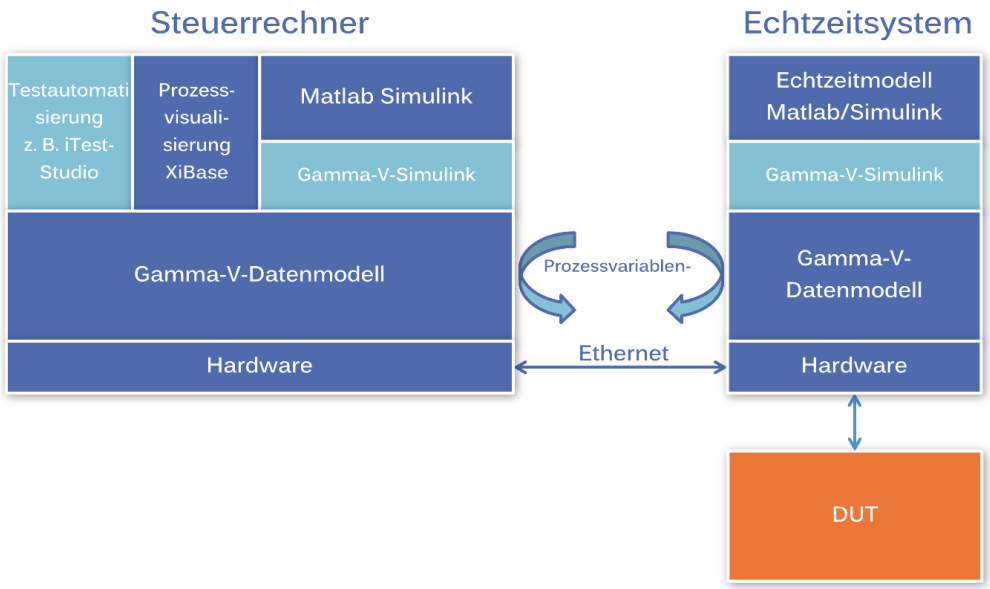


Abbildung 11: Interprozesssteuerung HiL-System

Die Testfälle werden in Pythonskripten umgesetzt und auf dem Steuerrechner durch das Testtool iTestStudio gestartet und verwaltet. Im Falle einer benötigten deterministischen Aufzeichnung von Prozessvariablen kann in einem Pythonskript eine Methode aufgerufen werden, welche einen Rekorder auf dem Echtzeitsystem ansteuert. Die so aufgezeichneten Werte einer oder mehrerer Variablen können dann bequem vom Steuerrechner abgeholt und ausgewertet werden.
Der Einsatz eines Fuzzers im Echtzeitsystem soll ebenfalls durch eine einfache Methode per Pythonskript angesteuert werden können.

Aus diesem Ansatz heraus wurde folgendes, in Abbildung 12 dargestelltes HiL-System definiert:

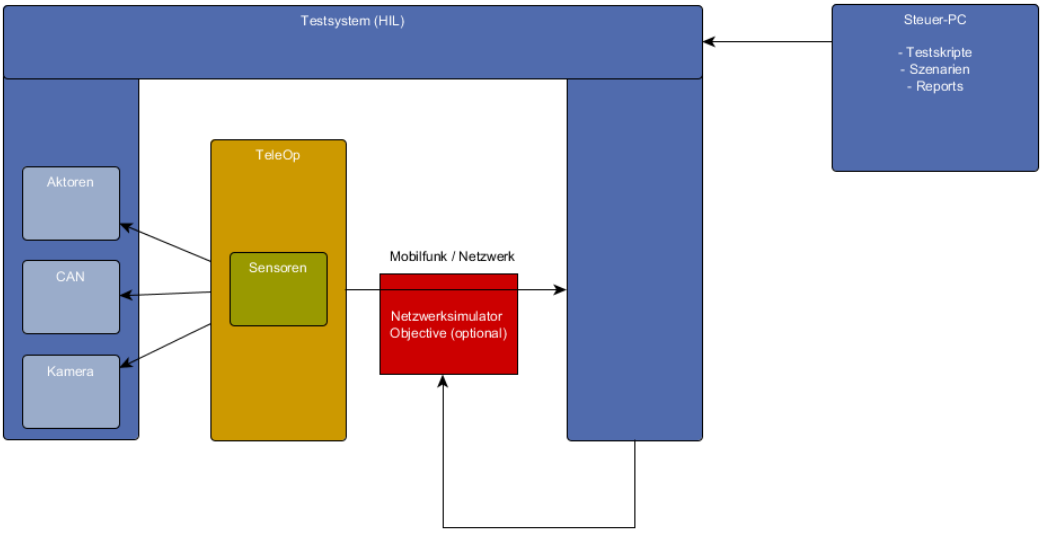


Abbildung 12: Definiertes HiL-Testsystem

Aus den Erfahrungen im funktionalen Testbereich in anderen Projekten der Fa. iSyst, wurde das HiL-Testsystem dann im Detail weiter geplant. In Abbildung 13 soll dies veranschaulicht werden.

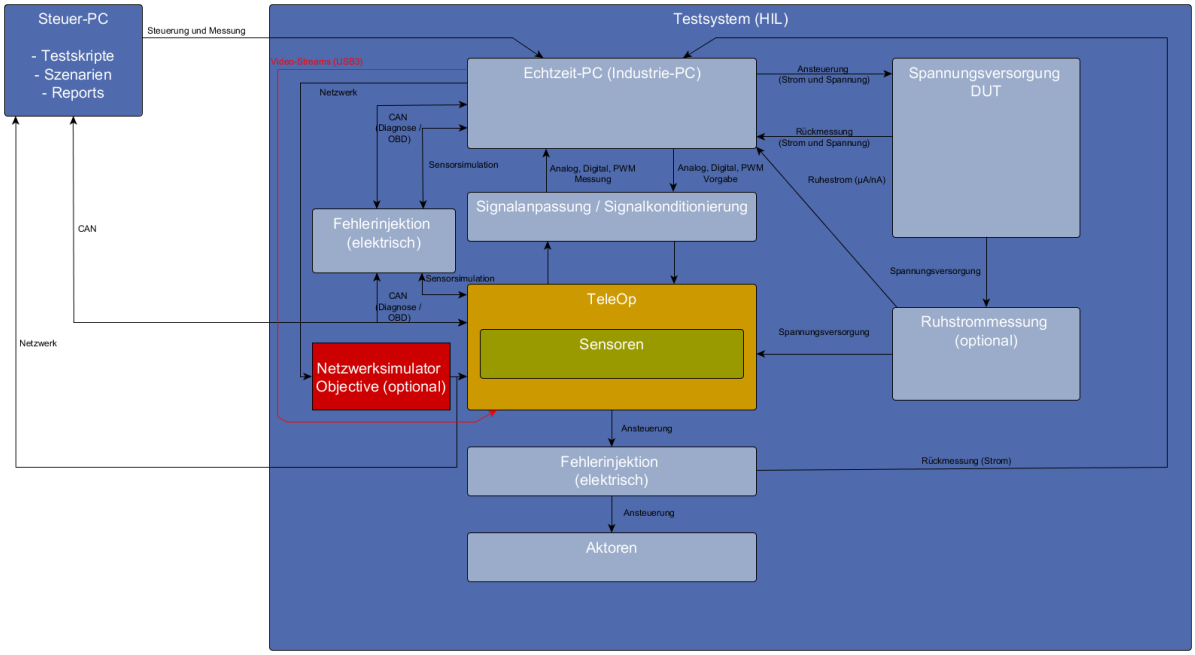


Abbildung 13: Detailplanung - HiL-Testsystem

Zu diesem Zeitpunkt war der tatsächlich erreichbare Testumfang des DUT noch nicht klar. Im weiteren Verlauf wird sich dieses Konzept noch an die Gegebenheiten anpassen können.

In den gezeigten Übersichten ist die Integration der neuen Security-Testverfahren noch nicht mit eingeflossen. Dies geschieht im weiteren Verlauf dieses Berichts. Des Weiteren ist der dargestellte Netzwerksimulator von Objective nicht umgesetzt worden, auch deshalb nicht, weil dieser nur für das Teleop-Module genutzt worden wäre.

Auf Basis der Erhebung der Anforderungen an ein HiL-Testsystem und den erarbeiteten HiL-Konzepten wurden folgende Komponenten beschafft und in Betrieb genommen:

Bezeichnung	Beschreibung	Funktion
Industrie-PC (IPC) (inkl. Zubehör wie Maus, Tastatur, RAM, SSD usw.)	Echtzeitsystem auf Linuxbasis mit Middleware GammaV zur Ansteuerung der HiL-Komponenten	Basis-HiL-Komponente (Herzstück)
TEWS TPMC530-10R	Analog-Einsteckkarte für IPC	Zur Anbindung von Analog I/O s
TPMC632-20R	FPGA-Einsteckkarte für IPC	Variable Funktionen – vorrangig als PWM-Karte eingesetzt
PCI-6509	Digital-Einsteckkarte für IPC	Zur Anbindung von Digital I/O s
CAN-IB200/PCIe	CAN Interface-Einsteckkarten für den IPC	Zur Anbindung von CAN-Bussen für Restbussimulationen
Intel-NUC	Barebone-PC	Als Mini-HiL-System für den Projektpartner THD
Matlab/Simulink mit Simulink Coder und Matlab Coder	Software zur Generierung von Simulationsmodellen	Erstellen von Restbussimulationen mit komplexem Hintergrund (CRC, Alivecounter- Berechnung)
Oszilloskop (MDO3034)		Manuelles Abtasten von Signalen vor Einbindung in das HiL-Testssystem und zum Abgleich der Signale im HiL-Testsystem
Notebook LENOVO A281403	Steuer-PC	Zur Ansteuerung des HiL-Testsystems per Ethernet

Tabelle 1: Grundbausteine des HiL-Testsystems

Mit den in Tabelle 1 genannten Komponenten konnte grundsätzlich schon ein skalierbares System generiert werden. Durch die Middleware GammaV der Fa. RST-Proway sind die Schnittstellen innerhalb des IPCs, insbesondere Digitalkarte, Analogkarte und sonstige Komponenten leicht mit einzubinden. Ein Simulink Modell kann die so angebundenen Schnittstellen entsprechend manipulieren und somit eine geforderte Restbussimulation erzeugen.
Andere Komponenten des HiL-Konzepts wurden zu diesem Zeitpunkt noch nicht weiter beschafft oder angefertigt, nachdem noch keine weiteren Informationen hinsichtlich des realen DUTs zur Verfügung standen.
Die Skalierbarkeit stellt sich insofern dar, dass durch den Einsatz einer Middleware verschiedene Teilaspekte des Test-Systems auch auf kleineren Architekturen wie z.B. mit einem Intel NUC Mini-PC betrachtet bzw. getestet werden können. Beispiel hier ist die reine Netzwerkkommunikation. Sofern es zu einem Gesamttestsystem kommt, kann durch eine Middleware auch ein Verbund-HiL-System generiert werden.

Der Intel-NUC zum Beispiel wurde dem Projektpartner THD zur Verfügung gestellt um die Simulationsumgebung mit der Software OMNeT++ [5] zu etablieren. OMNeT++ ist eine Software zur Simulation von Netwerkschnittstellen und Netzwerkknoten. Die Einbindung in das HiL-Testsystem kann grafisch wie in Abbildung 14 dargestellt werden.

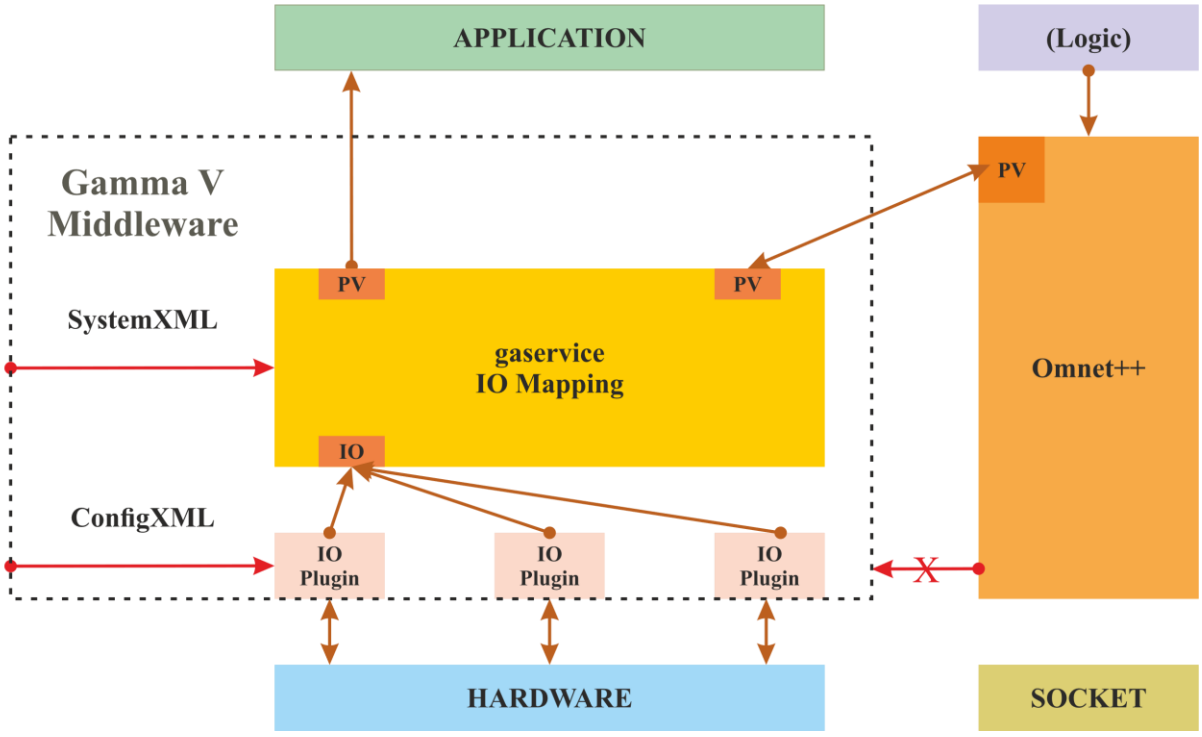


Abbildung 14: OMNeT++ Einbindung durch Middleware GammaV

Abbildung 14 zeigt einen Teil der Testumgebung von iSyst mit der GammaV Middleware. Die NED-Datei spezifiziert die Struktur der OMNeT++ Netzwerktopologie und wird in der Grafik als „(Logic)“ über der Omnet++ Applikation dargestellt.
Das durch OMNeT++ (bzw. OMNEST) simulierte Netz ist mit der Middleware (Gammaservice) über Prozessvariablen (PVs) verbunden. Diese Variablen werden kontinuierlich während der Anwendungslaufzeit synchronisiert (siehe Abbildung 14)
Es gibt zwei verschiedene XML-Files, die verwendet werden, um die Software zu konfigurieren. Die „ConfigXML“ konfiguriert Timing, Timeouts, virtuelles Netzwerk und die IP-Adressen. Die „SystemXML“ konfiguriert das Systemdatenmodell und das Zeitplanungsmodell. Diese Konfiguration wird immer für das lokale System verwendet. Die Anwendung kommuniziert mit der Hardware über die Middleware und ihre Prozessvariablen (PVs). Die E/A-Plugins sind mit der Middleware verbunden. Die gesamte Testumgebung ist bei der TH-Deggendorf auf einem Intel NUC installiert. Und wurde bei der Fa. iSyst auf den Industrie-PC übertragen bzw. auch dort etabliert.

2.1.3 AP3 – Konzipierung von Testfällen und Testvektoren

Um den Projektpartnern einen Einblick in das Feld der funktionalen Tests zu geben wurde ein Workshop abgehalten unter dem Titel, „Testfälle im Automotive Bereich“. [6] Hierbei wurden alle Aspekte berücksichtig, welche für das Projekt SHORT, bzw. für das DUT relevant sind. Weiterhin wurde erläutert, wie aus den gegebenen Testfeldern die Testfälle und Testvektoren ermittelt werden können.
Konkret für das Projekt SHORT wurden die Themen Kommunikation via CAN-Bus sowie Sensorik und Aktorik betrachtet.
Zur Konzipierung von Testfällen und Testvektoren ist die Analyse der Anforderungen an das System ein entscheidender Punkt. Daneben können auch physisch vorhandene Testaufbauten mit DUT an den Schnittstellen beobachtet werden und somit zu einer Ableitung von Testfällen führen.
Die automatisierte Generierung von Testfällen steht dabei an höchster Stelle, da hier viel Zeit im Entwicklungsprozess gespart werden kann. Dies funktioniert jedoch nur, wenn die Schnittstellen z.B. mittels einer Datenbank genau beschrieben sind.
Beispiel hierfür wäre eine DBC-Datei. Diese beinhaltet die Kommunikationsmatrix eines CAN-Busses wie er typischerweise im automotive Bereich zum Einsatz kommt. Aus dieser Matrix lassen sich automatisiert mittels Pythonskripten die Testfälle generieren. Wobei auch hier nicht allein auf die Daten der DBC-Datei zugegriffen wird, Daten über das Testsystem und die Schnittstelle müssen gleichermaßen vorhanden sein.

Konkret für das SHORT Projekt wurden die Requirements, welche von der Fa Luxoft zur Verfügung gestellt wurden untersucht. Weiterhin wurden die Schnittstellen mit in die Analyse eingebracht.
Der Aufbau des bis zu diesem Zeitpunkt existierenden, zu testende, Systems kann folgendermaßen dargestellt werden.

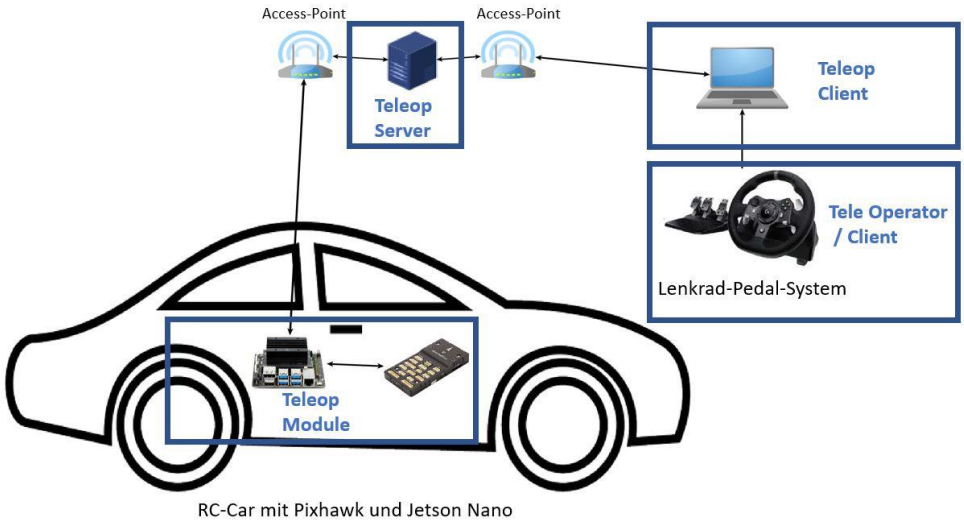


Abbildung 15: Systemaufbau bei Luxoft

Die einzelnen Teile des Teleop-Systems, wie in Abbildung 15 zu sehen, sind auch in der Anfangs gezeigten Konzeptübersicht in Abbildung 3 zu finden. Für die Schnittstellenbetrachtung sind die folgenden Komponenten jetzt genauer definiert:
Der Teleoperator (Person), welcher in der realen Welt das teleoperierte Fahrzeug fahren soll, sitzt an einem Lenkrad-Pedal-System. Dieses ist mit einem PC verbunden und stellt die Komponente *Teleop-Client* dar (Person, Lenkrad-Pedal-System, *Teleop-Client*-Software). Die Software des *Teleop-Servers* läuft in einer Infrastruktur oder auch Cloud.
Die Software des *Teleop-Modules* läuft auf einem Jetson Nano. An dem Jetson Nano sind weitere Komponenten verbaut, wie z.B. das Pixhawk[2] Modul, Kameras für den Videostream und ein WLAN-Modul. Auf einer ARM Cortex M7 Plattform stellt der Pixhawk verschiedenste Schnittstellen zur Verfügung. Im Falle des Teleop-Systems werden so die Sensorik und Aktorik Schnittstellen des RC-Cars bedient.

Die Schnittstellen welche erreichbar sind, werden wie folgt definiert:

1. Verbindung zwischen Teleop-Server und Teleop-Module (Übertragung via Funk)
2. Verbindung zwischen Teleop-Module und Pixhawk (Übertragung via USB)
3. Verbindung zwischen Teleop-Client und Teleop Server (Übertragung via Funk)
4. Verbindung zwischen Lenkrad-Pedal-System und Teleop-Client (Übertragung via USB)
5. Verbindung zwischen Pixhawk und Sensorik / Aktorik (Übertragung via Datenleitung)

Für den Systemtest vor allem im funktionalen Bereich ist es erforderlich, für das System eine adäquate Umgebung zur Verfügung zu stellen und die Schnittstellen entsprechend mit den richtigen Daten (bzw. im Test auch fehlerhafte Daten) zu Versorgen.
Die Schnittstellen für den funktionalen HiL-Test werden wie folgt fest definiert:

1. Verbindung zwischen Teleop-Client und Lenkrad-Pedal-System
2. Verbindung zwischen Pixhawk und Sensorik / Aktorik

Dabei wird das Lenkrad-Pedal-System auf der einen Seite simuliert wie auch die Sensorik / Aktorik auf der anderen Seite.

[2] Der Pixhawk 4: https://pixhawk.org/products/

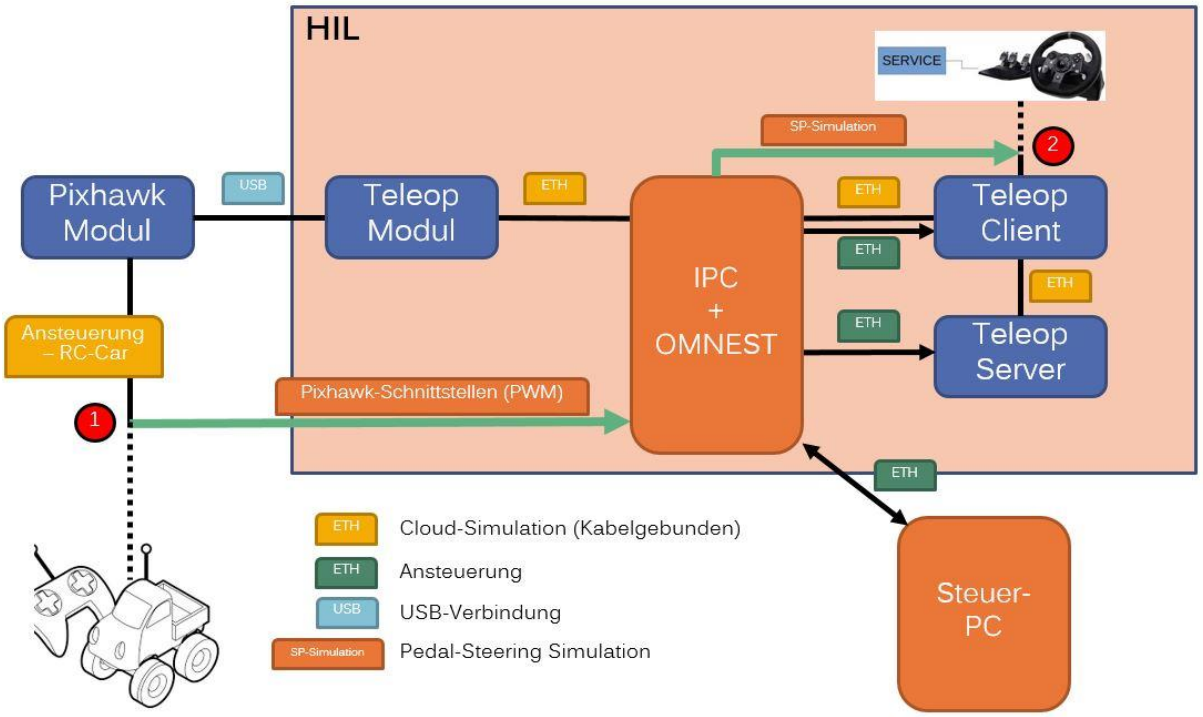


Abbildung 16: Testpunkte für den funktionalen Test

Die Angriffspunkte für den Test sind in Abbildung 16 durch rote, nummerierte Kreise gekennzeichnet.

- Angriffspunkt Nr. 1 würde dem Pixhawk-Modul ein real vorhandenes RC-Auto vorspielen
- Angriffspunkt Nr. 2 würde dem Teleop-Client ein real vorhandenes Pedal-Lenkrad-System für die Fernsteuerung vorspielen

Für den Test werden die Original-Leitungen gekappt und mit Leitungen mit dem HiL-System verbunden (grüne Pfeile).

Die von der Luxoft GmbH bereitgestellten funktionalen Anforderungen und Schnittstellenbeschreibungen wurden unter dem Aspekt der Testbarkeit an einem HiL-Testsystem und dem aktuellen Stand der Implementierung kategorisiert. Auf Basis dieser Einteilung wurde dann begonnen Testfälle zu spezifizieren und zu erstellen. Die nötigen Schnittstellen für die Testbarkeit wurden mit der Luxoft GmbH diskutiert und abgesprochen. Diese sind in der Abbildung 16 beschrieben. Die entstandenen Testfälle wurden so konzipiert, dass sie noch im Rahmen dieses Projekts durchführbar waren.

Zur Absicherung ob Testfälle mit falschen Daten tatsächlich einen Fehler generieren, wurde eine Diagnoseschnittstelle von der Fa. Luxoft bereitgestellt, welche per ZMQ-Protokoll abgefragt werden kann. An dieser Stelle wird an den Bericht der Fa. Luxoft verwiesen.

Die Entwicklung der funktionalen Tests basieren auf den Erfahrungen der Fa. iSyst und sind als trivial zu betrachten. Im Vordergrund stehen die Testfälle zur Safety und Security. Hierzu wurde von der TH-Deggendorf die Bedrohungslage für das System Under Test eruiert. Das HEAVENS- Modell (HEAling Vulnerabilities to ENhance Software Security and Safety) bietet hierzu den Ansatz der “Threat Analysis and Risk Assessment” (TARA) Methode.
Im Standard SAE J3061 “Cybersecurity Guidebook for Cyber-Physical Vehicle Systems” wird als Best Practice Methode ebenfalls TARA adressiert, wobei hier drei Schritte notwendig sind:

- Bedrohungsanalyse (Threat Analysis)
- Risikoabschätzung (Risk Assessment)
- Sicherheitsanforderungen (Security Requirements)

Für die genaue Vorgehensweise wird hier auf den Bericht der TH-Deggendorf verwiesen. Die Software von Microsoft, das „Threat Modeling Tool“, wurde dazu genutzt, alle Bedrohungsszenarien zu modellieren und im Anschluss daran, aufzuzählen. Die TH-Deggendorf diskutierte mehrfach mit den Projektpartnern die Ergebnisse und erweiterte das Modell entsprechend.

Für die Konzipierung von Testfällen konnten auch die Erfahrungen der Fa. iSyst hinsichtlich der Erstellung von Testspezifikationen mit einfließen. Eine interne Checkliste für das Anlegen von Testspezifikationen angelehnt an die Norm ISO/IEC/IEEE 29119-3:2021 [7] soll den Entwicklungsprozess auch für das Aufsetzen der Requirements weiter erleichtern.

Folgende Einträge sind in einer Testspezifikation niederzuschreiben (laut Standard [7]):

- Übersicht über das zu testende Objekt
- Eindeutige Bezeichnung des Testfalls (meist abgekürzt)
- Beschreibung des Testfalls
- Wichtigkeit
- Traceability (Wie ist der Testfall mit dem Requirement verknüpft
- Vorbedingungen
- Eingangsbedingungen
- Erwartetes Ergebnis

Auf Basis des Standards 29119-3 wurde bei der Fa. iSyst eine Checkliste erstellt, welche beim Bearbeiten der Requirements mit einbezogen wurde.

Für die Erstellung von Testskripten gilt in Kundenprojekten der Styleguide des Kunden, sofern dieser spezifiziert ist. Wenn nicht, wurde bei der Fa. iSyst auch ein interner Styleguide verabschiedet, mit Hilfe dessen eine leicht zu lesende Skriptform eingehalten werden soll.

2.1.4 AP4 – Auswahl, Erprobung, Erweiterung von Test Tools/-Methoden

Für die Testumgebung sind verschiedene Tools und Methoden nötig. In diversen Kundenprojekten der Fa. iSyst wird dabei auf eine große Auswahl zurückgegriffen, und dem Kunden angeboten. Je nach Entscheidung des Kunden werden die Tools dann in das entsprechende HiL-Testsystem mit integriert. Die Fa. iSyst nutzt dabei auch eigene Tools, welche speziell für den HiL-Test entwickelt wurden. Hier sei z.B. das iTestStudio genannt. Das iTestStudio ist eine Testsuite, mit der sich in erster Linie Testskripte leicht verwalten lassen. Die Gliederung von Testskripten (Einzel-Tests) zu Testserien basiert auf in Abbildung 17 dargestellter Ordnung:

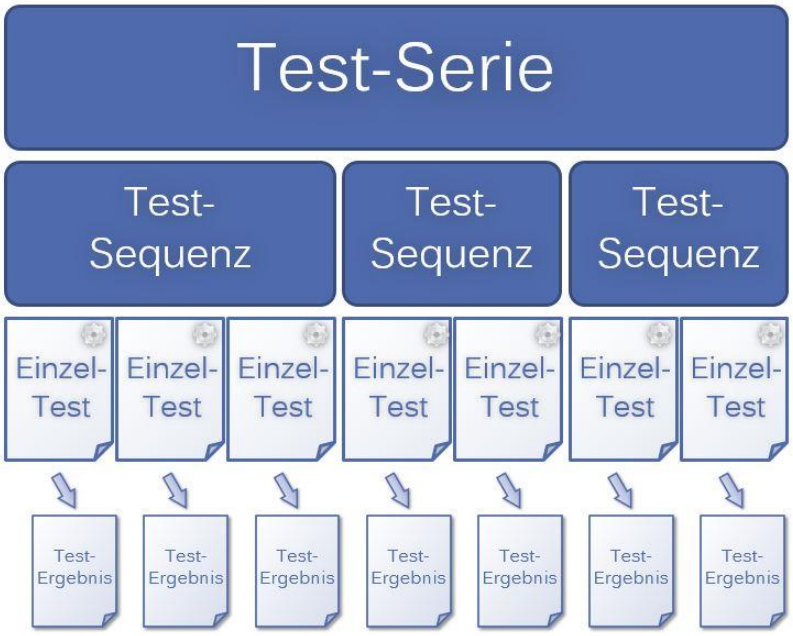


Abbildung 17: Übersicht Teststruktur iTestStudio

Wenn die Grundstruktur festgelegt ist, kann eine Test-Serie gestartet werden durch das iTestStudio und die einzelnen Testskripte werden entsprechend der Struktur ausgeführt. Des Weiteren übernimmt das iTestStudio auch das Zusammenstellen der Testergebnisse und das Weiterleiten an entsprechend weitere Tools in der Softwareentwicklungsumgebung wie z.B. Application-Management-Systeme oder auch Application-Lifecycle-Management-Systeme (ALM).

In Abbildung 18 ist der Workflow mit dem iTestStudio noch einmal dargestellt.

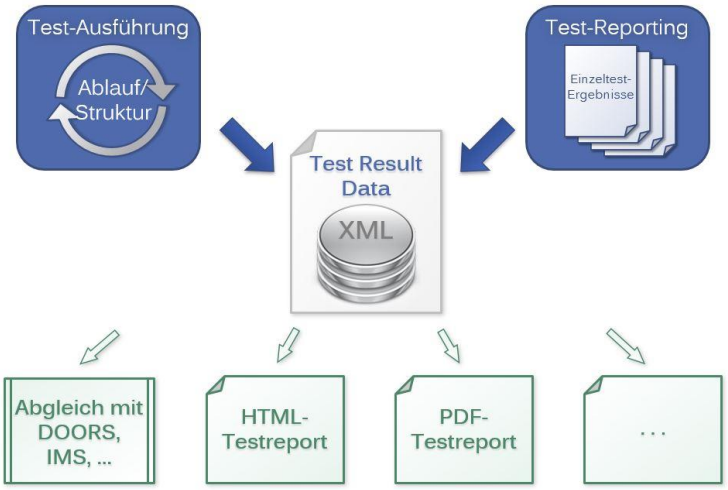


Abbildung 18: iTestStudio Workflow

Neben dem iTestStudio wurde bei der Fa. iSyst auch eine umfangreiche Bibliothek an Testfunktionen erstellt, welche in Pythonskripten aufgerufen werden können. Hier sind auch verschiedene Anbindungen an Testtools anderer Hersteller implementiert. Neue Testtools können meist durch die zur Verfügung gestellte API mit den Pythonbibliotheken verbunden werden. Dazu zählen auch die von der TH-Deggendorf ausgewählten Methoden für das Projekt SHORT.

Die Auswahl der neuen Methoden und Tools wurden von der TH-Deggendorf ausführlich eruiert und mit den Projektpartnern diskutiert. Einer der Methoden ist das Fuzzing mit KI-Anteilen. Als geeignetes Testtool wird der Fuzzer mit der zentralen Komponente des „Generative Adversarial Network“ (GAN) definiert.

Der Fuzzer, „BooFuzz“[3], der TH-Deggendorf soll auf dem Echtzeitrechner (IPC) laufen und per Python-Skript angesteuert werden können. Die Schnittstellen wurden zwischen der THD und der iSyst GmbH abgesprochen und wurden in einem Hands-On-Workshop, welcher Ende Mai 2022 stattfand, integriert. Neben den Testfällen für die Funkstrecke wurden dann auch die Testfälle für die funktionalen Anforderungen erprobt.

Die automatisierte Testfallerzeugung geht einher mit dem Fuzzingtool der TH-Deggendorf. Dies entwickelt selbstständig Testfälle, in dem der Datenverkehr ausgewertet und analysiert wird. Die Verknüpfung bestehender Datenbanken für Schwachstellen und dem Testtool (Fuzzer) und der KI dahinter ergeben eine Reihe von Testfällen, welche im Hands-On-Workshop weiterentwickelt wurden. Die Auswertung der entsprechenden Datenbanken fand durch die THD statt. Eine weitere Automatisierung zur Generierung von Testfällen konnte nicht evaluiert werden.

[3] Siehe Abschlussbericht der TH-Deggendorf

Ein Detail, welches sich bei der Einbringung eines Fuzzers mit KI-Anteil herauskristallisiert hat, ist die nicht Reproduzierbarkeit von Testfällen. Der gemeine HiL-Test soll neben der Automatisierung auch die Reproduzierbarkeit von Testfällen sicherstellen. Das Fuzzing-Tool kann eine Reproduktion allerdings nicht gewährleisten, da der Algorithmus des Fuzzers bzw. des GANs im Betrieb zufällige Daten generiert. Ein Fehlerfall würde zwar erkannt werden, jedoch kann nach einer Behebung des Problems der Testfall nicht direkt reproduziert werden.
Als Lösung wurde hier eine Aufzeichnungsfunktion vorgeschlagen, welche die „gefuzzten“ Pakete aufzeichnet. Bei Bedarf können diese abgespielt, und so der Testfall zumindest annähernd wieder nachgestellt werden.
Eine nähere Evaluierung dieses Problems konnte aus Zeitgründen nicht mehr durchgeführt werden.

In Absprache mit den Projektpartnern wurden die Schnittstellen für die Integration der entwickelten Testtools (Fuzzer) vor allem softwareseitig angelegt. Mittels Python-Skripte können alle erforderlichen Testtools erreicht werden. Das Auslesen der vom Fuzzer generierten SQ-Lite Datenbank zur Fehleraufzeichnung kann ebenfalls durch einfach Python-Skripte und FTP-Verbindungen umgesetzt werden. Die Testprotokolle können hierdurch zwar erst nach dem Test generiert werden, jedoch ist hier der zeitkritische Faktor beim Fuzzing selbst abgedeckt durch die interne Erstellung eben dieser SQ-Lite Datenbank. Die Fehleraufzeichnung funktioniert mittels der vom BooFuzz mitgebrachten Monitorkomponente, sowie der Diagnosefunktion des Teleop-Systems und Open Source Tools zum Monitoring wie z.B. Valgrind.
Valgrind ist ein Instrumentierungs-Framework zur Erstellung dynamischer Analysewerkzeuge. Es gibt Valgrind-Tools, die automatisch viele Speicherverwaltungs- und Threading-Fehler erkennen und ein detailliertes Profil von Programme erstellen können. Valgrind kann auch verwendet werden, um neue Tools zu erstellen [8].

Abbruchkriterien sollen durch eine weitere Komponente zur automatisierten Risikoabschätzung untersucht werden.

Für die genaue Umsetzung der Fuzzing-Komponente sei auf den Bericht der TH-Deggendorf verwiesen.

Testfälle zur Videomanipulation wurden nicht generiert, da der Fokus des Projekts SHORT auf die Kommunikation lag. Jedoch wurde von der TH-Deggendorf eine Abschlussarbeit umgesetzt, um den Videostream zu beeinflussen. Der Studierende Jakob Reitberger verfasste eine Masterarbeit mit dem Titel „Untersuchung von Echtzeit Video-Manipulation bei teleoperiertem Fahren“.

2.1.5 AP5 – Implementierung und erste HiL-System Erprobung

In einem Hands-On-Workshop in den Räumen der iSyst GmbH wurde die Zusammenführung der einzelnen Komponenten, Teleop-System, Fuzzer und HiL-Testsystem, verwirklicht.

2.1.5.1 Prototypischer HiL-Testaufbau

Der prototypische HiL-Testaufbau wurde umgesetzt wie in AP2 und AP3 erarbeitet und soll in Abbildung 19 noch mal genauer verdeutlicht werden.

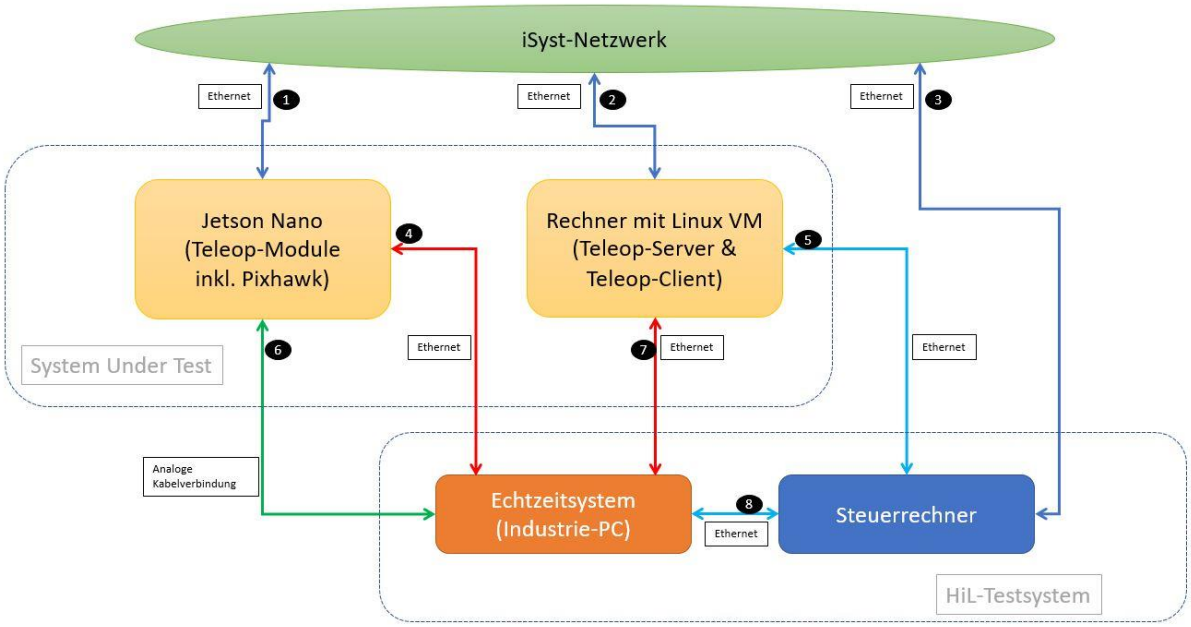


Abbildung 19: Finaler HiL-Aufbau – Netzwerktopologie und Verbindungen

Die Verbindungen des prototypischen HiL-Aufbaus sollen in Tabelle 2 kurz erläutert und im weiteren Verlauf genauer beschrieben werden.

Nr.	Beschreibung
1	Netzwerkverbindung zwischen Jetson Nano und dem iSyst Netzwerk. Wird vom Teleop-Module benötigt, um Updates zu erhalten.
2	Netzwerkverbindung zwischen LinuxVM und iSyst Netzwerk. Wird benötigt, um Updates an das Teleop-Module zu senden sowie um per Fernzugriff auf die LinuxVM zugreifen zu können.
3	Netzwerkverbindung zwischen Steuerrechner und iSyst Netzwerk. Wird benötigt, um per Fernzugriff auf die Testumgebung zugreifen zu können.
4	Netzwerkverbindung – MQTT- und ZMQ-Protokoll, zwischen Teleop-Module und Echtzeitsystem. Wird benötigt, um die Daten zwischen Teleop-Module und Teleop-Server weiter zu reichen. Dabei werden die Daten durch den Industrie-PC gereicht.
5	Netzwerkverbindung zwischen Steuerrechner und LinuxVM. Wird benötigt, um Steuersignale wie z.B. das Pedal-Lenkrad-System in das Teleop-Client System zu senden
6	Analoge Kabelverbindung – wird benötigt, um die Steuersignale, welche vom Module aus an ein Fahrzeug gesendet werden, abzugreifen.
7	Netzwerkverbindung, MQTT- und ZMQ-Protokoll, zwischen LinuxVM und Echtzeitsystem. Wird benötigt, um die Daten des Teleop-Servers an das Teleop-Module weiter zu reichen.
8	Netzwerkverbindung zwischen Steuerrechner und Echtzeitsystem. Wird benötigt für das Anstoßen der Tests sowie für das Übertragung von Testdaten.

Tabelle 2: Verbindungsdefinition HiL-Aufbau

Nachdem bis zum Abschluss des Projekts SHORT, das Teleop-Module mit dem Pixhawk das Ende der Wirkkette darstellt, wurden außer den Leitungen für die Erfassung der Lenkrad-Pedal-Daten keine weiteren Erfassungen für den funktionalen Test benötigt.
Der HiL-Aufbau könnte jedoch zusätzlich auch Daten eines real angebundenen Fahrzeugs erfassen oder simulieren, wie:

- CAN-Kommunikation
- Sensorik
- Aktorik
- Bildaufzeichnungsdaten

2.1.5.2 Verbindungsbeschreibung:

Die Verbindungen **1** und **2** werden als Support Kanal benötigt. Das Teleop-Module, genauso wie der Teleop-Client werden von außen (Netzwerk) mit Updates versorgt. Dies geschieht über Docker-Container. Für nähere Informationen hierzu sei auf den Bericht der Fa. Luxoft verwiesen. Des Weiteren kann die Verbindung **2** auch dazu genutzt werden, den Rechner auf dem die LinuxVM läuft, zu bedienen, die VM zu starten, Updates für den Rechner aufzuspielen oder weitere Testmethoden zu verwalten.

Die Verbindung **3** wird lediglich dazu benötigt, um mit Fernzugriff auf den Steuerrechner zuzugreifen um im Falle von z.B. wechselnden Mitarbeitern bzw. wechselnden Tätigkeiten oder Homeoffice, an dem System weiter arbeiten zu können.

Die Netzwerkverbindungen **4** und **7** stellen die Funkstrecke dar, da es nicht möglich war, eine Funkstrecke in den HiL-Aufbau zu integrieren – mangels Zeit und auch der monetäre Wert stünde nicht im Verhältnis zum Projekt. Nachdem die Verbindungen jedoch durch das Echtzeitsystem geleitet wurden, konnte ein Man-In-The-Middle Szenario simuliert werden. Die Daten der Protokolle ZMQ und MQTT, welche für das Teleop-System wichtig sind, konnten so aufgezeichnet und mit dem Fuzzer auch beeinflusst werden. Durch die Software Wireshark [9] können zu jedem Zeitpunkt auch Daten aufgezeichnet werden, welche über diese Leitungen durchgereicht werden.

Die Verbindung **5** wird benötigt, um die zu simulierenden Steuersignale eines Lenkrad-Pedal-Systems an den Teleop-Client-Dienst zu senden. Aus einem Testskript heraus können so direkt Lenkrad Winkel und Brems- oder Beschleunigungswerte angegeben werden. Des Weiteren können Funktionen der VM hier automatisiert bedient werden.

Mit der Verbindung **6**, in Abbildung 19 in grün dargestellt, können die Analogen Signale des Pixhawk erfasst werden, welche den Lenkwinkel oder die Beschleunigungsdaten repräsentieren. Dabei handelt es sich um PWM-Signale, welche im realen Aufbau an einen Servo-Motor-Controller gehen. Die Messung dieser Signale, bei gleichzeitiger Stimulierung im Teleop-Client stellt den funktionalen Test dar.

Abschließend wird mit der Verbindung Nummer **8** der Steuerrechner mit dem Echtzeitsystem verbunden. Über diese Verbindung funktioniert das Anstoßen der Tests und die Datenerfassung während und nach dem Testablauf. Hierzu wird auch eine Verbindung mit dem Service GammaV etabliert, welcher die Prozessvariablen der Middleware übertragen kann.

2.1.5.3 Komponentenbeschreibung:

Steuerrechner: Der Steuerrechner basiert auf einem Laptop mit einem Windows 10 Betriebssystem mit 16GB Arbeitsspeicher und einer normalen SSD-Festplatte. Auf ihm laufen verschiedene Softwaretools zum Bearbeiten und Starten von HiL-Tests. Folgende Software wurde hier installiert:

- iTest Studio [10] – zum Starten von Testserien und koordinieren von Testsequenzen und Testskripten
- ttk – TestToolkit [11] – Bibliothek zur einfachen Einbindung von Testfunktionen innerhalb von Python
- GammaV – Service zur Verbindung mit dem Echtzeitsystem zur Abfrage und Setzen von Prozessvariablen (GammaV ist ein Produkt der Fa. Proway)
- WINSCP [12] – Einfache Dateiübertragung zwischen Steuerrechner und Echtzeitrechner
- Putty [13] – Einfache Software zur Verbindung von zwei Rechnern

- Eclipse [14] – Python Entwicklungsumgebung zur Implementierung von Testfällen
- Tortoise [15] – SVN, Subversion Management zur Versionierung von Testskripten und mehr
- Mathwork / Simulink [16] – zur Gestaltung von Restbussystemen oder anderen Modellierungen, um die Umgebungsbedingungen abzusichern
- Powermanager – Programm zum Steuern von Steckdosen für die Stromversorgung
- Microsoft Remotedesktop – Programm für den Fernzugriff auf andere Rechner

Echtzeitsystem: Das Echtzeitsystem basiert auf einem Industrie-PC mit einem Linux-Debian Betriebssystem (Debian 8) und 8GB Arbeitsspeicher sowie einer normalen SSD-Festplatte. Das System ist mit einem Echtzeitkernel ausgestattet, welches ein diskretes Ausführen der Software GammaV erlaubt. Folgende Software wurde hier installiert:

- GammaV der Fa. RST Proway – Middleware
- Xrdp – Remote-Desktopzugriff
- Wireshark [9] – Tracetool für Netzwerkverbindungen
- Treiber für die jeweiligen I/O-Karten
- OMneT++ [5]- Software zur Simulation von Ethernet-Netzwerkknoten

Rechner mit Linux VM: Für die Integration des Teleop-Client und des Teleop-Servers wurde ein Linux-system auf Ubuntu Basis benötigt. Dies lief in einer Virtuellen Maschine (VM) auf einem normalen PC mit Windows 10, 16GB Arbeitsspeicher und einer normalen SSD-Festplatte. Die virtuelle Maschine wurde mit der Software VirtualBox[4] erstellt. Aufgrund von Zeitmangel wurde hier ein Rechner der Fa. iSyst zur Verfügung gestellt, um nicht extra Mittel umwidmen zu müssen und auf die durch die Chip-Mangel-Krise verlängerten Lieferzeiten zu verzichten.

Netzwerkadapter: Außer bei den direkten Verbindungen zum iSyst Netzwerk wurden ausschließlich USB zu Ethernet Adapter von Edimax[5] verwendet. Diese sind frei konfigurierbar und ermöglichen so eine genaue Definition des Netzwerks bzw. des Netzwerkknotens.

[4]Mehr Informationen zu VirtualBox: https://www.virtualbox.org/

[5] Mehr Informationen zum Ethernet-Adapter: https://www.edimax.com/edimax/merchandise/merchandise_detail/data/edimax/de/network_adapters_usb_adapters/eu-4208/

2.1.6 AP6 – Standardisiertes, automatisiertes Testen von Komponenten

Seit Anfang der Corona-Beschränkungen wurden im zweiwöchentlichen Abstand Teams-Meetings durchgeführt, um das Projekt trotz der Restriktionen und Umstände weiter voranzubringen. Im Verlauf hat sich jedoch gezeigt, dass eine direkte Zusammenarbeit innerhalb von Workshops die bessere Alternative ist. Aus diesem Grund wurden nach der Lockerung der Beschränkungen einige Gelegenheiten genutzt, wichtige Meilensteine und Punkte in einem Workshop zu diskutieren und zu bearbeiten.
Für das Aufbauen des Demonstrators fanden hierzu zwei mehrtägige Workshops bei der Fa. iSyst statt.
Der letzte Meilenstein war das Zusammensetzen der erarbeiteten Komponenten für einen Demonstrator bzw. eines prototypisches HiL-Testsystem.

Bei der Fa. iSyst galt es, den Industrie-PC sowie alle notwendigen sonstigen PCs vorzubereiten und die Netzwerkschnittstellen sowohl freizugeben, (internes iSyst-Netzwerk) als auch für andere Geräte wie das Jetson-Nano Modul Anbindungen vorzubereiten.
Benötigte Software wurde schon im Vorfeld des Workshops zur Integration aller Komponenten installiert und etabliert, um die Zeit während des Workshops nicht zu verringern.
Die Implementierung der Testfälle fand ebenfalls bereits im Vorfeld statt. Durch eine Auslagerung von spezifischen Daten wie z.B. IP-Adressen, Zugriffe und sonstige Daten der Infrastruktur, konnten die Umgebungsbedingungen an den prototypischen HiL-Aufbau schnell angepasst werden.
Für die Anbindung des Lenkrad-Pedal-Systems wurde von der Fa. Luxoft eine Schnittstelle im Teleop-Client geschaffen, welche die Daten entweder per USB angeschlossenes Lenkrad-Pedalsystem oder via Pythonskript und MQTT-Protokoll zuließ.

Durch die Unterstützung der TH-Deggendorf wurde das Fuzzer Tool in den HiL-Aufbau integriert und in Betrieb genommen.
Die Kontrollinstanzen in den jeweiligen Systemen – Teleop-Module, Teleop-Server und Teleop-Service wurden mit einem sogenannten Prozessmonitor auf dem Betriebssystem beaufschlagt. Dieser sollte eine Aufzeichnung von Programmabstürzen aufzeichnen, um eine negative Reaktion auf das Fuzz-Testen als Rückmeldung zu bekommen.

Wenn man sich nun das Bild aus dem Arbeitspaket AP5 nochmals näher ansieht, und die Komponenten mit hinzufügt ergibt sich folgendes Bild:

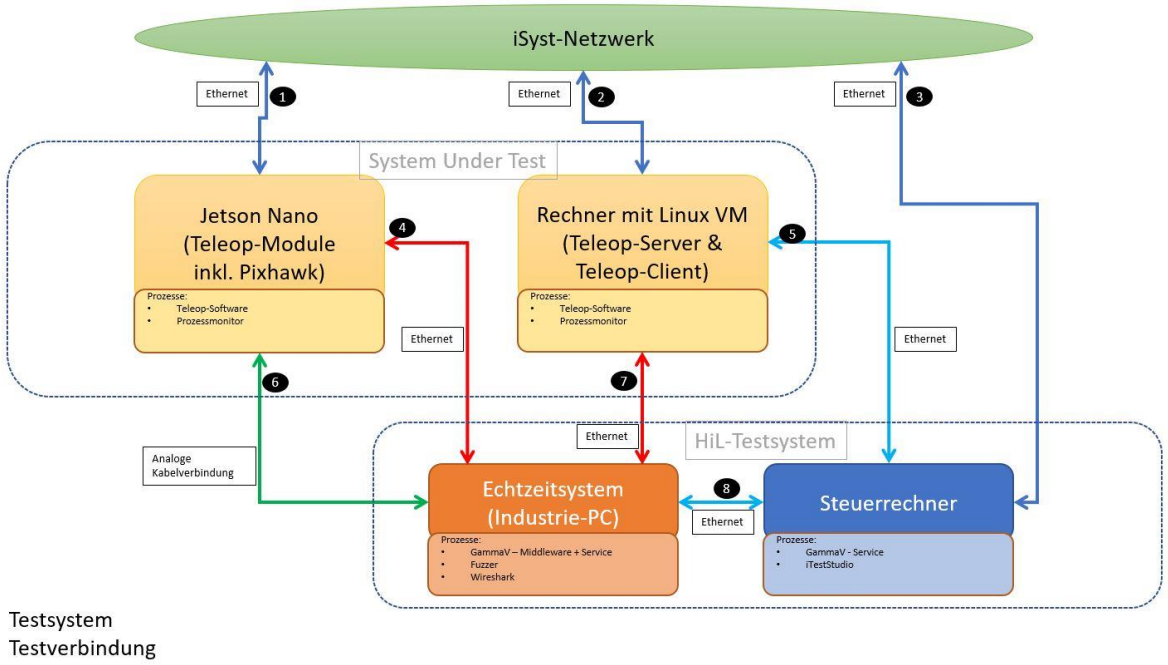


Abbildung 20: Testaufbau mit integrierten Test-Tools

In der Abbildung 20 wurden die Prozesse auf den einzelnen Rechnerinstanzen noch mit-eingefügt. Wie zuvor bereits erwähnt, wurde der Prozessmonitor zur Fehlererkennung mit eingebracht. Auf dem Echtzeitsystem wurde der Fuzzer als Prozess mit implementiert und kann vom Steuerrechner aus über Pythontestskripte angesteuert werden.
Der komplexe Vorgang mit resourcenaufwendigen Kompilier- und Trainingsmethoden zum Anlernen des KI-Modells konnte noch nicht automatisiert eingebaut werden, soll jedoch in den nächsten Schritten erfolgen.

2.2 Verwertbarkeit der Ergebnisse

Die Ergebnisse aus diesem Förderprojekt helfen der Fa. iSyst vorrangig, die schnelle Entwicklung im Bereich IT-Security sowie Cybersecurity zu verstehen und entsprechend die Produktwelt und. Beratungsleistungen zu erweitern.
Die Verwertungsabsichten von iSyst richten sich auf die Ebenen Produktentwicklung, Standardisierung und Dienstleistung. Dies schließt Veröffentlichungen und Einnahmen aus Lizenzen und Produktverkäufe (Testsysteme für den Entwicklerarbeitsplatz, z.B. Mini HiL oder Full Size HiL), Dienstleistung in Verbindung mit den entwickelten Technologien, ein. Die entwickelte HiL-Lösung wird auch über das Projekt hinaus betrieben und als Referenz für weitere Entwicklung zur Verfügung stehen. Dadurch werden neue Aufträge bei bestehenden Kunden generiert und zusätzliche Neukunden gewonnen. Durch die Vermarktung des erarbeiteten Testsystems und des erworbenen Wissens, kann die iSyst verschiedene Geschäftsfelder erweitern und ausbauen.
Es sollen weitere Abschlussarbeiten auf Basis dieses Förderprojekts ausgeschrieben werden, um die Integration des Fuzzings in den HiL-System Testbereich zu etablieren.

2.3 Fortschritt auf dem Gebiet bei anderen Stellen

Speziell für dieses Thema konnten keinen neuen Veröffentlichungen gefunden werden.

2.4 Erfolgte oder geplante Veröffentlichungen

Im Zuge des Projekts SHORT wurde eine Bachelorarbeit generiert. „Evaluierung der Einsatzmöglichkeiten künstlicher Intelligenz im Bereich des HIL-Tests“ welche von Kilian Forster im Zeitraum von Mai 2022 bis September 2022 bearbeitet wurde. Die Abschlussarbeit ist an der Technischen Hochschule Nürnberg – Fakultät Informatik einsehbar.

3 Anhang

3.1 Abbildungsverzeichnis

4 Literaturverzeichnis

[1] A. S. /. T. Linz, Basiswissen Softwaretest, 6., überarbeitete Auflage, Juni 2019 Hrsg., Berlin: dpunkt.verlag Heidelberg, Berlin, 2019.

[2] ISTQB. [Online]. Available: https://www.istqb.org/. [Zugriff am 03 2023].

[3] PARAVAN, „PARAVAN - Drive by Wire," [Online]. Available: https://www.paravan.de/technologie/paravan-drive-by-wire. [Zugriff am 03 2023].

[4] D. K. Yakdan, „www.informatik-aktuell.de," [Online]. Available: https://www.informatik-aktuell.de/betrieb/sicherheit/fuzzing-als-security-testing-methode.html#c36840. [Zugriff am 03 2023].

[5] OpenSim Ltd., „OMNET++," [Online]. Available: https://omnetpp.org/. [Zugriff am 03 2023].

[6] M. Jellen, „Testfälle im Automotive Bereich," 2020.

[7] IEEE Institute, *ISO/IEC/IEEE 29119-3 - Standard for Software testing - Part 3,* 2021.

[8] Valgrind™ Developers, „Valgrind," [Online]. Available: https://valgrind.org/. [Zugriff am 03 2023].

[9] Wireshark Foundation, „Wireshark," [Online]. Available: https://www.wireshark.org/. [Zugriff am 03 2023].

[10] iSyst GmbH, „iSyst," [Online]. Available: https://www.isyst.de/produkte/testsuite/iteststudio-ta/. [Zugriff am 03 2023].

[11] iSyst GmbH, „iSyst," [Online]. Available: https://www.isyst.de/produkte/testsuite/testtoolkit/. [Zugriff am 03 2023].

[12] WINSCP, [Online]. Available: https://winscp.net/eng/download.php. [Zugriff am 03 2023].

[13] putty foundation, „putty.org," [Online]. Available: https://www.putty.org/. [Zugriff am 03 2023].

[14] Eclipse Foundation, „Eclipse," [Online]. Available: https://www.eclipse.org/. [Zugriff am 03 2023].

[15] The TortoiseSVN team., „Tortoise SVN," [Online]. Available: https://tortoisesvn.net/index.de.html. [Zugriff am 03 2023].

[16] Mathworks, „simulink," [Online]. Available: https://de.mathworks.com/products/simulink.html. [Zugriff am 03 2023].

[17] D. K. Trenkel, *Gesamtvorhabensbeschreibung SHORT,* 2019.

Security
Hardware-in-the-Loop
Observing-AI
Risk-Awareness
Testing

Kurzbericht SHORT

Förderkennzeichen: 16KIS0965K

Ursprüngliche Aufgabenstellung

Teilvorhaben: Innovatives HiL-Testsystems zur Gewährleistung der Security von Automotive-Technologien und Einbettung in einen sicheren Entwicklungsprozess

Die wesentliche Hauptaufgabe des Teilprojekts der Firma iSyst war es, zum übergeordneten Gesamtprojekt beizutragen, sowie die Projektkoordination zu übernehmen.

Die Arbeiten innerhalb dieses Teilvorhabens befassten sich mit folgenden Punkten:

- Erarbeitung eines Konzeptes für die Verbindung von Security-Tests mit dem funktionalen Test von eingebetteten Systemen unter Berücksichtigung der Möglichkeiten von HIL-Testsystemen
- Entwicklung automatisierbarer Testmethoden für den Security-Test eingebetteter Systeme
- Entwicklung von Methoden der automatisierten Bewertung des Verhaltens des Device under Tests (DUT) bei der Ausführung von Security-Tests (automatisierte Bestimmung von Passed/Failed)
- Entwicklung eines umfassend Entwicklungs- und Testprozesses bei dem die Anforderungen an die Security und die Safety (funktionale Sicherheit) gleichermaßen berücksichtigt sind und die Testbarkeit aller Aspekte gewährleistet wird
- Implementierung von Testmethoden und Testtools für den automatisierten Security-Test von eingebetteten Systemen unter Berücksichtigung von Risikobewertungen
- Implementierung automatisierter Methoden zur Generierung von Testfällen und Testvektoren (Testdatenraum für einen Testparameter)
- Integration von bestehenden Datenbanken für Schwachstellen (z.B. Common Vulnerabilities and Exposures (CVE – deutsch: Bekannte Schwachstellen und Anfälligkeiten) und KI-Systeme für die automatisierte Erzeugung von Testfällen und Testvektoren
- Implementierung und Erprobung der entwickelten Tools und Methoden an einen praktischen Anwendungsfall in Form eines Demonstrators
-

Technischer Stand

Der funktionale Test im automotive Bereich beschäftigt sich vorwiegend mit der Kommunikation, Sensorik und Aktorik. Diese Bereiche können durch ein HiL-Testsystem auch mit harten Echtzeitanforderungen abgetestet werden. An diesem technischen Stand sollte angeknüpft werden, um Security-Tests zur Validierung kryptografischer Verfahren im Inter- und Intra-Fahrzeug-Kontext zu etablieren.

Ablauf Vorhaben

SHORT gliedert sich in sechs Arbeitspakete (AP 1 bis AP 6) und umfasst, mit der gewährten halbjährigen Verlängerung, eine Laufzeit von 42 Monaten (01.04.2019 bis 30.09.2022). Der Zusammenhang der Arbeitspakete ist schematisch in Abbildung 1 dargestellt.

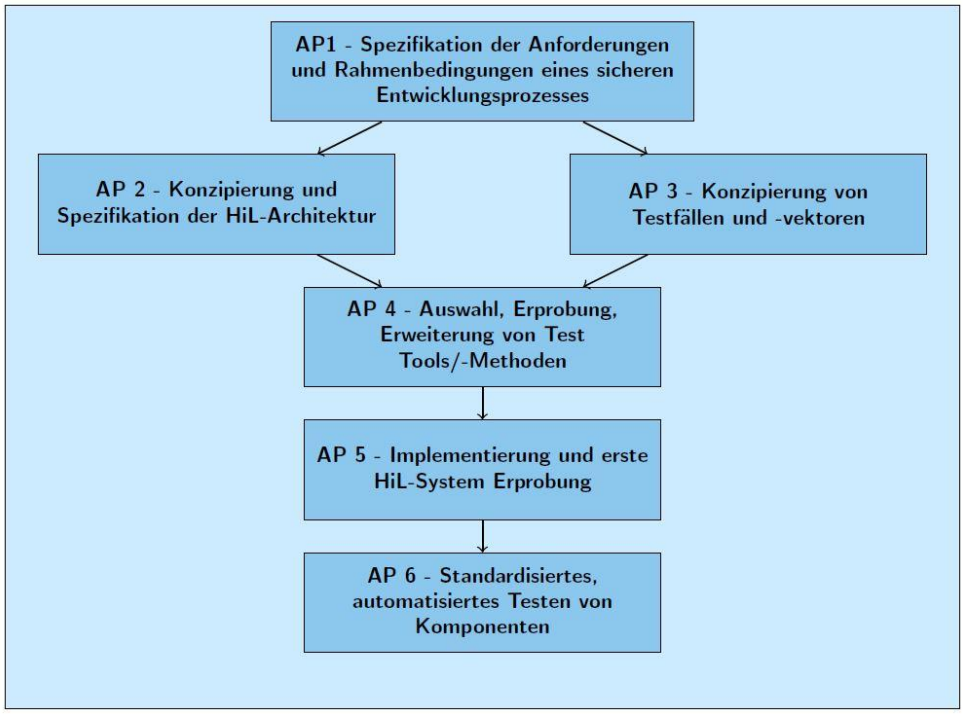


Abbildung 1: Übersicht der Arbeitspakete

Wesentliche Ergebnisse

Eines der Hauptergebnisse war die Integration eines Fuzzers mit KI-Anteilen, in ein HiL-Testsystem. Dabei ergeben sich für das HiL-Testen neue Möglichkeiten aber auch Herausforderungen, welche noch zu untersuchen sind.

Die Art und Weise des Testens mit einem KI-gestützten Tool unterscheidet sich von konservativen Methoden schon bei der Spezifikation von Testfällen bis hin zur Auswertung der Test-Ergebnisse.

Die folgenden Punkte können als Ergebnis aus dem SHORT-Projekt für die Fa. iSyst belegt werden:

- Bedrohungsanalyse eines zu testenden Objekts mit Hilfe von Softwaretools wie z.B. Microsofts® Threat Modeling Tool in den Softwareentwicklungsprozess mit einzubeziehen
- Analyse und Einbettung der IoT Protokolle ZMQ und MQTT in eine HiL-Testarchitektur
- Evaluierung neuer Schnittstellen und neuer Softwaretools wie dem „BooFuzz“-Fuzzer der TH-Deggendorf
- Integration der neuen Testtools – speziell die Integration des Fuzzers mit der cW-GAN-GP[1] Architektur
- Erlernen des Umgangs und der Erfahrungssammlung mit KI-unterstützten Werkzeugen

[1] Siehe Abschlussbericht der TH-Deggendorf