Band 403

Verlagsschriftenreihe des Heinz Nixdorf Instituts
Prof. Dr.-Ing. Roman Dumitrescu (Hrsg.)
Advanced Systems Engineering

Michael Hillebrand

Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Michael Hillebrand

Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme

Design Framework for the integration of self-healing properties into Intelligent Technical Systems

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Band 403 der Verlagsschriftenreihe des Heinz Nixdorf Instituts

ISSN (Online): 2365-4422
ISBN: 978-3-947647-22-4

Satz und Gestaltung: Michael Hillebrand

Geleitwort

Advanced Systems Engineering ist das Leitbild des Heinz Nixdorf Instituts und des damit verbundenen Fraunhofer-Instituts für Entwurfstechnik Mechatronik (Fraunhofer IEM).

Mit der Digitalisierung zeichnet sich ein zunehmender Wandel von mechatronischen Lösungen zu Intelligenten Technischen Systemen ab. Diese zukünftigen Systeme sind von einem hohen Grad an Autonomie, dynamischer Vernetzung und soziotechnischer Integration geprägt. Dabei müssen die Systeme im realen Einsatz in ihrer vorgesehenen Anwendungsdomäne verlässlich operieren und auch bei Funktionsstörungen ihre Aufgabe erfüllen. Die Verlässlichkeit und die Resilienz gehören daher zu den definierenden Merkmalen autonomer Systeme, ohne die ein praktischer Einsatz dieser Systeme nur eingeschränkt möglich ist. Erreicht wird dies durch die Integration von Self-X-Eigenschaften wie etwa Selbstoptimierung oder Selbstheilung. Dabei ist zu berücksichtigen, dass derartige komplexe Systeme durch das synergetische Zusammenwirken von vielen Fachdisziplinen entstehen. Die zunehmende Einbindung und Vernetzung dieser Fachgebiete sowie die damit einhergehende steigende Komplexität erfordert einen ganzheitliche Entwicklungsansatz.

Vor diesem Hintergrund hat Herr Hillebrand eine Systematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme erarbeitet. Die Systematik unterstützt ein interdisziplinäres Team in der frühen Phase der Produktentwicklung bei der kontinuierlichen Integration von Selbstheilungseigenschaften in ein Intelligentes Technisches System. Seine Systematik kombiniert hierzu ein Technologiekonzept zur Umsetzung von Selbstheilung, das Data-Driven Systems Engineering zur Entwicklung selbstheilender Systeme und den Einsatz von fortgeschrittenen Analysetechniken auf der Basis von Virtuellen Testbeds zur Eigenschaftsabsicherung der Selbstheilung. Dabei hat Herr Hillebrand Methoden, Hilfsmittel und Werkzeuge erarbeitet und in einem Vorgehensmodell zusammengeführt. Im Rahmen eines industriellen Anwendungsbeispiels erfolgte die Validierung.

Die Dissertation von Herrn Hillebrand adressiert ein hochspannendes und komplexes Themenfeld, das noch viele Jahre erforscht werden muss. Hierfür hat er einen sehr wertvollen Pionierbeitrag geleistet, um die Steigerung der Resilienz von Intelligenten Technischen Systemen zu ermöglichen. Die Ergebnisse seiner Arbeit sind ein herausragender wissenschaftlicher Meilenstein für unser Leitbild des Advanced Systems Engineerings. Ich wünsche Herrn Hillebrand weiterhin so viel Erfolg bei der Erforschung des Engineerings für Intelligente Technische Systeme.

Paderborn, im November 2021 Prof. Dr.-Ing. Roman Dumitrescu

Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme

zur Erlangung des akademischen Grades
DOKTOR DER INGENIEURSWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
Michael Hillebrand, M.Sc.
aus Paderborn

Tag des Kolloquiums:	22. April 2021
Referent:	Prof. Dr.-Ing. Roman Dumitrescu
1. Korreferent:	Prof. Dr. rer. nat. Otthein Herzog
2. Korreferent:	Prof. Dr.-Ing. habil. Walter Sextro

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Abteilung Produktentstehung am Fraunhofer-Institut für Entwurfstechnik (Fraunhofer IEM) in Paderborn. Die Arbeit ist das Ergebnis meiner wissenschaftlichen Arbeit im Rahmen von Forschungs- und Industrieprojekten.

Mein herzlicher Dank gilt Herrn Prof. Dr.-Ing. Roman Dumitrescu, der mich stets forderte und förderte. Ich danke ihm für sein großes Vertrauen in meine Arbeit, den Freiraum und die stete Unterstützung während der Promotion.

Herrn Prof. Dr. rer. nat. Otthein Herzog vom Lehrstuhl Intelligent Systems an der Universität Bremen und der Tongji Universität in Shanghai danke ich für die konstruktiven Gespräche und die vielen wertvollen Anregungen nicht nur als Korreferent, sondern auch als Mentor im Forschungsprojekt. Herrn Prof. Dr.-Ing. habil. Walter Sextro vom Lehrstuhl für Dynamik und Mechatronik an der Universität Paderborn danke ich herzlich für die Übernahme des Korreferats. Zudem möchte ich Herrn Prof. Dr.-Ing. Jürgen Gausemeier meinen besonderen Dank für den Vorsitz der Kommission aussprechen.

Allen aktuellen und ehemaligen Kollegen am Institut danke ich für die stets sehr gute Zusammenarbeit und das angenehme Arbeitsklima. Hervorzuheben sind Prof. Dr.-Ing. Lydia Kaiser und Dr.-Ing. Christian Tschirner. Vielen Dank für die vielen Gespräche auf beruflicher und persönlicher Ebene und eure Begeisterung für das Systems Engineering.

Mein Dank gilt auch Dr.-Ing. Christian Bremer, Dr.-Ing. Sebastian von Enzberg, Matthias Greinert und Dr.-Ing. Anja Schierbaum für die inhaltlichen Diskussionen, Ideen und dauerhafte Motivation. Ferner danke ich Florian Heiny und Pritha Gupta für die hervorragende interdisziplinäre Zusammenarbeit im Forschungsprojekt. Ferner allen weiteren Studierenden, die mich durch ihre Studien- und Abschlussarbeiten oder durch ihre studentische Hilfstätigkeit bei meiner Arbeit unterstützt haben.

Meiner Familie gebührt das allergrößte Dankeschön! Meinen Eltern Edelgard und Gerhard, die mir ein Vorbild sind und denen ich unendlich dankbar bin für ihre Fürsorge und Ausdauer bei der Erziehung und Ausbildung. Meiner Frau Lei danke ich für ihre Liebe, Zuneigung und Unterstützung - In den letzten Jahren musstest Du häufig zurückstecken. Trotzdem standest Du stets an meiner Seite und hast mir die Energie und die Ausdauer gegeben weiterzumachen. Dir gilt mein größtes Dankeschön. Schließlich widme ich Dir, mein lieber Friedrich, diese Arbeit.

你问我爱你有多深?
月亮代表我的心!

Paderborn, im November 2021 Michael Hillebrand

Vorveröffentlichungen

[BHB+18] BERNIJAZOV, R.; HILLEBRAND, M.; BREMERN, C.; KAISER, L.; DUMITRESCU, R.: Specification Technique for Virtual Testbeds in Space Robotics. In: 4th International Conference on System-Integrated Intelligence: Intelligent, Flexible and Connected Systems in Products and production (SysInt), 19.-20. June 2018, Hannover, Procedia Manufacturing, Vol. 24, 2018.

[BHH+15] BREMERN, C.; HILLEBRAND, M.; HASSAN, B.; DUMITRESCU, R.: Konzept für das ganzheitliche Testen komplexer mechatronischer Systeme. In: Digitales Engineering zum Planen, Testen und Bereiben technischer Systeme, 18. IFF Wissenschaftstage, 24.-25. Juni 2015, Magdeburg, Tagungsband, 2015.

[HEH21] HILLEBRAND, M., V. ENZBERG, S.; HERZOG, O.: Selbstheilende Systeme in der Smart Factory. Ein Konzept zur Steigerung der Resilienz und Autonomie. Industrie 4.0 Management, 2/2021, Gito Verlag, Berlin. 2021.

[HGD+20] HILLEBRAND, M.; GREINERT, M.; DUMITRESCU, R.; HERZOG, O.: Advanced Monkey Testing for connected autonomous systems. In: 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), 02.-04 June 2020, Budapest, 2020.

[HLD20] HILLEBRAND, M.; LAKHANI, M.; DUMITRESCU, R.: A design methodology for deep reinforcement learning for autonomous systems. In: 5th International Conference on System-Integrated Intelligence: Intelligent, Flexible and Connected Systems in Products and Production (SysInt), 11.-13. November 2020, Bremen, Procedia Manufacturing, Vol. 52, 2020.

[KEH20] KUMAR JHA, N.; V. ENZBERG, S.; HILLEBRAND, M.: Using Deep Learning for Anomaly Detection in Autonomous Systems. In: European Research Consortium for Informatics and Mathematics (ERCIM), ERCIM News 122, Biot, 2020.

[MHD+15] MICHAEL, J.; HILLEBRAND, M.; DUMITRESCU, R.; HENKE, C.; TRÄCHTLER, A.: Modellbasierte Mehrzieloptimierung zur Integration von Hausgeräten in SmartGrids. In: VDI Mechatronik 2015, 12.-13. März 2015, Dortmund, Tagungsband, 2015.

[MHW+16] MICHAEL, J.; HILLEBRAND, M.; WOHLERS, B.; HENKE, C.; DUMITRESCU, R.; TRÄCHTLER, A.: Implementing intelligent technical systems into smart grids by using model-based systems engineering and multi-agent systems. In: International Conference on Renewable Energies and Power Quality (ICREPQ), 04-06. May 2016, Madrid, 2016.

[RSR+15a] ROSSMANN, J.; SCHLUSE, M.; RAST, M.; HOPPEN, M.; DUMITRESCU, R.; BREMER, C.; HILLEBRAND, M.; STERN, O.; BLÜMEL, F.; AVERDUNG, C.: Architecture for Integrated Development of Complex Systems with Model-Based System Specification and Simulation. In: ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. 02.-05. August 2015, Boston, 2015.

[RSR+15b] ROSSMANN, J.; SCHLUSE, M.; RAST, M.; HOPPEN, M.; DUMITRESCU, R.; BREMER, C.; HILLEBRAND, M.; STERN, O.; BLÜMEL, F.; AVERDUNG, C.: Integrierte Entwicklung komplexer Systeme mit modellbasierter Systemspezifikation und -simulation. In: Wissenschaftsforum Intelligente Technische Systeme (WInTeSys), 23.-24. April 2015, Paderborn, 2015.

[RSR+17] ROSSMANN, J.; SCHLUSE, M.; RAST, M.; HOPPEN, M.; ATORF, L.; DUMITRESCU, R.; BREMER, C.; HILLEBRAND, M.; STERN, O.; SCHMITTER, P.: Integrierte Entwicklung komplexer Systeme mit modellbasierter Systemspezifikation und -simulation – Eine Fallstudie zur Sensorauslegung in der Raumfahrt. In: Wissenschaftsforum Intelligente Technische Systeme (WInTeSys), 11.-12. Mai 2017, Paderborn, 2017

Zusammenfassung

Intelligente Technische Systeme kombinieren Sensorik, Aktorik und die Selbstregulation zu intelligenten Regelkreisen. Diese Systeme lösen selbstständig komplexe Aufgaben in ihrer Anwendungsdomäne. Dabei agieren sie situativ und zielführend. Derartige Systeme eröffnen neue Möglichkeiten in der Produktion oder in der Mobilität. Mit diesen Nutzenpotenzialen geht die Zunahme der Komplexität einher. Sie stellen neue Anforderungen an die Verlässlichkeit. Ein Lösungsansatz bietet die Integration von Selbstheilungseigenschaften. Selbstheilende Systeme überwachen ihren Gesundheitszustand, detektieren selbstständig Beeinträchtigungen und können ihre intendierte Funktion aufrechterhalten oder wiederherstellen. Diese Systeme sind wandlungsfähig, robust, lernfähig, redundant und verteilt. Übergeordnetes Ziel ist die Steigerung der Resilienz und Autonomie. Eine durchgehende Systematik zur Entwicklung dieser selbstheilenden Systeme existiert bisher nicht. Diese Systematik ist Gegenstand der vorliegenden Arbeit. Ihr Einsatz ist für ein interdisziplinäres Entwicklungsteam konzipiert. Die Systematik führt die Bereiche Selbstheilung, Data-Driven Systems Engineering und Virtuelle Testbeds zusammen. Das Ergebnis ist ein Selbstheilungskonzept, eine Entwurfstechnik, sowie eine Werkzeugumgebung zum Szenario-basierten Trainieren und Testen. Ein Vorgehensmodell führt diese Bestandteile zu einer Systematik zusammen und macht sie für Dritte anwendbar. Die Anwendung der Systematik ist anhand eines autonomen Fahrzeugs gezeigt.

Summary

Intelligent Technical Systems combine sensors, actuators, and self-regulation to intelligent control loops. These systems solve complex tasks in their operational domain. They act situational and goal-oriented. Intelligent Technical Systems open up fascinating opportunities in multiple domains such as industrial production or mobility. These benefits come with an increase in system complexity and new requirements on dependability and resilience. One solution is the integration of self-healing properties. Self-healing systems are able to monitor their health state and recover from malfunction. These systems are versatile, robust, redundant, distributed and they can learn from experience. The overall goal is to increase systems resilience and autonomy. A design framework for self-healing systems does not yet exist. This framework is main contribution of the present work. The framework is designed for an interdisciplinary development team. It brings together the areas of self-healing concepts, data-driven systems engineering and virtual testbeds. The result is an architectural blueprint for self-healing systems, a design technique, and a tool environment for scenario-based training and testing using advanced monkey testing techniques. A process model brings these parts together and makes them applicable. Several demonstrators show the application of the design framework.

Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme

Inhaltsverzeichnis

Seite

Anhang

1 Einleitung

Die vorliegende Arbeit entstand im Rahmen der anwendungsorientierten Forschung am Fraunhofer IEM[1] in Paderborn. Die Ergebnisse basieren auf den Arbeiten aus verschiedenen Forschungsprojekten. Darunter sind insbesondere das BMBF-Forschungsprojekt *Validierung Künstlicher Immunsysteme für Autonome Systeme (KI4AS)* sowie das vom Deutschen Zentrum für Luft- und Raumfahrt (DLR) geförderte Forschungsprojekt *Integrierte Entwicklung komplexer Systeme mit Virtuellen Testbeds auf der Basis zentraler Weltmodelle und moderner Konzepte der eRobotik (INVIRTES)* zu nennen. Die vorliegende Arbeit ordnet sich in das Forschungsfeld Advanced Systems Engineering ein und beschreibt eine *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*.

1.1 Problematik

Die Digitalisierung wird die technischen Erzeugnisse von morgen entscheidend prägen. Diese Systeme werden eine inhärente Teilintelligenz aufweisen. Sie sind adaptiv, robust, vorausschauend und benutzungsfreundlich [Dum10, S. 41]. Derartige Intelligente Technische Systeme werden autonom agieren und dynamisch vernetzt sein, sie werden im Sinne von einem „System of Systems“ in einem dynamisch vernetzen Systemverbund kooperieren und auch autark voneinander operieren [GT18, S.7].

Diese **Systeme** lösen selbstständig komplexe Aufgaben innerhalb einer bestimmten Anwendungsdomäne. Sie agieren ohne Fernsteuerung oder weitere menschliche Eingriffe zielführend und an die Situation angepasst Dazu nehmen die Systeme über die Sensorik ihre Umgebung wahr, generieren proaktiv und situationsgerecht einen Handlungsplan und führen diesen über die Aktorik verlässlich aus [DGS+18, S. 12]. Die Systeme sind fähig im Betrieb auf neue Ereignisse zu reagieren und Aktionen anhand von Erfahrungswissen zu lernen. Hierfür werden zahlreiche technologische Bausteine benötigt, z.B. multimodale Sensorfusion, semantische Umfeldmodelle oder Planungsverfahren [FAS17, S. 135]. Diese Fähigkeiten eröffnen faszinierende Anwendungsperspektiven [FAS17], [DGS+18]:

- **Autonomes Fahren** ermöglicht neue Formen der Mobilität und stellt neue Anforderungen an das Verkehrssystem. Gleichsam ist die Automatisierung ein Schlüsselelement für mehr Sicherheit und Effizienz im Straßen- und Schienenverkehr bei gleichzeitiger Reduzierung der Umweltbelastung. CASE prägt die Mobilität der Zukunft: Vernetzt (Connected), Autonom (Autonomous), flexible Nutzung (Shared and Services) und elektrische Antriebe (Electric) [Lem16], [Dai19].
- Intelligente Verbraucher sind in einem **Smart Home** vernetzt und passen ihr Verhalten an die schwankende Energieverfügbarkeit an. Sie stimmen sich dezentral mit

[1] Fraunhofer-Institut für Entwurfstechnik Mechatronik

anderen Geräten ab, um ein Energie- und Preisoptimum bei volatilen Energiepreisen zu erzielen [MHW+16].

- Autonome Systeme können verlässlich in **Menschenfeindlichen Umgebungen** wie der Tiefsee oder dem Weltraum operieren. Dort übernehmen sie u.a. Aufgaben der automatischen Instandhaltung von Anlagen oder dienen zur planetaren Exploration [Wah17], [OT19].

Die Systeme beruhen auf dem synergetischen Zusammenwirken der Elektrotechnik, der Informatik, dem Maschinenbau, sowie der Künstlichen Intelligenz [VDI2206, S. 11ff.]. Neben diesen offensichtlichen Nutzenpotenzialen dieser Systeme, führt ihre Entwicklung zu einer Zunahme der Komplexität und neuen Anforderungen an die Verlässlichkeit [GRS+14, S. 12]: Intelligente Technische Systeme haben nicht zwangsläufige eine feste Systemgrenze. Ihre Funktionalität kann sich im Betrieb erweitern oder verändern (z.B. Nachrüstung, Updates) [GT18]. Dabei lernen sie innerhalb vorgegebener Randbedingungen und entwickeln sich selbstständig weiter. Die Systeme passen ihr Verhalten (mehr-)zielorientiert an [GRS14, S. 6]. Das Systemverhalten ändert sich und wird durch Umgebungsbedingungen beeinflusst. Dabei operieren diese Systeme in einer offenen, unstrukturierten und dynamischen Umgebung [FAS17]. Entwickler können diese Herausforderungen nur begrenzt antizipieren, müssen jedoch die Verlässlichkeit im Betrieb sicherstellen. Ein Ansatz ist die Integration von Eigenschaften der **Selbstheilung** [FAS17, S. 135]. Die Selbstheilung bietet das Potenzial, Funktionsstörungen zu detektieren, selbstständig auszugleichen und die übergeordnete Systemfunktion wieder herzustellen um so die Verfügbarkeit und Sicherheit zu steigern [GSR+07], [PS11]. Übergeordnetes Ziel ist die Steigerung der Resilienz.

Diese Herausforderungen müssen bereits frühzeitig in der Entwicklung begegnet werden. Ein vielversprechender Ansatz ist das **Systems Engineering**: Systems Engineering hat das Potenzial, Disziplinen, Entwicklungsaufgaben und die Akteure in der Entwicklung dieser komplexen Systeme zu orchestrieren. Es versteht sich als ein durchgängiges und Fachdisziplin-übergreifendes Paradigma zur Entwicklung komplexer Systeme. Es fehlen bislang noch durchgängige Ansätze sowie integrierte Entwicklungswerkzeuge zur Modellierung, Simulation sowie zur Visualisierung von Intelligenten Technischen Systemen [DGS+18, S. 18]. Simulationswerkzeuge müssen in Kombination mit realen Versuchsaufbauten aussagekräftige virtuelle Szenarien für das Training autonomer Systeme bereitstellen. Wie aussagekräftige virtuelle Szenarien jedoch auszusehen haben und welche Menge an Szenarien für den verlässlichen Betrieb autonomer Systeme ausreichend sind, bilden aktuelle Forschungsfragen [DGS+18, S. 19].

Diese skizzierte Problematik führt zu der Notwendigkeit, die Entwicklung von selbstheilenden Systemen methodisch zu unterstützen. Das Handlungsfeld befindet sich im **disziplinübergreifenden Systementwurf**. Hier fehlt ein systematisches Vorgehen sowie Techniken zur Modellierung und frühzeitigen Absicherung selbstheilender Systeme. Vor diesem Hintergrund ergeben sich die folgenden Fragestellungen:

- Wie kann eine Selbstheilungskonzept für Intelligente Technische Systeme aussehen?

- Welche Prinzipien unterstützen die Umsetzung der Selbstheilung?
- Welche Entwurfsaspekte sind im Systementwurf zu adressieren?
- Wie sieht ein Absicherungskonzept für selbstheilende Systeme aus?
- Wie können selbstheilende Systeme datengetrieben entwickelt werden?

1.2 Zielsetzung

Ziel der Arbeit ist somit eine *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*. Die Systematik unterstützt die Entwicklung selbstheilender Systeme über verschiedene Reifegrade des Systems (z.B. Funktionsmuster, Labormuster). Die Systematik ist für die Anwendung in einem interdisziplinären Team in der frühen Phase der Produktentwicklung konzipiert. Im Kern soll die Systematik die folgenden drei Bestandteilen umfassen:

- ein **Selbstheilungskonzept** zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme,
- eine Spezifikationstechnik zur **interdisziplinären Analyse, Synthese und Absicherung** selbstheilender Systeme,
- fortgeschrittene Analyse- und Simulationstechniken zum Training und zum Testen selbstheilender Systeme in einer **virtuellen Testumgebung**.

Das **Selbstheilungskonzept** soll den Selbstheilungsprozess für Intelligente Technische Systeme definieren und davon ausgehend eine Rahmenwerk beschreiben. Darüber hinaus ist Lösungswissen zur technischen Realisierung bereitzustellen. Die **Spezifikationstechnik** soll eine Methode, Sprache und ein Werkzeug zur Analyse, Synthese und Absicherung in der frühen Phase der Produktentstehung bereitstellen. Dabei sind Risikoanalyse und Absicherung in das Sprachkonzept zu integrieren. Die **Virtuelle Testumgebung** soll bei der Simulation und Szenario-basierten Absicherung selbstheilender Systeme unterstützen. Die **Entwicklungssystematik** fasst diese Bestandteile schließlich zusammen. Das Resultat der Systematik ist ein System mit integrierten Selbstheilungseigenschaften. Als System wird dabei nicht ausschließlich das fertige, real existierende Erzeugnis verstanden, sondern die zunehmende Konkretisierung in Form von Reifegraden. Reifegrade sind z.B. Labormuster, Funktionsmuster, ein virtueller Prototyp etc. [VDI2206, S. 30].

Die durchgängige Anwendbarkeit der Systematik soll anhand eines Demonstrators nachgewiesen werden. Hierbei handelt es sich um eine intra-logistische Transportaufgabe, die durch einen hoch-automatisierten mobilen Roboter realisiert wird.

1.3 Vorgehensweise

Die vorliegende Arbeit gliedert sich in sechs Kapitel. Das **Kapitel 2** konkretisiert die einleitende Problematik. Es werden die relevanten Begriffe definiert. Im Anschluss werden Funktionsweise und Aufbau von Intelligenten Technischen Systemen erläutert. Merkmal dieser Systeme ist die Verlässlichkeit und ihre Resilienz. Eine wesentliche Rolle spielt hier die Selbstheilung als Eigenschaft zur Steigerung der Resilienz. Ferner wird das Systems Engineering dieser Systeme skizziert. Der Fokus liegt auf den frühen Phasen der Entwicklung, dem sog. interdisziplinären Systementwurf. Im Rahmen der Problemabgrenzung erfolgt die Zusammenfassung der Handlungsfelder. Entlang der Handlungsfelder werden zuletzt die Anforderungen an die geforderte Systematik definiert.

Das **Kapitel 3** evaluiert den Stand der Forschung. Dazu werden ausgewählte Ansätze vorgestellt, die sich zur Erfüllung der Anforderung prinzipiell eignen. Zunächst werden Ansätze zur Realisierung der Selbstheilung erläutert. Der zweite Teil betrachtet Hilfsmittel zur Spezifikation selbstheilender Systeme. Der dritte Teil umfasst Techniken zur kontinuierlichen Absicherung von Systemeigenschaften. Abschließend werden die vorgestellten Ansätze hinsichtlich der Anforderungen aus Kapitel 2 bewertet. Daraus resultiert der Handlungsbedarf für die angestrebte Systematik.

Das **Kapitel 4** bildet den Kern der Arbeit. In diesem Kapitel wird die *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* erläutert. Zunächst wird ein Überblick über die Systematik gegeben. In den nachfolgenden Abschnitten werden die einzelnen Bestandteile der Systematik im Detail erläutert. Hierzu zählt das *Selbstheilungskonzept* für Intelligente Technische Systeme, die *Potenzialanalyse* und die *Entwurfstechnik* sowie das *Werkzeugkonzept* und die *Analysetechnik* für selbstheilende Systeme mittels Virtuellen Testbed. Das *Vorgehensmodell* führte diese Bestandteile schließlich zusammen.

Die Anwendung der Systematik erfolgt in **Kapitel 5**. Anhand eines Demonstrators wird die durchgängige Anwendung und der Nutzen der Systematik gezeigt. Das Kapitel schließt mit einer Bewertung der erarbeiteten Systematik hinsichtlich der definierten Anforderungen aus Kapitel 2.

Das **Kapitel 6** fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf den weiteren Forschungsbedarf. Der **Anhang** umfasst vertiefende Informationen zum Stand der Technik sowie zu der erarbeiteten Systematik.

2 Problemanalyse

2.1 Begriffsbestimmungen

Dieses Kapitel definiert die grundlegenden Begriffe. Ausgehend vom Systembegriff werden die für Autonome Systeme wichtigen Aspekte wie Parameter, Verhalten und Struktur definiert.

Die DIN 19226 definiert ein **System** als

> *„[...] eine in einem betrachteten Zusammenhang gegebene Anordnung von Elementen, die miteinander in Wechselwirkung stehen. Diese Anordnung wird aufgrund bestimmter Vorgaben gegenüber ihrer Umgebung abgegrenzt.“ [DIN19226]*

So verbindet ein mechatronisches System mindestens einen Sensor, einen Aktor, eine Informationsverarbeitung und ein Grundsystem zu einem Regelkreis.

Systeme bestehen aus **Elementen**, die ihrerseits wieder als Systeme betrachtet werden können. Elemente haben **Eigenschaften** oder eine Funktion. Die Elemente sind untereinander oder mit ihrer Umgebung durch **Beziehungen** verbunden. Bei der Beziehung kann es sich um einen Material-, Informations- oder Energiefluss sowie um Beziehung der Lage oder Wirkzusammenhänge etc. handeln [HFW+15, S. 32]. Eine Beziehung kann gerichtet oder ungerichtet sein. Gerichtete Beziehungen zwischen Systemelementen und Elementen der Umgebung werden als **Eingang** bzw. **Ausgang** bezeichnet [FGK+04, S. 16].

Die Elemente des Systems bilden mit ihren Beziehungen ein Wirkgefüge. Dieses wird als **Struktur** bezeichnet. Wird das System in mehrere Stufen gegliedert, ergibt sich ein hierarchischer Aufbau bzw. die **Systemhierarchie**. Ein System wird durch eine **Systemgrenze** von der Systemumgebung abgegrenzt. Unter der **Systemumgebung** werden Systeme oder Elemente verstanden, die außerhalb der Systemgrenze liegen und Einfluss auf das System nehmen bzw. durch das System beeinflusst werden [FGK+04, S. 16], [HFW+15, S. 33].

Ein **System von Systemen** (engl. System of Systems) besteht aus mehreren individuellen Systemen und zeichnet sich dadurch aus, dass jedes einzelne System unabhängig oder zusammen mit anderen Systemen betrieben werden kann. Jedes Einzelsystem besitzt also einen eigenen Zweck. Darüber hinaus wird jedes einzelne System dieses Systemverbundes unabhängig von den anderen Systemen entwickelt. Ein typisches Beispiel im Automobilbau ist die Kopplung von einem Mobiltelefon mit dem Fahrzeug. Die Karosserie, das Fahrwerk oder der Antrieb bilden in diesem Sinne kein System von Systemen [HFW+15, S. 35f.].

Einflüsse können den Zweck des Systems unterstützen, behindern oder verhindern. Behindernde bzw. Verhindernde Einflüsse sind Störgrößen. Weiterhin werden zwischen unstrukturierten, nicht vorhersehbaren, nicht geplanten, gezielten und geplanten Einflüssen auf das System unterschieden. Letztere werden als Vorgaben bezeichnet. Dem System

werden durch den Benutzer oder durch andere Systeme Vorgaben als Ziele gegeben. Neben den zweckunterstützenden Einflüssen können von dem Benutzer oder von anderen technischen Systemen Störgrößen auf das System einwirken. Neben den Einflüssen auf das System kann es auch zu ungewollten Ausgangsgrößen von dem System auf die Umgebung kommen [FGK+04, S. 17ff.].

Das **Verhalten** eines Systems wird durch dessen Eingangs- und Ausgangsgrößen in einem bestimmten Zustand charakterisiert. Das System befindet sich zu jedem Zeitpunkt in einem definierten **Zustand**. Dabei werden zwei Arten von Zuständen unterschieden. Zustandsgrößen sind physikalische Größen eines technischen Systems durch deren Wert zu einem beliebigen Zeitpunkt t_0 der Ablauf des Systems für $t > t_0$ eindeutig bestimmt ist, sofern die Eingangsgröße des Systems für $t < t_0$ gegeben sind [Foe13]. Bei ereignisdiskreten Systemen beschreibt ein Zustand im Zustandsautomaten eine Situation, in der eine bestimmte, unveränderliche Bedingung gilt. Eine **Funktion** beschreibt den allgemeinen, lösungsneutralen und gewollten Zusammenhang zwischen Eingang und Ausgang eines System, mit dem Ziel, eine Aufgabe zu erfüllen. Aus Sicht der Informationsverarbeitung wird das Systemverhalten als eine Abfolge von **Aktivitäten** verstanden. Eine Aktivität ist ein konkreter, von einem Systemelement durchgeführter Schritt (z.B. Sensorwerte erfassen). Aktivitäten repräsentieren die aktive Sicht auf das Systemverhalten. Die Aktivitäten der Systemelemente können auf Systemebene zu einer **Aktion** zusammengeführt werden (z.B. Fahrdynamik regeln). Ist einem System eine Funktion zugeordnet, so kann die Erfüllung dieser Funktion durch eine Aktion ausgedrückt werden [FGK+04, S. 16f.].

Mit der zunehmenden Komplexität technischer Systeme tritt das Selbstmanagement und die Emergenz stärker in den Vordergrund. Der Begriff **Self-X** wurde durch Initiativen wie *Autonomic Computing* [IBM05] und *Organic Computing* [SNS+11] geprägt. Dabei sind vier Aspekte des Selbstmanagements technischer Systeme zu unterscheiden, die sich unter dem Paradigma **Self-X-Eigenschaften** zusammenfassen: Die *Selbstkonfiguration (Self-Configuration)* umfasst die automatische Konfiguration von Elemente des Systems. Bei der *Selbstoptimierung (Self-Optimization)* suchen Elemente und Systeme kontinuierlich nach Möglichkeiten, ihre eigene Leistung und Effizienz zu steigern. Die *Selbstheilung (self-Healing)* beschreibt die Eigenschaft, Software- und Hardware-Probleme selbstständig zu entdecken, zu diagnostizieren und zu beseitigen. Schließlich der *Selbstschutz (Self-Protection* mit dem sich Systeme automatisch gegen Angriffe von außen verteidigen [IBM05]. Das Selbstmanagement kann zentral koordiniert werden, kann aber auch das Ergebnis der dezentral organisierten Interaktionen sein. Der letzte Fall wird als **Emergenz** bezeichnet, da sich das Selbstmanagement auf Systemebene allein aus dem lokalen Zusammenspiel der Elemente ergibt [SNS+11]. Mit dem Sonderforschungsbereich (SFB) 614 wurden diese Eigenschaften auf maschinenbauliche Erzeugnisse übertragen [GRS14].

2.2 Von der Mechatronik zu Autonomen Systemen

Dieses Kapitel erläutert den Wandel von mechatronischen Lösungen zu autonomen Systemen. Zunächst wird der grundlegende Aufbau mechatronischer Systeme vorgestellt (Kap. 2.2.1). Diese Grundstruktur bildet die Basis für selbstoptimierende und autonome Systeme (Kap. 2.2.2). Die Wandel zu autonomen Systemen vollzieht sich in der Informationsverarbeitung und der Vernetzung. Daher werden die Kern-, Umgebungs- und Querschnittstechnologien autonomer Systeme betrachtet. Die Konzepte münden in der Referenzarchitektur autonomer Systeme (Kap. 2.2.3).

2.2.1 Mechatronische Systeme

Die Mechatronik kombiniert mechanische, elektronische und informationstechnische Komponenten zu einem System. Mechatronische Systeme sind heute in jeder technischen Branche im Einsatz z.B. die Abstandsregelung im Fahrzeug oder Handhabungstechniken in der Automatisierungstechnik. Dies sind nur zwei repräsentative Beispiele für ihre Anwendung.

Die Grundstruktur eines mechatronischen Systems besteht aus einem Grundsystem, der Sensorik und Aktorik sowie einer Informationsverarbeitung. Diese vier Einheiten bilden zusammen einen systeminternen Regelkreis (Bild 2-1) [VDI2206, S. 14]:

- Das **Grundsystem** ist eine mechanische, elektromechanische, hydraulische oder pneumatische Struktur bzw. eine Kombination aus diesen Strukturen als ein physikalisches System.
- Die **Sensorik** erfasst die physikalischen oder chemischen Eigenschaften und/oder die stoffliche Beschaffenheit der Umgebung bzw. die Zustandsgrößen des Grundsystems. Sensoren können physisch vorhandene oder virtuelle Messwertaufnehmer sein.
- Die **Aktorik** wandelt die Sollvorgaben der Informationsverarbeitung in physikalische Effekte um, insbesondere in mechanische Bewegungen. Aktionen können auch rein digital umgesetzt werden.
- Die **Informationsverarbeitung** ist das zentrale Element der logischen Ebene. Ihre Aufgabe ist die Ermittlung aller relevanten Einwirkungen, um die Zustandsgrößen des Grundsystems mittels der Aktorik entsprechend der Vorgaben zu beeinflussen.

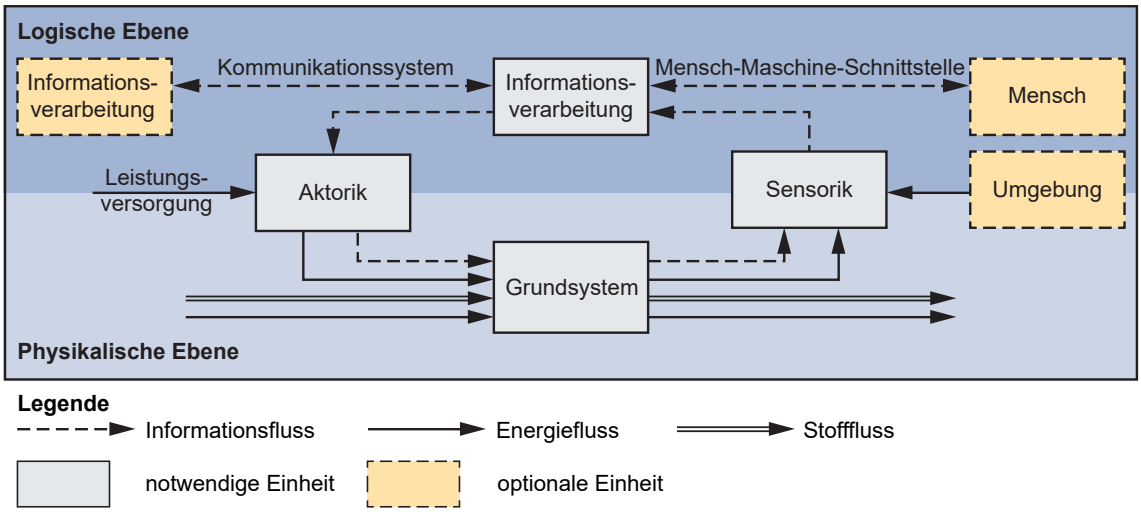


Bild 2-1: Grundstruktur mechatronischer Systeme [VDI2206, S. 14]

Diese Grundstruktur ist die Basis für die darauf aufbauende Strukturierung vernetzter mechatronischer Systeme nach LÜCKEL ET AL. [LKS00]:

- **Mechatronisches Funktionsmodul (MFM):** Dies bildet das Basiselement eines mechatronischen Systems und weist alle vier Elemente der Grundstruktur eines mechatronischen Systems auf.
- **Autonomes Mechatronisches System (AMS):** Informationstechnisch und/ oder mechanisch gekoppelte MFM führen zu autonomen mechatronischen Systemen. In der Informationsverarbeitung werden übergeordnete Aufgaben wie die Überwachung und die Fehlerdiagnose realisiert.
- **Vernetztes Mechatronisches System (VMS):** Durch die Interaktion mehrerer AMS entsteht ein VMS. Auf dieser Ebene findet ausschließlich eine informationstechnische Kopplung statt.

2.2.2 Selbstoptimierende Systeme

Der Begriff Selbstoptimierende Systeme wurde durch den SFB[1] 614 „Selbstoptimierende Systeme des Maschinenbaus“ geprägt. Grundlage dieser Systeme ist die Mechatronik. Der Ausgangspunkt für die Selbstoptimierung ist die Regelungstechnik und die Informationsverarbeitung. Die klassische Regelung hat die Aufgabe Sensorsignale auszuwerten und über verschiedene Reglerstrukturen die Sollgrößen in entsprechende Stellsignale umzuwandeln. Adaptive Regelungen gehen weiter und passen z.B. Reglerparameter an veränderte Umweltbedingungen an. Die Selbstoptimierung erweitert diese Eigenschaft erheblich. Der Begriff Selbstoptimierung wird wie folgt definiert:

„Unter Selbstoptimierung eines technischen Systems wird die endogene

[1] Förderung von 2002 bis 2013

> *Änderung der Ziele des Systems auf veränderte Einflüsse und die daraus resultierende zielkonforme autonome Anpassung der Parameter und ggf. der Struktur und somit des Verhaltens dieses Systems verstanden. Damit geht Selbstoptimierung über die bekannten Regel- und Adaptionsstrategien wesentlich hinaus. Selbstoptimierung ermöglicht handlungsfähige Systeme mit inhärenter 'Intelligenz', die in der Lage sind, selbstständig und flexibel auf veränderte Betriebsbedingungen zu reagieren." [GRS14, S. 5]*

Diese Anpassung des Verhaltens wird durch den Selbstoptimierungsprozess erreicht. Dieser Prozess ist in Bild 2-2 definiert [GRS14, S. 5ff.]:

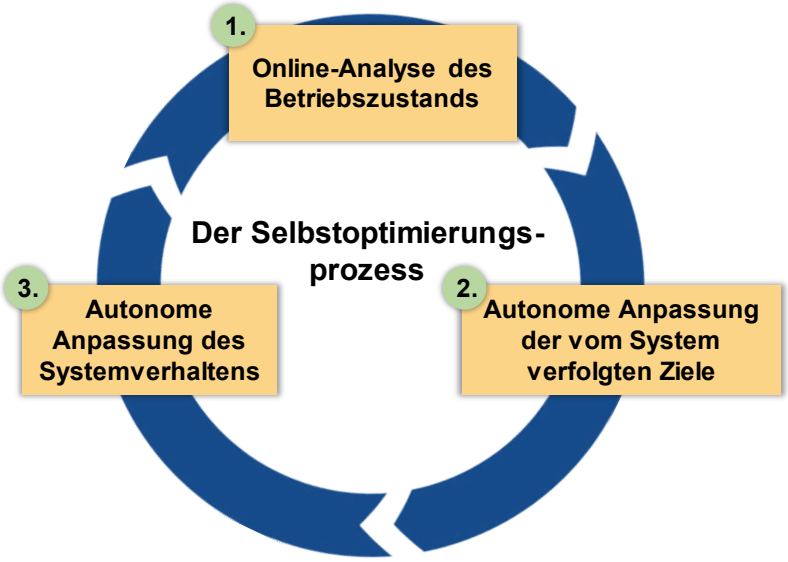


Bild 2-2: Prozess der Selbstoptimierung [GRS14, S. 7]

In der ersten Aktion wird die Situation analysiert. Dies beinhaltet die Online-Analyse des Betriebszustandes. Neben der Analyse bzw. Diagnose der aktuellen Situation und des aktuellen Systemzustandes kann über eine Zustandsprognose eine frühzeitige Verhaltensanpassung ermöglicht werden. Darüber hinaus erfolgt eine situationsadäquate Bewertung der internen Systemziele. Die autonome Anpassung der vom System verfolgten Ziele ist die zweite Aktion der Selbstoptimierung. Die dritte Aktion bildet die autonome Anpassung des Systemverhaltens. Die Verhaltensanpassung wird schließlich durch Parameter- oder Strukturanpassung realisiert [GRS14, S. 6f.].

2.2.3 Autonome Systeme

Die Informationsverarbeitung vollzieht den Wandel von mechatronischen zu autonomen Systemen. Mechatronische Systeme besitzen eine starre Kopplung zwischen Sensorik und Aktorik. Autonome Systeme besitzen darüber hinaus eine flexible Kopplung zwischen Wahrnehmung und Handlung. Systeme gelten als autonomen:

> *„[...] wenn sie ohne menschliche Steuerung und expliziter Programmierung ein vorgegebenes Ziel selbstständig und an die Situation angepasst erreichen. Autonome Systeme haben die Fähigkeit, sich der Umwelt anzupassen, zu lernen und gegebenenfalls mit anderen Systemen zu kooperieren. Sie nehmen ihre Umgebung über Sensoren wahr, generieren proaktiv, situationsgerecht und in Echtzeit eine angemessene Aktion und führen diese über Aktoren aus.“ [DGS+18, S. 9]*

Ein autonomes System ist ein lernendes System. Die Lernfähigkeit ist für die Anpassung des Handlungsablaufs notwendig. Allerdings sind nicht alle lernende Systeme autonom, sondern werden teilweise weiterhin bewusst von Menschen gesteuert (z.B. intelligente Prothesen). Die Lernfähigkeit wird maßgeblich durch den Einsatz von Methoden des maschinellen Lernens realisiert [PLS18-ol].

Die Selbstregulation ist die **Kerntechnologie** von autonomen Systemen. Entsprechend der Definition der PLATTFORM LERNENDE SYSTEME bezeichnet die **Selbstregulation**:

> *„die Fähigkeit [...] sich an die Umgebung oder das Verhalten von Menschen anzupassen und eigene Aktionen zu korrigieren. Unterstützt wird die Selbstregulation durch Wahrnehmung und Interpretation, Planung und Planerkennung, Lernen und Schlussfolgern sowie Kommunikation und Kollaboration. Neben der Aktorik und Sensorik ist die Selbstregulation eine der drei Hauptkomponenten von autonomen Systemen“ [PLS18-ol].*

Somit können autonome Systeme mit Situationen umgehen, die von Vagheit, Unsicherheit, Unvollständigkeit und Ressourcenbeschränkung geprägt sind [Wah17]. Wesentliche Fähigkeiten sind der Selbstschutz, die Selbsterhaltung sowie die Selbstheilung [DGS+18, S. 13]. Unterstützt wird die Selbstregulation durch Fähigkeiten der Wahrnehmung und Interpretation, der Planung und Planerkennung, des Lernens und Schlussfolgerns sowie der Kommunikation und Kollaboration [PLS18-ol], [Wah17], [DGS+18, S. 14f.]:

- Die **Wahrnehmung und Interpretation** realisiert die Erkennung von Objekten und Situationen in der Umgebung. Ein Beispiel ist die Erkennung von Verkehrsschildern durch das maschinelle Lernen. Das Bild 2-3 zeigt die Erkennung verschiedener Verkehrsschilder durch ein Neuronales Netz und die Konfusionsmatrix[2] zur Bewertung des verwendeten Klassifikators. Die Umfelderkennung muss dabei möglichst in Echtzeit erfolgen. Für eine sichere Erfassung, werden Sensordaten fusioniert und/ oder verschiedene Sensoren eingesetzt und mit Verfahren zur Datenanalyse oder Mustererkennung kombiniert.

[2] Beschreibt den Fehler eines Klassifikators, ordnet das Bild einer falschen Klasse zu.

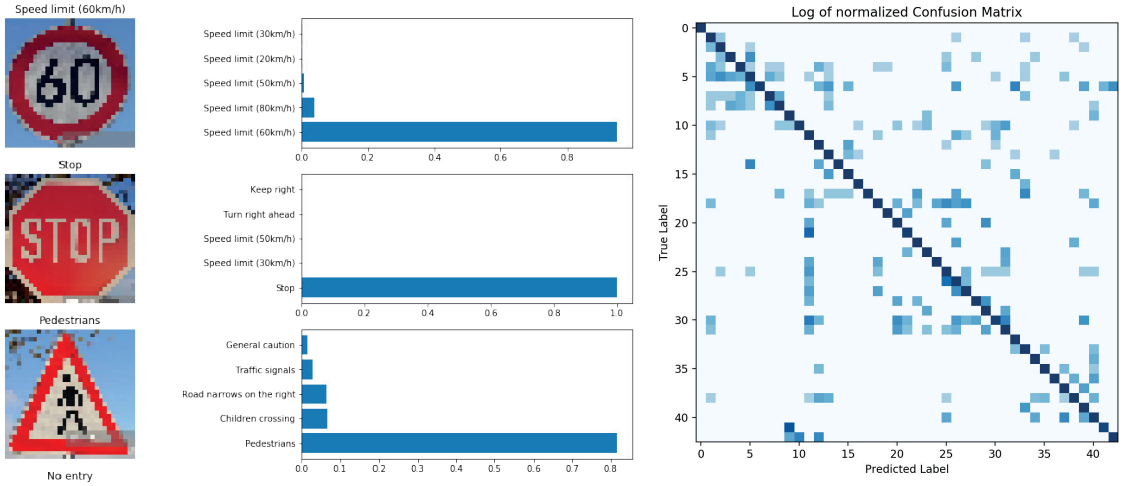


Bild 2-3: Erkennung von Verkehrsschilder durch maschinelles Sehen

- Das **Lernen** ermöglicht es, anhand vorgegebener Entscheidungs- und Bewertungsproblemen Strukturen in den Daten zu erfassen und später auf einen gleichen oder hinreichend ähnlichen Anwendungsfall anzuwenden. Das automatische temporale oder räumliche **Schlussfolgern** ergänzt diese Lernverfahren in Situationen, die einen Mangel an Trainingsdaten aufweisen.
- Die **Planung** beschäftigt sich mit der automatischen Auswahl zielgerichteter Aktionen. Der Ansatz erfolgt modellbasiert, d.h. das System verfügt über ein formales Modell der Welt, des aktuellen Weltzustandes, der verfügbaren Aktionen, einer Zielbedingung sowie diverser Randbedingungen bzgl. Ressourcen, Fristen und Sicherheitsanforderungen. Anhand des Modells simuliert ein Planungsalgorithmus die möglichen Alternativen und priorisiert bzw. selektiert eine geeignete Handlungsstrategie. Bei der **Planerkennung** geht es darum, aus den beobachteten Aktionen eines handelnden Agenten mithilfe einer Planbibliothek auf dessen Ziele bzw. Pläne zu schließen und eine Vorhersage über die Folgeaktionen zu treffen. Die erkannten Pläne und Intentionen werden genutzt, um eine geeignete Unterstützung zu bieten oder in der eigenen Handlungsplanung zu berücksichtigen.
- Die **Kommunikation** eines autonomen Systems wird durch eine multimodale Kommunikation von Sprache, Gestik, Mimik und Blickbewegungen realisiert. Zur **Kollaboration** mit Menschen und anderen autonomen Systemen werden Technologien aus dem Bereich Multiagentensysteme eingesetzt.

Die **Umgebungstechnologien** autonomer Systeme umfassen die Sensorik, die Aktorik sowie eine taktile und sichere Kommunikation. Die **Sensorik** dient zur Erfassung der Umgebung. Die **Aktorik** ermöglicht die Ausführung der Handlungspläne und sorgt für eine Zielerreichung [DGS+18, S. 12].

Die **Kommunikation mit der Umgebung** umfasst die Vernetzung aller notwendigen technischen Komponenten untereinander sowie mit der Infrastruktur. Hierzu sind stan-

dardisierte, systemübergreifende Kommunikationsmechanismen erforderlich. Über die **Kommunikation mit dem Menschen** interagiert das System mit dem Nutzer. In unbekannten Situationen erfolgt ein Transfer der Kontrolle. Eine technische Rückfallebene minimiert die Gefährdung von Menschen bei einem Ausfall zentraler Funktionen. Diese Rückfallebene versetzt das System in einen sicheren Betriebszustand (fail-safe oder fail-operational) ([Wah17], [FAS17, S. 136]).

Neben diesen technologischen Aspekten sind **Querschnittsthemen** für die Realisierung Autonomer Systeme zu adressieren. Hierzu zählen die **Infrastruktur** mit Cloud-Computing, Verfügbarkeit von Geodaten zur Bereitstellung von präzisen Umgebungsmodellen und die flächendeckende Kommunikationsinfrastruktur, wie z.B. der 5G-Mobilfunk bzw. das taktile Internet. Ebenso wichtig ist der Aspekt **Safety and Security**. So ist die Sicherheit vor Manipulation und der Schutz vor nicht autorisierten Zugriffen eine wesentliche Herausforderung. Die funktionale Sicherheit und der Schutz gegenüber Angriffen muss im Betrieb gewährleistet sein. Dabei sind neue Zertifizierungen und Testumgebungen einzusetzen. **Regulierungen und Standards** erforderlich, um in den verschiedenen Anwendungsdomänen eine Kompatibilität und Vernetzung der Systeme zu ermöglichen. Zur Entwicklung und Absicherung dieser Systeme sind geeignete **Werkzeuge** erforderlich. So können mit Hilfe von Referenzarchitekturen und abgeleitete Software-Bibliotheken neue Anwendungen entwickelt und konfiguriert werden. Virtuelle Szenarien und Simulation dienen dazu, Algorithmen zu entwickeln, zu analysieren und zu validieren [DGS+18, S. 19].

Die Referenzarchitektur stellt die Kerntechnologien, Umgebungstechnologien und Querschnittsthemen zusammenfassend dar. Sie ist in Bild 2-4 visualisiert.

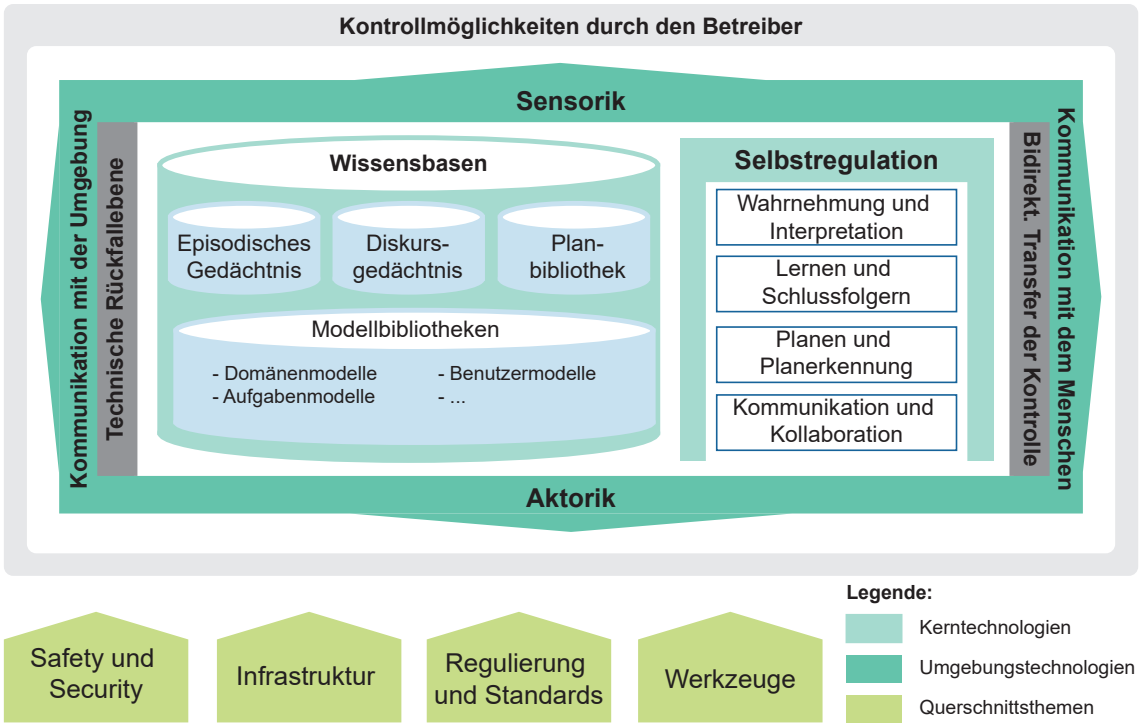


Bild 2-4: Referenzarchitektur Autonomer Systeme [FAS17, S. 18], [DGS+18, S. 12]

Die Entwicklung von mechatronischen Systemen zu autonomen Systemen erfolgt evolutionär und wird nach dem Grad der Automatisierung in Stufen eingeteilt. Das Spektrum reicht von ferngesteuerten Systemen bis hin zu autonomen Systemen (vgl. Bild 2-5).

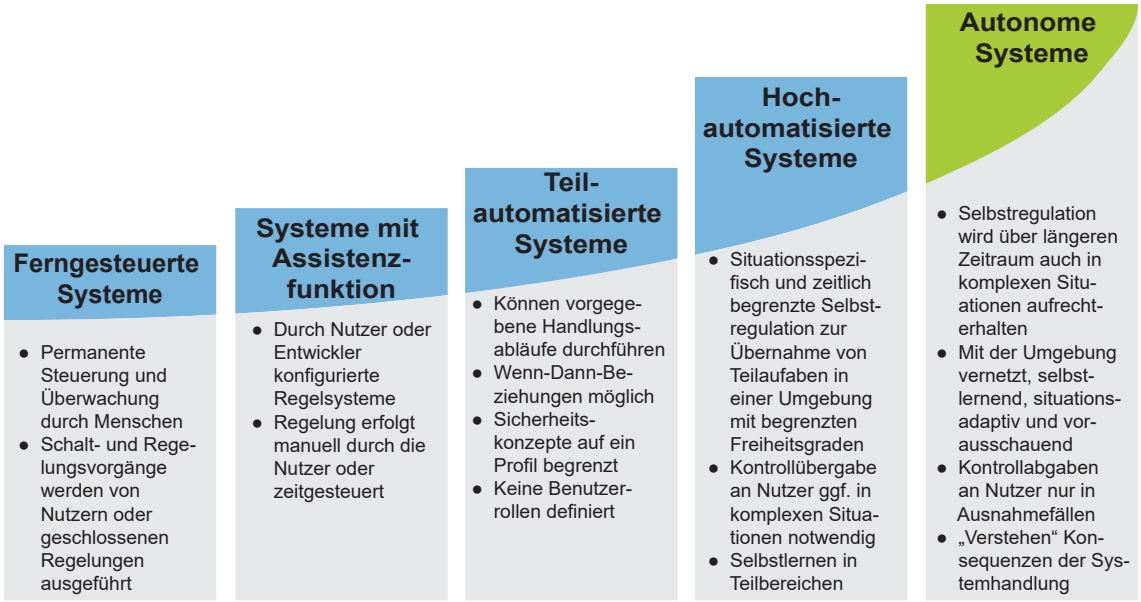


Bild 2-5: Entwicklungsstufen automatisierter Systeme [FAS17, S. 67]

2.3 Verlässlichkeit

Die Verlässlichkeit spielt bei autonomen Systemen eine entscheidende Rolle. Ohne Verlässlichkeit ist ein realer Einsatz kaum vorstellbar. Zunächst werden daher die zentralen Begriff der Verlässlichkeit eingeführt (Kap. 2.3.1). Anschließend werden entlang der skizzierten Referenzarchitektur aus Kap 2.2.3 mögliche Beeinträchtigungen mit Auswirkung auf die Verlässlichkeit erläutert (Kap. 2.3.2). Autonome Systeme stellen hohe Anforderungen an die Verlässlichkeit im Betrieb. So werden Strategien zur Steigerung der Verlässlichkeit (Kap. 2.3.3) dargestellt und eine Roadmap zu resilienten technischen Systemen skizziert. Ein Mittel zur Realisierung der Resilienz ist die Selbstheilung (Kap. 2.3.4). Dabei können nicht alle Betriebssituationen zur Entwicklungszeit antizipiert werden. Ein möglicher Lösungsansatz ist die Integration der Anomalie Detektion in den Selbstheilungsprozess (Kap. 2.3.5).

2.3.1 Begriffe der Verlässlichkeit

Die Terminologie im Bereich der Verlässlichkeit wird im deutschen Raum u.a. durch die VDI-Richtlinie 4001-2 „Terminologie der Zuverlässigkeit“ [VDI4001] bzw. 4003 „Zuverlässigkeitsmanagement“ [VDI4003] definiert. Diese Richtlinien berücksichtigen jedoch nicht den zunehmenden Anteil von Software in mechatronischen Systemen [Son15, S. 11]. Daher werden in dieser Arbeit, in Anlehnung an der Terminologie des SFB 614, die Begriffe in der Verwendung von LAPRIE ET AL. [Lap92] bzw. AVIZIENIS ET AL.

[ALR+04, S. 4] verwendet. Die Definition der Verlässlichkeit erfolgt über die Aspekte Beeinträchtigung, Kenngrößen und Mittel (vgl. Bild 2-6)

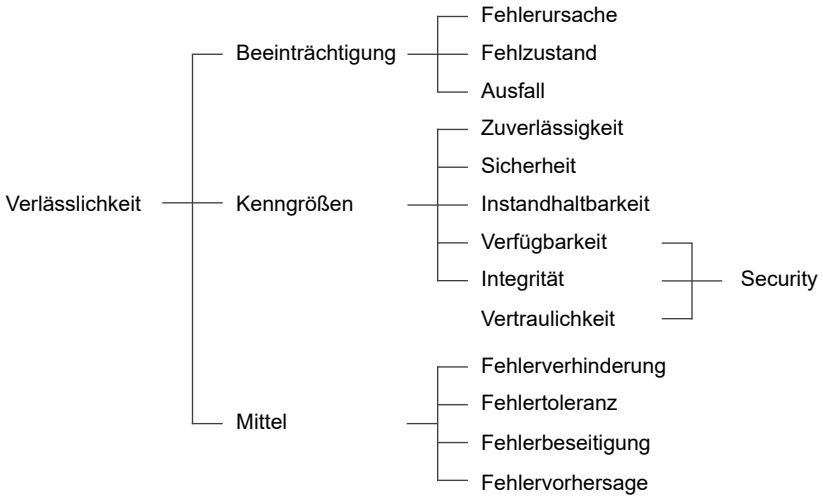


Bild 2-6: Verlässlichkeitsbaum [ALR+04, S. 4]

Beeinträchtigungen

LAPRIE unterscheidet drei Formen der Beeinträchtigung: Die Fehlerursache, der Fehlerzustand und der Ausfall [Lap92]. Dies soll am Beispiel eines Elektromotors in einer Waschmaschine verdeutlicht werden. Eine **Fehlerursache** kann zum Beispiel eine hohe Dauerbelastung des Motors durch eine Umwucht in der Trommel sein. Dies führt in einem Dauerbetrieb zu erhöhten Vibrationen und Verschleiß des Motors. Dies stellt ein **Fehlzustand** dar. Die Vibrationen und der Verschleiß können dazu führen, dass der Motor funktionsunfähig wird. Dies wird als **Ausfall** bezeichnet. Im Sinne einer Fehlerpropagierung kann dieser Ausfall eine Fehlerursache für den Fehlzustand des übergeordneten Systems darstellen. Sollte eine Umwuchtkontrolle oder ein redundanter Motor verbaut sein, muss dies nicht direkt zum Ausfall des übergeordneten Systems führen [ALR+04]. Eine weitere Beeinträchtigung wird durch eine **Störung** dargestellt. Dieser Begriff repräsentiert eine unbeabsichtigte Unterbrechung der Funktionserfüllung einer Einheit. Eine Störung kann durch externen Einflüsse verursacht sein. Ist die Störung die Folge eines Schadens von einer Systemeinheit, sind Ausfall und Störung synonym verwendete Begriffe [DIN17359, S. 15].

Kenngrößen

Die Verlässlichkeit komplexer Systeme muss im Systementwurf modelliert, analysiert und mittels Kenngrößen bewertet werden. Die Verlässlichkeit gliedert sich nach [ALR+04, S. 4ff.] in fünf Kenngrößen: Zuverlässigkeit, Sicherheit, Instandhaltbarkeit, Verfügbarkeit und Integrität:

BERTSCHE definiert die **Zuverlässigkeit** wie folgt:

> *„Zuverlässigkeit ist die Wahrscheinlichkeit dafür, dass ein Produkt während einer definierten Zeitdauer unter gegebenen Funktions- und Umge-*

bungsbedingungen nicht ausfällt.“ [BGJ+09, S. 10]

Zur Bewertung der Zuverlässigkeit können Abschätzungen anhand von Betriebsdaten oder mittels Lebensdauerversuchen an physischen Prototypen durchgeführt werden [Son15, S. 12]. Eine Dichtefunktion f(t) approximiert das Ausfallverhalten und ermittelt den Ausfallzeitpunkt probabilistisch. Kumuliert man die Ausfälle über die Lebensdauer, erhält man die Verteilungsfunktion F(t), auch **Ausfallwahrscheinlichkeit** (vgl. Gleichung 2-1) genannt.

$$F(t) = \int_0^t f(\tau)d\tau$$

Gleichung 2-1: Ausfallwahrscheinlichkeit[BGJ+09, S. 10]

Komplementär steht die **Überlebenswahrscheinlichkeit**. Sie ergibt sich aus der Ausfallwahrscheinlichkeit (vgl. Gleichung 2-2).

$$R(t) = 1 - F(t)$$

Gleichung 2-2: Überlebenswahrscheinlichkeit [BGJ+09, S. 10]

Die **Ausfallrate** λ(t) repräsentiert das Ausfallrisiko einer Systemeinheit. Zur Ermittlung der Ausfallrate wird die Dichtefunktion mit der Überlebenswahrscheinlichkeit ins Verhältnis gesetzt (vgl. Gleichung 2-3).

$$\lambda(t) = \frac{-dR(t)/dt}{R(t)} = \frac{f(t)}{R(t)}$$

Gleichung 2-3: Ausfallrate [BGJ+09, S. 12]

Das Ausfallverhalten der untersuchten Systemeinheit wird über Verteilungsfunktionen angenähert. Die beiden bekanntesten Verteilungsfunktionen sind die Exponentialverteilung und die Weibullverteilung. Die Exponentialverteilung wird häufig zur Bewertung von elektronischen Einheiten verwendet, da mit dieser Verteilungsfunktion Früh- und Zufallsausfälle sehr gut abgebildet werden können. Die Weibullverteilung wird für mechanische Einheiten verwendet, da diese wesentlich durch Verschleißausfälle und Alterungseffekte charakterisiert sind. Für die zweiparametrige Weibullverteilung wird die charakteristische Lebensdauer T und der Formparameter b verwendet. Der Parameter T gibt den Zeitpunkt an, bei dem 63,2% der Einheiten ausgefallen sind. Der Formparameter wird genutzt, um die Weibullverteilung an den Versuchsdaten anzupassen. Bei $b < 1$ sind in dem Versucht

vermehrt Frühausfälle aufgetreten. Die Exponentialverteilung wird mit $b \approx 1$ abgebildet. Verschleiß- oder Alterungsausfälle werden bei $b > 1$ approximiert [Son15, S. 12ff.]. Die Weibullverteilung mit verschiedenen Formparametern zeigt Bild 2-7. Zusätzlich zu diesen

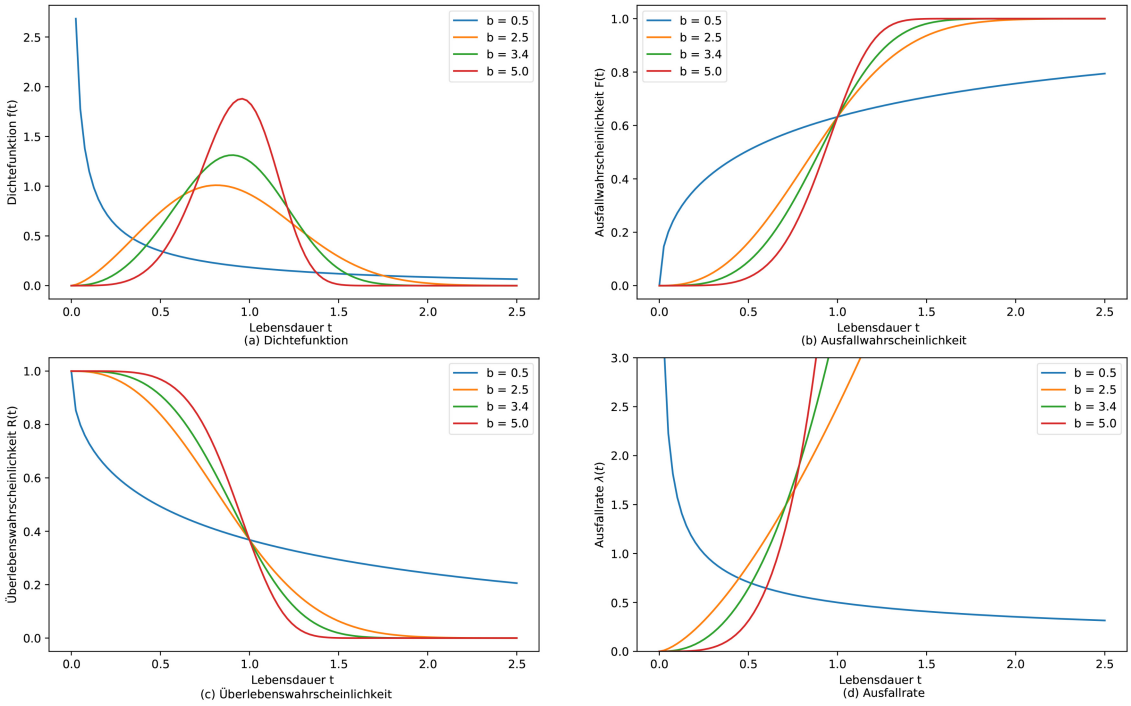


Bild 2-7: Weibullverteilung mit verschiedenen Formparametern

Kenngrößen existieren weitere etablierte statistische Maßzahlen. Handelt es sich zum Beispiel um eine reparierbare Einheit, wird zwischen der mittleren Zeit des Auftreten des ersten Fehlers (*Mean Time To First Failure*, MTTFF) und der mittleren Zeit zwischen Fehlern (*Mean Time Between Failure*, MTBF) unterschieden.

Die zweite Kenngröße ist die **Sicherheit**. Sie beschreibt die Eigenschaft eines Systems, weder Menschen noch die Umwelt oder andere Systeme zu gefährden [DIN17359, S. 9]. Eng mit damit verbunden ist der Begriff Gefahr. Die Gefahr beschreibt einen potenziellen Schaden, der durch eine Beeinträchtigung im System in einem Betriebsszenario auftreten kann. In der IEC 61508 wird Sicherheit als die *„Freiheit von unangemessenen Risiken“* formuliert. Die funktionale Sicherheit beschreibt das Fehlen von unangemessenen Risiken, die durch ein fehlerhaftes Verhalten im System verursacht werden. Die funktionale Sicherheit wird durch das zu erwartende Risiko abgeschätzt. Das Risiko ergibt sich aus dem Schweregrad, der Wahrscheinlichkeit und der Beherrschbarkeit einer Gefahrensituation [IEC61508a, S. 7].

Die ISO/PAS 21448 kategorisiert Betriebsszenarien und Systemverhalten entsprechend Bild 3-18. Ein Betriebsszenario ist bekannt oder unbekannt, das Systemverhalten sicher oder unsicher. So können Autonome Systeme selbst ein unvorhergesehenes Verhalten an den Tag legen, etwa infolge einer Fehlklassifikation oder von unsicheren Modellen. Die Modellunsicherheit wird unterteilt in aleatorische und epistemische Unsicherheit. Die

aleatorische Unsicherheit beschreibt Unsicherheiten infolge natürlicher Variablität. Die *epistemische Unsicherheit* beschreibt Unsicherheit infolge von unvollständigem Wissen [Trö18, S.32f.]. Aus der Kombination von Systemverhalten und Betriebsszenario ergeben sich vier Quadranten.

Anhand der Matrix werden verschiedene Maßnahmen zur Steigerung der Resilienz und Verlässlichkeit verortet. Ziel der Entwicklungsaktivitäten ist die Maximierung des nominalen Verhaltens, d.h. sicheres Systemverhalten in einem bekannten Betriebsszenario sowie die Minimierung der unbekannten und unsicheren Anteile [ISO 21448, S. 7]. Dies erfolgt zum Beispiel durch Mittel der Prävention, d.h. frühzeitige Risikoanalyse, integrierte Sicherheitskonzepte und Testen auf Sicherheitsanforderungen in den jeweiligen Betriebsszenarien. Die Resilienz ist nun ein Ansatz, um mit unbekannte Betriebsszenarien und unsicheren Systemverhalten umzugehen. Es umfasst dabei Maßnahmen zur Fehlertoleranz und zur Robustheit.

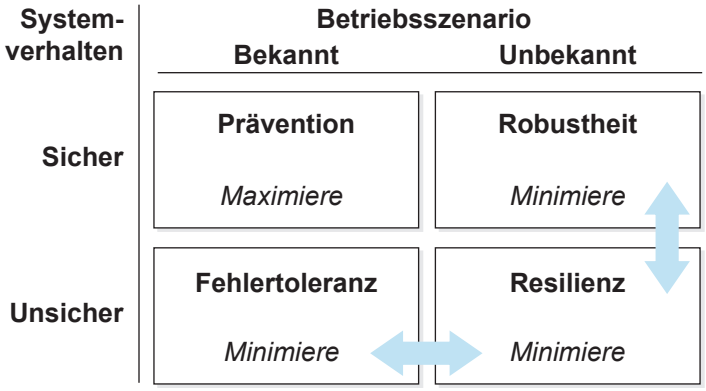


Bild 2-8: Kategorisierung von Betriebsszenarien, nach [ISO 21448, S. 7]

Die **Instandhaltbarkeit** ist die dritte Kenngröße und ist in der DIN 17359 definiert als

> *„Fähigkeit einer Einheit, in dem Zustand erhalten zu werden oder in diesem zurückversetzt zu werden, in dem es eine geforderte Funktion erfüllen kann.“ [DIN17359, S. 40]*

Die **Verfügbarkeit** kombiniert die Kenngrößen Zuverlässigkeit und Instandhaltbarkeit. Bei nicht reparierbaren Systemen entspricht die Verfügbarkeit der Zuverlässigkeit des Systems. BIROLINI definiert die Kenngröße wie folgt:

> *„Verfügbarkeit ist die Wahrscheinlichkeit dafür, dass die Betrachtungseinheit zu einem bestimmten Zeitpunkt die geforderte Funktion unter vorgegebenen Arbeitsbedingungen ausführt.“ [Bir91, S.7]*

Die fünfte Kenngröße ist die **Integrität**. AVIZIENIS bezeichnet diese als die Abwesenheit von unzulässigen Veränderungen des Systemzustandes [ALR+04, S. 14]. Die Definition bezieht sich auf die Manipulation oder die unzulässige Löschung von Daten.

Mittel

Die Mittel zur Steigerung der Verlässlichkeit sind: Fehlerverhinderung, Fehlertoleranz, Fehlerbeseitigung und die Fehlervorhersage [ALR+04, S. 15ff.].

Die **Fehlerverhinderung** bezieht sich auf einen fähigen Entwicklungsprozess und dem begleitenden Qualitäts- und Zuverlässigkeitsmanagement. Hinsichtlich der Systemgestaltung können darüber hinaus kritische Einheiten überdimensioniert oder weniger störanfällige Einheiten ausgewählt werden [Son15, S. 18].

Die **Fehlertoleranz** beinhaltet Methoden zur Überwachung und Fehlererkennung, sowie Methoden zur Aufrechterhaltung oder Wiederherstellung der Funktionalität [ALR+04, S. 15]. In den Bereich der Fehlertoleranz fallen auch Redundanzkonzepte. Dazu gehören funktionale, diversitäre, strukturelle, und analytische Redundanz. Das Ziel der Fehlertoleranz ist die Verhinderung des Ausfall des übergeordneten Systems. Eine technische Rückfallebene dient dazu, das System bei einem Ausfall in einem verfügbaren Zustand zu halten und katastrophale Auswirkungen abzuwenden. Hierzu zählen die Strategien Fail-Operational, Fail-Soft und Fail-Safe. Unter Fail-Operational fallen die Redundanzkonzepte. Fail-Soft bedeutet eine verringerte Funktionsfähigkeit. Bei Fail-Safe wird ein definierter sicherer Zustand eingenommen [Son15, S. 18].

Methoden zur **Fehlerbeseitigung** finden bei der Absicherung technischer Systeme ihre Anwendung [ALR+04, S. 17]. Dazu werden verschiedene Test- und Simulationsverfahren eingesetzt. Ein wichtiges Verfahren hierbei ist die Fehlerinjektion, d.h. das kontrollierte Einbringen von Fehlern zum Testen des Systemverhaltens. Weitere Verfahren sind erweiterte Funktionstests oder Worst-Case-Analysen [IEC61508b]. Während der Betriebsphase werden Ausfälle durch entsprechende Instandhaltung beseitigt [Son15, S. 56ff.].

Die **Fehlervorhersage** bezeichnet die Vorhersage des Verhaltens im Fehlerfall [ALR+04, S. 19]. Zur Abbildung und Bewertung werden Methoden zur Fehlerbeschreibung und -propagierung verwendet. Die Bewertung erfolgt qualitativ oder quantitativ [Lap92].

2.3.2 Beeinträchtigungen

Mit dem Wandel von mechatronischen Systemen zu autonomen Systemen geht ein wesentlicher Nutzenzuwachs in den verschiedenen Anwendungsbereichen einher. Damit verbunden steigt die Komplexität der Systeme. Es treten zahlreiche Beeinträchtigungen mit Auswirkungen auf die Verlässlichkeit, Security und Langzeiteffekte in den Kernkomponenten autonomer Systeme auf. Das Bild 2-9 zeigt mögliche Beeinträchtigungen. So kann zum Beispiel die Funktion der Sensorik durch Reflexion und Interferenz beeinträchtigt sein. Die Beispiele basieren auf einer Literaturrecherche: RATASICH ET AL. analysieren mögliche Beeinträchtigungen in Cyber-Physischen Systemen [RKG+18]. AMODEI ET AL. zeigen Sicherheitsprobleme in KI-basierten Systemen auf [AOS+16]. PETIT ET AL. erläutern potentielle Cyber-Angriffe auf vernetze Fahrzeugsysteme [PS15]. Weitere Beispiele geben LAPRIE ET AL. [Lap92], GOODFELLOW ET AL.[GPH+19-ol] und BEHZADAN ET AL. [BM17]. Eine Herausforderung liegt darin, diese Beeinträchtigungen und ihre Varianten in

der Entwicklung zu antizipieren und geeignete Strategien zur Steigerung der Verlässlichkeit und Resilienz zu integrieren.

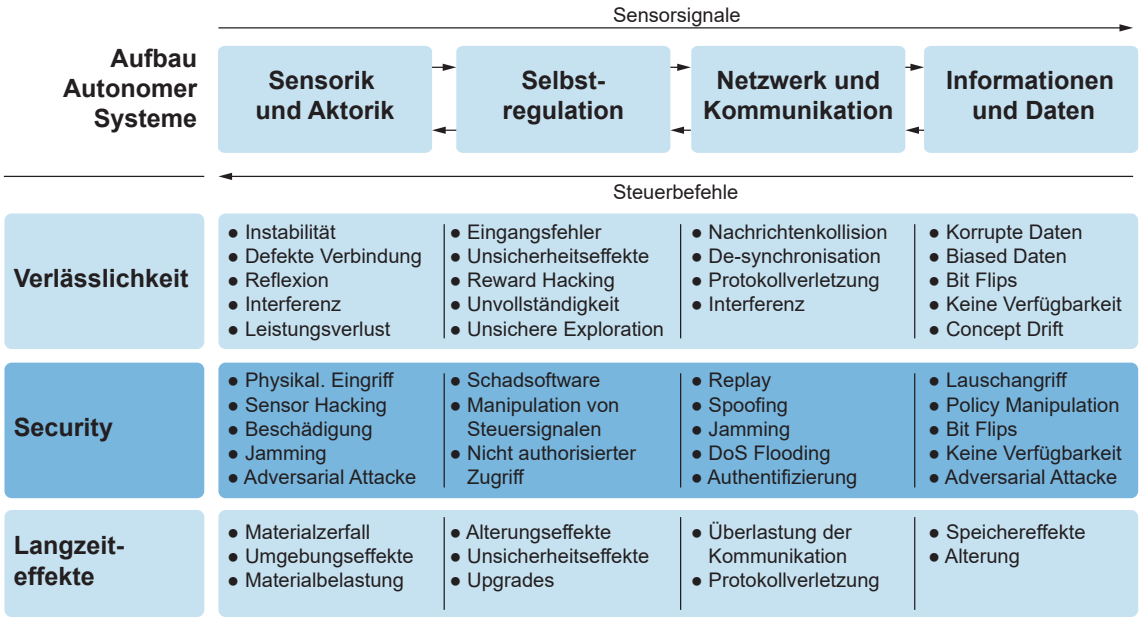


Bild 2-9: Beeinträchtigung in Autonomen Systemen, in Anlehnung an [RKG+18, S. 5]

2.3.3 Strategien zur Steigerung der Verlässlichkeit

Ziel der Verlässlichkeit ist die Steigerung der Verfügbarkeit und Sicherheit technischer Systeme. Zur Steigerung der Verfügbarkeit unterscheiden LEE AT AL. Strategien der Instandhaltung [LWK+08, S. 2ff.] (vgl. Bild 2-10). Die **ungeplante Instandhaltung** ersetzt korrektiv eine ausgefallene Einheit, um schwerwiegendere Folgeausfälle (Notfall-Instandhaltung) zu vermeiden. Die **präventive Instandhaltung** dient der Vermeidung von Ausfällen durch eine geplante Instandhaltung. Dabei lassen sich die periodische und alterungsbedingte Instandhaltung unterscheiden. Die periodische Instandhaltung geht zum Beispiel von einem festen Wartungsintervall aus (z.B. Wartungs- und Serviceintervalle beim Fahrzeug). Die alterungsbedingte Instandhaltung basiert auf der ermittelten Lebensdauer einer Einheit. Die **prädiktive Instandhaltung** untergliedert sich in die zuverlässigkeitszentrierte Instandhaltung (*reliability centered maintenance*, RCM) und in die zustandsorientierte Instandhaltung (*condition based maintenance*, CBM). Bei der CBM werden Systemparameter wie z.B. Schwingung, Drehzahl, Temperatur überwacht, um den aktuellen Systemzustand zu bewerten und den Zustandsverlauf der Einheit zu prognostizieren. Die CBM setzt sich aus der Zustandsüberwachung und der Prognose zusammen [Son15, S. 19 ff.].

Ein Aspekt der **Zustandsüberwachung** ist die Festlegung von Schwellwerten. Die Festlegung erfolgt häufig a priori durch Domänenexperten. Die Überprüfung der Schwellwertüberschreitung erfolgt durch den Vergleich mit den gemessenen Zustandsgrößen. YAM ET AL. beschreiben die Ausprägung der Schwellwerte graphisch [YTL+01]. In Bild 2-11 wird

der Verlauf einer Zustandsgröße zur Betriebszeit exemplarisch dargestellt. Es werden vier Systemzustände definiert: Normaler Zustand, Degradierter Zustand, Warnender Zustand und Gefährlicher Zustand. Diese sind durch entsprechende Schwellwerte voneinander abgegrenzt. Jedem Systemzustand wird die entsprechende Instandhaltungsmaßnahme zugeordnet. Ein Alarm deutet auf eine notwendige Instandhaltung hin. Wird dies nicht berücksichtigt, ist eine Notfall-Instandhaltung erforderlich. Für die Prognose der Systemparameter bieten sich dynamische Kenngrößen an, da diese einen Trend hinsichtlich des zukünftigen Zustandes ableiten lassen. Eine dynamische Kenngröße ist zum Beispiel die Änderungsrate eines Zustandsgröße [Son15, S. 21f.].

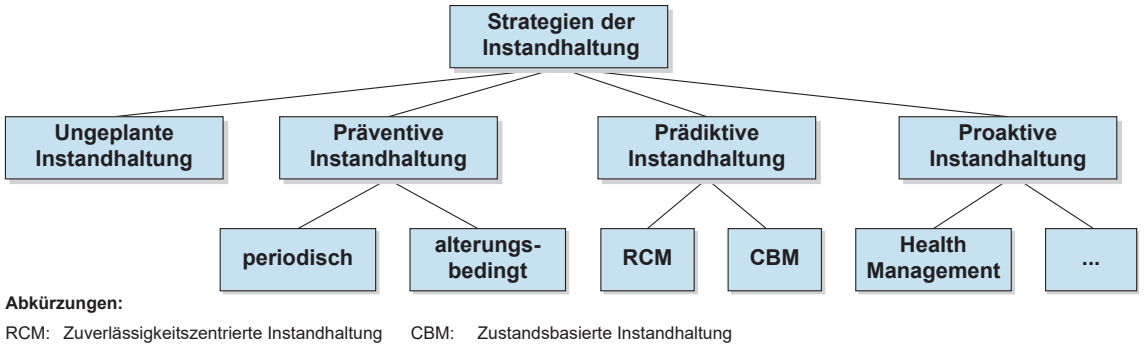


Bild 2-10: Strategien der Instandhaltung [LWK+08, S. 2]

Mit Überschreiten der Schwellwerte wird eine **Diagnostik** durchgeführt. Die Diagnostik umfasst Aktivitäten zur Identifikation, zur Lokalisierung und zur Quantifizierung von Beeinträchtigungen [DIN17359, S. 6]. Auf dieser Basis werden die notwendigen Maßnahmen zur Instandhaltung ermittelt. Bei der Diagnostik werden die relevanten Zustandsgrößen aufbereitet und interpretiert. Zur Interpretation dienen z.B. mathematische Fehlermodelle. Die ISO 13381 teilt diese Fehlermodelle in vier Klassen: Verhaltensbasierte Modelle, Statistische Modelle, Probabilistische Modelle und Neuronale Netze. Verhaltensbasierte Modelle nutzen physikalische Gesetzmäßigkeiten zur Detektion des Fehlers. Statistische Modelle nutzen Daten über die Fehlerhäufigkeiten in der Vergangenheit. Probabilistische Modelle und Neuronale Netze schätzen die Auftrittsswahrscheinlichkeiten weiterer Fehler auf Basis der detektierten Fehler [ISO13381, S. 12f.].

Gegenstand der **Prognose** ist die Analyse des Fortschritts vorhandener Schäden, des Auftretens und Fortschritts neuer Schäden sowie der zu erwartender Ausfälle. Die ISO 13381 beschreibt ein generisches Vorgehen zur Prognose von Maschinendaten. Das Vorgehen besteht aus vier Schritten: Vorverarbeitung der Messwerte, Prognose auf Basis der bisherigen Fehlzustände, Prognose auf Basis zukünftiger Fehlzustände und Gegenmaßnahmen [ISO13381, S. 13ff.]. Die Prognose kann durch eine Extrapolation oder einer Projektion erfolgen. Die Extrapolation nutzt historische Daten, um die Zeit bis zum Ausfall der Einheit zu ermitteln. Die Projektion schätzt den zukünftigen Systemzustand und durch eine Kurve approximiert [Son15]. Dies erfolgt zum Beispiel durch eine Schätzung der Restlebensdauer (engl. Remain Useful Life) oder eine Abnutzungsprognose der betrachteten Einheit [DIN17359, S. 36]. Hierzu werden Schadensakkumulationshypothesen und Degradations-

modelle verwendet, welche die Auswirkung der bisherigen Belastung berücksichtigen. Degradationsmodelle lassen sich einteilen in: Modellbasiert, datenbasiert, wissensbasiert oder erfahrungsbasiert [Son15, S. 21]. Auf der Basis dieser Modelle können die Instandshaltungsstrategien optimiert werden. In Verbindung mit dem Ersatzteilmanagement und dem Autragsbestand können so Wartungsintervalle und Bedarfsforderungen der Maschine terminiert werden.

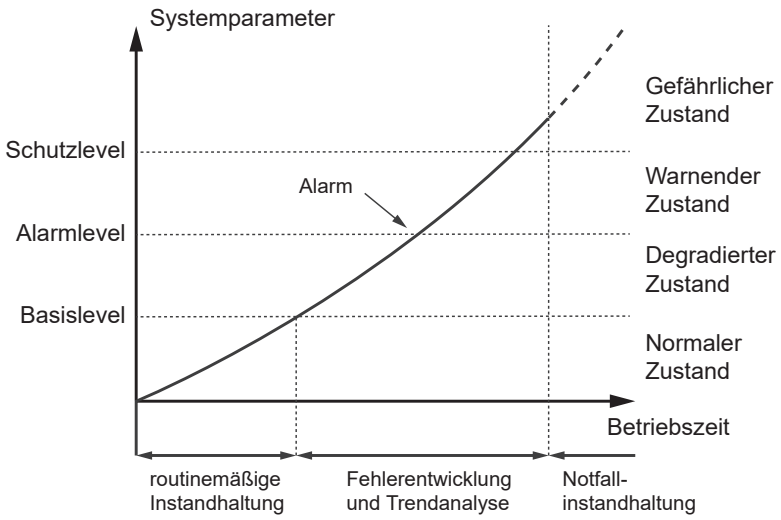


Bild 2-11: Zustandsdefinition [YTL+01, S. 389]

Das Bild 2-12 visualisiert die Entwicklung konvergierender Technologiestränge technischer Systeme. Es verdeutlicht, wie sich mechatronische Systeme durch die Integration von Künstlicher Intelligenz, Interoperabilität und Konnektivität sowie der Verlässlichkeit und des Computings zu resilienten technischen Systemen entwickeln [RKG+18]. Eng mit der Instandhaltung verknüpft ist das **Health Management**. Es stellt verschiedene Verfahren der Diagnose und Prognose bereit und ermöglicht eine Bewertung des aktuellen und zukünftigen Systemzustandes [Lee10, S. 2], [Son15, S. 19]. Das Health Management führt im Betrieb Maßnahmen zur Steigerung der Verfügbarkeit durch. Als Grundlage dient die funktionale Redundanz und fortgeschrittene Regelstrategien [Son15, S. 23].

Ausgehend vom Health Management gibt LEE einen Ausblick auf zukünftige Strategien der Verlässlichkeit. Eine zunehmende Bedeutung besitzt die **Resilienz**. Resiliente Systeme widerstehen Störungen und können sich nach einer Auslenkung so anpassen, dass wieder eine stabile Funktion oder Struktur erreicht wird. Ein resilientes System muss somit robust und gleichzeitig in der Lage sein, sich im Schadensfall selbst zu re-initialisieren [Aca13, S. 105].

Künstliche Immunsysteme gehen deutlich weiter. Künstliche Immunsysteme adaptieren die Funktionsweise des biologischen Immunsystems und übertragen es auf technische Systeme. Als Grundsatz sieht LEE die Self-X-Fähigkeiten: Selbstkonfiguration Selbstoptimierung, Selbstheilung und Selbstschutz. Selbstheilung bezieht sich dabei auf das eigenständige Erkennen und Beheben von Fehlzuständen [Lee10, S. 2].

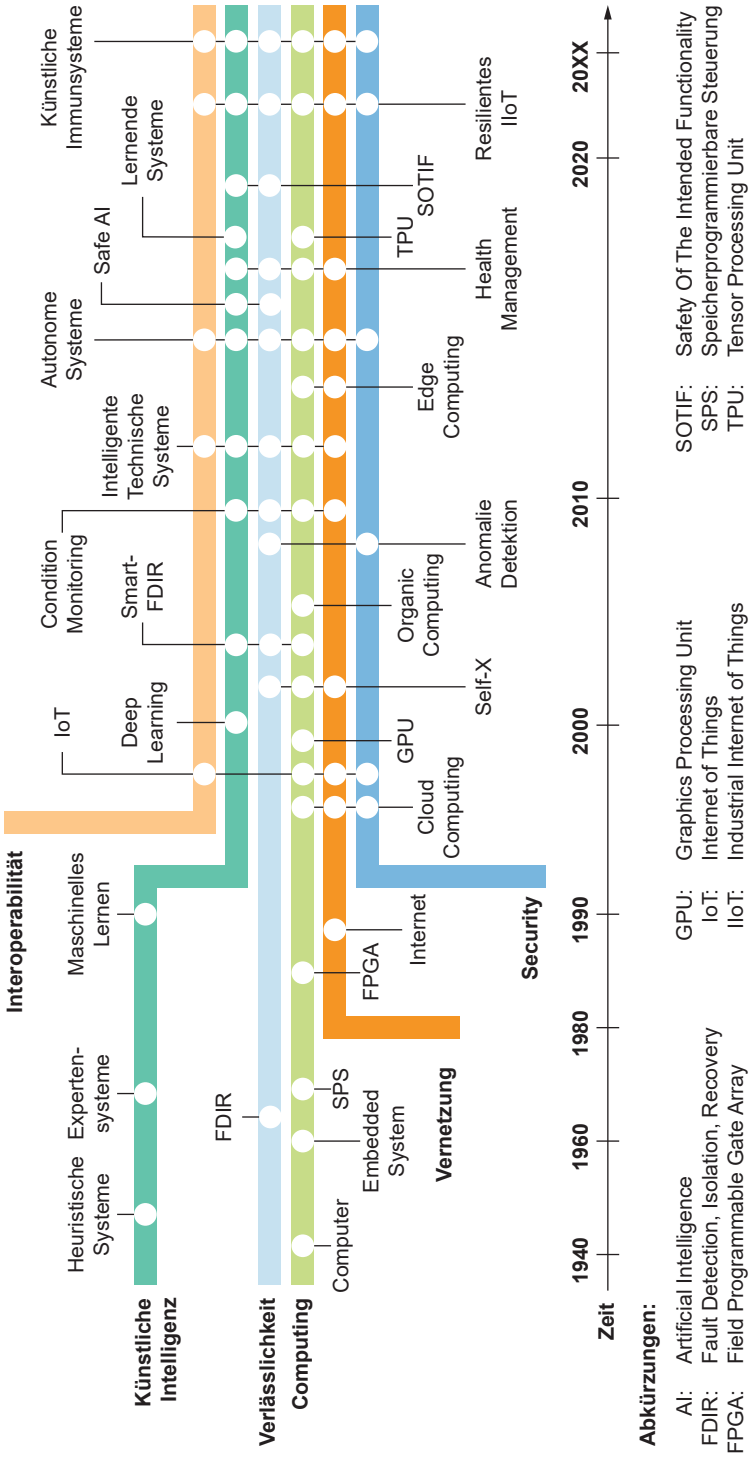


Bild 2-12: Roadmap zu resilienten Systemen, in Anlehnung an [RKG+18, S. 14]

2.3.4 Selbstheilung

Der Wandel von der Mechatronik zu Intelligenten Technischen Systemen führt zu einer Zunahme der Komplexität. Hinzu kommt die verstärkte Einbettung und Vernetzung von sicherheitskritischen Teilsystemen (z.B. Vernetztes Fahren) und die offene Betriebsumgebung. Diese Komplexität stellt in Verbindung mit der Sicherstellung der Verlässlichkeit

eine Herausforderungen für die Technologie und für die Entwicklung dieser Systeme dar. Ein Mittel zur Beherrschung dieser Komplexität sind Self-X-Fähigkeiten: Selbstkonfigurierend, selbst-optimierend, selbst-heilend, selbst-erklärend und selbst-schützend. Der Begriff und das Konzept der Selbstheilung wird in der Literatur und in den Domänen unterschiedlich verwendet [PS11, S. 45]. Eine Übersicht zu den Ansätzen in den Domänen wie den Materialwissenschaften, der Elektronik und Informatik oder der Robotik liefert FREI ET AL. [FMD+13]. Im Leitprojekt Autonomic Computing wurde der Begriff von IBM geprägt und definiert:

> *„Can discover, diagnose and react to disruptions. Self-healing components can detect system malfunctions and initiate policy-based corrective action without disrupting the [...] environment. Corrective action could involve a product altering its own state or effecting changes in other componentes in the environment. The [...] system as a whole becomes more resilient because day-to-day operations are less likely to fail.“ [IBM05, S. 5]*

Systeme mit dieser Fähigkeit erkennen Störungen eigenständig, diagnostizieren ihre Ursache und führen selbstständig Maßnahmen durch. Ziel ist die Steigerung der Verfügbarkeit und Sicherheit komplexer Systeme durch Selbstorganisation. In dem DFG Schwerpunktprogramm (SPP) 1183 Organic Computing wurde der Anwendungsbereich deutlich erweitert. Dabei wurden biologische Mechanismen der Selbstorganisation und Emergenz untersucht und für das Computing adaptiert. Eine neuere Definition findet sich bei GHOSH ET AL.:

> *„[...] a self-healing system should recover from the abnormal (or unhealthy) state and return to the normative (healthy) state, and function as it was prior to disruption.“ [GSR+07, S. 2167]*

PSAIER ET AL. definieren einen generischen Selbstheilungsprozess für technische Systeme. Dieser Prozess besteht aus den Schritten: Detektion, Diagnose und Wiederherstellung. Mit der Detektion (*detecting*) wird der Betriebszustand kontinuierlich überwacht und Beeinträchtigungen erkannt. Die Diagnostik (*diagnosing*) ermittelt mögliche Ursachen der Beeinträchtigung. Ebenso wird ein Plan zur Wiederherstellung des normalen Betriebszustandes abgeleitet. Die Wiederherstellung (*recovery*) führt den Plan aus [PS11, S. 45].

Bild 2-13: Prozess der Selbstheilung nach [PS11, S. 45]

Eine Herausforderung der Selbstheilung ist die Erkennung von Anomalien im Systemverhalten. Anomalien sind Indikatoren für mögliche Fehlzustände und können z.B. als Folge von Verschleiß, unbekannten Störungen oder aufgrund einer gewollten Verhaltensanpassung auftreten [UTY13]. Es ist notwendig diese Anomalien zu unterscheiden. Die Anomalie Detektion ermöglicht so die Erkennung und Diagnose von Beeinträchtigungen bevor sie im System propagieren [Sta15].

2.3.5 Anomalie Detektion

Die Anomalie Detektion ist ein interdisziplinäres Forschungsfeld. Es kombiniert Data Mining, Maschinelles Lernen und Statistik. Anwendungen der Anomalie Detektion finden sich u.a. in der Eindringungserkennung (*Intrusion Detection*) in Netzwerken oder in der Schadenserkennung (*Damage Detection*) [CBK09, S. 5]. TAN ET AL. beschreiben die Anomalie Detektion als das Problem, Beobachtungen aus Daten zu extrahieren, deren Charakter signifikant von den restlichen Beobachtungen abweicht [TSK14]. Nach CHANDOLA ET AL. beschreibt die Anomalie Detektion *„die Fähigkeit, Abweichungen von normalen Verhalten zu detektieren“*[CBK09, S. 2]. Verwandte Begriffe sind die Novelity Detection oder die Outlier Detection [CBK09], [Agg13], [HA14].

Entsprechend der Definition nach CHANDOLA ET AL. basiert die Anomalie Detektion auf:

- Lernen von normalen Verhaltensprofilen,
- Beobachtungen der Signalverläufe, sowie der
- Identifikation und Bewertung von Abweichungen.

Bei der Anomalie Detektion sind verschiedene Aspekte zu berücksichtigen. Hierzu zählen die Art der Beobachtung, die Gestalt der Anomalie und das Ergebnis der Detektion.

Die **Art der Beobachtung** charakterisiert die Daten und Signale. Jede Beobachtung enthält Datenpunkte (Merkmale, Variablen). Diese Daten können kategorische oder numerisch und diskret oder kontinuierliche Daten sein. AGGARWAL unterscheidet fünf Kategorien von Daten. Dies sind: Kategorische Daten und Textdaten (z.B. Namen, URL), Zeitreihen (z.B. Aktienkurs) und Mehrdimensionale Streaming-Daten (z.B. Videos), Diskrete Sequenzen (z.B. Aufrufsequenzen von Tasks), Räumliche Daten (z.B. Verkehrsdaten) sowie Graph- und Netzwerk-basierte Daten (z.B. Internet-Seiten) [Agg13, S. 2]. Ein Zeitreihen-Signal wird unterteilt in periodisch (z.B. Schwingungen), stochastisch (z.B. Sprachsignal) oder quasi-linear (z.B. Temperatur).

Hinsichtlich der Anomalie Detektion ist die Auflösung, Filterung und die Veränderung der Eigenschaften eines Signals relevant. Zur Detektion einer Anomalie ist die korrekte Auflösung und Abtastung eines Signals festzulegen. Beispielsweise ob die Anomalie in Sekunden, Stunden, Tagen etc. auftreten kann [CBK09]. Ebenso sind Anomalien vom Rauschen (engl. noise) abzugrenzen. Ein Ansatz zur Unterscheidung ist die Filterung (engl. noise removal) des Signals sowie eine Bewertung der Signifikanz der Anomalie [Agg13, S. 3]. Die Eigenschaften eines Signals und damit die Definition einer Anomalie können sich über die Zeit, mit dem Systemzustand oder mit der Umgebung verändern. Dies erfordert eine Überwachung der Eigenschaften und die Fähigkeit zur Online Adaption der Anomalie Detektion. Um dies in der Detektion und dem zugrunde liegenden Modell zu berücksichtigen, werden in GAMA ET AL. Ansätze zum Concept Drift erläutert [GZB+14, S. 5ff.].

Bild 2-14 visualisiert verschiedene **Arten von Anomalien**. Eine **Punktanomalie** tritt auf, wenn eine Beobachtung signifikant von den anderen Beobachtungen abweicht. Die Punktanomalie ist in dem Datensatz oben links in der Grafik abgebildet. Die **Kollektive Anomalie** beschreibt ein Muster, bei dem eine Gruppe von Beobachtungen im Vergleich zu den restlichen periodischen Beobachtungen anormal ist. Der untere Teil der Grafik verdeutlicht dies am Beispiel einer EKG-Messung. Zur Detektion dieser Anomalie ist es erforderlich, die Beobachtungswerte innerhalb eines Zeitfenster zu analysieren. Neben der Veränderung eines Wertes kann die Anomalie in dem Signal auch durch eine Veränderung des Verlaufes, durch einen Drift oder durch eine Veränderung der Periodendauer charakterisiert sein. **Kontextuelle Anomalien** treten auf, wenn die Beobachtungen in einem spezifischen räumlichen, zeitlichen oder logischen Kontext anormal sind. Dies verdeutlicht die Temperaturmessung in der Grafik oben rechts. So stellt ein Temperatursturz im Sommer eine Anomalie dar. Die Definition des Kontextes und des normalen Verhalten in diesem Kontext ist daher Domänenwissen [CBK09, S. 7], [Agg13].

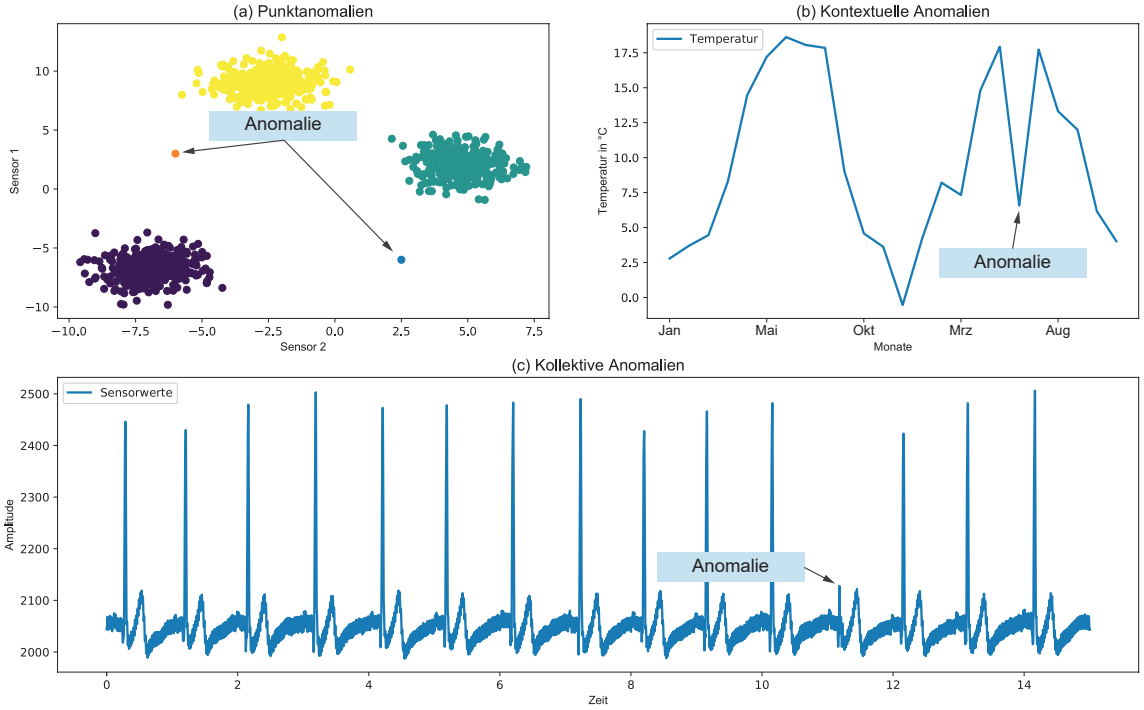


Bild 2-14: Arten von Anomalien (Auswahl) (vgl. [CBK09, S.7ff.])

Neben diesen Arten der Anomalien dient das Lernen von Zusammenhängen zwischen den Signalen dazu, anormales Verhalten zu lokalisieren und zu interpretieren. Dies kann auf verschiedenen Wegen erfolgen: Zum einen können *anormale Ähnlichkeiten* gelernt werden, d.h. je häufiger in verschiedenen Signalen zur selben Zeit Anomalien detektiert werden, desto wahrscheinlicher ist es, dass diese zusammenhängen. Beispielsweise verändern sich die Körpertemperatur, pH-Wert, Herzschlagfrequenz etc. bei einer Krankheit. Zum anderen können *normale Ähnlichkeiten* bspw. bei Schlaf, Sport, Arbeit z.B. durch Korrelationen erlernt werden.

Das **Ergebnis einer Anomalie Detektion** kann ein *binärer Wert* als Label der Beobachtung sein, z.B. {normal, anormal} oder ein *Anomalie-Score* sein, der weitere *Kontextinformationen* über die Art und Dauer der Anomalie beinhaltet. Als Score dient etwa die Signifikanz als die Distanz der Beobachtung zum normalen Systemverhalten (z.B. Distanz zum Cluster). Eine Herausforderung liegt darin - insb. bei höheren Dimensionen - die Entscheidungsgrenze für diese Bewertung festzulegen.

Die **Detektionsverfahren** unterteilen sich in drei Kategorien [CBK09, S. 10]:

- **Überwachte Verfahren**: Dieser Ansatz setzt voraus, dass die Trainingsdaten jede mögliche Anomalie repräsentieren. In diesem Fall handelt es sich um ein Klassifikationsproblem mit zwei oder mehr Klassen. Jeder Beobachtungswert in den Signalen wird einer dieser Klassen zugeordnet.
- **Semi-Überwachte Verfahren**: Bei diesem Ansatz wird das Normalverhalten des Systems anhand der Trainingsdaten gelernt und durch ein Modell abgebildet. Das

Modell wird anschließend genutzt, um Abweichungen in den Signalen zu detektieren. Eine Validierung erfolgt durch die Abbildung von Anomalien in den Testdaten.

- **Unüberwachte Verfahren**: Dieser Ansatz erfordert keine gekennzeichneten Trainingsdaten. Die wesentliche Annahmen bei diesem Ansatz ist, dass das Verhalten des Systems im wesentlich normal ist und nur wenige Anomalien auftreten.

2.4 Datengetriebener Systementwurf

2.4.1 Systems Engineering

Die Realisierung von Intelligenten Technischen Systemen setzt das synergetische Zusammenwirken von verschiedenen Fachdisziplinen voraus. Dabei müssen zahlreiche Entwicklungsaspekte wie Funktionalität, Verhalten, Benutzungsfreundlichkeit, Herstellbarkeit oder Verlässlichkeit interdisziplinär betrachtet werden. Die integrative und kontinuierliche Absicherung dieser Aspekte im Systementwurf stellt sicher, dass die tatsächlichen Eigenschaften des Systems mit den geforderten Eigenschaften übereinstimmen und das System im Betrieb verlässlich funktioniert. Dies erfordert einen durchgängigen und übergreifenden Entwicklungsansatz sowie integrierte Entwicklungswerkzeuge zur Simulation, Planung und Entwurf, Konfiguration und zur Visualisierung [DWF18], [DGS+18].

Ein Ansatz der diesen Herausforderungen gerecht wird, ist das Systems Engineering. Systems Engineering hat den Anspruch, die beteiligten Disziplinen und vielfältigen Entwicklungsaspekte zu integrieren und die beteiligten Akteure zu orchestrieren. Der Ansatz hat seinen Ursprung in der Systemtheorie und entwickelt sich stetig weiter [DWF18]. Das International Council on Systems Engineering (INCOSE) definiert Systems Engineering als:

> *„Systems Engineering ist ein interdisziplinärer Ansatz und stellt die Methoden und Prozesse für die Realisierung von erfolgreichen Systemen zur Verfügung. Es ist auf die frühzeitige Klärung der Kundenbedürfnisse und der erforderlichen Funktionalität ausgerichtet und fordert die Dokumentation sämtlicher Anforderungen, bevor mit dem Entwurf und der Validierung eine ganzheitliche Lösung erarbeitet wird[...]. Systems Engineering berücksichtigt gleichermaßen die geschäftlichen und technischen Anforderungen mit dem Ziel der Erzeugung eines Qualitätsprodukts, das auf die Nutzerbedürfnisse zugeschnitten ist.“[INC18-olb]*

Das V-Modell repräsentiert ein verbreitetes Vorgehensmodell zur Gestaltung komplexer technischer Systeme als *Top-Down/ Bottom-Up*-Makroprozess (Bild 2-15) [VDI2206, S. 29]. Ausgangspunkt bildet die strategische Produktplanung. Aus dieser resultiert, unter Berücksichtigung der Marktanforderungen und zum Beispiel regulativer Vorgaben, ein Entwicklungsauftrag. Im Rahmen der Anforderungsanalyse wird die Aufgabenstellung präzisiert und in Form von Anforderungen dokumentiert. Die Anforderungen bilden den Maßstab, anhand dessen das zu entwickelnde System zu bewerten ist.

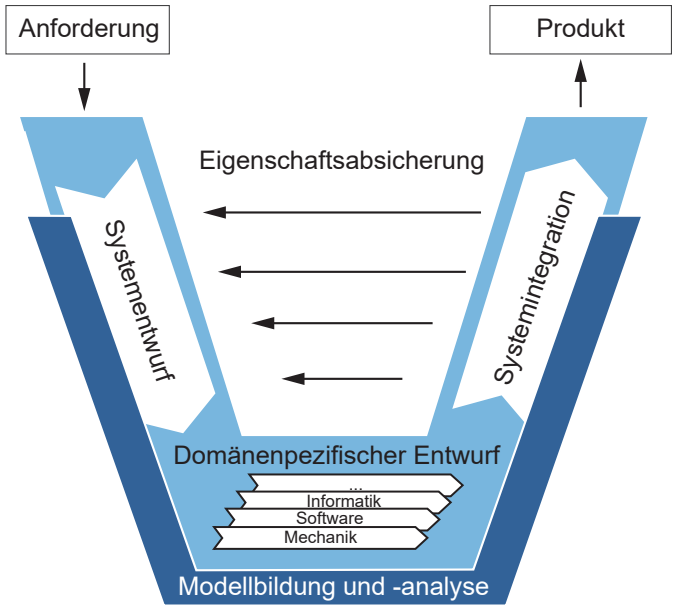


Bild 2-15: V-Modell [VDI2206, S. 29]

Die nächste Phase ist der **Systementwurf**. Das Ziel dieser Phase ist ein disziplinübergreifendes Lösungskonzept, das die wesentlichen physikalischen und logischen Wirkungsweisen des Produkts beschreibt. Hierzu wird die Gesamtfunktion des Produkts in wesentliche Teilfunktionen zerlegt. Diesen Teilfunktionen werden geeignete Wirkprinzipien bzw. Lösungselemente zugeordnet. Auf der Basis des disziplinübergreifenden Lösungskonzepts erfolgt in der **disziplinspezifischen Ausarbeitung** die Konkretisierung. Detaillierte Auslegungen, Berechnungen und Simulationen sind notwendig, um bei kritischen Funktionen die Funktionserfüllung sicherzustellen. Bei der **Systemintegration** werden die einzelnen Entwicklungsartefakte zu einem Gesamtsystem integriert und das Zusammenwirken untersucht. Die **Eigenschaftsabsicherung** dient zur fortlaufenden Überprüfung des Produktes anhand der Anforderungen. Dabei ist sicherzustellen, dass die tatsächlichen Systemeigenschaften mit den geforderten Eigenschaften übereinstimmen [GDE+19], [VDI2206].

Die beschriebenen Phasen werden durch eine **Modellbildung und -analyse** unterstützt. Mit Hilfe von Modellen und rechnergestützten Werkzeugen zur Simulation werden die Systemeigenschaften abgebildet und untersucht.

Das Ergebnis des Makroprozesses ist eine Produkt. Dabei wird unter Marktleistung nicht ausschließlich das fertige, real existierende Erzeugnis verstanden, sondern die zunehmende Konkretisierung des zukünftiges Produkts entlang verschiedener Reifegrade wie z.B. das Labormuster, das Funktionsmuster oder die Vorserie [VDI2206, S. 31].

2.4.2 Data-Driven Systems Engineering

Mit dem Wandel von mechatronischen Systeme hin zu Intelligenten Technischen Systemen und der Digitalisierung der Wertschöpfungsprozesse ändert sich auch das Systems Engineering. Kern dieses Wandels ist die Nutzung von Daten und die Wissensextraktion für

die Produktentwicklung. So werden in der Produktentstehung und im Betrieb eine große Menge an Daten generiert (z.B. Simulations- und Maschinendaten) [Die14]. Dies führt zur einer konsequenten Weiterentwicklung des Model-Based Systems Engineering (MBSE) hin zum Data-Driven Systems Engineering.

Model-Based Systems Engineering

Das MBSE beschreibt die Marktleistung auf Basis von durchgängigen Modellen, von dem Systementwurf über den gesamten Produktlebenszyklus. Das INCOSE definiert den Begriff Model-Based Systems Engineering wie folgt:

> *„The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.“ [INC18-ola]*

Modell und Sicht

Ein Modell ist im Allgemeinen eine Abbildung eines Originals. Es dient zur Strukturierung und zum besseren Verständnis. Ebenso können mit Hilfe von Modellen Experimente durchgeführt werden, die nicht sinnvoll am Original durchgeführt werden könnten [GTS14]. Dabei werden zahlreiche Vereinfachungen und Abstraktionen verwendet, um die Komplexität des Originals zu verringern. Modelle werden für einen bestimmten Zweck erstellt und abstrahieren Eigenschaften, die für den Zweck nicht relevant sind. Ganz allgemein definiert STACHOWIAK und ROPOHL den Begriff Modell wie folgt:

> *„Ein Modell ist eine beschränkte Repräsentation (Abstraktion, Verkürzung) von Entitäten und Beziehungen der wirklichen Welt (Abbildung) mit einer eindeutigen Korrespondenz für ein festes Zeitintervall und für einen vorher festgelegten Zweck (pragmatisches Merkmal).“[Rop12]*

Entscheidend für die Interpretation des Modells ist, dass diejenigen Eigenschaften des Originals, die für den Zweck relevant sind, möglichst gut im Modell abgebildet werden. Alle Operationen und Experimente, die auf einem Modell ausgeführt werden, müssen ebenso auf dem Original ausführbar sein und zu gleichen oder ähnlichen Verhalten führen. Nur wenn dies der Fall ist, handelt es sich um ein korrektes Modell und die gewonnen Erkenntnisse sind auf die Realität übertragbar [GTS14].

Charakteristisch für das MBSE ist das Prinzip der Trennung von Modell und Sicht. Eine Sicht ist die Projektion eines Modells. Diese Projektion visualisiert das Modell von einer bestimmten Perspektive oder einem Standpunkt und abstrahiert Aspekte, die für diese Perspektive nicht relevant sind. Das bedeutet, es werden Ausschnitte des Modells in einer Sicht dargestellt. Solche Sichten werden als Diagramm bezeichnet [Kai14], [Alt12]. Ein Standpunkt definiert die Konventionen und die Verwendung von Sichten, um spezifische Anliegen zu formulieren. Ein Anliegen beschreibt ein Interesse an einem System, das für einen oder mehrerer Akteure des Systems relevant sind. Dementsprechend definiert die

IEEE 1471 in Übereinstimmung mit der IEEE 42010 [IEE42010]:

> *„A model usually offers different views in order to serve different purposes. A view is a representation of a system from the perspective of related concerns or issues.“ [ISO1471]*

Beispielsweise sind bei der Entwicklung eines mechatronischen Systems Konstrukteure und Softwarentwickler als Akteure beteiligt. Für den Konstrukteur sind Aspekte wie Bauraum, Wirkflächen oder Kräfte und Belastungen relevant. Sein Standpunkt ist die physikalische Struktur des Systems. Dabei wird weitestgehend von der Informationsverarbeitung abstrahiert. In der Sicht werden die mechanischen Systemelemente und ihre Beziehungen dargestellt. Modelle der Softwaretechnik (z.B. Zustandsautomaten) abstrahieren hingegen von der konkreten Physik [GTS14]. Durch die Trennung von Modell und Sicht kann dasselbe Modellelement mehrfach als Instanz in verschiedenen Sichten für verschiedene Akteure verwendet werden, oder unterschiedliche Aspekte auf unterschiedlichen Sichten herausgearbeitet werden [Alt12].

Data-Driven Systems Engineering

Das Data-Driven Systems Engineering erweitert MBSE um den Bestandteil Data Analytics. Data Analytics beschreibt den Einsatz maschineller Lernverfahren, um aus strukturierten oder unstrukurierten Daten versteckte Muster, unbekannte Korrelationen und andere Informationen zu extrahieren. Durch die Vernetzung von diesen Informationen wird Wissen generiert [Bit18], [Gartner20-ol]. In der Arbeit wird Data-Driven Systems Engineering wie folgt definiert:

> *„The formalized and continuous application of systems modeling and data scienc to support product planning, system requirements, design, analysis, verification and validation activities beginning in the product definition phase and continuing throughout development and later life cycle phases while considering different product generations.“*

Die Anwendung von Data-Driven Systems Engineering erfordert eine abgestimmte Kombination von Sprache, Methode, Datenanalyse und unterstützende Werkzeuge (Bild 2-16):

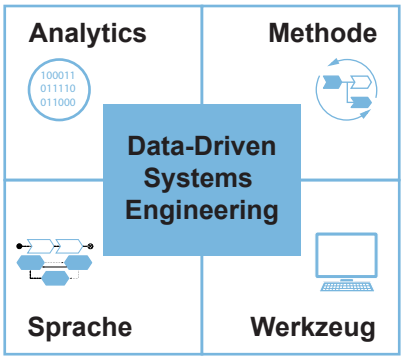


Bild 2-16: Data-Driven Systems Engineering erfordert das Zusammenwirken von Methode, Sprache, Werkzeug und Data Analytics

Die **Modellierungssprache** dient zur Formalisierung der disziplinübergreifenden Inhalte des Spezifikationsmodell. Die Sprache wird durch ihre Syntax (Notation) und Semantik (Bedeutung) definiert. Die **Syntax** wird unterteilt in abstrakte und konkrete Syntax. Die abstrakte Syntax ist die Grammatik der Sprache. Darunter fallen die syntaktischen Elemente und Konstrukte sowie entsprechende Regeln zur Bildung größerer Konstrukte. Die konkrete Syntax beschreibt die Notation der Konstrukte. Die **Semantik** legt die Art der zugelassenen Verknüpfungen zwischen den Konstrukten fest. Die Regeln werden in Form von Bedingungen festgehalten und stellen einen Teil der statischen Semantik dar. Die abstrakte Syntax und die statische Semantik werden in einem Meta-Modell festgelegt. Die Bedeutung der Konstrukte sowie deren Verknüpfung wird in der dynamischen Semantik definiert. Mit der Syntax und Semantik werden Modelle konsistent generiert und zu integriert [GDE+19, S. 407], [Kai14, S. 34f.].

Die Modellierungssprache ist ein reines Ausdrucksmittel. Ein wesentlicher Baustein zur Anwendung von MBSE ist die **Modellierungsmethode**. Die Methode bildet das Bindeglied zwischen Sprache und Vorgehensmodell und legt die Nutzung einer Modellierungssprache fest. Dabei gibt die Methode vor, welche Aspekte zu welchem Zeitpunkt wie detailliert von welchem Akteur zu modellieren sind [GDE+19, S. 412].

Das **Autorenwerkzeug** ermöglicht eine rechnerinterne Abbildung des Spezifikationsmodell und ist die Bedingung für computergestützte Analysen und Auswertungen. Es existieren verschiedene Werkzeugklassen, die sich z.B. durch die jeweils unterstützte Modellierungssprache und ihren Integrationsgrad mit anderen Entwicklungswerkzeugen unterscheiden. Zu nennen sind hier kursorisch Graphikwerkzeuge, Dezidierte Modellierungswerkzeuge (z.B. Cameo, iQuavis, Eclipse Papyrus) oder PDM-Systeme (z.B. 3DEXPERIENCE) von Dassault Systèmes, Siemens Teamcenter, PTC Windchill) [GDE+19, S. 414].

Data Analytics unterstützen die Modellierung des Systems auf Basis von Daten. Werden diese Daten mit Semantik angereichert ergeben sich Informationen. Werden diese Informationen miteinander vernetzt (z.B. Kontext und Erfahrung) ergibt sich das Domänenwissen [NBS16]. So können Simulationsdaten oder Betriebsdaten für die Modellierung und Optimierung maschineller Lernverfahren genutzt werden (z.B. Training von Anomalie Detektoren oder Klassifikatoren) oder zur Identifikation von Anforderungen und Testfällen.

Modellbasierte Systemspezifikation

Die **modellbasierte Systemspezifikation** ermöglicht eine ganzheitliche und disziplinübergreifende Betrachtung des zu entwickelnden Systems. Dabei werden verschiedene Aspekte in einem domänenübergreifenden Modell abgebildet. Die Literatur benennt hierzu übereinstimmend drei Aspekte: Anforderungen, Systemstruktur und das Systemverhalten [Kai14], [Alt12]:

- **Anforderungen** bilden den Kern einer Entwicklungsaufgabe und definieren die Entwicklungsziele sowie die gewünschten Produkteigenschaften aus Kundensicht

[GDE+19, S. 406]. Man unterscheidet funktionale und nicht-funktionale Anforderungen. Funktionale Anforderungen beschreiben das geforderte Verhalten des Systems, bzw. dessen Komponenten; nicht-funktionale Anforderungen definieren die Eigenschaften des Systems wie Verlässlichkeit, Benutzbarkeit, Effizienz [Alt12], [GDE+19]

- Die **Struktur** beschreibt die wesentlichen Elemente eines Systems und wie diese in Beziehung zueinander stehen. Dies umfasst ebenso die Beschreibung der Schnittstellen zwischen den Systemelementen und zur Systemgrenze(vgl. Kap. 2.1). Systemelemente repräsentieren Funktionseinheiten, Softwarekomponenten und Module, bzw. Baugruppen. Die Struktur repräsentiert eine statische Betrachtung des Systems, ohne Berücksichtigung ihrer zeitlichen Veränderung, z.B. Bewegungsverhalten [GDE+19, S. 406], [Alt12], [Kai14, S, 27].
- Das **Verhalten** konkretisiert die funktionalen Anforderungen. Neben den Funktionen erfolgt die (semi-)formale Beschreibungen des Verhaltens. So unterscheidet man das dynamische, zeitliche oder statische Verhalten des Systems, bzw. seiner Komponenten (vgl. Kap. 2.1) [GTS14, S. 12].

Basierend auf dem Entwicklungsauftrag werden Anforderungen, Struktur und Verhalten des Systems semi-formal erarbeitet. Die formale Modellierung des Systemverhaltens erfolgt darauf folgenden aufbauend, z.B. zur frühzeitige Absicherung der Anforderungen oder zur Machbarkeitsanalyse. Das Spezifikationsmodell wird im Zusammenspiel aller beteiligten Disziplinen erarbeitet und dient in diesem Sinne zur Kommunikation, Kooperation und Koordination der Akteure. Das Spezifikationsmodell bildet den Ausgangspunkt für die domänenspezifische Ausarbeitung (Bild 2-17). Im Zuge dieser Konkretisierung wird die modellbasierte Systemspezifikation aktualisiert und ggf. verfeinert. Dabei ist die Konsistenz der Artefakte sowohl vertikal (zwischen dem disziplinübergreifenden Modell und den domänenspezifischen Modellen) als auch horizontal (über alle Aspekte der modellbasierten Spezifikation) sicherzustellen [Kai14], [GDE+19].

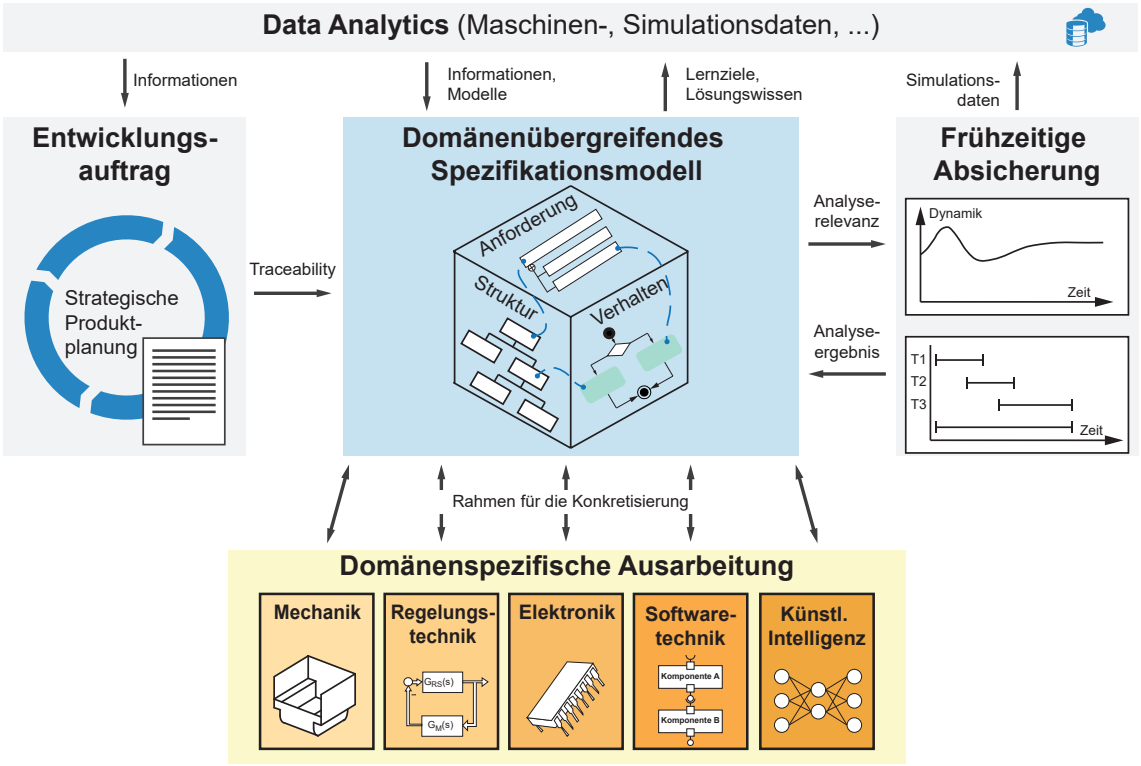


Bild 2-17: Anwendung von MBSE im Systementwurf nach [Kai14, S .32]

Mehrwerte von Data-Driven Systems Engineering

Das datengetriebene Vorgehen bietet mehrere Vorteile: Es ermöglicht eine interdisziplinäre Systembetrachtung und trägt dabei zum **einheitlichen Verständnis** aller beteiligten Disziplinen und zur Beherrschung der Komplexität bei. Dabei werden die systemischen Zusammenhänge transparent dargestellt. So sind zum Beispiel sämtliche Aspekte mit den Anforderungen verknüpft. Änderungen und Abhängigkeiten können so nachvollzogen werden. Die formale Modellierung stellt die **Konsistenz, Korrektheit und Vollständigkeit** der Informationen sicher. Darüber hinaus wird das Systemmodell als Plattform zu **Kommunikation und Kooperation** der beteiligten Akteure gesehen. Durch die gemeinsame Erstellung in der frühen Phase dient das Systemmodell als Basis für den **Übergang in die Konkretisierung**. Durch die rechnerinterne Beschreibung dient es als Bindeglied zwischen allen Produktdaten. Dies führt zu einer besseren **Erfassung und Verwendung von Wissen** durch Wiederverwendbarkeit und Standardisierung. Das Spezifikationsmodell schafft die Voraussetzungen für die Abbildung von Testfällen und zur kontinuierlichen Absicherung der Systemeigenschaften [Kai14, S. 26f.], [INC18-olb], [GDE+19, S. 405f.].

2.4.3 Absicherung komplexer technischer Systeme

Der Bedarf an einer frühzeitigen und kontinuierlichen Absicherung ergibt sich durch ihre Hebelwirkung. Durch eine frühzeitige Absicherung werden Entwicklungsrisiken begrenzt und kostenintensive Iterationen vermieden. Die Änderungskosten steigen von einer Lebenszyklus-Phase zur nächsten um den Faktor 10 (rule-of-ten) (vgl. Bild 2-18) [EM17].

Demgegenüber reduziert sich die technische Unsicherheit entlang der Phasen. Eine frühzeitige Absicherung und Analyse verringert zum einen die technische Unsicherheit und zum anderen wird durch eine virtuelle Absicherung ein Frontloading von Entwicklungsrisiken und Änderungskosten ermöglicht [EM17, S. 168ff.].

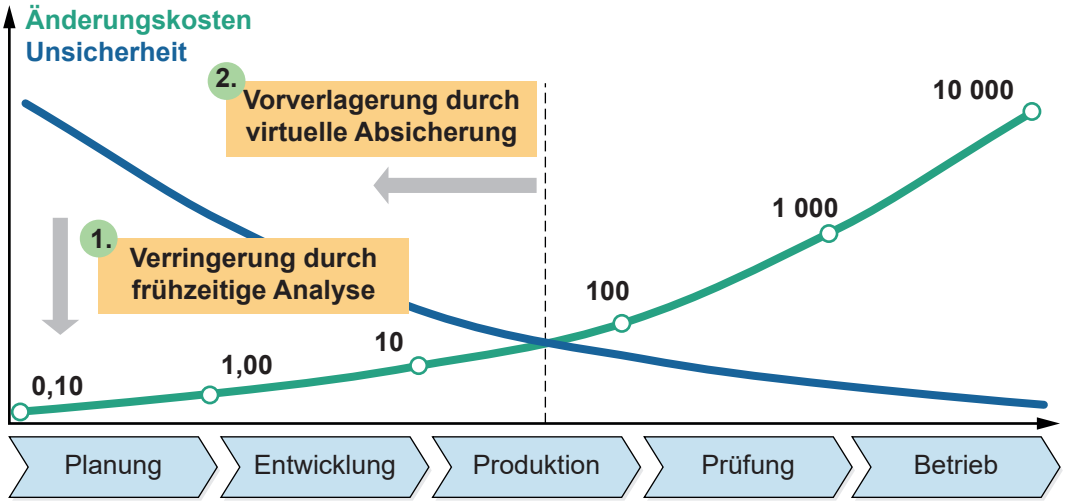


Bild 2-18: Änderungskosten und Unsicherheit im Lebenszyklus nach EHRLENSPIEL ET AL. *[EM17, S. 186]*

Die Absicherung umfasst die Verifikation und die Validierung [VDI2206, S. 38]. In der DIN EN ISO 9000 werden die beiden Begriff eingeführt:

- **Verifikation**: *„Bestätigung durch Bereitstellung eines objektiven Nachweises, dass festgelegte Anforderungen erfüllt worden sind.“* [DIN9000, S. 49]
- **Validierung**: *„Bestätigung durch Bereitstellung eines objektiven Nachweises, dass die Anforderungen für einen spezifischen beabsichtigten Gebrauch oder eine spezifische beabsichtigte Anwendung erfüllt worden sind.“*[DIN9000, S. 50]

Mit der Verifikation erfolgt somit der Nachweis, dass die festgelegten Anforderungen erfüllt sind, unabhängig von der Sinnhaftigkeit der Anforderungen. Bei der Validierung muss hingegen sichergestellt werden, dass die richtigen Anforderungen ermittelt und dokumentiert sind und diese auch erfüllt bzw. überprüft werden können [ABK+16, S.542].

Die Validierung kann anhand von Tests, Experimenten, Versuchen, Simulationen etc. durchgeführt werden. Eine klare Unterscheidung der Begriffe ist schwer. Eine Möglichkeit ist die Unterscheidung nach dem Validierungsziel. Es können zum einen konkrete technische Anforderungen oder vage Hypothesen überprüft werden. Liegt der Fokus der Validierung auf der Überprüfung von Hypothesen, wird der Begriff Experiment bevorzugt. In den anderen Fällen wird eher der Begriff Test verwendet [ABK+16, S. 542].

Der Begriff **Test** wird in der Literatur folgendermaßen definiert:

> *„Ein Test ermittelt Systemeigenschaften eines [zu untersuchenden] Systems [...] und liefert Erkenntnisse über das System, insbesondere ob das*

> *System zuvor definierte Anforderungen ganz, teilweise oder gar nicht erfüllt. Ein Test umfasst stets einen Testfall, eine Testumgebung und eine Testinterpretation.“ [Ebe15, S. 136]*

Der **Testfall** repräsentiert ein oder mehrere Anwendungsfälle. Er definiert die Start- und Randbedingungen sowie das erwartete Verhalten des Systems [ABK+16, S. 554]. Ein Testfall wird über die **Testumgebung** realisiert. Nach EBEL umfasst die Testumgebung:

> *„[...] die Gesamtheit aller physischen und virtuellen Modelle bzw. Originale, die notwendig sind, um einen oder mehrere Testfälle durchzuführen und das erwartete Systemverhalten zu erfassen.“ [Ebe15, S. 136]*

Die **Testinterpretation**

> *„überführt das in einem durchgeführten Testfall erfasste Systemverhalten in eine oder mehrere zugrundeliegende Systemeigenschaften und liefert Erkenntnisse über das System und den Test, insbesondere ob die zuvor definierte Ziele, Anforderungen oder Hypothesen ganz, teilweise oder nicht erfüllt werden.“[Ebe15, S. 136]*

Ziel ist es, die Bandbreite der späteren Anwendungen eines komplexen technischen Systems in einem Testfall abzubilden [ABK+16, S. 554ff.]. Eine notwendige Basis für die Definition der Testfälle bilden die Identifikation und die Beschreibung der Anwendungsfälle aus denen konkrete Anforderungen an das System und auch die Testumgebung abgeleitet werden [ABK+16, S. 555]. Bild 2-19 stellt die Zusammenhänge von Anwendungsfällen, Anforderungen, Zielen und Testfällen anschaulich dar.

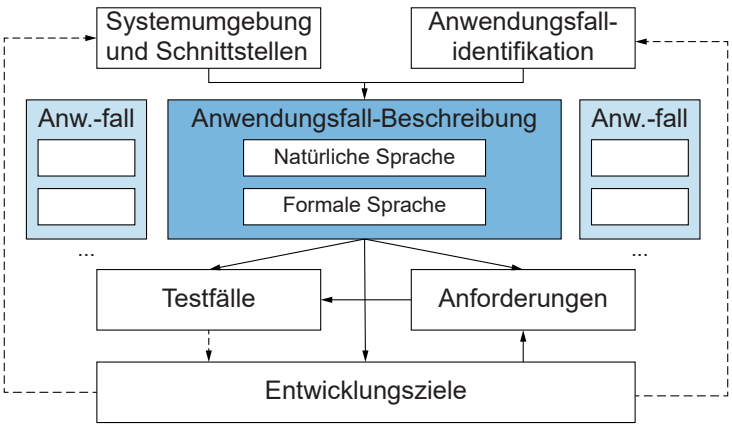


Bild 2-19: Zusammenhang von Anwendungsfällen, Anforderungen, Entwicklungszielen und Testfälle [Ebe15, S. 137]

Mit der Absicherung fortgeschrittener mechatronischer Systeme kommt dem Szenariobasierten Testen eine zunehmende Bedeutung zu. Ein Szenario stellt für spezielle Validierungsziele jeweils eine Referenzanwendung im Betrieb des Systems dar [ABK+16]. ULBRICH ET AL. definieren die Begriffe **Szene** und **Szenario**:

> *„Eine Szene beschreibt eine Momentaufnahme des Umfelds, welche die Szenerie, dynamische Elemente, die Selbstrepräsentation aller Akteure und Beobachter wie auch die Verknüpfung dieser Entitäten umfasst. [...].“ [UMR+15, S. 983]*

Entsprechend der Definition besteht eine Szene aus dynamischen Entitäten (auch Virtuellen Objekte), der Szenerie und der Selbstrepräsentation sowie aus den beteiligten Akteuren und Beobachtern. Dynamische Entitäten besitzen Zustände und Attribute. Die Szenerie beschreibt alle räumlich stationären Aspekte eine Szene. Dies sind Aspekte wie z.B. stationäre virtuelle Objekte, Umweltbedingungen oder topologische Informationen. Die Selbstrepräsentation spezifiziert Fähigkeiten sowie die Zustände und Attribute der Akteure. Bild 2-20 zeigt eine Szenenrepräsentation [SPA+18]. Ein Szenario verknüpft Szenen durch Aktionen und Ereignisse in einer zeitlichen Abfolge (z.B. Fahrt auf einer Durchgangsspur).

> *„Ein Szenario beschreibt die zeitliche Entwicklung von Szenenelementen innerhalb einer Folge von Szenen, welche mit einer Startszene beginnt. Aktionen und Ereignisse ebenso wie Ziele und Werte können spezifiziert werden, um diese zeitliche Entwicklung in einem Szenario festzulegen. Im Gegensatz zu Szenen decken Szenarien [...] Zeitspannen ab.“ [UMR+15, S. 986]*

Zur formalen Beschreibung von Szenarien werden Beschreibungssprachen wie etwa *OpenSCENARIO*[1] verwendet [Vir18-ol]. Neben der Untersuchung und der Absicherung von Eigenschaften dienen Simulationen dazu, Daten zum Training und zum Testen für das maschinelle Lernen bereitzustellen. Hierzu werden virtuelle Szenarien generiert, mit denen Funktionen der Selbstregulation in dem jeweiligen Kontext trainiert und untersucht werden können [DGS+18, S. 19]. Ein Konzept zur Analyse des Systemverhaltens in virtuellen Szenarien sind Virtuelle Testbeds.

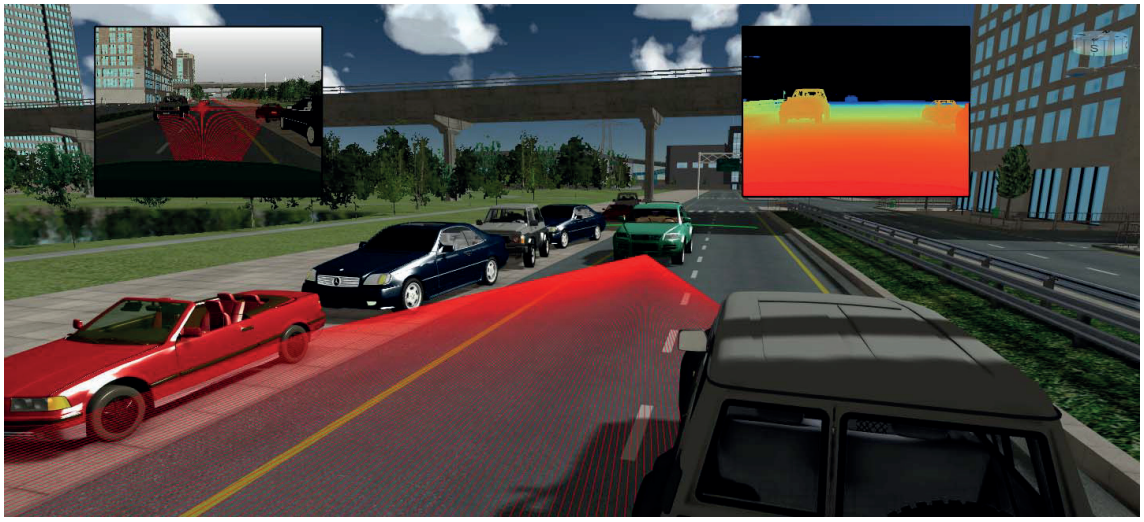


Bild 2-20: Repräsentation einer Szene im Virtuellen Testbed [SPA+18]

[1] Spezialisierter Anwendungsbereich zur Beschreibung von Szenarien im Autonomen Fahren

2.4.4 Virtuelle Testbeds

Die Simulation und Absicherung von Intelligenten Technischen Systemen stellt eine Herausforderung dar. Dies liegt zum Beispiel an den folgenden drei Faktoren:

- **Komplexe Operationsumgebung**: Autonome Systeme operieren in einer offenen, unstrukturierten und dynamischen Umgebungen. Eine geschlossene und vollständige Funktionsspezifikation des Systems und somit eine vollständige Testabdeckung aller Situationen ist nicht möglich [SSL15], [Ras15].
- **Selbstregulation**: Systeme generieren ihren Handlungsplan unter Berücksichtigung der Systemziele und der Umgebung. Die Verifikation und Validierung der Selbstregulation stellen hierbei wesentliche Herausforderungen dar. Die Selbstregulation operiert in einem komplexen Zustandsraum [HSS16].
- **Nicht-Deterministisches Verhalten**: Autonome Systeme basieren auf lernenden Algorithmen. Über die Zeit kann sich ihre Reaktion auf gleiche Zustandsgrößen verändern, da sich ihr Wissen über die Zeit verändert. Dies bedeutet auch, dass ein erfolgreicher Test nicht garantiert, dass das System beim nächsten Testlauf den gleichen Test besteht [HSS16].

Zur Absicherung dieser Systeme werden in der Literatur **Virtuelle Testbeds** vorgeschlagen [SSL15], [HSS16], [Tho08]. Virtuelle Testbeds bieten zahlreiche Vorteile: Sie ermöglichen kritische Szenen in einer sicheren Umgebung reproduzierbar zu analysieren, um daraus Implikationen für die Weiterentwicklung eines Systems abzuleiten. Ebenso können Simulationen auf verteilten Computersystemen parallelisiert und synchronisiert werden [HSS16], [Ras15, S. 10]. Zudem können konsistente virtuelle Szenarien durch den Rechner generiert und simuliert werden. Dies erhöht die Testabdeckung des Systems [Tho08]. Virtuelle Testbeds sind hierbei ein Konzept zur virtuellen Integration und Simulation. RAST nutzt folgende Definition eines Virtuellen Testbeds:

> *„Ein Virtuelles Testbed (VTB) ist ein Software-System zur system-, disziplin- und anwendungsübergreifenden Entwicklung komplexer Systeme. Es stellt Methoden zum reproduzierbaren Test, sowie zur Verifikation und Validierung dieser Systeme unter frei wählbaren Randbedingungen bereit. Hierzu werden Systemmodelle, Simulationsalgorithmen, datenverarbeitende Algorithmen und reale System für die ganzheitliche Betrachtung des dynamischen Gesamtsystems einschließlich aller Wechselwirkungen in gewählter Detaillierung integriert.“[Ras15, S. 7]*

Das Bild 2-21 beschreibt eine Referenzstruktur für Virtueller Testbeds in Anlehnung an [GRS14]. Ausführbare **physikalisch-mathematische Modelle** bilden die unterste Ebene. Entsprechend ihres Detaillierungsgrades bilden diese Modelle ein plausibles Verhalten oder ein physikalisch-korrektes Verhalten ab. Diese Modelle sind Teil des Virtuellen Prototypen. Ein **Virtueller Prototyp** umfasst verschiedene Aspekte wie etwa die Gestalt und die Struktur des Systems. Zur Modellierung werden 3D-CAD Werkzeuge und das

Produktdatenmanagement (PDM) verwendet. Der Virtuelle Prototyp integriert zudem weitere Modelle wie etwa die Kinematik und Dynamik, Modelle der Sensorik und Aktorik, sowie der Selbstregulation. Das Virtuelle Testbed kombiniert Virtuelle Prototypen und ihre **Virtuelle Umgebung** zur einer Simulation des Gesamtsystems. Dies erfolgt durch die Modellierung von Szenarien in einer interaktiven Umgebung. Hierzu ist es notwendig Simulationstechniken, wie z.B. Starrkörperdynamik, Bodenmechanik oder Partikelsysteme zu integrieren [Ras15, S. 86]. Virtuelle Testbeds simulieren den virtuellen Prototypen und dessen Interaktion mit der virtuellen Umgebung in einem spezifischen Szenario. Die virtuelle Umgebung approximiert die reale Operationsumgebung [SSL15].

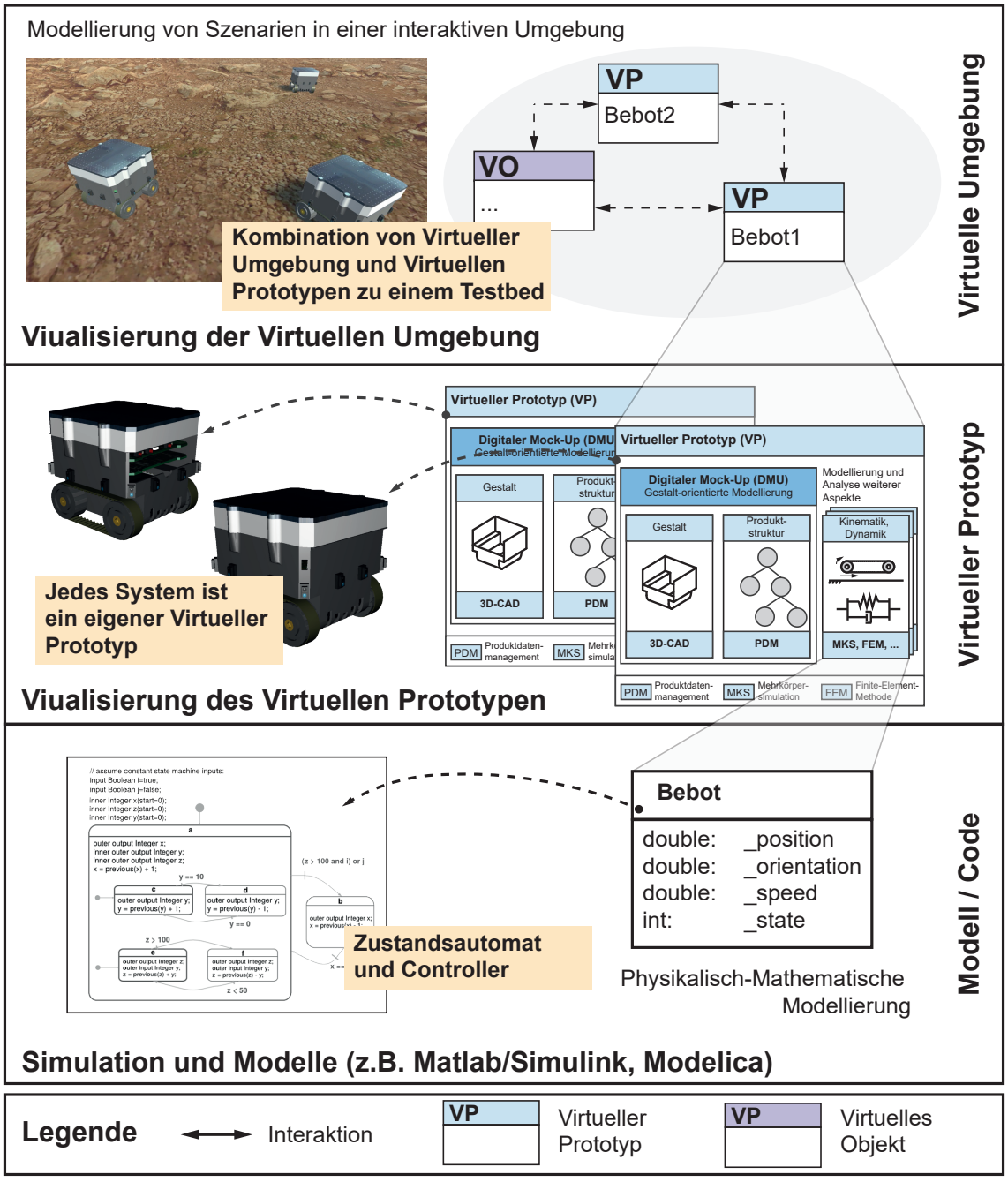


Bild 2-21: Struktur Virtueller Testbeds nach [GRS14, S. 311]

2.5 Problemabgrenzung

Mechatronische Systeme basieren auf dem synergetischen Zusammenwirken von Mechanik, Elektronik, Softwaretechnik und Regelungstechnik. Mit der zunehmenden Durchdringung der Digitalisierung und der Künstlichen Intelligenz wandeln sich diese System zu Intelligenten Technischen Systemen. Derartige Systeme sind autonom, interaktiv und dynamisch vernetzt (vgl. Kap. 2.2.3). Autonome Systeme sind selbstlernend und erreichen ein vorgegebenes Ziel weitestgehend ohne menschliche Eingriffe und expliziter Programmierung. Sie agieren zielorientiert, selbstständig und situationsadaptiv [DGS+18, S. 9]. Autonome Systeme erfordern und ermöglichen ein hohes Maß an Resilienz (vgl. Kap. 2.3). Die Integration von Eigenschaften der Selbstheilung adressiert diese Anforderungen und stellt somit eine Notwendigkeit zur Steigerung der Autonomie dar (vgl. Kap. 2.3.4). Konkret weisen selbstheilende Systeme **fünf Merkmale** auf:

- **Wandlungsfähig:** Die Systeme können sowohl ihr Verhalten als auch ihr Umfeld entsprechend der Situation und der vorgegebenen Systemziele autonom anpassen. Zur Laufzeit können sie sich in einem vom Entwickler vorgesehenen Rahmen weiterentwickeln.
- **Robust:** Die Systeme können auch in einem dynamischen Umfeld flexibel und autonom agieren. Sie besitzen ein robustes Verhalten in Hinblick auf unerwartete und vom Entwickler nicht berücksichtigte Situationen. Sie können darüber hinaus mit unbekannten Situationen umgehen, die von Unsicherheit, Unvollständigkeit und Vagheit geprägt sind.
- **Lernfähig:** Die Systeme können sich innerhalb vorgegebener Randbedingungen weiterentwickeln. Sie lernen aktiv aus erfolgreichen der misslungenen Handlungen. Mit Hilfe von Algorithmen kann das System im Betrieb auf Basis von Daten lernen. Sie verbessern vorab trainierte Modelle und erweitern ihre Wissensbasis.
- **Redundant:** Das System verfügt über funktional gleiche oder vergleichbare Ressourcen. Darüber hinaus verfügt es um eine technische Rückfallebene mit einer sicheren Funktion im Schadensfall oder in Notsituationen.
- **Verteilt:** Das System besteht aus einer Ansammlung unabhängiger Einheiten aus Hardware und/oder Software. Die Einheiten kommunizieren und agieren in einem dynamischen Verbund und bilden ein einzelnes kohärentes System.

Selbstheilende Systeme erfordert ein ganzheitliches Lösungskonzept und sind das Ergebnis der Synergetik aus den Disziplinen der Künstlichen Intelligenz, der Verlässlichkeit und der Mechatronik. Selbstheilende Systeme sind hochkomplex. Das gleiche gilt für ihre Entwicklung und die Orchestrierung der beteiligten Disziplinen. Konkret sind vier **Herausforderungen** zu adressieren:

- **Herausforderung 1: Sicherstellen der Resilienz der intendierten Funktionen**
  Selbstheilende Systeme können auch bei beeinträchtigter Funktionalität ihre Aufgabe

erfüllen. Sie müssen daher in der Lage sein, ihr Verhalten anzupassen, um unzureichende oder ausgefallene Einheiten zu kompensieren. Eine lernfähige Selbstheilung muss eine inhärente Eigenschaft dieser Systeme sein. Es fehlt dabei ein einheitliches Begriffsverständnis und ein Lösungsmodell wie diese Eigenschaften systematisch in die Systeme integriert werden und welche Entwurfsaspekte dabei zu adressieren sind. Die Sicherstellung der Resilienz stellt somit eine Herausforderung dar, die bereits im Systementwurf berücksichtigt werden muss [SMK+17, S. 17f.], [RKG+18].

- **Herausforderung 2: Zunehmende Interdisziplinarität**
  Die Integration der Selbstheilung kann nicht aus dem Blickwinkel einer Disziplin erfolgen, sondern muss systemisch in der frühen Phase der Produktentwicklung auf Ebene der Architektur erarbeitet werden. Ein einheitliches Systemverständnis über die Struktur und das Verhalten ist daher erforderlich. Die frühzeitige Einbindung von Experte der Verlässlichkeit und der Künstlichen Intelligenz stellt daher eine zentrale Herausforderung dar [DGS+18, S. 67].

- **Herausforderung 3: Methodisches Vorgehen**
  Selbstheilende Systeme basieren auf vergleichsweisen neuen und komplexen Technologien. Es fehlt dabei noch an durchgängigen Ansätze zur Integration von Selbstheilung. Dies umfasst integrierte Entwicklungswerkzeuge zur Modellierung, Simulation und zur Visualisierung [DGS+18, S. 18]. Die zielgerichtete Anwendung von Entwicklungsmethoden und integrierten Entwicklungswerkzeugen bildet die Grundlage für die Integration der Selbstheilung.

- **Herausforderung 4: Szenario-basierte Absicherung**
  Die Absicherung selbstheilender Systeme auf der Basis von virtuellen Testbeds ist eine zentrale Aufgabe. Die Herausforderung liegt darin, das Absicherungskonzept integrativ mit dem Systemkonzept zu beschreiben und geeignete Methoden zur frühzeitigen Eigenschaftsabsicherung der Selbstheilung einzusetzen. Potenzial bietet der kombinierte Einsatz von szenariobasierten Simulation und Methoden der Fehlerinjektion in virtuellen Testbeds [Ras15, S. 7ff.].

Die Integration von Selbstheilung wird die Autonomie von Intelligenten Technischen Systemen deutlich steigern. Dabei müssen die Entwurfsaspekte der Selbstheilung systematisch für den Systementwurf aufbereitet sein und die genannten Herausforderungen adressieren. Es besteht der *Bedarf* an einer *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*. Diese Systematik sollte die folgenden Bestandteile umfassen:

- **Selbstheilungskonzept:** Ein Selbstheilungskonzept für Intelligente Technische Systeme. Das Konzept dient als Referenzmodell und weist die oben genannten Merkmale auf. Es definiert den Prozess der Selbstheilung, beschreibt die Konstruktionsprinzipien und stellt Modellbibliotheken zur Realisierung selbstheilender Systeme zur Verfügung.

- **Hilfsmittel für den datengetriebenen Systementwurf selbstheilender Systeme:** Die Entwicklung benötigt eine interdisziplinäre Beschreibung des zu entwickelnden Systems. Insbesondere sind die Meta-Aspekte Struktur, Verhalten und Anforderungen unerlässlich und mit einer systemischen Methode zur Analyse von Potenzialen für die Integration von Eigenschaften der Selbstheilung zu kombinieren. Zudem ist das Absicherungskonzept des Systems integrativ zu beschreiben. Nur so ist eine disziplinübergreifende Spezifikation des selbstheilenden Systems zu erreichen.
- **Strukturiertes Vorgehensmodell:** Kern der Systematik ist ein Vorgehensmodell, das die Integration von Eigenschaften der Selbstheilung in ein Intelligentes Technisches System in der frühen Phase der Entwicklung beschreibt. Das Vorgehen soll die notwendigen Tätigkeiten als wiederkehrende Prozessbausteine und entsprechende Hilfsmittel integrieren. Die erforderlichen Rollen und ihre Verantwortlichkeiten im Systementwurf sind zu definieren.
- **Konzept zur Szenario-basierten Absicherung:** Die Systematik sollte ein Konzept zur Szenario-basierten Analyse selbstheilender Systeme auf der Basis von virtuellen Testbeds bereitstellen. Hier ist ein Entwicklungsumgebung mit Analysetechniken umzusetzen. Diese Umgebung stellt Daten für das maschinelle Lernen als auch für die Selbstheilung optimierte virtuelle Szenarien bereit. Es unterstützt eine virtuelle Inbetriebnahme selbstheilender Systeme. Kern bilden automatisierte Analysetechniken, die Fehler und Beeinträchtigungen auf Modell- und Szenarioebene simulieren.

2.6 Anforderungen an die Systematik

Entsprechend der vorangegangenen Problemanalyse und den resultierenden Handlungsfeldern werden im folgenden die Anforderungen an eine *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* definiert.

Anforderungen an ein Framework selbstheilender Systeme

A1) Ganzheitliches Selbstheilungskonzept: Die Systematik muss ein geeignetes Framework zur Realisierung von selbstheilenden Systemen bereitstellen und relevante interdisziplinäre Entwurfsaspekte definieren. Das Framework muss sich dabei an einem Referenzprozess der Selbstheilung orientieren. Mögliche Leistungsstufen der Selbstheilung sind dabei zu berücksichtigen.

A2) Bereitstellung von Konstruktionsprinzipien: Die Systematik muss Lösungswissen bereitstellen. Dieses muss sich an dem Selbstheilungskonzept orientieren und Konstruktionsprinzipien für die *Überwachung*, die *Diagnose* und für *Selbstheilungsaktionen* bereitstellen. Domänenexperten sind bei der Auswahl von Lösungswissen zu unterstützen.

Anforderungen an die Spezifikationstechnik selbstheilender Systeme

A3) Interdisziplinäre Anwendbarkeit: Die Systematik muss für den interdisziplinären

Systementwurf konzipiert und anwendbar sein. Dabei sind neben der Mechanik, Elektrotechnik und der Informationstechnik auch die Künstliche Intelligenz und die Verlässlichkeit zu berücksichtigen. Die Systematik muss ein interdisziplinär anwendbares Sprachkonzept für die Systemgestaltung intelligenter technischer Systeme mit Eigenschaften der Selbstheilung bereitstellen (vgl. Bild 2-16).

A4) Identifikation von Potenzialen zur Selbstheilung: Die Integration von Eigenschaften der Selbstheilung eignet sich für kritische Funktionen eines Intelligenten Technisches Systems. Die Systematik muss daher bei der systemischen Analyse von Potenzialen unterstützen. Dabei sind Unsicherheit, Komplexität und Risiken zu bewerten. Ebenso muss die Systematik eine Notation bereitstellen, um funktionale und temporale Abhängigkeiten zwischen Beeinträchtigung und Gefährdungen zu modellieren.

A5) Beschreibung der Absicherung: Die frühzeitige Absicherung selbstheilender Systeme und dessen Eigenschaften dient zur Überprüfung des spezifizierten Lösungskonzeptes. Die Systematik muss daher die Modellierung eines Absicherungskonzeptes unterstützen. Kern bilden Testfälle und die Szenarien (vgl. 4.3.2). Beide Aspekte sind mit den Anforderungen auf der Modellebene zu verknüpfen, um die Kohärenz und Konsistenz der Artefakte zu gewährleisten.

Anforderung an ein strukturiertes Vorgehensmodell

A6) Orientierung am Systementwurf: In dem Systementwurf wird ein übergreifendes Spezifikationsmodell als Grundlage für die Ausarbeitung erarbeitet. Die Systematik muss sich an diesem Vorgehen orientieren und die frühe Phase der Entwicklung fokussieren. Ihre Bestandteile sind entsprechend zu gestalten.

A7) Systematische und interdisziplinäre Vorgehensweise: Die Integration von Selbstheilung ist ganzheitlich auf Systemebene zu betrachten. Die Systematik muss einen zielgerichteten und systematischen Entwurf ermöglichen. Das Vorgehen muss sich in das etablierte Vorgehen zum Entwurf von Intelligenten Technischen Systemen einordnen.

Anforderung an die szenario-basierte Absicherung

A8) Bereitstellung von Analysetechniken: Die Analyse und Absicherung von selbstheilenden Systemen und dessen Eigenschaften erfordert entsprechende Analysetechniken. Diese Techniken müssen in der frühen Phase der Entwicklung möglichst automatisiert anwendbar sein und Auswertungen bereitstellen. Dabei sind Beeinträchtigungen in der Software, Hardware oder Umgebung reproduzierbar zu simulieren.

A9) Werkzeugunterstützung durch ein Virtuelles Testbed: In Anlehnung an das Konzept Virtueller Testbeds nach RAST (vgl. Abschnitt 2.4.4) ist eine virtuelle Validierungsumgebung bereitzustellen. Die Umgebung muss die Analysetechniken integrieren und somit aussagekräftige Simulationsdaten für das Training und das Testen datengetriebener Modelle bereitstellen. Das Virtuelle Testbed ist prototypisch umzusetzen.

3 Stand der Technik

Dieses Kapitel untersucht einen ausgewählten Stand der Technik. Zunächst gibt Kap. 3.1 einen strukturellen Überblick. Anschließend wird der betrachtete Stand der Technik dargestellt. Das Kapitel schließt mit einer Bewertung vom Stand der Technik gegenüber den Anforderung und stellt den Handlungsbedarf für eine *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* in Kap. 3.5 heraus.

3.1 Überblick zum Stand der Technik

Das Bild 3-1 gibt einen Überblick zu den untersuchten Ansätzen aus dem Stand der Technik. Diese gliedern sich in drei Bereiche: Kap. 3.2 fokussiert die **Integration der Selbstheilung** in ein komplexes technisches System. Es werden Strukturkonzepte fortgeschrittener mechatronische Systeme und technologische Ansätze der Selbstheilung und Steigerung der Verlässlichkeit untersucht. Das Kap. 3.3 betrachtet **Hilfsmittel zur Spezifikation** von Selbstheilung im Systementwurf. Hierzu zählen Sprachen, Analysetechniken und Vorgehensweisen. Das Kap. 3.4 analysiert **Techniken zur kontinuierlichen Absicherung** der Selbstheilung mittels Virtueller Testbeds.

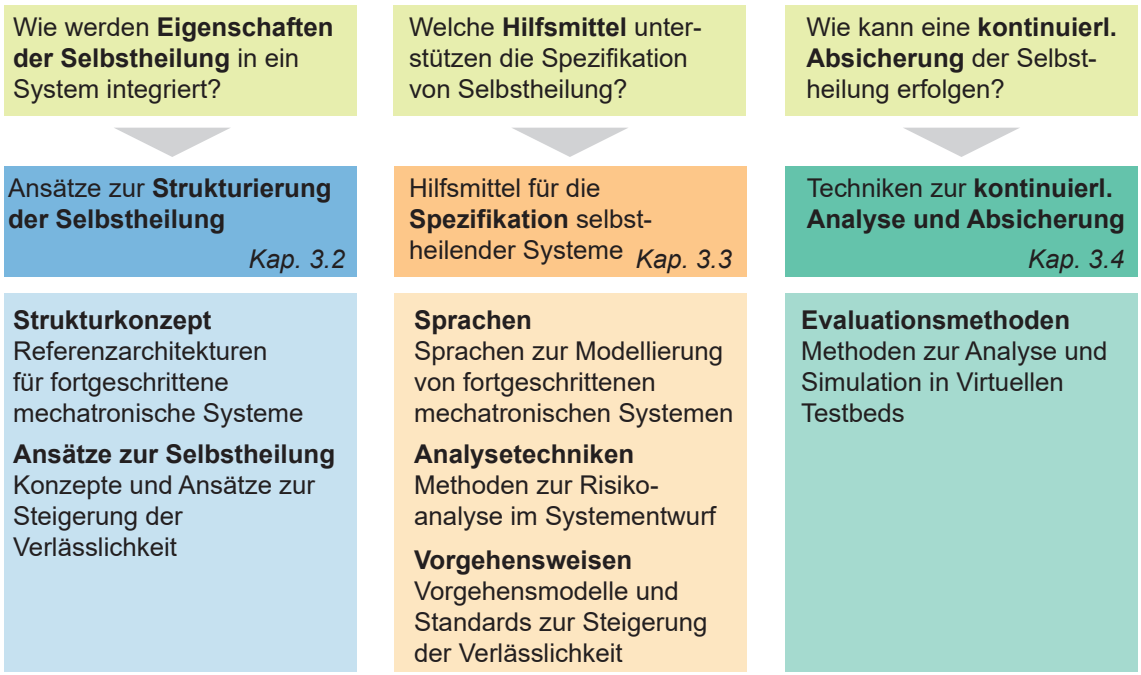


Bild 3-1: Strukturierung vom ausgewählten Stand der Technik

3.2 Ansätze zur Strukturierung der Selbstheilung

3.2.1 Strukturkonzepte autonomer Systeme

Die Literatur umfasst zahlreiche Ansätze zur Strukturierung der Informationsverarbeitung komplexer technischer Systeme. In diesem Abschnitt wird ein ausgewählter Stand der Technik betrachtet und anhand der Anforderungen bewertet.

3.2.1.1 Steuerungsarchitekturen

Die **Deliberative Architektur** repräsentiert eine klassische Steuerungsarchitekur in der Robotik. Dabei werden die (Sensieren-Planen-Agieren) Einheiten hierarchisch angeordnet. Aus den Sensordaten wird ein Modell der Welt erstellt, aus denen Informationen abgerufen werden. Aus diesen Informationen wird ein Plan zur Lösung einer Aufgabe generiert. Der Plan wird anschließend durch die Aktorik ausgeführt. Die **Reaktive Architektur** verzichtet auf die Planung. Sie ermöglicht somit eine schnelle Reaktion auf Sensorinformationen. Das Verhalten des Roboters wird durch Stimulus-Response Regeln definiert (Sensieren-Agieren). Die sensorischen Eingaben werden über die Regeln iteriert. Die Ausgaben dieser Regeln bestimmen dann das Verhalten des Systems [RN16], [SSN11]. Eine der bekanntesten reaktiven Architektur ist die **Subsumption Architektur**. Sie wurde 1986 von BROOKS veröffentlicht. Das Verhalten ist nach Abstraktionsebenen in Module gegliedert. Ein Koordinator priorisiert die Module. Die **Hybride Architektur** kombinieren reaktives und deliberatives Verhalten. So wird z.B. eine lokale Hindernisvermeidung durch eine reaktive Architektur umgesetzt. Die Navigation zu einem Ziel über die deliberative Verhaltensebene [Bro85].

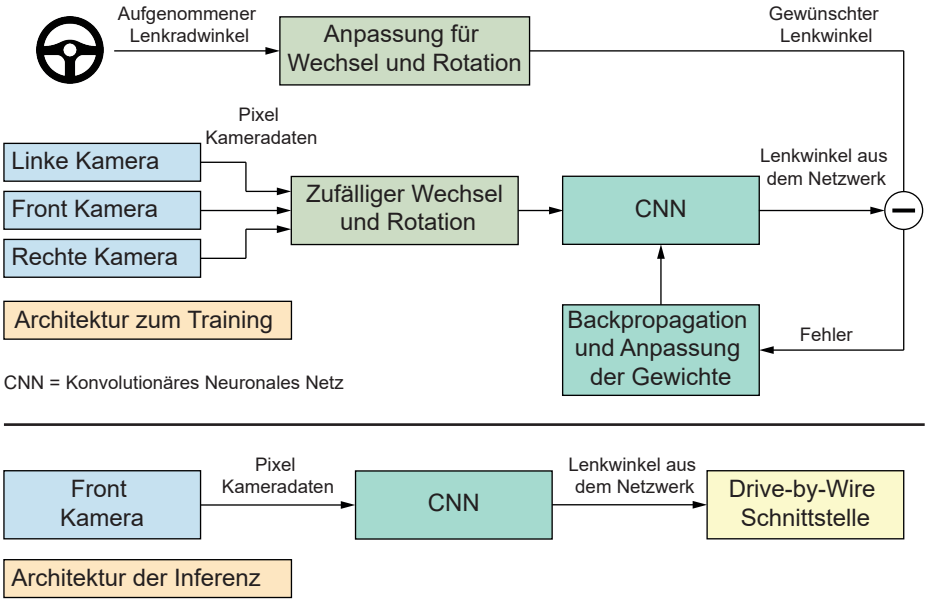


Bild 3-2: Dave-2 Architektur als konnektivistischer Ansatz [BDD+16, S. 3f.]

Konnektivistische Ansätze repräsentieren neuere Ansätze. Sie bilden kognitives Verhalten

ab [GSS14]. So werden Ansätze mit *Deep Reinforcement Learning* wie das **DAVE-2 System** in der Robotik erforscht. Dieser Ansatz bildet die sensorischen Eingaben auf die Aktorik ab. Das Lernen erfolgt durch Imitation oder Belohnung [BDD+16] (vgl. Bild 3-2. Deep Reinforcement Learning (DeepRL) kombiniert tiefe Neuronale Netze und Verstärkendes Lernen, um ein spezifisches Verhalten zu lernen [ABC+18], [Wah17].

Bewertung: Klassischen Architekturen sind in der Robotik erprobt. Die Anwendung konnektivistischer Architekturen ist jedoch noch eingeschränkt. Diese Architekturen sind für spezifische Anwendungsbereiche vielversprechend. Die Integration von Aspekten der Selbstheilung oder der Verlässlichkeit in die Architektur wird aber nicht adressiert. Ein Lösungskonzept wird entsprechend nicht bereitgestellt oder durch die Architekturen unterstützt.

3.2.1.2 Neurorobotik Architektur

Die Neurorobotik bildet die Schnittstelle zwischen den Neurowissenschaften und der Robotik. Diese Schnittstelle wird mit dem Feld Neurorobotics im *Human Brain Project* (HBP) erforscht. Das Forschungsprojekt fasst Erkenntnisse über den Aufbau und die Funktionsweise zentraler Einheiten des Nervensystems zusammen. Mittels Modellen und Simulationen wird diese Funktionsweise nachgebildet. Ein Ergebnis sind neue Robotertechnologien. Mit der Neurorobotics Platform wird eine Plattform bereitgestellt, über die Robotersysteme über diese Modelle angesteuert werden. Technologisch sind dies Gepulste Neuronale Netze [KRA+17], [HBP18-ol]. Gepulste Neuronale Netze (spiking neural nets) sind vom Nervensystem inspiriert und beschreiben, wie Aktivierungspotentiale starten und durch ein neuronales Netzwerk propagieren [Maa97].

Das Bild 3-3 visualisiert eine Anwendung der Plattform für ein visuelles Tracking. Der Roboter wird mit seinem Grundsystem (z.B. dem Bewegungsapparat), der Sensorik und der Aktorik über Transferfunktionen mit dem Neuronalen Netz verknüpft. Die Sensoren aktivieren Pulse (Spikes) im Neuronalen Netz, diese Pulse propagieren im Netz und steuern die Aktorik des Roboters über die Transferfunktionen.

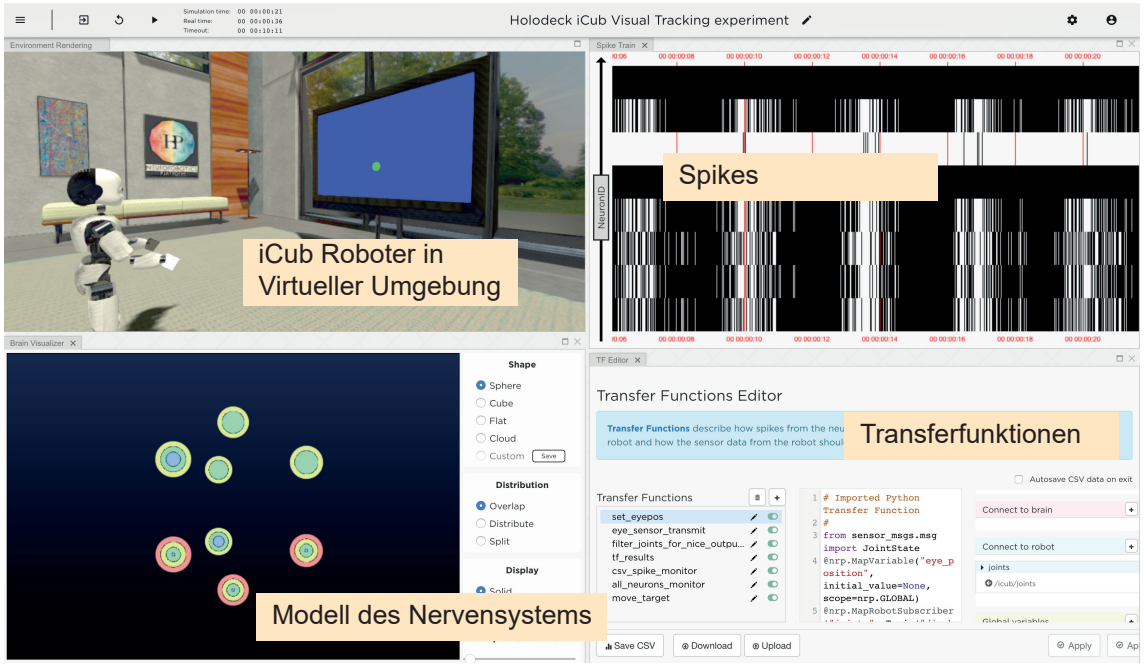


Bild 3-3: Neurorobotics Platform [HBP18-ol]

Bewertung: Die Neurorobotik ist ein vergleichsweise neues Forschungsfeld. In dem Projekt wird eine Plattform als Virtuelles Testbed bereitgestellt. Hierdurch werden biologisch inspirierte Neuronale Netze mit der Sensorik und Aktorik des Systems gekoppelt. Aspekte der Selbstheilung werden nicht adressiert.

3.2.1.3 Operator-Controller-Modul

Im SFB 614 Selbstoptimierende Systeme des Maschinenbaus wird ein Strukturierungskonzept für die Informationsverarbeitung selbstoptimierender Systeme vorgestellt. Zentraler Ansatz ist die Verhaltensanpassungen von Reglerkomponenten als Voraussetzung der Selbstoptimierung. Die konkrete Umsetzung dieses Ansatzes ist das **Operator-Controller-Modul** (OCM). Das OCM umfasst drei Schichten: Controller, Reflektorischer Operator und Kognitiver Operator [GRS14, S. 10ff.]:

Der **Controller** bildet die unterste Schicht und hat direkten Durchgriff auf die Regelstrecke. Der Regelkreis verarbeitet die Messsignale, ermittelt die Stellsignale und gibt diese an die Aktorik weiter. Dies wird als *„Motorischer Kreis“* bezeichnet. Die Informationsverarbeitung auf dieser Ebene operiert quasi-kontinuierlich, d.h. Messwerte werden zeit-kontinuierlich eingelesen, verarbeitet und unter harter Echtzeit ausgegeben. Der Controller kann sich aus mehreren Reglern und Umschaltmechanismen zusammensetzen.

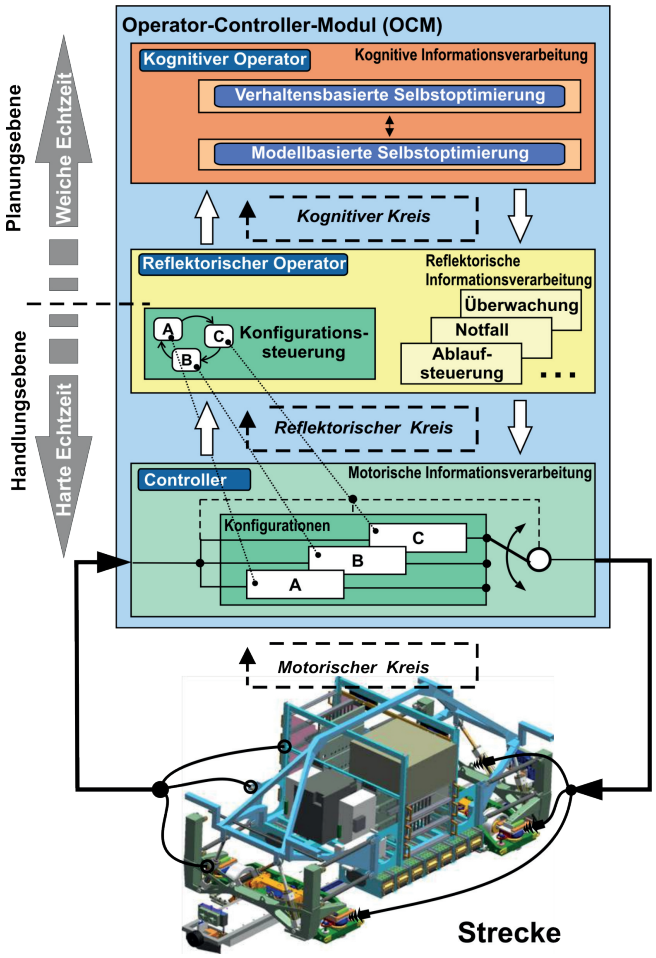


Bild 3-4: Operator-Controller-Modul (OCM) [GRS14, S. 10f.]

Der **Reflektorische Operator** überwacht und steuert den Controller. Der Operator modifiziert den Controller, indem er Parameter- oder Strukturveränderungen initiiert. Diese Konfigurationssteuerung definiert, bei welchem Systemzustand welche Konfiguration gültig ist und unter welchen Bedingungen zwischen den Konfigurationen umgeschaltet wird. Eine wichtige Funktion ist die Zustandsüberwachung des Controllers. Sie ist mit entsprechenden Notfall-Routinen gekoppelt. Der reflektorische Operator arbeitet ereignisorientiert. Als Verbindungsschicht bietet der Reflektorische Operator die Schnittstelle zwischen dem echtzeitfähigen Controller und dem Kognitiven Operator. Die Schicht ist weiterhin für die Kommunikation zwischen mehreren OCMs verantwortlich.

Der **Kognitive Operator** bildet die oberste Schicht des OCM. Auf dieser Ebene erfolgt die Optimierung. Die Anwendung basiert auf vielfältigen Methoden (Lernverfahren, Wissensbasierte Systeme, etc.). Der Schwerpunkt liegt auf den kognitiven Fähigkeiten zur Durchführung der Selbstoptimierung. Die *Modellbasierte Optimierung* erlaubt eine vorausschauende und vom realen System zeitlich entkoppelte Optimierung, während die *Verhaltensbasierte Optimierung* Funktionen zur Planung und zur Bewertung der aktuellen Zielvorgaben liefert. Der Kognitive Operator kann asynchron zur Realzeit arbeiten und unterliegt weicher Echtzeit.

Das OCM definiert die Referenzarchitektur für selbstoptimierende Systeme. DUMITRESCU erweitert das OCM zu Intelligenten Technischen Systemen. Diese orientieren sich an dem Kognitionsmodell nach [S. 9]Strube [Dum10, S. 112], [Str98]. Das Schichtenmodell besteht aus der Nicht-kognitive Regulierung, der Assoziativen Regulierung und der Kognitiven Regulierung. WESTERMANN erweitert diese Architektur um spezifische Merkmale von Cyber-Physischen Systemen (CPS) wie die Vernetzung, Daten und Dienste. Das Ergebnis ist die CPS-Referenzarchitektur [Wes17, S.93].

Bewertung: Das Operator-Controller-Modul bietet einen geeigneten Strukturierungsrahmen für fortgeschrittene mechatronische Systeme. Der reflektorische Operator sieht eine Zustandsüberwachung sowie eine Konfigurationssteuerung zur Verhaltensanpassung vor. Ebenso sind Notfall-Routinen integriert. Die Architektur bietet eine Voraussetzung zur Integration von Selbstheilung. Verfahren und Funktionen der Selbstheilung werden nicht adressiert.

3.2.1.4 Referenzarchitektur Autonomer Systeme

Im FACHFORUM AUTONOME SYSTEME wurde eine Referenzarchitektur für Autonome Systeme definiert. Den Rahmen bilden die Sensorik, die Aktorik und die Selbstregulation. Autonome Systeme besitzen eine selbstregulierende Informationsverarbeitung und verschiedenen Wissensbasen und Planbibliotheken, die ausgehend von einer initialen Konfiguration, selbstständig lernen [FAS17, S. 134]. Die technologischen Bausteine zur Realisierung der Basisarchitektur Autonomer Systeme visualisiert Bild 3-5 [Wah17], [FAS17, S. 135ff.].

Den Kern bildet die **Selbstregulation**. Die **Wahrnehmung und Interpretation** sorgt durch Sensordatenfusion für das maschinelle Sehen von relevanten Objekten und die Analyse von Situationen. Das **Lernen und Schlussfolgern** ermöglicht eine automatische Modellbildung. Verfahren zur temporalen und räumlichen Inferenz ergänzen das maschinelle Lernverfahren. Die **Planung** von Handlungen befasst sich mit der Auswahl zielgerichteter Aktionen. Die modellbasierte Planung basiert auf einem formalen Modell der Welt, des Weltzustandes, der verfügbaren Aktionen, einer Zielbedingung sowie diversen Randbedingungen bzgl. Ressourcen, Fristen und Sicherheit. Anhand des Modells simuliert der Planungsalgorithmus die möglichen Entwicklungen und findet eine adäquate Handlungsstrategie. Die **Planerkennung** ermöglicht das Erkennen von Handlungen und antizipiert Intentionen anderer Agenten. Die **Kommunikation und Kollaboration** ermöglicht eine multimodale Interaktion mit Nutzern. Die Kollaboration nutzt Verfahren aus Multiagentensystemen, um Aufgaben auszuhandeln und Planausführungen zu koordinieren [FAS17, S. 135ff.].

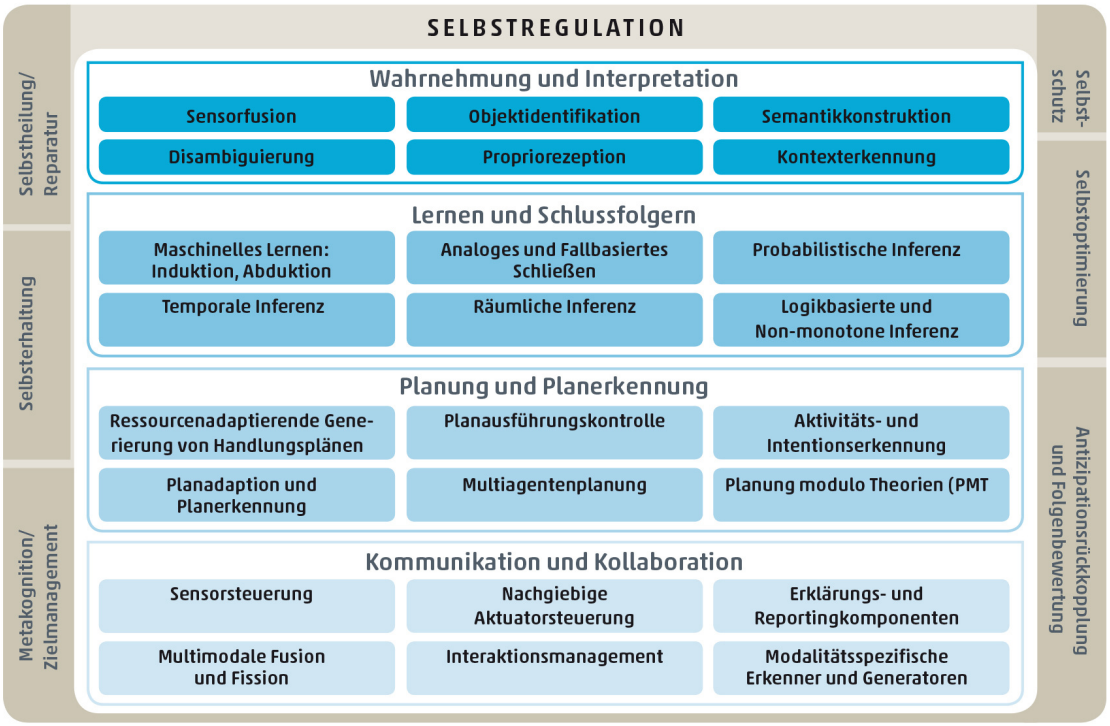


Bild 3-5: Technologie Bausteine Autonomer Systeme [FAS17, S. 135]

Die Wissensbasen beinhalten ein **episodisches Gedächtnis** und ein **Diskursgedächtnis**. Das episodische Gedächtnis ist ein Langzeitspeicher für Ereignisse. Das *Diskursgedächtnis* speichert die Kommunikation des Systems mit anderen Akteuren. Eine **Planbibliothek** speichert Pläne für verschiedene Probleme. Das **Domänenmodell** enthält Modelle von relevanten Objekte, Relationen, Zustände und Ereignisse. Das **Aufgabenmodell** umfasst Aufgaben des Systems. Das **Kollaborationsmodell** beinhaltet Schemata für das Zusammenspiel mit anderen Akteuren. **Benutzermodelle** sind für die Personalisierung erforderlich und beinhalten Präferenzen oder Fähigkeiten eines Systemnutzers [Wah17].

Bewertung: Die Referenzarchitektur definiert einen umfassenden Rahmen an Funktionen und technologischen Bausteinen. Die Selbstheilung ist dort explizit als Eigenschaft hervorgehoben. Es ist hingegen offen, wie die Funktionen und Bausteine umgesetzt werden und zusammenwirken. Auch Hilfsmittel und Vorgehensmodelle werden zwar motiviert, aber nicht bereitgestellt.

3.2.1.5 Organic Design Pattern

Im DFG Schwerpunktprogramm (SPP) 1183 *Organic Computing* (OC) wurden Mechanismen der Selbst-Organisation für komplexe technische Systeme untersucht [DFG18-ol]. Ein Ergebnis ist das **Organic Design Pattern** (ODP). Das ODP ist ein Konstruktionsprinzip für Self-X-Systeme die aus unabhängige Komponenten bestehen, miteinander interagieren und Ressourcen verarbeiten. Das ODP besteht aus den folgenden Klassen und Objekten: Ein **Agent** ist eine Komponente des Systems. Die **Capability** ist die Fähigkeit, die ein

System bei einer Ressource anwenden kann. Eine **Ressource** ist ein Datenobjekt oder ein Werkstück. Ein **Task** sind sequenzielle Aktionen die bei einer Ressource durchgeführt werden sollen, z.B. verbrauchen, herstellen, verarbeiten. Die **Role** definiert die Fähigkeit, die ein Agent im System besitzt. Die **Condition** definiert die Vor- und Nachbedingungen für einen Task. Der **Observer/Controller** ist ein Container für die Rekonfiguration [SNS+10, S. 85f.]. Dass Bild 3-6 zeigt die Struktur vom ODP als Klassendiagramm.

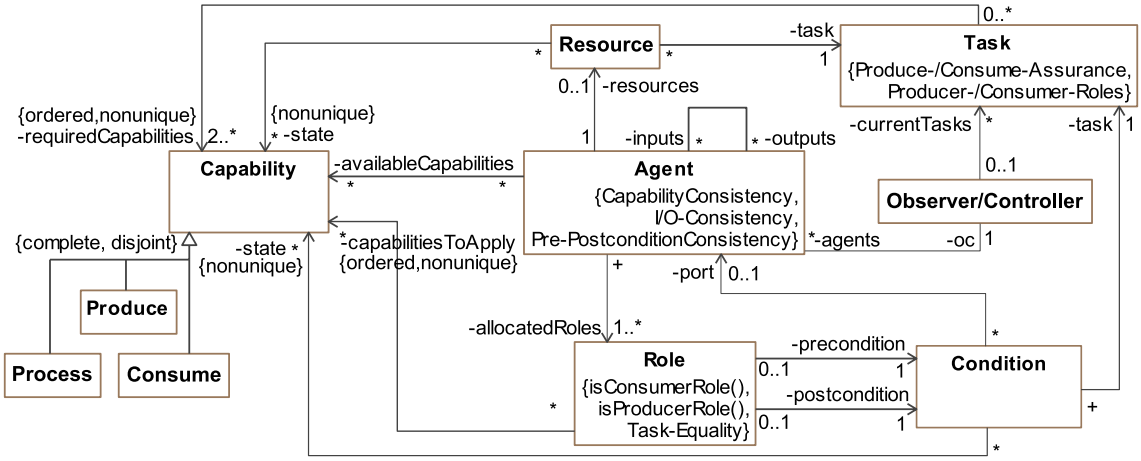


Bild 3-6: Organic Design Pattern [SNS+10, S. 85f.]

Der Agent ist in seinem Verhalten über **Constraints** eingeschränkt und überwacht diese permanent. Im Falle einer Störung, muss der Agent das System durch Rekonfiguration heilen. Die Rekonfiguration erfolgt durch eine Neuzuordnung von Rollen zu den Agenten. Die auf Systemebene definierten Constraints definieren, wie eine korrekte Allokation von Rollen aussieht und definieren so einen korrekten Ressourcenfluss innerhalb des Systems. Korrektheit bedeutet, dass jede in das System eintretende Ressource das System verlässt und von allen Fähigkeiten bearbeitet wird, die die Ressource benötigt. Die Allokation kann durch Mechanismen wie dem Constraint Solver berechnet werden. Das Konstruktionsprinzip ist in [SNH+10] in ein Adaptive Cruise Control (ACC) integriert.

Bewertung: Das ODP folgt einem Agenten-basierter Ansatz. Der Schwerpunkt der Anwendung liegt auf Software-intensiven und verteilten Systemen. Die Selbstheilung erfolgt durch Rekonfiguration und Neuzuordnung von Rollen. Über den Observer/Controller erfolgt eine Überwachung und eine Rekonfiguration. Verfahren zur Selbstheilung in fortgeschrittenen mechatronischen Systemen werden nicht konkretisiert.

3.2.1.6 AUTOSAR Adaptive Plattform

Die Automotive Open System Architecture (AUTOSAR) ist ein internationales Konsortium, bestehend aus Automobilherstellern, Zulieferern und Unternehmen aus der Elektronik, Halbleiter und Softwaretechnik. Das Konsortium wurde 2003 gegründet. Das Ziel des Konsortiums ist eine standardisierte Softwarearchitektur für elektronische Steuergeräte.

Im Jahr 2017 hat das Konsortium die AUTOSAR Adaptive Plattform veröffentlicht. Sie

liegt derzeit in der Version 19.11 vor [AUT20-ol] (Stand: 2020). Diese Plattform trägt den Anforderungen vernetzter Steuergeräte mit zunehmenden Automatisierungsgrad der Fahrzeuge Rechnung. Dies sind u.a. Update- und Upgradefähigkeit durch dynamisches Nachladen von Softwarekomponenten, die Verwendung von Standard-Bibliotheken z.B. der Bildanalyse, selbstständiges Lernen, sowie die Anbindung von digitalen Diensten über ein IT-Backend (vgl. Bild A-1). Damit ermöglicht die AUTOSAR Adaptive Plattform flexiblere E/E-Architekturen im Fahrzeug. Die Architektur der Plattform zeigt Anhang A1.1 [AUT17].

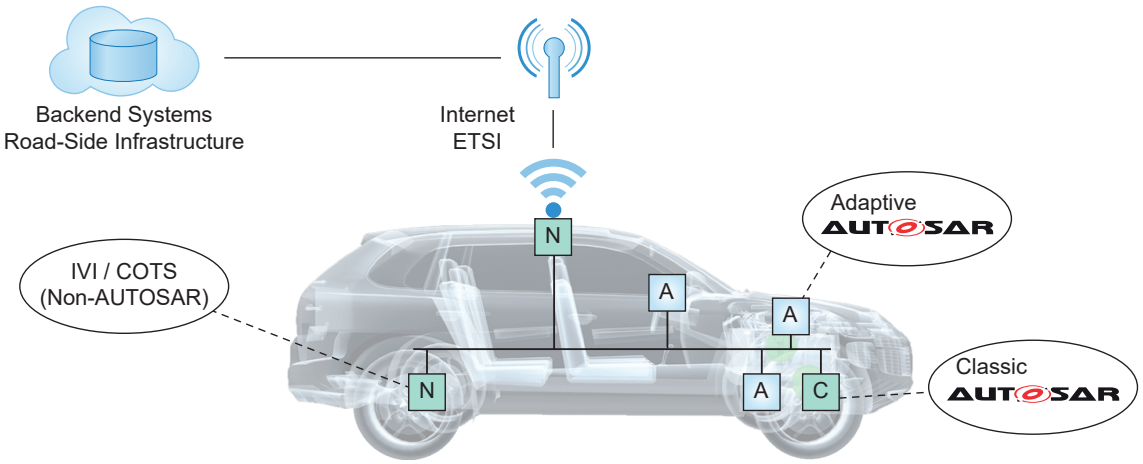


Bild 3-7: Flexible E/E-Architektur mit der AUTOSAR Adaptive Plattform [AUT17]

In der Softwarearchitektur sind die Module *Platform Health Management* und *Diagnostics* in der Adaptive Platform Foundation integriert. Das Platform Health Management unterstützt fail-operational Anwendungen wie z.B. das Health Monitoring im Steuergerät. Der Standard definiert weitergehend Anforderungen and die E/E-Architektur und umfasst eine Software-Spezifikation zur Realisierung von Anwendungen. Ebenso werden Akzeptanztests durch den Standard vorgegeben [AUT17].

Bewertung: Die Plattform definiert eine standardisierte Softwarearchitektur und stellt Anforderungen und eine Software-Spezifikationen bereit. Die Diagnostik und das Health Monitoring sind explizite Bestandteile dieser Softwarearchitektur. Der Standard definiert ferner Vorgaben für Akzeptanztests, die durch die Entwicklungswerkzeuge bereitgestellt werden. Weitere Funktionen der Selbstheilung, wie etwa die Anomalie Detektion, werden nicht adressiert. Ebenso ist der Standard auf der interdisziplinären Systemebene nicht anwendbar.

3.2.2 Ansätze zur Selbstheilung

3.2.2.1 Mehrstufiges Verlässlichkeitskonzept

SEXTRO ET AL. haben im SFB 614 ein mehrstufiges Verlässlichkeitskonzept umgesetzt [GRS+14, S.55]. Das Konzept dient der Steigerung der Verfügbarkeit und Sicherheit in einem selbstoptimierenden Systems. Innerhalb des Operator-Controller-Moduls wird das

Konzept im Reflektorischen Operator positioniert [Son15, S.50]. Dort beeinflusst es die Optimierung im Kognitiven Operator und ermöglicht eine Verbindung zur Konfigurationssteuerung.

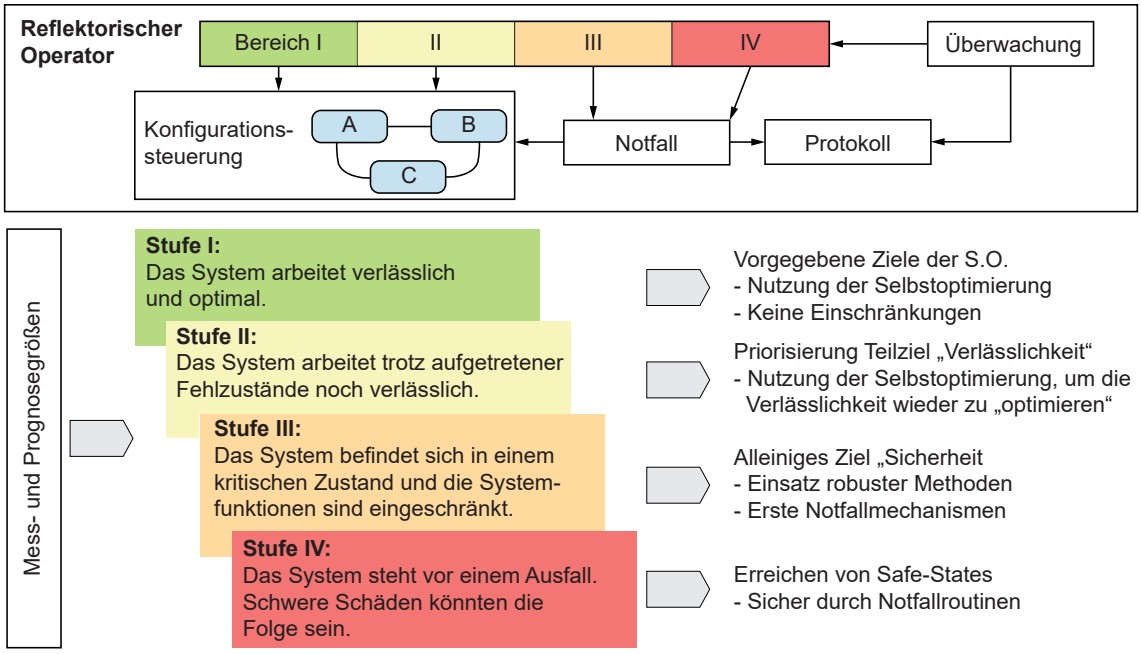


Bild 3-8: Mehrschichtiges Verlässlichkeitskonzept [Son15, S. 50]

Mittels eines Moduls zur Zustandsüberwachung werden die Sensordaten verarbeitet und eine Bewertung des Betriebszustands durchgeführt. Der Betriebszustand unterscheidet vier Bereiche. Jeder Bereich definiert Maßnahmen hinsichtlich Sicherheit und Verlässlichkeit. In **Stufe I** ist das System in einem verlässlichen Zustand. Die Ziele hinsichtlich der Verlässlichkeit sind gleichrangig mit den anderen Systemzielen. In der **Stufe II** wird eine Verschlechterung des Systems hinsichtlich der definierten Verlässlichkeitskenngrößen festgestellt. Ein Schwellwert der ersten Stufe wurde überschritten. Das Teilziel Verlässlichkeit wird priorisiert. Bei **Stufe III** ist ein schwerer Fehler aufgetreten (z.B. Ausfall mehrerer Sensoren) und das System ist in einem kritischen Zustand. Um einen drohenden Schaden am System oder am Menschen zu verhindern, werden erste Notfallroutinen initiiert. Das Teilziel Sicherheit wird priorisiert. In der **Stufe IV** sind kritische Komponenten ausgefallen. Das System geht in einen sicheren Zustand und führt Notfallroutinen aus. Dies verhindert unkontrolliertes Verhalten [GRS+14, S. 56]. Das Verlässlichkeitskonzept zeigt Bild 3-8.

Bewertung: Das Konzept adressiert die Zustandsüberwachung und die Verhaltensanpassung für selbstoptimierende Systeme des Maschinenbaus. Es ist interdisziplinär anwendbar. Hervorzuheben ist die Integration in das OCM. Verlässlichkeit und Sicherheit sind inhärente Ziele der Selbstoptimierung. Die Zustandsüberwachung wird über Schwellwert-basierte Verfahren umgesetzt. Einen Ansatz zur Detektion und Bewertung von Anomalien im Verhalten wird nicht berücksichtigt adressiert. Ebenso erfolgt kein Lernen und Adaption auf unbekannte Situationen.

3.2.2.2 Fault Detection, Isolation and Recovery

Zur Steigerung der Verlässlichkeit für sicherheitskritische Systeme der Luft- und Raumfahrt wird das Konzept **Fehler Detektion, Isolation und Recovery** (FDIR) verwendet. Zur Umsetzung von FDIR werden z.B. analytische Methoden, Probabilistische Methoden, Neuronale Netze oder Fuzzy-basierte Ansätze [WF13] eingesetzt. In dem von der ESA[1] geförderten Forschungsprojekt ARPHA wurde ein Prototyp zur Fehler Detektion, Isolation und Recovery für autonome Systeme entwickelt und validiert [PCG+12]. ARPHA nutzt zur Inferenz dynamische probabilistische graphische Modelle. Diese Modelle werden aus RAMS[2] Analysen abgeleitet [PC11]. Das Bild 3-9 zeigt das UML-Zustandsdiagramm für die Diagnostik, Prognose und Recovery. Die **Diagnostik** überwacht den Betriebszustand des Systems. Dies erfolgt durch Dynamische Bayessche Netze. Wird ein Fehler oder eine Anomalie erkannt, wird über die **Recovery** eine Strategie zur Wiederherstellung gewählt. Dies erfolgt auf der Basis von dynamischen Entscheidungsnetzen. Die **Prognose** dient zur Analyse des zukünftigen Betriebszustandes und zur Durchführung von proaktiven Strategien. Der Prototyp wurde am Beispiel der Energieversorgung von dem Mars Rover validiert [PCG+12].

Bewertung: FDIR ist ein umfassendes Lösungskonzept zur Steigerung der Verlässlichkeit. Der Einsatz von Methoden aus der Künstlichen Intelligenz steigert die Autonomie der Systeme. Das hierarchische Konzept berücksichtigt zudem die Struktur des Systems und legt Funktionen für die jeweiligen Ebenen fest. ARPHA basiert auf dem Domänenwissen aus dem Systementwurf. Verfahren zur Anomalie Detektion oder zum Online Lernen werden jedoch nicht als Lösungswissen bereitgestellt. Der Fokus liegt auf der Software.

---

[1] European Space Agency

[2] Reliability, Availability, Maintainablity, Safety

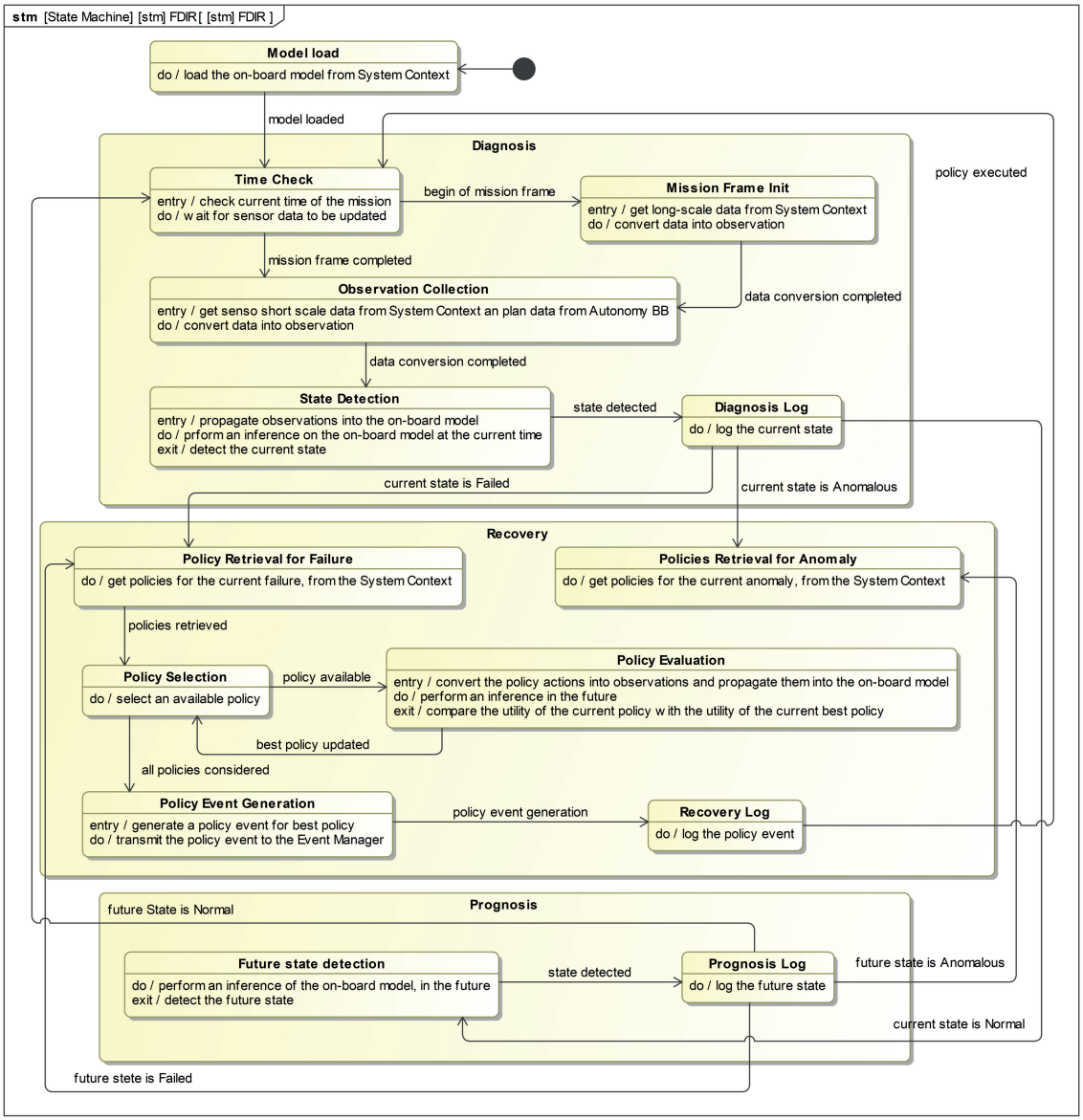


Bild 3-9: ARPHA UML-Zustandsdiagramm [PCG+12]

3.2.2.3 Organic Computing

Im DFG Schwerpunktprogramm (SPP) 1183 wurde eine Referenzarchitektur für O/C-Systeme entwickelt. Die Architektur nutzt den **Observer**, um den Systemzustand zu erfassen und einen **Controller**, um das System entsprechend externer Zielvorgaben zu beeinflussen. Zusammen mit dem **System under Observation and Control (SuOC)** formen Observer und Controller ein O/C-System [BMM+06] (vgl. Bild 3-10).

Das **SuOC** repräsentiert ein komplexes technisches System. Die Strukturierung und Partitionierung von Elementen zu einem SuOC erfolgt nach Kriterien wie z.B. der funktionale Abhängigkeit oder nach der Nutzung von gemeinsamen Ressourcen [Ric09]. Bild 3-6 in Kap. 3.2.1.5 zeigt die Integration vom O/C in das ODP. Der **Observer** beobachtet den Betriebszustand vom SuOC. Der Monitor erfasst die Zustandsgrößen. Anschließend wer-

den über Metriken, Prädiktor und Aggregator die Zustandsinformationen analysiert. Der **Controller** vergleicht die Zustandsinformationen mit den Zielen und greift korrigierend in das Verhalten vom SuOC ein. Der Controller besteht aus zwei Ebenen: Die erste Ebene dient der Aktionsauswahl und nutzt ein regelbasiertes Mapping zwischen Zustand und Aktion. Eine Evaluation bewertet jede Regel. Die zweite Ebene dient zum Lernen und Planen von Regeln. Das Mapping wird optimiert oder neue Regeln durch Mutation und Kombination generiert. Das Lernen erfolgt durch *extended Learning Classifier Systeme* (XCS). Ein Simulationsmodell bewertet jede Regel durch eine Fitness-Funktion [Ric09, S. 78], [BMM+06].

Die Realisierung von O/C-Architekturen in O/C-Systemen hängt von der Anwendungsdomäne und der Komplexität ab. RICHTER unterscheidet verschiedene Varianten zur Umsetzung: Eine zentrale Architektur eignet sich für einfache Systeme. Sie setzt für das komplette System einen O/C-Modul um. Bei verteilten Systemen eignen sich dezentrale O/C-Module, die für jedes Teilsystem eine Observer/Controller Instanz besitzen. Bei komplexen Systemen bieten sich hybride Strukturen, d.h. Mehr-Ebenen oder hierarchische Strukturen an [Ric09, S. 23].

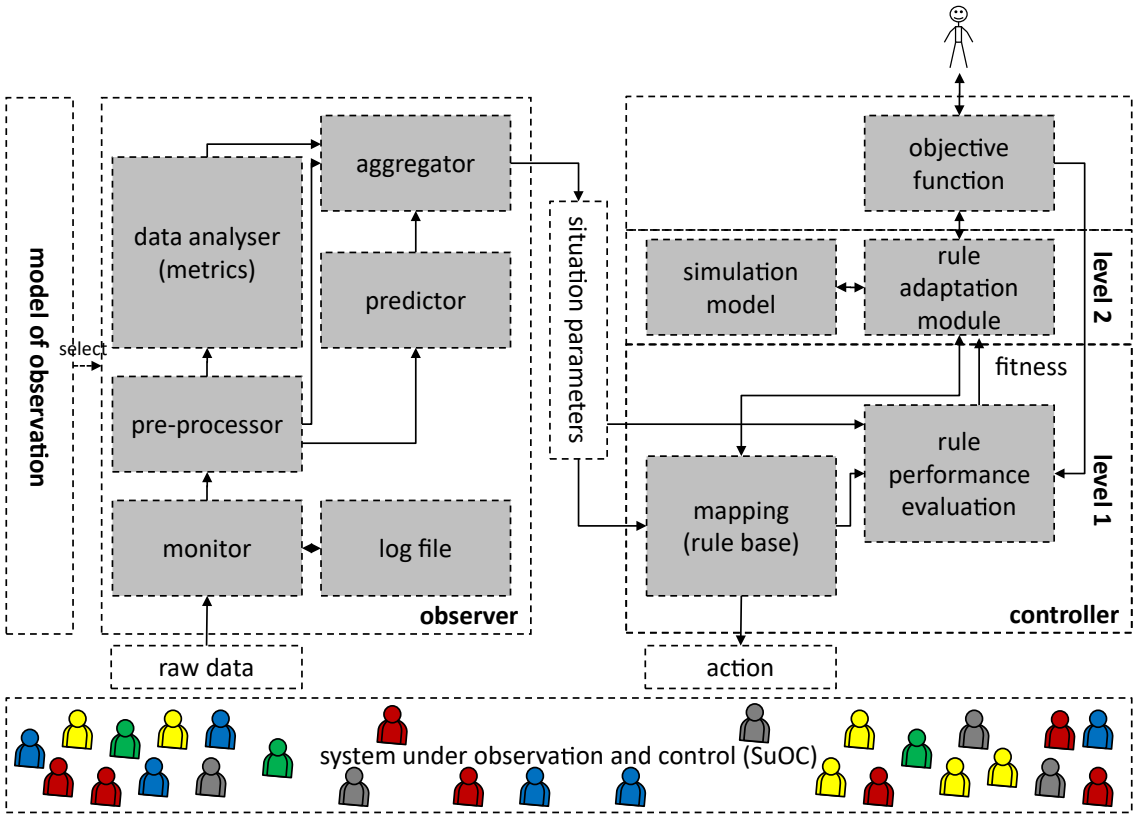


Bild 3-10: Observer Controller Architektur [Ric09, S. 28]

Bewertung: Die O/C-Architektur stellt eine Referenzarchitektur für Self-X-Fähigkeiten bereit. Fokus liegt auf der Fähigkeit zum Online Lernen und Adaption von Reaktionsstrategien durch XCS. Für verschiedene generische Systemklassen werden Architekturkonzept empfohlen. Der Ansatz liefert keine Methode zur Spezifikation und Modellierung. Ebenso

werden Reaktionsstrategien nicht nach Kritikalität berücksichtigt. Hier bietet sich eine Kombination mit dem Verlässlichkeitskonzept (vgl. Kap. 3.2.2.1) an. Vorgehensmodelle und Analysetechnik werden nicht bereitgestellt.

3.2.2.4 Organic Robot Control Architecture

Die **Organic Robot Control Architecture** (ORCA) ist eine dezentrale, modulare und auf den Prinzipien des Organic Computings basierende Architektur. ORCA beinhaltet zwei Module: Die **Basic Control Unit** (BCU) und die **Organic Control Unit** (OCU), die sowohl hierarchisch als auch heterarchisch organisiert sind. BCU und OCU kommunizieren und interagieren über einheitliche Schnittstellen. Die BCU ist das Funktionsmodul zur Realisierung einer Steuerung. Die OCU dient zur Überwachung der Funktionsmodule. Die ORCA-Architektur ist am Beispiel eines Laufroboters in Bild 3-11 visualisiert [Auf10].

Die Funktionsweise einer OCU ist durch die Funktionsweise des biologischen Immunsystems inspiriert. Wie bei dem biologischen Immunsystem wird von einem „gesunden“ Zustand ausgegangen und auf eine Störungen reagiert [Auf10]. Die OCU besteht aus einem Monitor, einem Memory und einem Reasoner. Der **Monitor** dient zur Überwachung der von der BCU generierten Signale. Der Monitor überprüft, ob sich das Signal in einem Normalbereich befindet, oder sich anormale Muster in dem Signal abzeichnen. Sofern keine Abweichung vorliegt, verhält sich die OCU passiv. Bei einer Abweichung wird die Funktionalität der BCU als eingeschränkt, fehlerhaft oder nicht verfügbar eingestuft. Der **Reasoner** greift reaktiv ein [Auf10]. Eine Reaktion kann eine Anpassung von Parameter, eine Modifikation oder eine Deaktivierung und Isolation bis hin zu einer Zielanpassung sein [Auf10], [MSU11]. Die **Memory** dient zum Lernen und zur Selbstoptimierung.

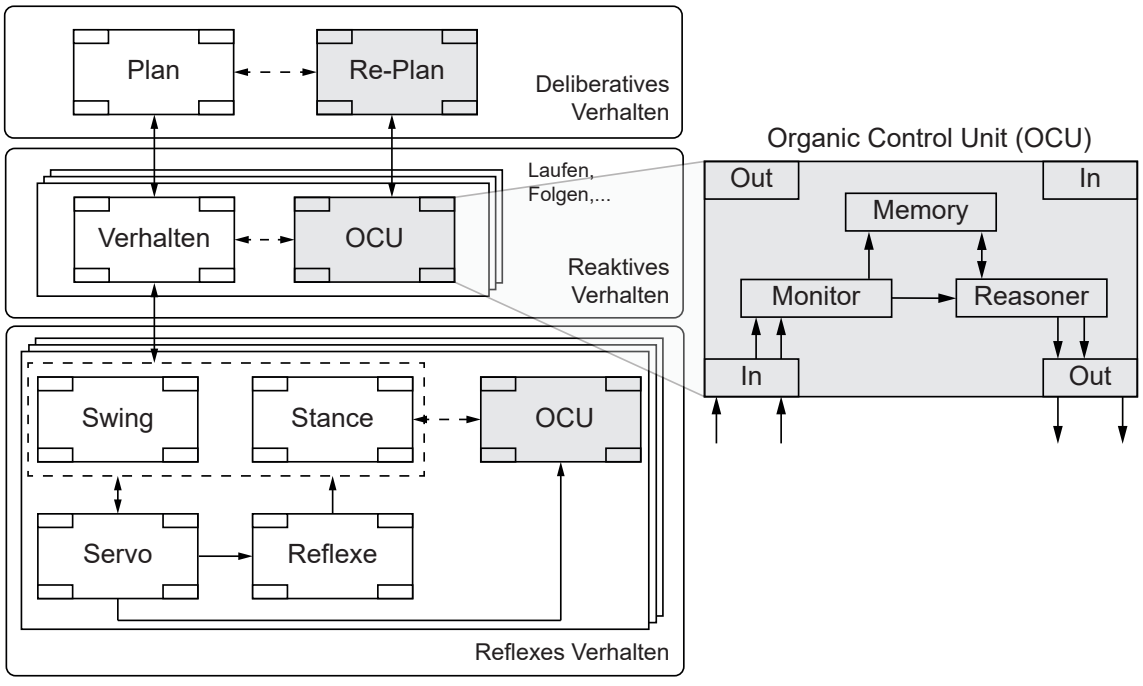


Bild 3-11: Organic Robot Control Architecture [ALM08]

Bewertung: ORCA ist eine integrierte und dezentrale Architektur nach dem Vorbild des biologischen Immunsystems. Es repräsentiert ein ganzheitliches Lösungskonzept. Für das Monitoring werden Verfahren bereitgestellt. Die Validierung erfolgt an einem fortgeschrittenen mechatronischen Systems. Ein Vorgehensmodell zum Entwurf von ORCA und begleitende Analysetechniken werden nicht bereitgestellt.

3.2.2.5 Self-Healing Framework für Cyber-Physische Systeme

RATASICH ET AL. präsentieren ein Framework zur Umsetzung von Selbstheilung in Cyber-Physischen Systemen. Ziel ist die Steigerung der Resilienz. Die Selbstheilung erfolgt durch strukturelle Anpassung, d.h. durch Hinzufügen, Entfernen von Komponenten oder durch das Ändern ihrer Interaktion [RHI+17]. Das Framework umfasst Anforderungen an die Architektur, Entwurfsregeln sowie eine Plattform zur Umsetzung von Selbstheilung. Das Bild 3-12 konkretisiert das Konzept der Selbstheilung. Der **Monitor** überwacht die **Anwendung**. Wird ein Ausfall erkannt, triggert der Monitor die **Rekonfiguration**. Die Funktion der Rekonfiguration ist es, den Ausfall durch eine strukturelle Anpassung zu kompensieren.

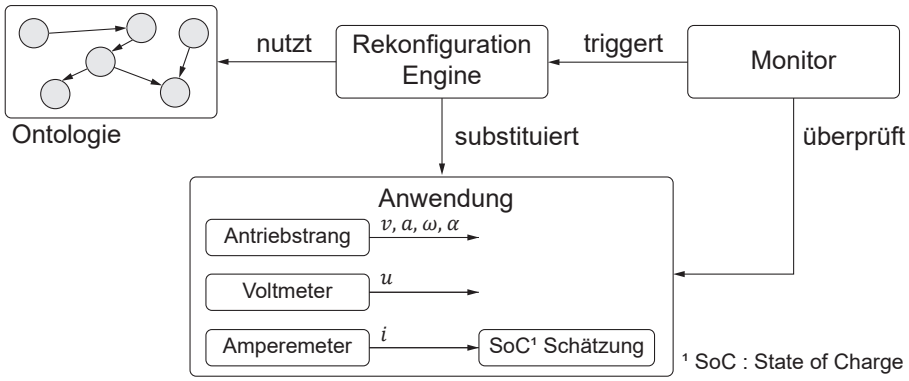


Bild 3-12: Konzept zur Umsetzung von Selbstheilung [RHI+17]

Für die strukturelle Anpassung nutzt die Rekonfiguration eine **Ontologie**. Die Ontologie umfasst semantisches Wissen über die Struktur des Systems, die Eigenschaften der Komponenten sowie Transferfunktionen. Mit Hilfe der Ontologie werden Substitute für eine ausgefallene Einheit oder Funktion ermittelt. Die Ontologie ist in HÖFTBERGER ET AL. vorgestellt [HO13].

Bewertung: Das Framework definiert Anforderungen, Entwurfsregeln und eine Referenzarchitektur zur Umsetzung von Selbstheilung in Cyber-Physischen Systemen. Hervorzuheben ist die Ontologie, die das System als Wissensbasis zur strukturellen Anpassung bei einem Ausfall nutzt. Eine systematische Integration in die Architektur fortgeschrittener mechatronischer Systeme erfolgt nicht. Daher liefert es kein ganzheitliches Lösungskonzept. Ebenso werden keine Verfahren als Lösungswissen bereitgestellt. Eine begleitende interdisziplinäre Entwurfsmethode fehlt.

3.3 Hilfsmittel zur Integration von Selbstheilung

Die Integration von Eigenschaften der Selbstheilung in die Architektur von fortgeschrittenen mechatronischen Systemen erfordert ein systematisches Vorgehen. Dazu werden Hilfsmittel und Vorgehensmodelle für den Systementwurf benötigt. Kap. 3.3.1 erläutert Sprachen zum interdisziplinären Systementwurf. Kap. 3.3.2 untersucht Methoden zur Potenzialanalyse für Selbstheilung in einem technischen System. Kap. 3.3.3 bewertet abschließend Vorgehensmodelle zur Steigerung der Verlässlichkeit.

3.3.1 Sprachen zur Spezifikation komplexer Systeme

3.3.1.1 OMG SysML

Die Unified Modeling Language (UML) hat sich als objektorientierte Modellierungssprache in der Softwaretechnik etabliert. Die UML ist ein internationaler Standard, der von der Object Management Group (OMG) spezifiziert und als ISO/IEC-Standard 19501 veröffentlicht ist. Trotz zahlreicher Initiativen, die Prozesse im Systems Engineering zu standardisieren (ISO/IEC 15288, EIA-632) hat sich bisher noch keine einheitliche Modellierungssprache für das Systems Engineering ergeben. Als möglicher Standard zeichnet sich die OMG Systems Modeling Language (OMG SysML) ab [Wei14, S. 23]. Die SysML wurde im Jahr 2006 von der OMG veröffentlicht und basiert auf der UML v2.5. Seit 2019 ist die SysML in der Version v1.6 veröffentlicht.

Die SysML ist eine objektorientierte Beschreibungssprache und richtet sich an den Ansätze des Systems Engineerings (Spezifikation, Analyse, Entwurf, Validierung und Verifikation) aus. Mit der SysML lassen sich Struktur, Verhalten und Anforderungen eines Systems beschreiben und in Beziehung setzen [GTS14, S.37]. Generell unterscheidet SysML Modelle und Diagramme. Die SysML gliedert sich in neun Diagramme (vgl. Bild 3-13). Das Zusicherungsdiagramm, das Blockdefinitionsdiagramm, das interne Blockdiagramm und das Paketdiagramm gehören zu den *Strukturdiagrammen*. Das Anwendungsfalldiagramm, Zustandsdiagramm, sowie Sequenzen und Aktivitäten gehören zu den *Verhaltensdiagrammen*. Zusätzlich gibt es das Anforderungsdiagramm. Die Diagramme repräsentieren Sichten auf das Modell. Das Modell umfasst alle Aspekte. Das Konzept der SysML-Sprache ist so aufgebaut, dass die Sprache für unterschiedliche Modellierungszwecke angepasst werden kann und so Dialekte gebildet werden können. Hierzu können Sprachprofile, oder Modellsichten definiert werden [FMS12, S. 357ff.], [Wei14, S. 23ff.].

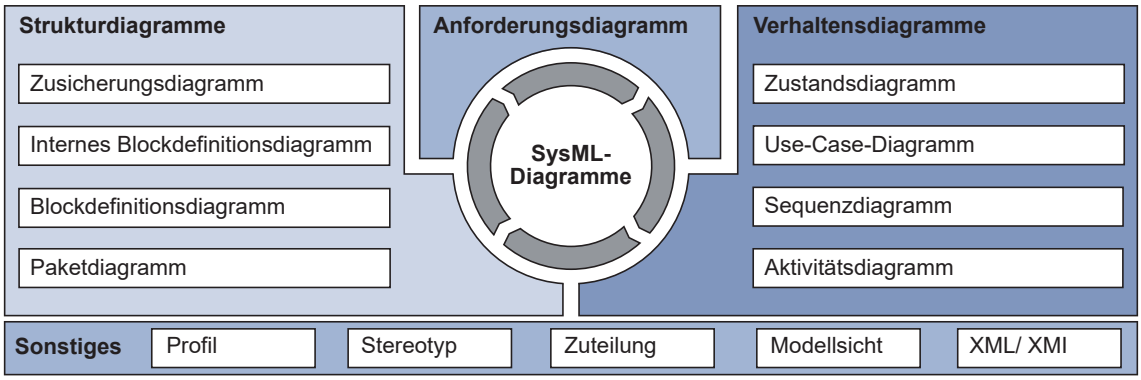


Bild 3-13: Diagramme der SysML [GTS14, S. 37]

Bewertung: Die SysML liefert ein umfangreiches und standardisiertes Sprachkonzept zur Spezifikation komplexer technischer Systeme. Die Trennung von Modell und Sicht reduziert die Komplexität. Die Formalisierung unterstützt die Durchführung einfacher Analysen. Durch Profilbildung besitzt die SysML eine hohe Anpassungsfähigkeit für verschiedene Modellierungszwecke. So können Aspekte der Verlässlichkeit oder der Absicherung durch ein Profil umgesetzt werden. In der praktischen Anwendung zeigt die SysML einige Defizite: Sie ist zunächst eine reine Sprache. Eine standardisierte Methode existiert nicht. Die abstrakten Darstellung der Konstrukte reduziert die Lesbarkeit und Akzeptanz für ihre interdisziplinäre Anwendung.

3.3.1.2 CONSENS

Die Spezifikationstechnik CONSENS[3] wurde am Heinz Nixdorf Institut mit DFG SFB 614 *Selbstoptimierende Systeme des Maschinenbaus* entwickelt [GFD+08] und seitdem kontinuierlich um weitere Aspekte erweitert [GDE+19], [GTS14, S. 37 ff.]. Die Spezifikationstechnik liefert ein Set an Beschreibungsmitteln zur ganzheitlichen und disziplinübergreifenden Modellierung und Analyse eines mechatronischen Systems [GDE+19]. Das Ziel ist die integrative und modellbasierte Erarbeitung der Prinziplösung im Rahmen des interdisziplinären Systementwurfs. Die Sprache setzt sich aus sieben kohärenten Aspekten zusammen, den sog. Partialmodellen. Die Aspekte sind *Anwendungsszenarien, Umfeld, Anforderungen, Funktionen, Wirkstruktur, Gestalt* und *Verhalten*. Der Aspekt Verhalten umfasst *Zustandsautomaten, Sequenzen* und *Aktivitäten*. Die Erarbeitung der Aspekte erfolgt iterativ. Erkenntnisse aus einem Aspekt (z.B. Funktion) sind mit den anderen Aspekten (z.B. Verhalten, Umfeld) abzugleichen. Eine Übersicht der Aspekte zeigt Bild 3-14 [GFD+08], [GFD+08], [GTS14].

[3] CONceptual design Specification technique for the ENgineering of complex Systems

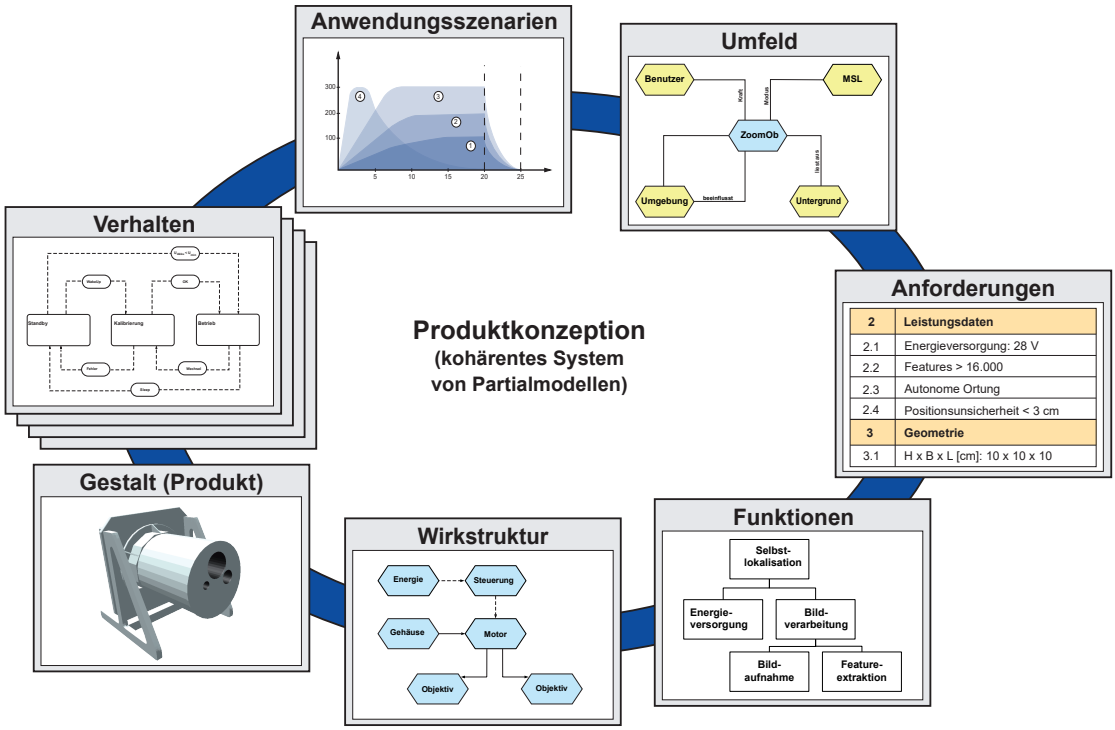


Bild 3-14: Aspekte zur Beschreibung der Prinziplösung [GFD+08]

Die Menge der Partialmodelle repräsentiert die Prinziplösung des mechatronischen Systems. Sie bilden die Basis für die disziplinspezifische Ausarbeitung. Die Sprachkonstrukte können durch ein SysML-Profil für SysML-Autorenwerkzeuge nutzbar gemacht werden. Erweiterung der Sprache ermöglichen die integrativen Konzipierung von Dienstleistung und Produktionssystem [GDE+19].

Bewertung: CONSENS ist eine semi-formale Sprache zur interdisziplinären Beschreibung fortgeschrittener mechatronischer Systeme. Dabei orientiert sich die Sprache am Systementwurf. Die Anwendung ist intuitiv und für die beteiligten Disziplinen einfach zu verstehen. Modellierungsregeln und Richtlinien stellen die Vergleichbarkeit, Vollständigkeit und Richtigkeit sicher [Kai14]. Die Sprache ist leicht zu erweitern und Aspekte zur Absicherung oder Verlässlichkeit können integriert werden. CONSENS besitzt bisher kein eigenständiges Autorenwerkzeug.

3.3.1.3 Object Process Methodology

Die *Open Process Methodology* (OPM) wurde 2002 von DORI veröffentlicht [Dor16]. Die Sprache vereinigt objekt- und prozessorientierte Aspekte der Systembeschreibung in einem Sprachkonzept. OPM ist in der ISO/PAS 19450 AUTOMATION SYSTEMS AND INTEGRATION standardisiert [ISO19450]. Die OPM nutzt verschiedene Sprachkonstrukte, die in einem Meta-Modell definiert sind: Zur Modellierung von Struktur und Verhalten dienen Objekte und Prozesse. Objekte sind physikalische Dinge, die einen Zustand besitzen. Prozesse sind Dinge, die Objekte transformieren, d.h. erstellen, zerstören oder

deren Zustand ändern. Prozesse und Objekte werden über Verbindungen verknüpft. Die OPM unterscheidet prozedurale und strukturelle Verbindungen. Prozedurale Verbindungen verbinden Prozesse und Objekte und beschreiben eine Transformation. Strukturelle Verbindungen verknüpfen Objekte zu einer Struktur. Die Sprachkonstrukte werden in OPM in einem Diagramm visualisiert[Dor16, S.19ff.]. Dies ist der wesentliche Unterschied zur SysML, die auf dem Konzept der Trennung von Sicht und Modell aufbaut [CCS16, S. 59]. Mit den Open Source Werkzeugen OPCAT und der OPCloud können Systeme mit der OPM modelliert werden. Das Bild 3-15 zeigt die Anwendung von OPM am Beispiel eines Automatischen Nofallsystems.

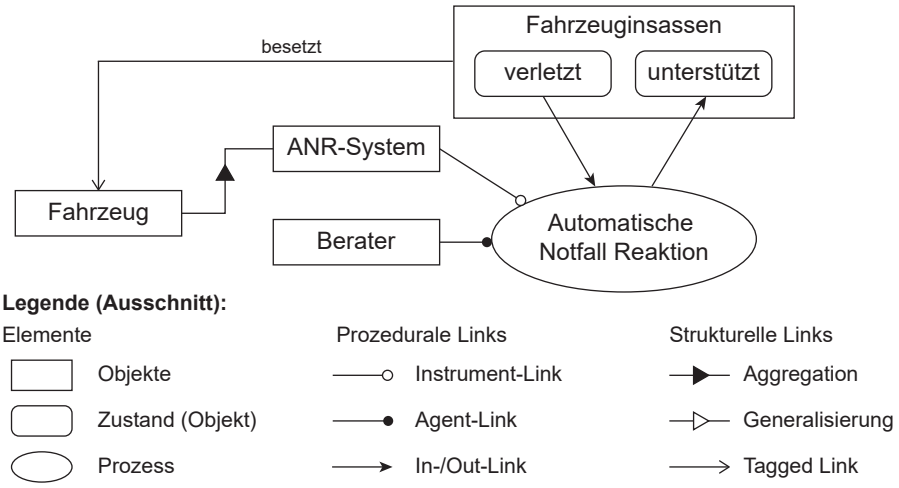


Bild 3-15: Modellierung eines Notfallsystems mit OPM [Dor16, S. 22]

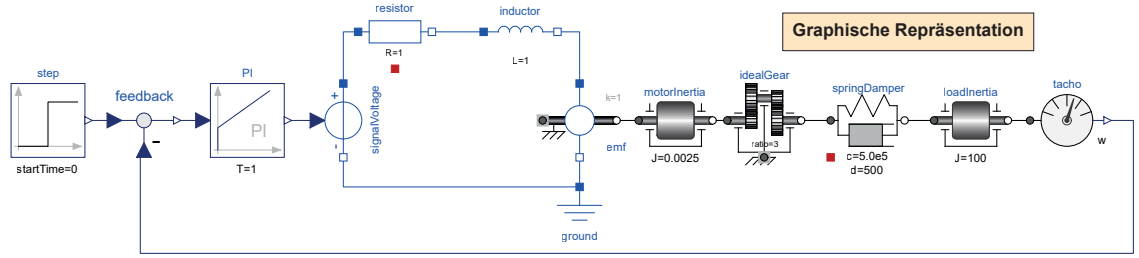
Bewertung: OPM bietet eine einfache und disziplinübergreifende Sprache zur Beschreibung komplexer Systeme und Architekturen. Die Sprache unterstützt die Synthese im Systementwurf. Umfangreiche Modelle können an Übersichtlichkeit verlieren. Ebenfalls ist es nicht möglich, Anforderungen an das System zu beschreiben. Zuständen und Prozessen beschreiben das Verhalten. Absicherungskonzepte und Aspekte der Verlässlichkeit können mit OPM nicht spezifiziert werden. Ebenso fehlen Analysetechniken.

3.3.1.4 Modelica

Modelica ist eine offene objekt- und topologieorientierte Sprache zur Modellierung und Simulation komplexer Systemen. Die Sprache wird seit 1997 durch die *Modelica Association* weiterentwickelt und ist in der Version 3.4 (Stand 2017) veröffentlicht. Mit der Sprache werden komplexe physikalische Modelle aus verschiedenen Domänen erstellt und miteinander verknüpft [Mod18-olb]. Die Verknüpfung der Energieflüsse erfolgt über Ports, d.h. nach ihrer Topologie. Informationsflüsse, wie etwa Sollwerte einer übergeordneten Regelung, werden dagegen nach ihrem Signalfluss modelliert. Neben Parametern und Komponenten finden sich in den objektorientierten Modellklassen die mathematischen Gleichungen. Dies können Differentialgleichungen, algebraische Gleichungen oder diskrete Gleichungen sein [Mod18-olb]. Zusätzlich zur Sprache stellt die Modelica Association

eine *Modelica Standard Library* zur Verfügung. Diese Open-Source Bibliothek umfasst eine Vielzahl an Domänen, z.B. Elektronik, Thermodynamik, oder Mechanik.

Zur Simulation der Modelle sind kommerzielle und freie Werkzeuge verfügbar (z.B. SimulationX, Dymola, OpenModelica). Diese Werkzeuge generieren aus den Modellen die erforderlichen Gleichungen und optimieren diese mit Hilfe spezieller Algorithmen. Zudem bieten die meisten Werkzeuge grafische Hilfestellungen, z.B. durch grafische Modellierung oder durch die Visualisierung der dreidimensionalen, mechanischen Anteile des Systems [GTS14]. Bild 3-16 zeigt das Multidomänen-Modell von einem Elektromotor in der graphischen und textuellen Repräsentation sowie das Ergebnis der Simulation.



model Antriebssystem
EMF emf(k = 1);
SpeedSensor tacho;
Inertia motorInertia(J = 0.0025);
Inertia loadInertia(J = 100);
IdealGear idealGear(ratio = 3);
SpringDamper springDamper(c = 5.0e5, d = 500);
...
equation
connect(loadInertia.flange_b, tacho.flange);
connect(springDamper.flange_b, loadInertia.flange_a);
connect(idealGear.flange_b, springDamper.flange_a);
connect(motorInertia.flange_b, idealGear.flange_a);
connect(emf.flange, motorInertia.flange_a);
...
end Antriebssystem;

Textuelle Repräsentation

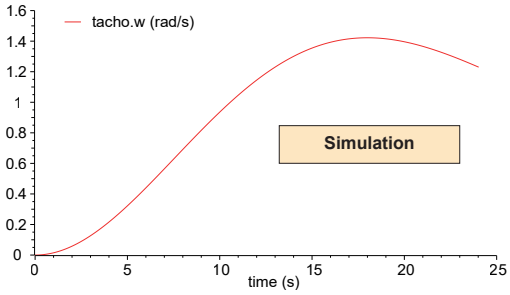


Bild 3-16: Multi-Domänenmodell von einem Elektromotor in Open Modelica

Die modelltechnische Verknüpfung von Modelica und der UML wird durch die *Open Modelica* Initiative vorangetrieben [Sch11-ol]. Die *ModelicaML* erweitert die graphische Repräsentation von Modelica um weitere UML-Diagramme (Struktur, Verhalten). Dies ermöglicht eine integrative Simulation von diskreten und dynamischen Verhalten. ModelicaML ermöglicht die Formalisierung und Evaluierung von Anforderungen durch Simulation [Sch11-ol]. Die Sprache unterstützt den Standard FMU/FMI (*functional mock-up unit/ functional mock-up interface*). Durch FMI lassen sich die Modelle verschiedener Werkzeuge und Solver koppeln und zu einem ausführbaren Systemmodell zusammensetzen [Mod18-ola], [Mod18-olb].

Bewertung: Modelica ist eine leistungsfähige formale Sprache zur multidomänen Modellierung und Analyse komplexer Systeme. Die Modelle können mit unterschiedlichem Detaillierungsgrad erstellt werden. Dies ermöglicht es, Verhalten auf Basis plausibler Modelle frühzeitig zu analysieren. Modellica besitzt eine umfangreiche Bibliothek. Mittels FMU/FMI werden weitere Domänenmodelle gekoppelt. Modelica eignet sich zur Analyse des Verhaltens und dient somit als Ergänzung der Systemspezifikation. Ein einheitliche Methode zur Modellierung und Analyse mit Modelica existiert nicht.

3.3.2 Ausgewählte Ansätze zur frühzeitigen Analyse

3.3.2.1 Zuverlässigkeitsanalysen

Zuverlässigkeitsanalysen ermitteln und bewerten die erwartete Zuverlässigkeit eines Systems. Die Tabelle 3-1 fasst ausgewählte Methoden zur Analyse der Zuverlässigkeit, die Art der Analyse alphabetisch zusammen.

Tabelle 3-1: Ausgewählte Methoden zur Analyse und Bewertung der Zuverlässigkeit

Name	Beschreibung	Art	Quelle
DFT	Die dynamische Fehlzustandsbaumanalyse erweitert die FTA, um funktionale Abhängigkeiten, Redundanzen/ Reserven und zeitliche Abfolge.	Fehlerrate	[DBB92a]
FMEA/ FMECA	Die Fehler-Möglichkeits-Einfluss-Analyse ist ein qualitatives, induktives Verfahren, welches Ausfallarten einer Einheit, die Ausfallfolge auf andere Einheiten sowie die Ausfallursache aufzeigt und mit einem Risikowert bewertet.	Fehlerart	[BL04]
FTA	Die Fehlzustandsbaumanalyse ist ein deduktives Verfahren, um mögliche Kombination von Ereignissen festzustellen, die zu einem unerwünschten Hauptereignis führen.	Fehlerart/ Fehlerrate	[DIN61025]
Markov	Eine Einheit wird durch Zustände und Übergänge dargestellt. Jeder Zustand entspricht einer Kombination aus funktionsfähigen und ausgefallenen Bestandteilen. Jeder Übergang ist durch eine Wahrscheinlichkeit repräsentiert. So kann die Wahrscheinlichkeit für einen Systemzustand analytisch bestimmt werden.	Fehlerrate	[VDI4008]
Vorhersagetechniken	Bezeichnet Zuverlässigkeitstechniken zur probabilistischen Vorhersage von Ausfallraten auf Basis von Verteilungsfunktionen (z.B. Weibull-Verteilung). Ziel ist die Optimierung der Funktionsfähigkeit eines Designs im Vergleich mit anderen Kenngrößen.	Fehlerrate	[BL04]
„Worst-Case“-Analysen	Nachweis einer Einheit, dass sie trotz bestimmter Änderung von Parametern ihre Funktion in der vorgegebenen Umgebung erfüllt.	Fehlerrate	[ECSS-Q-ST-30C]

Die Methoden dienen zur präventiven Erkennung und Beseitigung von Schwachstellen im System. Diese Analysen sind im Systementwurf systematisch anzuwenden und die resultierenden Maßnahmen sind in der Anforderungsanalyse und der Synthese zu berücksichtigen. Es werden zwei Formen der Analyse verwendet: **Qualitative Analysen** unterstützen bei

der Ermittlung von Fehlerarten. **Quantitative Analysen** dienen der Berechnung vorausgesagter Zuverlässigkeit in Form von Fehlerraten [BGJ+09]. In der Literatur existiert eine Vielzahl an Methoden und Techniken die z.B. von der ISO 61508, ISO 26262 oder der ECSS-Q-30 gefordert werden [IEC61508b], [ISO26262], [ECSS-Q-ST-30C].

Bewertung: Die Methoden zur Analyse der Zuverlässigkeit finden ihre Anwendung im interdisziplinären Systementwurf. Durch den Einsatz von qualitative Methoden können Schwachstellen frühzeitig im System identifiziert werden. Dynamische Fehlerbäume können in probabilistische Netze überführt werden und später für die Diagnostik im System verwendet werden [Son15]. Die Anwendung der Methoden erfolgt meistens isoliert, d.h. nicht integrativ mit dem Spezifikationsmodell.

3.3.2.2 Sicherheitsanalyse

Der Begriff Sicherheitsanalyse umfasst die Gefahrenanalyse, Gefährdungsanalyse und die Risikoanalyse. Diese Analysen reichen von der Identifikation und qualitativen sicherheitstechnischen Bewertung möglicher Gefahren auf Mensch, Maschine und Umwelt bis hin zu quantitativen Analysen, bei denen Wahrscheinlichkeiten und Schadensschwere ermittelt werden. Das gemeinsame Ziel ist die Vermeidung von Gefahren durch präventive Maßnahmen [VDI4003].

Die **Environmental Survey Hazard Analyse** (ESHA) eine qualitative Sicherheitsanalyse für mobile Roboter. Der Fokus dieser Methode liegt in der Analyse von unbeabsichtigten Interaktionen zwischen dem System und seiner Umwelt und die Bewertung von resultierenden Gefahrensituationen [DGH+14]. Die ESHA umfasst eine Taxonomie, Assoziationslisten sowie ein Formblatt zur Bewertung und Dokumentation. Die Taxonomie klassifiziert die Umgebung in drei Bereiche:

- **Umgebung und Umwelt:** Beschreiben die Umgebung in dem der Roboter und andere Objekte existieren. Dies sind statische Merkmale wie das Terrain oder dynamische Merkmale wie das Ambiente, z.B. Witterung.
- **Hindernisse und Objekte:** Entitäten, die innerhalb der statischen Umgebung platziert sind. Sie können in der Umgebung statisch, beweglich oder aktiv sein. Das Verhalten ist nicht zielorientiert.
- **Agenten und Akteure:** Aktive und dynamische Entitäten in der Umgebung mit einem zielorientierten Verhalten. Dies sind z.B. automatisiert, autonome Systeme, Tiere oder Menschen.

Für diese Bereiche stellt die ESHA Leitfragen und Assoziationslisten bereit. Sie unterstützen bei der Identifikation von Interaktionen. Das Formblatt ist exemplarisch in Bild 3-17 dargestellt. Zunächst werden relevante Aspekte der Umgebung anhand der Taxonomie identifiziert. In der nächsten Spalte werden die Details zur Interaktion ermittelt. Diese können sich zum Beispiel aus der Aufgaben- oder Missionsanalyse des Systems

ergeben. Generische Leitwörter unterstützen bei der Identifikation von Störungen. Die Details beinhalten eine Beschreibung der Störung. Die Konsequenz beschreibt mögliche Gefahren für den Roboter und der Umgebung. Abschließend wird für jede Konsequenz eine Sicherheitsmaßnahme definiert [DGH+14].

System:		Mobiler Saugroboter				
Bearbeiter:		Michael Hillebrand	Revision:	1.3	Datum:	Dez. 2018
Ref.	Objekte	Interaktion	Fehler / Störung	Details	Konsequenz	Sicherheitsmaßnahme
1.	Treppe	Kein Treppensteigen möglich	Treppe nicht erkannt	Roboter fährt vorwärts	Roboter stürzt die Treppe hinunter	Kamerasensor zur Umfelderkennung (Treppen)
2.	Wand	Abstand erfassen	Abstandserfassung ausgefallen	Roboter fährt gegen die Wand	Dellen am Roboter	Integration von Bumpern zur Kollision, Material
...	...	...	...	...	...	...

Bild 3-17: Formblatt am Beispiel Mobiler Roboter [DGH+14]

Bewertung: Die Methode bietet einen pragmatischen Ansatz zur systematischen Identifikation von Gefahrensituationen. Mit der Methoden können Sicherheitsmaßnahmen frühzeitig im Systementwurf definiert und textuell dokumentiert werden. Charakteristisch ist der interdisziplinäre Einsatz auf der Systemebene. Die Methode unterstützt nur eingeschränkt den modellbasierten Systementwurf. Eine Gefahrenbewertung findet nicht statt.

3.3.2.3 Sneak-Analyse

Ziel der Sneak-Analyse ist die Identifikation von „Sneak-Kreisen“. Sneak-Kreise sind unerwartete Pfade für einen Fluss von Masse, Energie oder einer logischen Abfolge, die aufgrund von latenten Zuständen oder Bedingungen im System auftreten. Diese können eine unerwünschte Funktion auslösen oder eine erwünschte Funktion verhindern [ECSS-Q-40A-1].

Die Begriff **Sneak-Analyse** repräsentiert eine Gruppe analytischer Techniken zur Ermittlung von Sneak-Kreisen und Schwachstellen in einem System: Die **Sneak-Pfadanalyse** ist eine Technik, welche auf der Identifikation von Pfaden zwischen „Senken “ und „Quelle“ und der Anwendung von Suchmerkmalen beruht. Die **Schwachstellenanalyse** basiert auf der Anwendung von Suchmerkmalen. Suchmerkmale sind spezifische Fragen, die auf eine latenten Funktionsstörung hindeuten. Dabei beziehen sich Suchmerkmale auf Pfade, Einheiten oder einer Kombination von beiden, wie zum Beispiel die Frage: „*Können bei Umschaltvorgängen vorübergehende Strompfade auftreten?*“ [ECSS-Q-40-4A-2], [ECSS-Q-40-04A].

Die Durchführung der Sneak-Analyse erfolgt in drei Schritten: In der **Vorbereitung** wird der Umfang der Analyse festgelegt, das System in Funktionen und in Einheiten gegliedert, sowie die Betriebsarten und Zustände der Einheiten festgelegt. Die **Analyse** ist der zweite Schritt. In der Sneak-Pfadanalysen werden Senken und Quellen sowie deren Pfade identifiziert. Die Suchmerkmale werden auf die im Pfad enthaltenen Einheiten angewendet. Anschließend erfolgt die Schwachstellenanalyse durch Anwendung von

Suchmerkmalen. Abschließend werden die Folgen der Schwachstellen und Sneak-Pfaden bewertet und Maßnahmen festgehalten. Zuletzt erfolgt die **Dokumentation** der Ergebnisse und der entsprechenden Maßnahmen [ECSS-Q-40-04A].

Bewertung: Mit der Sneak-Analyse werden Schwachstellen im System identifiziert. Hervorzuheben ist die Analyse der Flussbeziehungen, die mit Hilfe von Suchmerkmalen bewertet werden. Fokus liegt auf pneumatischen, elektronischen, hydraulischen und softwareintensiven Systemen. Die Analyse ist interdisziplinär anwendbar. Die detaillierten Fragestellungen setzen eine detaillierte Systemgestaltung voraus. Die Anwendung im Systementwurf ist daher eingeschränkt. Es fehlt an einem modellbasierten und integrativen Ansatz mit dem die Sneak-Analyse mit den Spezifikationsmodell gekoppelt werden kann.

3.3.2.4 System-Theoretic Process Analysis

Die **System-Theoretic Process Analysis** (STPA) unterstützt die Gefährdungsanalyse komplexer Systeme in der frühen Phase des Systementwurfs. Die Methode wurde 2004 von LEVESON veröffentlicht. Die STPA basiert auf einem systemtheoretischen Verständnis von Gefährdungen. Demnach entstehen Unfälle nicht nur durch einen Fehlzustand von Komponenten und deren Ereignisfolgen, sondern vielmehr aus der Interaktion der Komponenten untereinander und mit der Umwelt. Verlässlichkeit, insbesondere Sicherheit und Zuverlässigkeit, ist in diesem Verständnis eine emergente Eigenschaft, die aus dieser Interaktion entsteht. Ziel der STPA ist die Identifikation kritischer Gefährdungszenarien die zu Unfällen führen [Lev04].

Die STPA besteht aus vier Schritten: Im ersten Schritt werden mögliche Unfälle und Gefährdungen identifiziert. Im zweiten Schritt wird die sozio-technische Kontrollstruktur des Systems definiert. Anschließend werden mit Hilfe der Kontrollstruktur inadäquate Kontrollaktionen ermittelt, die zu den Gefährdung führen können. Zum Beispiel kann eine nicht ausgeführte, oder eine nicht gewollte Kontrollaktion zu einer Gefahr oder einem Unfall führen. Zur Unterstützung dieses dritten Schritts stellt STPA Leitwörter zur Verfügung mit denen die Kontrollaktionen analysiert werden. Im letzten Schritt werden kausale Faktoren ermittelt, die in ihrer Kombination zum Fehlverhalten der Kontrollaktion führen könnten. Aus diesen werden die bestehenden Sicherheitsanforderungen abgeleitet und verfeinert [Lev14]. Zur Unterstützung der Anwendung von STPA für sicherheitskritische Softwaresysteme wurde das Werkzeug XSTAMPP entwickelt [Abd17].

Bewertung: Die STPA betrachtet die Kontrollstruktur sozio-technischer Systeme sowie dessen Wechselwirkung mit der Umgebung. Dadurch können Gefährdungen bereits im Systementwurf ermittelt werden. Dies stellt einen wesentlichen Vorteil z.B. gegenüber der Fehlerbaumanalyse dar, die die Funktionsfähigkeit einer Komponente fokussiert. Die Anwendung erlaubt eine Priorisierung von Gefährdungsszenarien und den damit verbundenen Sicherheitsanforderung. Eine integrative Modellierung mit dem Spezifikationsmodell wird nicht unterstützt.

3.3.3 Vorgehen zur Steigerung der Verlässlichkeit

3.3.3.1 ISO/PAS 21448: SOTIF

Die ISO/PAS 21448 Safety Of The Intended Functionality (SOTIF) befasst sich mit der Sicherheit der intendierten Funktionen in autonomen Fahrzeugen ab dem zweiten Automatisierungsgrad [ISO 21448] (vgl. Kapitel 2.2.3). Damit ergänzt SOTIF die ISO 26262, welche die Standards für die funktionalen Sicherheit von Straßenfahrzeuge für sicherheitskritische E/E-Systeme festlegt [ISO26262]. SOTIF adressiert Risiken, die sich aus unbekannten und unsicheren Systemzuständen wie etwa den folgenden ergeben [ISO 21448, S. 7f.]:

- Beeinträchtigungen, die sich aus Grenzen der Sensorik oder Aktorik ergeben.
- Beeinträchtigungen, die sich aus der limitierten Leistung eines Systems ergeben.
- Beeinträchtigungen, die sich aus einer Fehlanwendung des Benutzers ergeben.

Kern der ISO/PAS 21448 ist das in Bild 3-18 dargestellte Vorgehen. Das Vorgehen integriert sich in das V-Modell und umfasst mehrere aufeinander aufbauende Phasen. Es ergänzt somit die Schritte der ISO 26262.

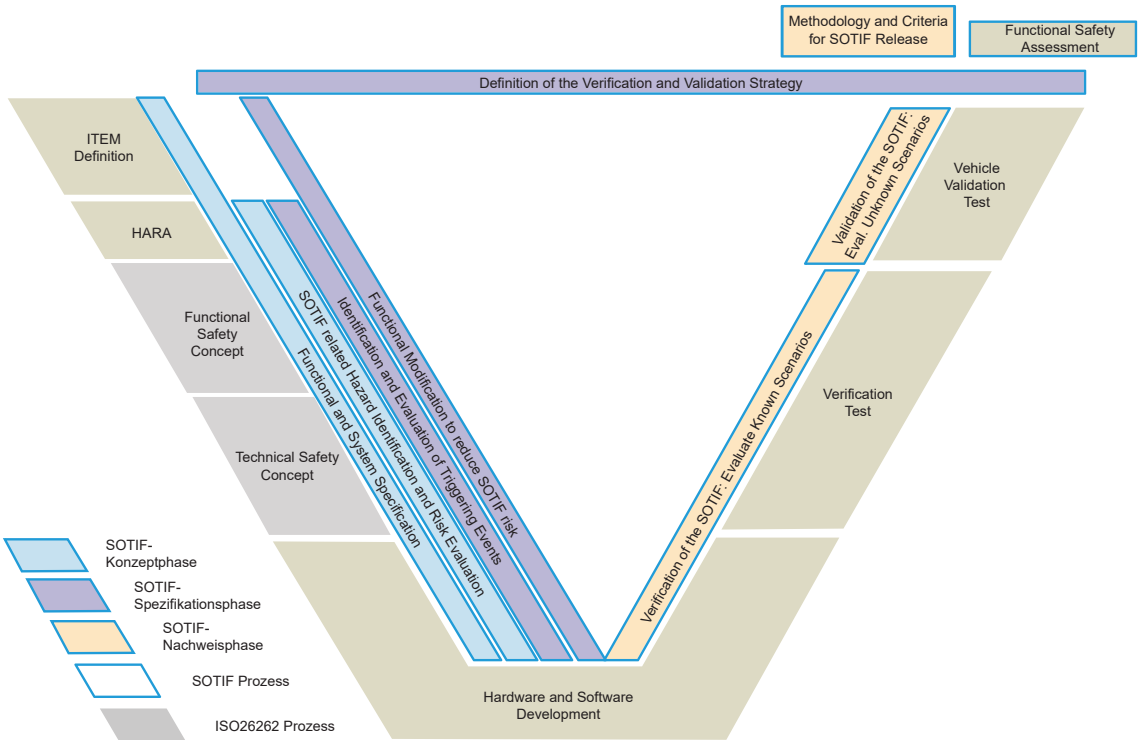


Bild 3-18: SOTIF-Vorgehen in der ISO/PAS 21448 [ISO 21448, S. 10]

Das Vorgehen gliedert sich in drei Phasen: Kern der **Konzeptphase** ist die Erstellung der Funktions- und Systemspezifikation in Verbindung mit der Identifikation von SOTIF-Risiken. In der **Spezifikationsphase** erfolgt die Planung der Verifikation und Validierung sowie die Identifikation und Bewertung von gefährlichen Anwendungsfällen. Auf dieser

Basis werden Maßnahmen zur Reduktion der SOTIF-Risiken identifiziert. In der **Nachweisphase** erfolgt die Verfikation, Validierung und die Freigabe. Nach der Freigabe werden strukturiert Felddaten erfasst mit denen Abweichungen der spezifizierten Sollfunktion erfasst und analysiert werden. Für die einzelnen Phasen empfiehlt die ISO/PAS 21448 Methoden und Entwurfsechniken [ISO 21448, S. 10ff.], [SH20, S. 15ff.].

Bewertung: Die ISO/PAS 21448 stellt ein gutes Vorgehensmodell zur Realisierung sicherheitskritischer Funktionen für das autonome Fahren bereit. Hervorzuheben ist die frühzeitige Verfikation und Validierung sowie die konsequente Fokussierung auf Fahrszenarien. Dabei fokussiert es unsichere und unbekannte Systemzustände. Das Vorgehen stellt jedoch keine Methoden und Architekturen als Referenz zur Verfügung. Es bleibt jedoch vage, wie die Schritte erfolgen und die Hilfsmittel zusammengreifen.

3.3.3.2 Zustandsüberwachung für S.O.-Systeme

SONDERMANN-WÖLKE stellt eine Systematik zum Entwurf und zur Anwendung einer erweiterten Zustandsüberwachung in dem mehrschichtigen Verlässlichkeitskonzept für selbstoptimierende Systeme vor (vgl. Kap. 3.2.2.1). Das Vorgehen orientiert sich an der DIN ISO 17359. Ziel ist eine durchgehende Methode zur Bewertung der Verlässlichkeit von der Entwurfs- bis zur Betriebsphase selbstoptimierender Systeme [Son15, S. 40ff.]. Das Vorgehen ist in Kapitel A1.2 dargestellt. Es gliedert sich in die Entwurfsphase und in die Betriebsphase

In der **Entwurfsphase** werden die Eigenschaften und die Aspekte der Verlässlichkeit des Systems auf Basis der Prinziplösung analysiert und bewertet. Kern sind Strategien zur Zustandsüberwachung für die unterschiedlichen Konfigurationen des Systems mittels des mehrstufigen Verlässlichkeitskonzeptes. Damit verbunden ist die Verlässlichkeitsanlyse und -bewertung sowie die Ausarbeitung der Konfigurationssteuerung. Die Spezifikation der Prognose und Diagnose erweitern die Spezifikation der Prinziplösung. Anschließend erfolgt die Ausarbeitung in der domänenspezifischen Ausarbeitung [Son15, S. 43f.].

In der **Betriebsphase** werden die relevanten Zustandsgrößen überwacht und in Kenngrößen der Verlässlichkeit umgerechnet. Diese Kenngrößen werden anschließend mit den Schwellwerten des Verlässlichkeitskonzeptes verglichen. Bei einer Über- bzw. Unterschreitung dieser Schwellwerte wird eine Diagnose und Prognose durchgeführt. Zudem werden Verhaltensanpassungen im System initiiert (z.B. Konfigurationssteuerung, Adaption von Regelstrategien) oder eine Instandhaltungsmaßnahme geplant. Die Auswertung der Prognose gewährleistet ein proaktives Handeln des selbstoptimierenden Systems [Son15, S. 57]. Die erweiterte Zustandsüberwachung erhöht die Verfügbarkeit und die Sicherheit des Systems [Son15, S. 43].

Bewertung: Das Vorgehen bietet einen guten Ansatz zur Zustandsüberwachung selbstoptimierender Systeme. Es sieht eine frühzeitige Analyse auf Basis der Prinziplösung vor und unterstützt so die Potenzialanalyse. Durch das mehrstufige Verlässlichkeitskonzept werden

Instandhaltungsstrategien und das Notfallmanagement im interdisziplinären Systementwurf berücksichtigt. Aspekte des Online Lernens und der Anomalie Detektion werden im Konzept nicht adressiert. Fortgeschrittene Analysetechniken zur frühzeitigen Absicherung werden nicht betrachtet.

3.3.3.3 Integriertes FDIR Design nach BITTNER ET AL.

BITTNER ET AL. stellen ein Vorgehen zur systematischen Betrachtung von FDIR [4] im Entwicklungsprozess vor. Das Vorgehen orientiert sich an der ECSS-E-ST-10 C: SPACE ENGINEERING: SYSTEMS ENGINEERING GENERAL REQUIREMENT [ECSS-E-ST-10C]. Es umfasst sechs Schritte: In der **Analyse der Benutzeranforderungen** werden die Anforderungen an RAMS[5] definiert und die Missionsphasen und die Betriebsarten des Systems definiert. Im zweiten Schritt erfolgt die **Partitionierung und Allokation**. Die Anforderungen werden auf die Missionsphasen und Betriebsarten des Systems allokiert. Die FDIR-Architektur (Funktionale Dekomposition, HW/SW Partitionierung, Redundanzen und FDIR-Levels) des Systems wird erarbeitet. Die **Analyse der Fehlerfortpflanzung** erfolgt durch TFP-Modelle (Time Failure Propagation Models) (siehe [AKM+09]). Das Modell nutzt die Ergebnisse der RAMS Analysen wie etwa FMEA, Gefahrenanalyse, oder die FTA. Die **Definition der FDIR Ziele und Strategien** erfolgt im vierten Schritt. Die FDIR-Ziele definieren das Systemverhalten im Falle einer Störung oder eines Ausfalls. In der **FDIR Ausarbeitung** werden die relevanten Parameter und Bereiche für die Überwachung festgelegt und Aktionen zur Recovery definiert. Die **FDIR Implementierung** umfasst die Ausarbeitung von FDIR in Hardware und Software sowie die Absicherung. Das Vorgehen wird durch die formale Sprache SLIM [BCR+09] und das Modellierungswerkzeug FAME [COM19-ol] unterstützt. Das Bild 3-19 zeigt das Vorgehen, dessen Integration in den Entwicklungsprozess sowie die Unterstützung durch ein Autorenwerkzeug.

[4] Fault Detection, Isolation und Recovery

[5] Reliability, Availability, Maintainability, Safety

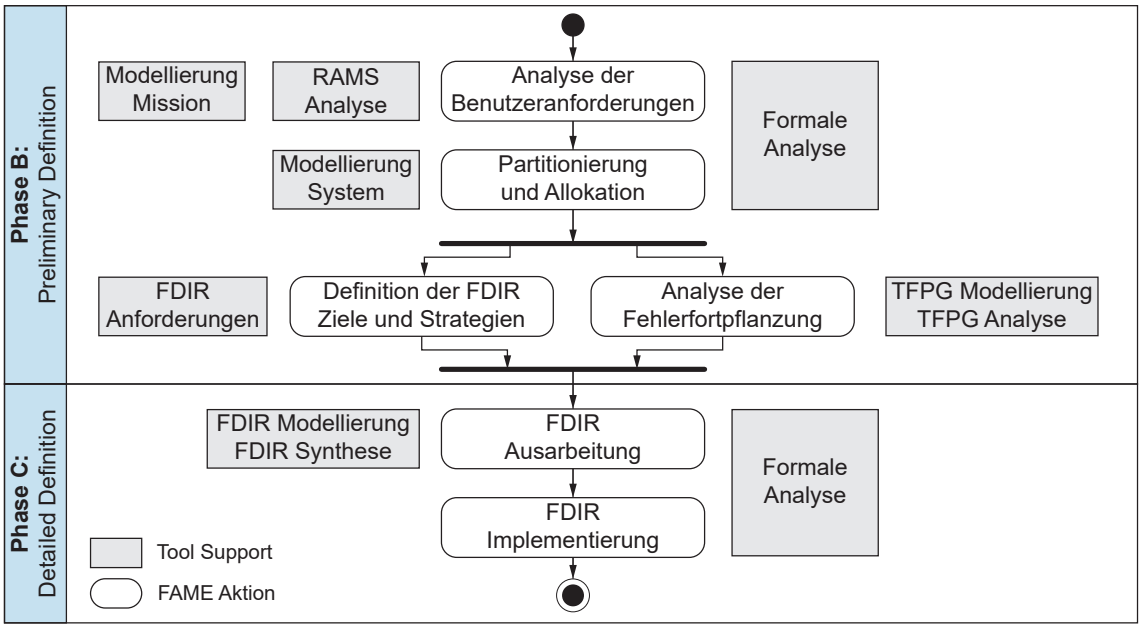


Bild 3-19: Vorgehensmodell nach BITTNER ET AL. *[BBC+14]*

Bewertung: Das Vorgehensmodell von BITTNER ET AL. orientiert sich an dem Entwicklungsprozess in der Luft- und Raumfahrt. Das Vorgehen wird durch eine Modellierungssprache und einem Autorenwerkzeug unterstützt. Kern der Modellierung sind Modelle der Fehlerpropagation. Verfahren zur Integration von FDIR-Fähigkeiten werden nicht adressiert. Ebenso werden Techniken zur Analyse oder zur Absicherung der FDIR-Eigenschaften nicht berücksichtigt. Eine Beschreibung der Absicherung erfolgt nicht.

3.3.3.4 Integration kognitiver Funktionen nach DUMITRESCU

Die Integration kognitiver Funktionen in mechatronische Systeme verspricht vielfältige Nutzenpotenziale. Diese Integration findet nach DUMITRESCU nicht systematisch statt. Daher stellt DUMITRESCU eine **Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme** vor. Die Entwicklungssystematik umfasst vier Bestandteile: Ein **Vorgehen**, welches die Tätigkeiten strukturiert und notwendige Hilfsmittel, wie z.B. Methoden und Werkzeuge zur Verfügung stellt. Eine Technik zur **Systembeschreibung**, die generische Entwurfsschablonen für die Spezifikation der Informationsverarbeitung bereitstellt. **Wiederverwendbares Lösungswissen** in Form von Lösungsmuster. Sowie ein Konzept zur **Werkzeugunterstützung**, das die Dokumentation, Suche, Analyse und Auswahl von Lösungswissen unterstützt [Dum10, S. 95ff.].

Das Vorgehensmodell besteht aus vier Phasen (vgl. Bild 3-20). In der **Systemanalyse** werden die Potenziale kognitiver Funktionen ermittelt.

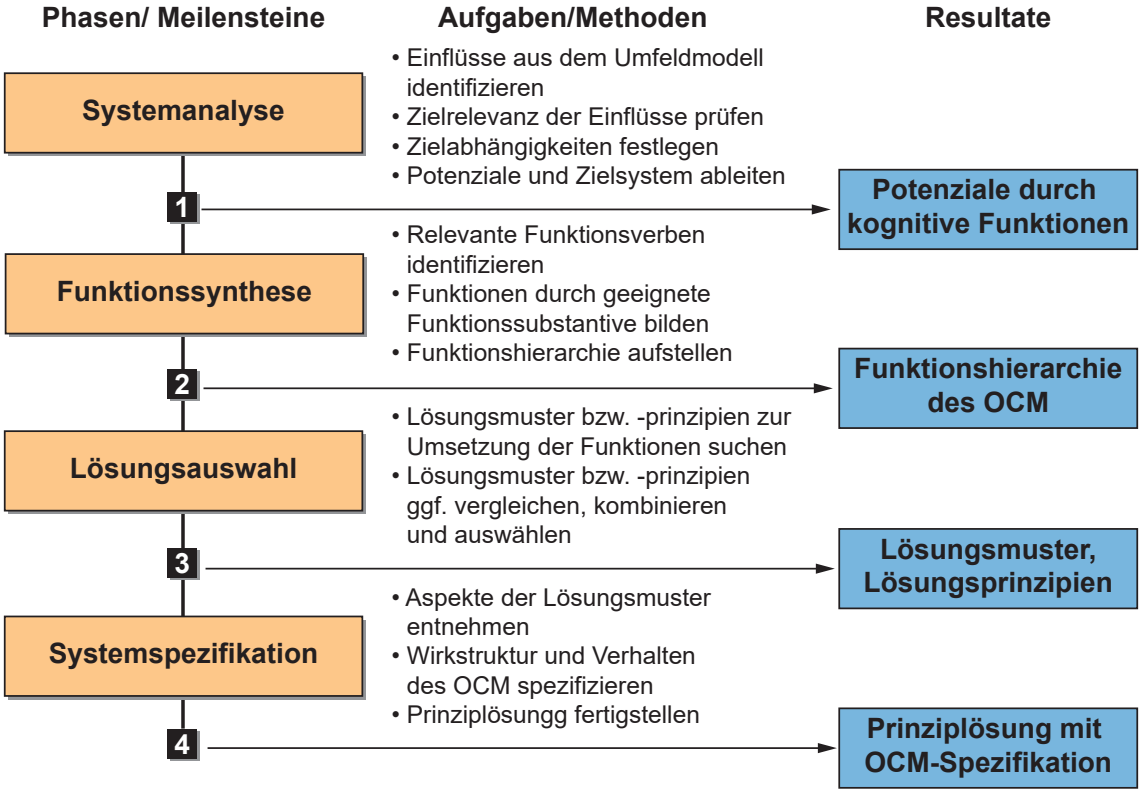


Bild 3-20: Vorgehensmodell zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme [Dum10, S. 99]

Basierend auf den Potenzialen erfolgt die **Funktionsynthese**. In der Synthese wird eine lösungsneutrale funktionale Beschreibung der Informationsverarbeitung erarbeitet. Dieser Schritt wird durch einen Katalog kognitiver Funktionen unterstützt. Mit der Funktionshierarchie werden in der **Lösungsauswahl** geeignete Lösungsmuster zur Umsetzung der kognitiven Funktionen ermittelt. Hierzu stellt DUMITRESCU einen Katalog an Lösungsmuster zur Verfügung. Die **Systemspezifikation** bildet die letzte Phase im Vorgehensmodell. Basierend auf den Lösungsmustern wird die Wirkstruktur und das Verhalten der Informationsverarbeitung spezifiziert und in die Prinziplösung des Systems integriert [Dum10, S. 99ff.].

Bewertung: DUMITRESCU präsentiert eine Systematik zur Integration kognitiver Funktion in die Informationsverarbeitung Intelligenter Technischer Systeme. Die Systematik orientiert sich an dem interdisziplinären Systementwurf und basiert auf dem Strukturkonzept Intelligenter Technischer Systeme. Dabei wird die Synthese durch ein Katalog von Funktionsverben und durch Lösungsmuster unterstützt. Eine Betrachtung der Verlässlichkeit erfolgt nicht. Analysetechniken und die Absicherung des Systems sind nicht betrachtet.

3.3.3.5 Potenzialanalyse zur Steigerung der Intelligenz in mechatronischen Systemen

IWANEK stellt ein **Systematik zur Steigerung der Intelligenz in fortgeschrittenen mechatronischen Systemen** des Maschinen- und Anlagenbaus vor. Die Systematik besteht aus einem **Stufenmodell** zur Steigerung der Intelligenz mechatronischer Systeme, einer **Planung der Umsetzung** von intelligentem Verhalten, einem **Vorgehensmodell**, sowie **Hilfsmittel** zur Förderung der Kommunikation und der Spezifikation [Iwa17, S. 91]. Das Stufenmodell fokussiert die Anwendung der Selbstoptimierung in der Informationsverarbeitung und orientiert sich an dem Operator-Controller-Modul (vgl. Kap. 3.2.1.3).

Bild 3-21 visualisiert das Stufenmodell. Es besteht aus den Funktionsbereichen Sensorik *Messen* und Aktorik *Agieren* sowie der Informationsverarbeitung. Die Informationsverarbeitung wird in die Bereiche *Steuern und Regeln*, *Identifizieren*, *Adaptieren* sowie *Optimieren* gegliedert. Die Funktionsbereiche bauen aufeinander auf. Jeder Funktionsbereich umfasst Leistungsstufen. Mit jeder Stufe steigt die Intelligenz des Systems [Iwa17, S. 93ff.].

Die entwickelte Systematik unterstützt Unternehmen bei der Potenzialanalyse hinsichtlich der Steigerung der Intelligenz. Das Vorgehensmodell gliedert sich in vier Phasen: Die erste Phase ist die **Disziplinübergreifende Systemspezifikation**. In dieser Phase erfolgt die Spezifikation des Prozesses, des Systems sowie der Nutzerinteraktionen. Zum Einsatz kommt OMEGA[6] zur Modellierung von Geschäftsprozessen und CONSENS zur Beschreibung des mechatronischen Systems. Darüber hinaus wird der aktuelle Leistungsstand entsprechend des Stufenmodells bewertet. Die zweite Phase ist die **Identifikation von Potenzialen**. Hierzu wird eine Schwachstellenanalyse mit Hilfe von Methoden aus dem Bereich Sicherheits- und Zuverlässigkeitstechnik durchgeführt. Das Resultat ist ein modifizierter Fehlerbaum, der einen Überblick über Schwachstellen, Ursachen und Folgen sowie abgeleiteten Potenzialen gibt. In der dritte Phase erfolgt die **Spezifikation von Lösungsideen**. In dieser Phase werden die Potenziale zu Lösungsideen konkretisiert und durch Ideensteckbriefe spezifiziert. Abschließend erfolgt die **Bewertung und Auswahl**. Die Lösungsideen werden anhand von Nutzen und Aufwand bewertet und in einem Portfolio visualisiert. Abschließend erfolgt die Priorisierung und Selektion [Iwa17, S. 132ff.].

[6] Objektorientierte Methode zur Geschäftsprozessmodellierung und -analyse

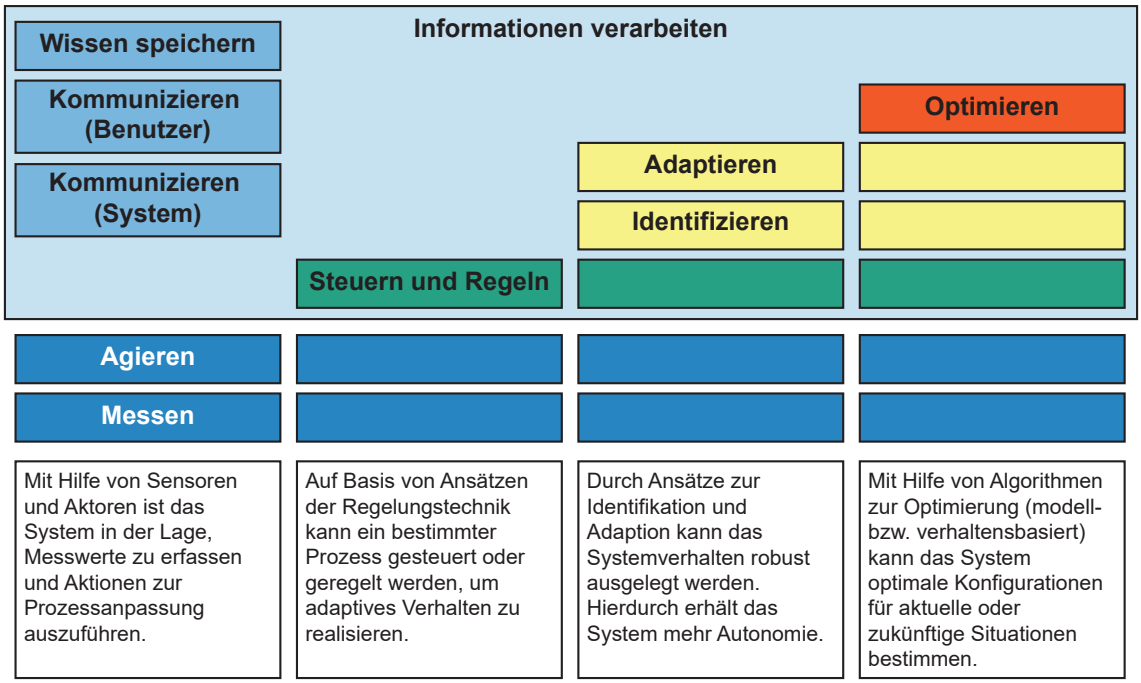


Bild 3-21: Funktionsbereiche zur Steigerung der Intelligenz [Iwa17, S. 93]

Bewertung: Die Systematik stellt eine Methode zur erweiterten Analyse der Verlässlichkeit mechatronischer Systeme bereit. Das Stufenmodell kann unterstützend eingesetzt werden. Der erweiterte Fehlerbaum ermöglicht eine systematische und frühzeitige Identifikation von Schwachstellen, Ursachen und Fehlerfolgen. Ein Lösungskonzept zur Integration von Selbstheilung wird nur indirekt durch die Funktionsbereiche und Leistungsstufen adressiert. Ebenso erfolgt lediglich eine qualitative Bewertung von Lösungsideen. Eine Beschreibung der Absicherung und Analyse erfolgt nicht.

3.3.3.6 Entwurf von Organic Computing Systemen

Organic Computing Systeme besitzen Self-X-Fähigkeiten. Dabei können Situationen auftreten, die nicht zur Entwurfszeit antizipiert wurden. In diesen Situationen kann sich das System unvorhersehbar oder unsicher verhalten. Aufgrund dieser Herausforderungen schlagen TOMFORD ET AL. einen inkrementellen Entwurfsprozess für Organic Computing Systeme vor [THC13]. Dieser Entwurfsprozess adressiert drei Bereiche: Das produktive System (SuOC), den Kontrollmechanismus (Observer/Controller) sowie die Mechanismen zur Adaption und Rekonfiguration in einem O/C-System [MT17, S. 260ff.].

Im Entwurf von O/C-Systemen sind verschiedene Entwurfsentscheidungen zu treffen (vgl. Bild 3-22. Zunächst ist festzulegen, welche Merkmale und Zustandsgrößen für den Observer relevant sind. Diese Größen dienen dem Observer zur Zustandsüberwachung des SuOC. Ein weiterer Aspekt sind die Konfigurationen und Parameter sowie die Randbedingungen für die Selbst-Adaption des Systems im Betrieb.

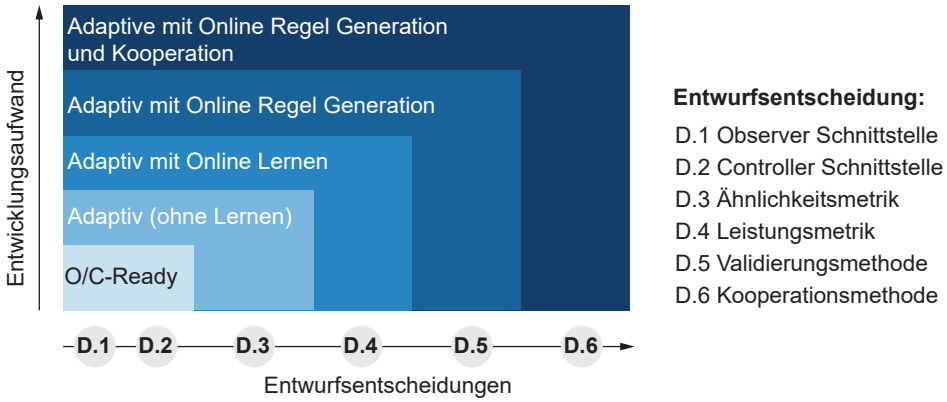


Bild 3-22: Design Entscheidungen und OC-Fähigkeiten [MT17, S. 272ff.]

Die Zuordnung von Zustand auf Adaption erfolgt durch ein Regelwerk. Zum Lernen dieses Regelwerks ist eine Ähnlichkeitsmetrik und eine Leistungsmetrik erforderlich. Die Ähnlichkeitsmetrik bewertet die Ähnlichkeit von Zuständen, z.B. über ein Distanzmaß. Diese Ähnlichkeit wird genutzt, um einem unbekannten Zustand eine geeignete Adaption im Betrieb zuzuordnen. Die Leistungsmetrik bewertet das Verhalten des System in Hinblick auf die Ziele und dient daher zur Bewertung der Regeln. Der fünfte Aspekt adressiert das Lernen, das Optimieren und das Explorieren des Regelwerks durch kombinierte Verfahren des Verstärkenden Lernens und der evolutionäre Algorithmen. Die soziale Interaktion von O/C-Systemen ist der sechste Aspekt. Dies adressiert insbesondere Kooperationsmodelle zwischen O/C-Systemen [MT17], [THC13], [TM13].

Die Realisierung von O/C-Systemen erfolgt in drei Phasen: Vorbereitung, Evaluierung und Anwendung. In der **Vorbereitung** werden die O/C-Fähigkeiten definiert und durch Modelle, Metriken und Methoden spezifiziert. Ebenso werden die Schnittstellen implementiert. Das System ist *O/C-Ready*. In der **Evaluierung** werden Observer und Controller in das System integriert. Das System läuft in einem *offenen O/C-Kreis*. Die Evaluierung erfolgt durch die Datenaufnahme, -analyse. Das adaptive Verhalten wird anhand der Leistungsmetriken bewertet. Das System operiert in einem assistierten und überwachten Modus. In der **Anwendung** wird das O/C-System in Betrieb genommen. Das System operiert unüberwacht in einem *geschlossenen O/C-Kreis*[MT17, S. 277ff.].

Bewertung: Der Ansatz unterstützt ein ganzheitliches Lösungskonzept. Es werden Leistungsstufen und Entwurfsregeln definiert. Es wird zwar ein generisches Vorgehen vorgeschlagen, dieses berücksichtigt jedoch nicht das interdisziplinäre Vorgehen. Ebenso fehlt eine Orientierung am Systementwurf. Die Analyse ist Bestandteil der Realisierung. Der Ansatz adressiert jedoch nicht, wie die Analyse umgesetzt ist und welche Techniken dafür zum Einsatz kommen.

3.4 Analysetechniken

Eine Systematik zur Integration von Selbstheilung in fortgeschrittene mechatronische Systeme muss Methoden zur frühzeitige Analyse und Absicherung von Funktionen und Eigenschaften der Selbstheilung bereitstellen. Das Kap. 3.4.1 analysierte dazu Verfahren der Fehlerinjektion. Das Kap. 3.4.2 bewertet das Simian Army Testing. Evolutionäre Funktionstests sind in Kap. 3.4.3 dargestellt. Das Kap. 3.4.4 stellt eine Techniken zur Analyse von Operationen der Selbstheilung vor.

3.4.1 Verfahren zur Fehlerinjektion

Verfahren zur Fehlerinjektion simulieren Fehler oder Fehlerfolgen in einem System. Ziel ist die Absicherung von Verfügbarkeit und Sicherheit. So kann zum Beispiel die Energieversorgung von einem Verbraucher abgetrennt und die Fehlerfolgen untersucht werden [IEC61508b], [AAA+90].

SVENNINGSSON ET AL. definieren generelle Anforderungen an die Verfahren: Die *Kontrollierbarkeit* fordert die Kontrolle wann und wo ein Fehler in das System eingebracht wird. Die *Beobachtbarkeit* ist die Fähigkeit, die Folgen des Fehlers zu beobachten, zu messen und nachvollziehen zu können. *Reproduzierbarkeit* bedeutet, dass der Fehler und die Folge statistisch reproduzierbar ist [SVE+10].

In der Literatur werden drei Methoden zum Fehlerinjektion diskutiert: Hardware-, Software- und Modell-basierte Verfahren. **Hardware-basierte Verfahren** werden bei physischen Prototypen angewendet. Sie simulieren temporale oder permanente Fehler in der Hardware (z.B. Ausfall eines Relais) oder der Umgebung (z.B. Strahlung). Ein Tool ist MESSALINE. Das Werkzeug injiziert komplexe logische Fehler auf einer Elektronik und variiert dabei die Dauer und die Frequenz des Fehlers [AAA+90]. **Software-basierte Verfahren** simulieren Fehler in einer Software. Dies sind z.B. Fehler in der Funktion, den Algorithmen, dem Timing, der Serialisierung sowie in der Deklaration von Variablen. FERRARI ist z.B. ein Tool zum Testen von Software, indem es Fehler in der CPU, dem Speicher oder dem Bus simuliert [ZAV04]. **Modell-basierte Verfahren**[7] werden eingesetzt, um Fehler in den Modellen und Virtuellen Prototypen von Software und Hardware (z.B. in VHDL[8] für Elektronik oder Matlab/Simulink für Zustandsautomaten und Regler) einzubringen. Die Fehlerszenarien sind reproduzierbar. Zudem können diese Verfahren bereits im Systementwurf eingesetzt werden [HTR97], [AAA+90], [ZAV04].

Bewertung: Modell-basierte Verfahren sind im Systementwurf einsetzbar und simulieren verschiedene Fehlerarten. Ihr Anwendungsbereich ist spezialisiert auf domänenspezifische Modelle (z.B. VHDL). Eine systematische Beschreibung der Absicherung und das szenariobasierte Testen mit Virtuellen Testbeds wird nicht unterstützt.

---

[7] auch Simulations-basierte Verfahren

[8] VHDL: Very High Speed Integrated Circuit Hardware Description Language

3.4.2 Simian Army Testing

Simian Army Testing ist ein Toolkit von automatisierten Testverfahren. Das Ziel ist die Steigerung der Resilienz einer Cloud-Infrastruktur gegenüber Ausfällen und Störungen. Im Gegensatz zu den genannten Verfahren zur Fehlerinjektion werden die Verfahren auch im Betrieb des Systems eingesetzt. Das Toolkit umfasst verschiedene Verfahren, die in Form von sog. *Monkeys* umgesetzt sind [Net11-ol]. Die Verfahren gliedern sich in drei Kategorien: Fehlergenerierung, Konformitätschecks und Ressourcenprüfung.

Die **Fehlergenerierung** simuliert einen Ausfall oder eine Störung in einer Anwendung, einer Plattform oder der Infrastruktur durch einen *Chaos Monkey*. Das Bild 3-23 visualisiert die Aktivitäten von einem Chaos Monkey. Nach der Initiierung des Verfahrens wird eine aktive Instanz im Netzwerk durch ein Zufallsgenerator ausgewählt und abgeschaltet [OI10]. Der *Latency Monkey* generiert Latenzen in der Kommunikation im Netzwerk. Er simuliert eine Degradation oder einen Ausfall [Bue12-ol]. **Konformitätschecks** überprüfen die

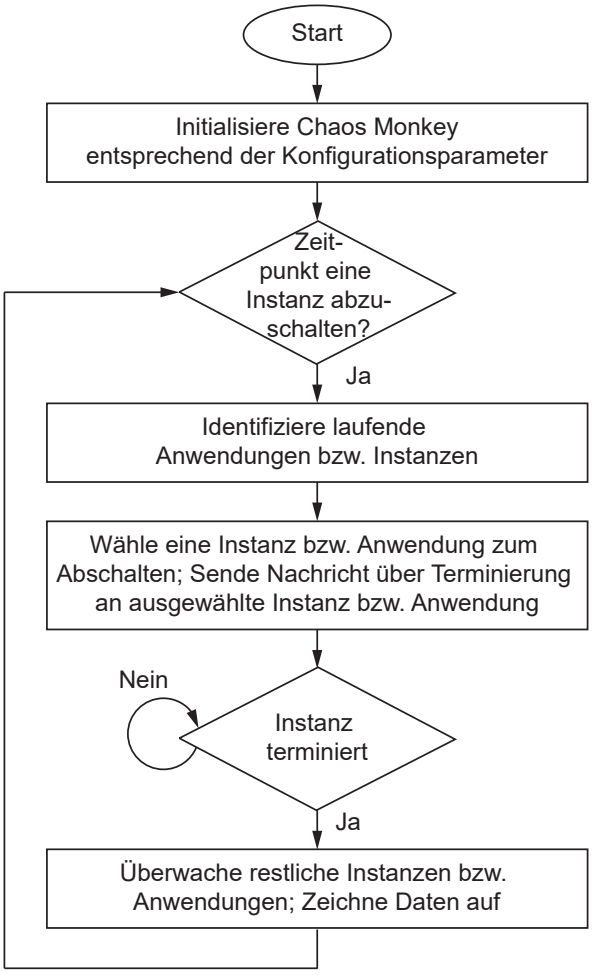


Bild 3-23: Ablauf Chaos Monkey [OI10]

Einhaltung von Standards hinsichtlich Security und Designrichtlinien. Der *Security Monkey* sucht Sicherheitslücken oder Schwachstellen im Netzwerk. Der *Conformity Monkey* vergleicht die Instanzen mit den Designrichtlinien. Die Ressourcenprüfung überprüft mit

dem *Doctor Monkey* die Health Checks, die sich auf jeder Instanz befinden und überwacht zudem Eigenschaften, wie z.B. CPU-Auslastung. Fehlerhafte Instanzen werden automatisch entfernt. Der *10-18 Monkey* erkennt Probleme in der Konfiguration oder im Laufzeitverhalten. Die **Ressourcenprüfung** erfolgt mit dem *Janitor Monkey*. Er identifiziert ungenutzte Ressourcen und gibt überschüssige Instanzen frei [Bue12-ol], [Gre19-ol]. Das Simian Army Toolkit ist als Open Source Anwendung veröffentlicht [Net19-ol].

Bewertung: Die Simian Army liefert einen automatisierten Ansatz zur Analyse der Resilienz in komplexen Systemen mit hohen Anforderungen an die Verfügbarkeit. Das Toolkit ist modular und erweiterbar. Es adressiert unterschiedliche Aspekte der Resilienz. Es bietet eine Werkzeugunterstützung. Die Anwendung ist bisher auf Netzwerke beschränkt. Eine Übertragung auf fortgeschrittene mechatronische Systeme ist möglich.

3.4.3 Evolutionärer Funktionstest

Funktionstests dienen zur Absicherung des Verhaltens [IEC61508a], [WKG09]. Evolutionäre Funktionstests suchen Testfälle, bei denen das Testobjekt seine funktionalen Anforderungen verletzt [Lin12]. Dazu werden Testfälle in ein Optimierungsproblem transformiert, welches dann durch heuristische Suchverfahren gelöst wird [WKG09].

Das Bild 3-24 stellt den Ablauf von evolutionären Funktionstest dar. Zunächst werden die Parameter und ihre Wertebereiche definiert. Dies spannt den Suchraum für das Optimierungsproblem auf. In der **Initialisiert** wird durch einen Zufallsgenerator aus dem Suchraum eine Menge von Individuen erstellt. Die **Evaluierung** bewertet die Individuen im Hinblick auf das Optimierungsziel. Ist das Ziel erreicht, erfolgt der **Abbruch** der Suche, andernfalls beginnt die Optimierung: In der **Selektion** wird ein Teil der Individuen aus der aktuellen Generation ausgewählt. Diese stellen das Erbgut für die nachfolgende Generation zur Verfügung. Durch **Rekombination** und **Mutation** werden die Individuen verändert. Jedem Individuum wird eine Fitness durch die Evaluation zugewiesen. Die Evaluierung erfolgt durch die Durchführung und Bewertung des Funktionstests - entsprechend der vorgegebenen Individuen (Testdaten). Die Fitness repräsentiert die Güte der Individuen in Hinblick auf das Optimierungsziel. Die Fitness dient der Selektion, zum Vergleich der Individuen und zur Optimierung der Suche. Die **Wiedereinfügung** wählt ein Teil der Individuen aus der Elterngeneration aus und übernimmt diese unverändert in die nächste Generation. Die Iteration erfolgt bis entweder das Optimierungsziel oder ein Abbruchkriterium erfüllt ist [Lin12], [WKG09].

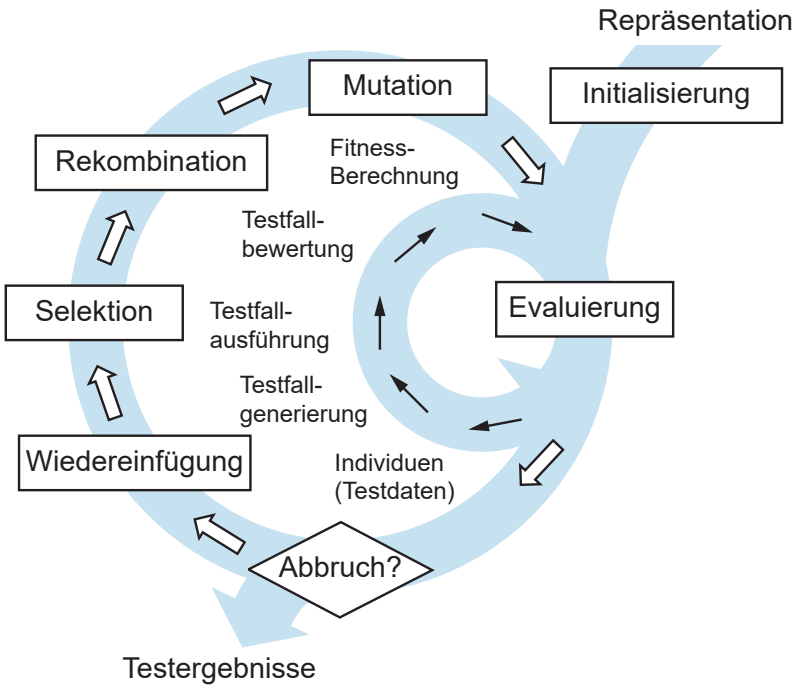


Bild 3-24: Ablauf Evolutionärer Funktionstests [Rei10-ol], [Lin12]

Im EU-Projekt EvoTest wurde das **Automated Evolutionary Testing Framework** entwickelt. Das Framework unterstützt bei der Durchführung, der Überwachung und der Auswertung sowie bei der Fitnessbewertung der erzeugten Testdaten. WEGENER ET AL. kombinieren dieses Framework mit dem Testwerkzeug Messina, sowie mit Matlab/Simulink und modularHiL. Diese Testinfrastruktur kombiniert evolutionäre Funktionstests mit XiL-Techniken[9] [WKG09].

Bewertung: Evolutionäre Funktionstest bieten einen Ansatz, um das Verhalten in kritischen Szenarien zu untersuchen. Durch die automatisierte Generierung von Testfällen wird der Aufwand reduziert und die Testabdeckung erhöht. Eine wesentliche Herausforderung liegt darin, eine geeignete Fitness-Funktion zu identifizieren und formal zu beschreiben. Ebenso besteht die Gefahr einer lokalen Optimierung. Durch die Kombination mit XiL-Techniken unterstützt das Verfahren die virtuelle Inbetriebnahme. Die Parameter könnten aus der Spezifikation abgeleitet werden.

3.4.4 Analyse von Operationen der Selbstheilung

PRIESTERJAHN stellt ein Verfahren zur Bewertung von Operationen der Selbstheilung in selbstoptimierenden Systemen vor. Grundlage bildet eine Gefahren- und Risikoanalyse des Systems. Die Risikobewertung ermöglicht eine Priorisierung und Umsetzung von Operationen der Selbstheilung. Nach PRIESTERJAHN resultiert eine Gefahrensituationen aus der Propagierung von Fehlern im System. Den Kern des Verfahrens bilden daher

[9] XiL: Umfasst Model-in-the-Loop (MiL), Software-in-the-Loop (SiL) und Hardware-in-the-Looop (HiL)

Modelle, mit denen die Fehlerfortpflanzung zeitlich beschrieben werden kann [Pri13, S.43ff.], [GRS+14, S. 102ff.].

Der Ablauf des Verfahrens visualisiert Bild 3-25. Grundlage bilden das MechatronicUML Deployment Diagramm und der Zustandsautomat des Systems. Das Deployment Diagramm beschreibt die Architektur der Informationsverarbeitung. Der Zustandsautomat das diskrete Verhalten des Systems. Aus diesen Diagrammen wird ein **Modell der Fehlerausbreitung generiert**. Darauf aufbauen wird für jedes Teilsystem eine **Gefahren- und Risikoanalyse** durchgeführt. Für jedes Risiko werden die Minimalschnitte ermittelt. Minimalschnitte bilden minimale Ereigniskombinationen, die zu einem Risiko führen. Im dritten Schritt erfolgt die **Analyse der Selbstheilung** durch Simulation und Methoden der Fehlerinjektion. Dabei wird untersucht, ob die Operation der Selbstheilung das Ausfallrisiko des Systems reduziert. Die **Bewertung** erfolgt durch die Analyse des kritischen Pfades. Der kritische Pfad ist die Zeitspanne zwischen der Detektion und der Rekonfiguration des Systems im Vergleich zur Propagierungszeit des Fehlers. Ist die maximale Detektions- und Rekonfigurationszeit geringer als die minimale Zeit der Fehlerausbreitung, reduziert sich das Sicherheitsrisiko [Pri13, S. 91ff.], [GRS+14, S. 105ff.].

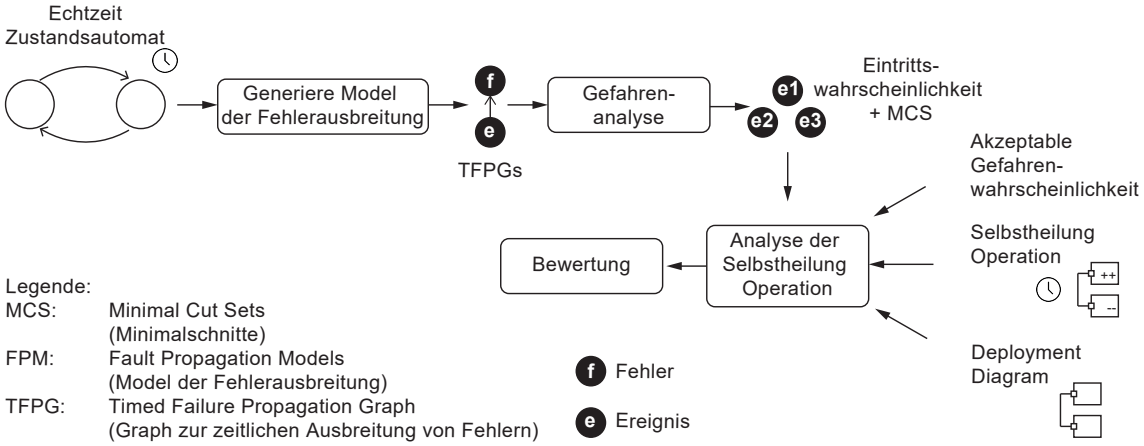


Bild 3-25: Ablauf zur Analyse von Operationen der Selbstheilung [GRS+14, S. 105]

Bewertung: Das Verfahren bietet einen umfassenden Ansatz zur Analyse von Selbstheilung in selbstoptimierenden Systemen. Die Gefahrenanalyse ermöglicht eine Potenzialanalyse für Selbstheilung in fortgeschrittenen mechatronischen Systemen. Hilfsmittel zum Entwurf der Selbstheilung werden nicht adressiert. PRIESTERJAHN liefert ein systematisches Vorgehen für den domänenspezifischen Softwareentwurf. Der Systementwurf wird nicht adressiert.

3.5 Handlungsbedarf

Der Abgleich vom untersuchten Stand der Technik mit den Anforderungen aus dem Kapitel 2.6 führt zu der folgenden Bewertung. Das Bild 5-41 fasst die Ergebnisse zusammen.

A1) Ganzheitliches Selbstheilungskonzept: Die Ansätze aus dem Organic Computing, insbesondere die Observer/Controller-Architektur (vgl. Kap. 3.2.2.3), erfüllen diese Anforderungen. Sie müssen aber in die Struktur Intelligenter Technischer Systeme integriert werden. Das Strukturkonzept Intelligenter Technischer Systeme liefert hierfür einen umfassenden Rahmen (vgl. Kap. 3.2.1.3).

A2) Bereitstellung von Konstruktionsprinzipien: Keine der Ansätze stellt Verfahren in Form von Lösungswissen zum Monitoring, Analyse und zur Wiederherstellung bereit. DUMITRESCU liefert eine Schablone mit der die Verfahren als Lösungswissen aufbereitet und dokumentiert werden können (vgl. Kap. 3.3.3.4).

A3) Interdisziplinäre Anwendbarkeit: Die Sprachen CONSENS und die SysML sind interdisziplinär anwendbar. Aspekte der Verlässlichkeit werden jedoch nicht explizit in der Sprache adressiert. Ebenso kann das zielorientierte und situationsabhängige agieren von intelligenten Systemen nicht modelliert werden.

A4) Identifikation von Potenzialen zur Selbstheilung: IWANEK liefert einen Ansatz zur Potenzialanalyse auf Basis eines modifizieren Fehlerbaums (vgl. Kap. 3.3.3.5). Es fehlt an einem Ansatz der Unsicherheit, Komplexität und Risiko in einer Potenzialanalyse vereint. Ebenso sind von dem Risiko Kausalszenarien abzuleiten, die neben den funktionalen auch die temporalen Abhängigkeiten berücksichtigen. Hierzu ist der dynamische Fehlerbaum mit der STPA (vgl. Kap. 3.3.2.4) zu kombinieren.

A5) Beschreibung der Absicherung: Keine der untersuchten Ansätze unterstützt die integrative Spezifikation der Absicherung durch *Testfälle*, *Testumfeld* und *Virtuelle Szenarien*. CONSENS und die SysML bieten jedoch Möglichkeiten zur Erweiterung durch Sprachprofile. Ebenso können darüber Testfälle mit den Anforderungen verknüpft werden.

A6) Orientierung am Systementwurf: Die Sprachen CONSENS und SysML unterstützen diese Anforderung. Ergebnis ist die Prinziplösung des Systems. Die SysML wird durch Autorenwerkzeuge unterstützt. Hier ist ein UML/SysML-Profil für CONSENS erforderlich.

A7) Systematische und interdisziplinäre Vorgehensweise: BITTNER ET AL., DUMITRESCU und TOMFORD ET AL. liefern jeweils Teilaspekte. Hervorzuheben sind die Leistungsstufen des Organic Computing Systems. Sie ermöglichen Entwurfsentscheidungen. Keiner der betrachteten Ansätze berücksichtigt jedoch die Phasen Analyse, Synthese und frühzeitige Absicherung in einem interdisziplinären Vorgehen.

A8) Bereitstellung von Analysetechniken: Das Simian Army Testing liefert einen vielversprechenden Ansatz zur automatisierten Simulation von Beeinträchtigungen. Der Ansatz ist auf die Anwendung für Intelligente Technische Systeme zu erweitern. Dabei sind die

Anforderungen *Reproduzierbarkeit, Beobachtbarkeit* und *Kontrollierbarkeit* (vgl. Kap. 3.4.1) zu berücksichtigen.

A9) Werkzeugunterstützung durch ein Virtuelles Testbed: Ein Virtuelles Testbed zur Absicherung der Selbstheilung in Intelligenten Technischen Systemen wird in keinem der betrachteten Ansätze berücksichtigt. PRIESTERJAHN simuliert zwar mit Hilfe von TFPG, die Anwendung ist aber auf die MechatronicUML beschränkt (vgl. Kap. 3.4.4). Die Systematik muss ein Konzept für ein Virtuelles Testbed bereitstellen, mit dem ein Virtuelles Szenario simuliert und der virtuelle Prototyp untersucht werden kann. Ebenso fehlt ein Ansatz, um Daten für das maschinelle Lernen synthetisch zu generieren.

Keiner der untersuchten Ansätze und auch keine einfache Kombination erfüllt die Anforderungen an eine Entwicklungssystematik. Die entscheidende Schwachstelle ist die mangelnde Verzahnung zwischen dem Systementwurf, der Integration von Selbstheilung sowie der Werkzeugunterstützung für die Synthese und Analyse. Zudem fokussieren viele Ansätze nur Teilaspekte des Gesamtproblems. Es besteht Handlungsbedarf für eine *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*.

Bewertung der untersuchten Ansätze hinsichtlich der gestellten Anforderungen.			Anforderungen								
			A1	A2	A3	A4	A5	A6	A7	A8	A9
Ansätze zur Strukturierung von Selbstheilung	Struktur-konzepte	Steuerungsarchitekturen	○	○	◑	○	○	○	○	○	○
		Neurorobotik Architektur	○	○	◑	○	○	○	○	◑	◑
		Operator-Controller-Modul	◑	◑	◑	○	○	○	○	○	○
		Referenzarchitektur Autonome Systeme	◑	◑	◑	○	○	○	○	○	○
		Organic Design Pattern	●	◑	◑	○	○	○	○	○	○
		AUTOSAR Adaptive Plattform	◑	◑	○	○	◑	○	○	◑	◑
	Ansätze zur Selbstheilung	Mehrstufiges Verlässlichkeitskonzept	◑	◑	●	○	○	○	○	○	○
		Fehler Detektion, Isolation und Recovery	◑	◑	◑	○	○	◑	○	○	○
		Observer/Controller-Architektur	●	◑	◑	○	○	○	○	○	○
		Organic Robot Control Architecture	●	◑	◑	○	○	◑	○	○	○
		Self-Healing Framework f. Cyber-Phyische Systeme	◑	◑	◑	○	○	◑	○	○	○
Hilfsmittel für die Spezifikation selbstheilender Systeme	Sprachen zur Spezifikation	OMG Systems Modelling Language (SysML)	○	○	◑	○	◑	●	○	◑	◑
		CONSENS	○	○	●	○	○	●	○	○	◑
		Object Process Methodology (OPM)	○	○	●	○	○	◑	○	○	◑
		Open Modelica	○	○	◑	○	◑	◑	○	◑	◑
	Methoden zur frühz. Analyse	Methoden zur Analyse der Zuverlässigkeit	○	◑	●	◑	○	◑	○	○	○
		Environmental Survey Hazard Analysis	○	○	●	◑	○	◑	○	○	○
		Sneak-Analyse	○	○	●	◑	○	◑	○	○	○
		System-Theoretic Process Analysis (STPA)	○	○	●	◑	○	●	○	○	○
	Vorgehens-modelle	ISO/PAS 21448 SOTIF	○	○	◑	◑	◑	●	●	◑	○
		Zustandsüberwachung für S.O.-Systeme	◑	◑	●	◑	○	●	◑	○	○
		Integriertes FDIR Design nach BITTNER ET AL.	◑	◑	◑	○	◑	◑	◑	○	○
		Integration kognitiver Funktionen nach DUMITRESCU	○	◑	●	○	◑	●	◑	○	○
		Potenzialanalyse zur Steigerung der Intelligenz	◑	◑	●	○	○	●	◑	○	○
		Entwurf von Organic Computing Systemen	◑	◑	○	◑	○	◑	◑	◑	◑
Analyse und Bewertung	Verfahren und Techniken	Verfahren zur Fehlerinjektion	○	○	○	◑	○	◑	○	◑	◑
		Simian Army Testing	○	○	○	◑	○	○	○	●	◑
		Evolutionäre Funktionstests	○	○	○	◑	◑	○	○	◑	◑
		Analyse von S.-H.-Operationen nach PRIESTERJAHN	○	○	◑	◑	◑	◑	◑	◑	◑

Bild 3-26: Bewertung vom Stand der Technik anhand der Anforderungen

4 Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung

4.1 Bestandteile der Entwicklungssystematik

Der Abgleich der Anforderungen mit dem ausgewählten Stand der Technik zeigt den Bedarf an einer *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*. Die Systematik kombiniert die bisher isoliert betrachteten Bereiche **Selbstheilung** zur Steigerung der Resilienz technischer Systeme, das **Data-Driven Systems Engineering** (DDSE) zum disziplinübergreifenden Systementwurf sowie **Virtuelle Testbeds** zur kontinuierlichen und szenariobasierten Analyse und Absicherung. Diese drei Kernbereiche und ihre Schnittmengen zueinander ergeben die sechs kohärenten Bestandteile der Systematik (Bild 4-1).

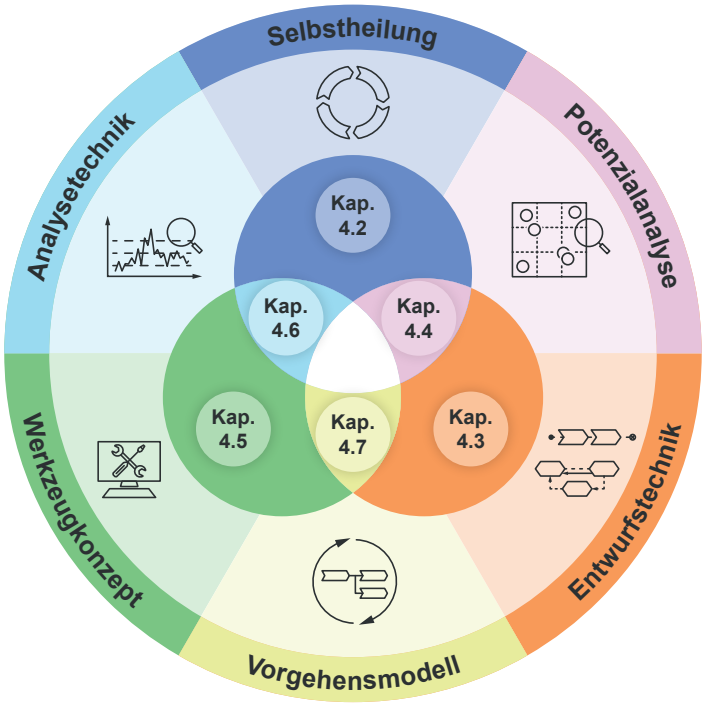


Bild 4-1: Bestandteile der Systematik

Der erste Bestandteil der Systematik ist ein Rahmenmodell **selbstheilender Systeme**. Das Rahmenmodell basiert auf Leistungsstufen der Selbstheilung. Die Leistungsstufen dienen der Steigerung der Autonomie und Resilienz durch die Lernfähigkeit. Zur Umsetzung einer Leistungsstufe sind spezifische Entwurfsaspekte im Systementwurf zu berücksichtigen. Das Rahmenmodell ist Gegenstand von Kapitel 4.2. Weitergehend werden Strategien zur Integration der Selbstheilung in ein Intelligentes Technisches System nach DUMITRESCU erläutert. Basierend auf den Entwurfsaspekten wird Lösungswissen zur Umsetzung der Selbstheilung vorgestellt. Die grundsätzliche Kombinierbarkeit des Lösungswissens ist durch das Rahmenmodell gegeben.

Das Kapitel 4.3 **Entwurfstechnik** beschreibt einen weiteren Bestandteil der Systematik. Die Entwurfstechnik stellt eine Sprache zur Modellierung von selbstheilenden Systemen bereit und kombiniert diese mit einem modellbasierten Absicherungskonzept. Die Sprache basiert auf CONSENS (vgl. Kap. 3.3.1.2) und erweitert diese um Modellelemente und Partialmodelle. Die erweiterte Entwurfstechnik ermöglicht eine integrierte Modellierung von System- und Absicherungskonzept. Das Absicherungskonzept definiert geeignete Testfälle und Szenarien zur Absicherung und Analyse fortgeschrittener mechatronischer Systeme auf der Basis von kohärenten Partialmodellen. Die Sprache ist als Profil der UML/SysML in einem Autorenwerkzeug umgesetzt.

Der dritte Bestandteil ist die **Potenzialanalyse** in Kapitel 4.4. Die Potenzialanalyse bildet die Schnittmenge zwischen Entwurfstechnik und Selbstheilung. Ziel der Potenzialanalyse ist die Priorisierung und Auswahl von kritischen Funktionen und Einheiten in einem technischen System für die ein Selbstheilungskonzept umgesetzt werden muss. Die Bewertung ermöglicht eine Aussage über die anzustrebende Leistungsstufe der Selbstheilung. Die Analyse ist für ein interdisziplinäres Team konzipiert. Die Anwendung erfolgt im Systementwurf.

Das **Werkzeugkonzept** bildet den vierten Bestandteil. Dieses ermöglicht die kontinuierliche Integration und Analyse der Eigenschaften durch *Scenario-in-the-Loop*-Testing. Das Konzept bietet eine fortgeschrittene Validierungsumgebung, mit der die Eigenschaften durch ein Kennzahlensystem bewertet werden. Mit Hilfe synthetischer Daten können die Modelle des maschinellen Lernens trainiert und getestet werden. Das Werkzeugkonzept wird in Kapitel 4.5 dargestellt.

Analysetechniken zur virtuellen Absicherung bilden den fünften Bestandteil der Systematik. Die entwickelten Verfahren kombinieren Techniken der modellbasierten Fehlerinjektion mit einem agentenbasierten Testverfahren in einem virtuellen Testbed. Der Teil bildet die Schnittmenge zwischen Werkzeugkonzept und Selbstheilung. Das Kapitel beschreibt die Konzipierung und die prototypische Implementierung des Monkey Testing Ansatzes für selbstheilende Systeme. Die Techniken können fortlaufend im Vorgehensmodell angewendet werden. Die Analysetechniken sind Kern von Kapitel 4.6.

Das **Vorgehensmodell** verbindet die Bestandteile der Systematik. Das Vorgehen orientiert sich an dem Systementwurf mechatronischer Systeme (vgl. VDI 2206 [VDI2206]), sowie der ISO/PAS 21488 [ISO 21448]). Es besteht aus wiederkehrenden Prozessbausteinen. Die Prozessbausteine definieren konkrete Artefakte und ein Rollenkonzept. Das Vorgehensmodell ist in Kapitel 4.7 beschrieben.

4.2 Selbstheilende Systeme

Dieses Kapitel stellt die Integration von Eigenschaften der Selbstheilung in ein Intelligentes Technisches System vor. Das Kap. 4.2.1 definiert den Selbstheilungsprozess und die Entwurfsaspekte entlang von Leistungsstufen. Das Kap. 4.2.2 stellt die Referenzarchi-

tektur selbstheilender Systeme vor. Das Kap. 4.2.3 stellt Konstruktionsprinzipien für die Bestandteile der Referenzarchitektur bereit.

4.2.1 Selbstheilungsprozess

4.2.1.1 Definitionen

Selbstheilende Systeme und begleitende Begriffe sind wie folgt definiert:

Definition 1: *Eine* ***Beeinträchtigung*** *im eigentliche Sinne liegt vor, wenn eine für die Verfügbarkeit oder für die Sicherheit des Systems relevante Störung der intendierten Funktion besteht. Neben solchen Beeinträchtigungen gibt es eine Vielzahl an* ***Störungen****, die sich nicht auf die sicherheitsrelevante Funktionsfähigkeit des Systems, der Umwelt oder den Bediener auswirken. Eine Klassifikation von Störungen erfolgt nach intern, extern und statisch, dynamisch sowie nach bekannt und unbekannt in Art und Größe.*

Intelligente Technische Systeme operieren in offenen Betriebsumgebungen. Dabei können sich nicht antizipierte Betriebsszenarien und unerwartete Wechselwirkungen der Teilsysteme untereinander oder mit dem Nutzer ergeben. Vor dem Hintergrund unbekannter Risiken und der Anforderungen an die Verlässlichkeit im Systemverhalten fällt der Resilienz eine zunehmende Bedeutung zu. Im Fokus stehen dabei die verteilten Regelstrukturen, beginnend bei den Sensoren über die in den Steuergeräten zu verarbeitenden Sensordaten bis hin zu der Aktorik. Vernetzte Systeme greifen auch auf Daten und Dienste von externen Systemen zu. Die Informationen werden fusioniert, aggregiert und dann für komplexe Regelungen und Entscheidungen herangezogen.

Definition 2: *Die* ***Resilienz*** *beschreibt die Eigenschaft eines Systems, einer Störung in diesen Regelstrukturen zu widerstehen oder sich so anzupassen, dass nach einer Beeinträchtigung die intendierte Funktion oder eine funktionsfähige Struktur wiederhergestellt wird. Im Kern geht es um den unbedingten Erhalt der Verfügbarkeit, Sicherheit und Robustheit der intendierten Funktion.*

Definition 3: *Ein* ***selbstheilendes System*** *besitzt die Fähigkeit, seinen Gesundheitszustand online zu überwachen, die Ursache einer Beeinträchtigung selbstständig zu diagnostizieren und eine situierte Selbstheilungsstrategie (policy) unter Berücksichtigung vorgegebener Ziele zu entwickeln. Dabei entwickelt sich ein selbstheilendes System unter vorgegebenen Randbedingungen im Betrieb weiter. Ziel ist die Steigerung der Resilienz durch Wandlungsfähigkeit, Redundanz, Robustheit, Verteilung und Lernfähigkeit.*

Definition 4: *Der* ***Gesundheitszustand*** *klassifiziert den Zustand eines technischen Systems anhand von Gesundheitssignalen. Ein System ist gesund, wenn die Funktions- und Leistungsfähigkeit in einem „normalen“ Zustandsbereich ist.*

Das Bild 4-2 verdeutlicht die Definition selbstheilender Systeme anhand der Systemgesundheit.

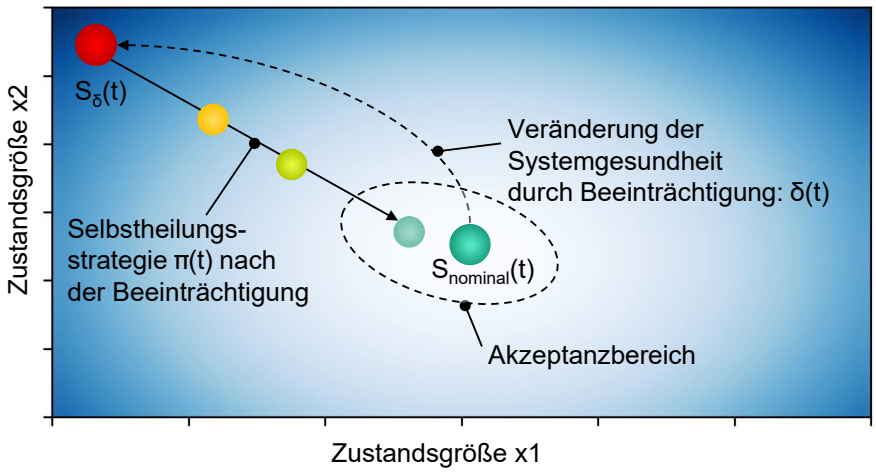


Bild 4-2: Prinzip der Selbstheilung

Entsprechend der Definitionen besteht der Selbstheilungsprozess aus den Funktionen: Überwachung, Diagnostik, Planung, Ausführung und Lernen. Bild 4-3 visualisiert den Selbstheilungsprozess.

- **Überwachung**: Die Überwachung umfasst eine Anomalie Detektion für das Erkennen von unbekannten Systemverhalten oder Signalmuster mittels maschineller Lernverfahren, sowie einen Gesundheitsmonitor zur Überwachung des Gesundheitszustandes aufgrund vordefinierter Regeln und Expertenwissen. Das Zusammenspiel von maschinellen Lernen und vordefinierten Systemwissen erlaubt die Umsetzung des Lernparadigmas. Aus den möglicherweise abweichenden Ergebnissen von Anomalie Detektion und Gesundheitsmonitor wird ein Gesundheitszustand ermittelt, der einen Kompromiss zwischen Robustheit und Sensitivität gewährleistet.
- **Diagnostik:** Die Diagnostik ordnet einer Beeinträchtigung automatisiert eine mögliche Ursache zu. Dabei kommen Klassifikations- oder Regressionsverfahren aus dem maschinellen Lernen zum Einsatz. Die Ausgabe der Überwachung dient als Eingangsgröße für die Diagnostik. Ergebnis ist die Diagnose. Domänenwissen zu möglichen Ursachen kommen aus dem Systementwurf und der Potenzialanalyse.
- **Planung:** Anhand der Diagnose erfolgt die situierte Auswahl einer Selbstheilungsstrategie (policy). Die Strategien können, je nach Ebene in der sie im selbstheilenden System stattfinden, sehr unterschiedlich ausfallen. Die Strategie wird in ein Handlungsplan übersetzt.
- **Ausführung**: Basierend auf dem Handlungsplan werden die Selbstheilungsaktionen im System under Observation and Control (SuOC) über Operationen ausgeführt. Bei einem kritischen Gesundheitszustand werden Notfallroutinen initiiert.
- **Lernen**: Das Lernen ist entscheidend für die Anwendbarkeit der Selbstheilung. Dazu muss umfassendes Systemwissen beispielsweise zu vorhandenen Systemkomponenten, typisches Normalverhalten, vorgesehene Signaleigenschaften, aber auch

mögliche Selbstheilungsstrategien vorhanden sein. In Betriebssituationen kann die Bewertung des Gesundheitszustandes und auch die gewählte Selbstheilungsstrategie aufgrund dieses Wissens angepasst werden.

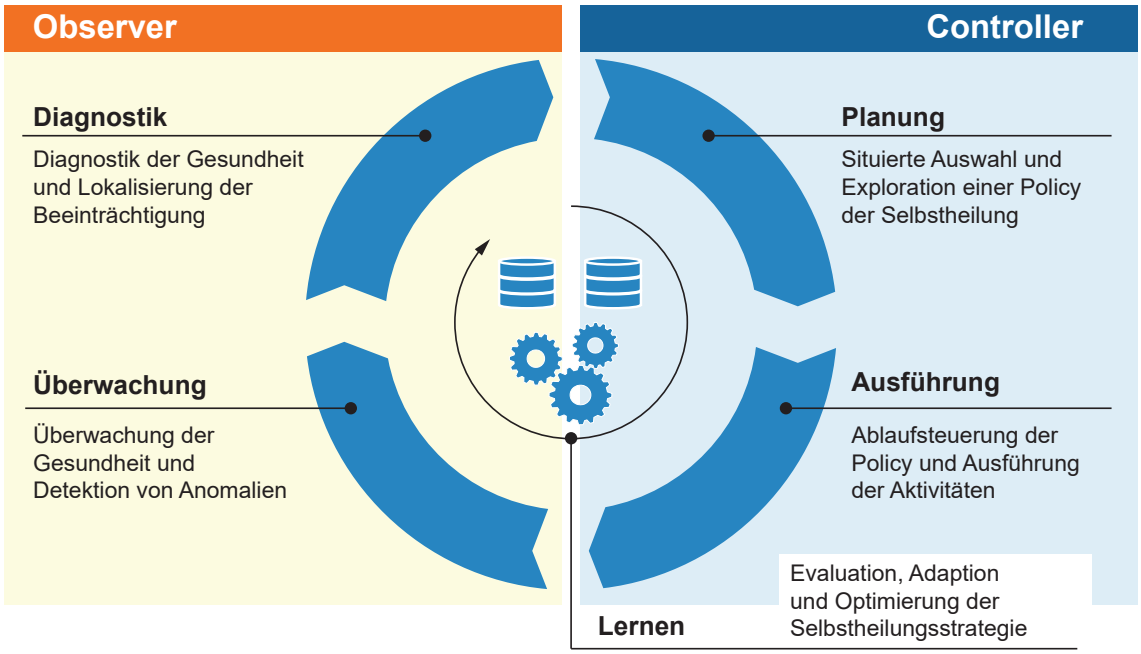


Bild 4-3: Selbstheilungsprozess

4.2.1.2 Leistungsstufen und Entwurfsaspekte

Zur Integration von Selbstheilungseigenschaften in ein Intelligentes Technische System bieten sich Leistungsstufen in Anlehnung zum Organic Computing an (vgl. Kap. 3.3.3.6). Das Modell wird in dieser Arbeit erweitert und auf selbstheilende Systeme übertragen. Zur Umsetzung der Leistungsstufen sind Entwurfsaspekte im Systementwurf zu berücksichtigen. Diese sind in Bild 4-4 zusammengefasst.

Leistungsstufe 1: Regelbasierte Selbstheilung. Die Regelbasierte Selbstheilung ist in der Assoziativen Regulierung verortet. Sie operiert unter Echtzeit-Bedingungen und initiiert Notfallroutinen, wie z.B. *Fail-Safe*. Verfahren zur Überwachung und Diagnostik der Systemgesundheit sind integriert. Aktionen der Selbstheilung und Notfallmechanismen sind im System umgesetzt. Der Controller operiert auf Basis statischer Regeln.

Leistungsstufe 2: Adaptive Selbstheilung mit Online Lernen. Die Adaptive Selbstheilung mit Online Lernen erweitert den regelbasierten Ansatz in der Assoziativen Regulierung um das inkrementelle Lernen. Zudem integriert diese Ebene die Evaluation, Adaption und Optimierung von Selbstheilungsstrategien. Zur Evaluation ist ein Mechanismus zur Leistungsbewertung und Belohnung erforderlich.

Leistungsstufe 3: Adaptive Selbstheilung mit Exploration der Wissensbasis. Die Leistungsstufe fokussiert das Planen. Die Exploration ist Teil der kognitiven Regulierung. Es werden neue Strategien zur Selbstheilung auf Wissensbasis generiert und bewertet. Es sind

Verfahren zur Exploration, Simulation und Optimierung erforderlich.

Leistungsstufe 4: Adaptive Selbstheilung mit Exploration und Kooperation in einem vernetzten System. Diese Leistungsstufe erfolgt auf Ebene des vernetzten Systems. Dabei kommen Ansätze zur Selbstorganisation zum Einsatz. Bei Ausfall eines Systems werden die Aufgaben durch andere Systeme übernommen oder ein beeinträchtigtes System im Netzwerk isoliert.

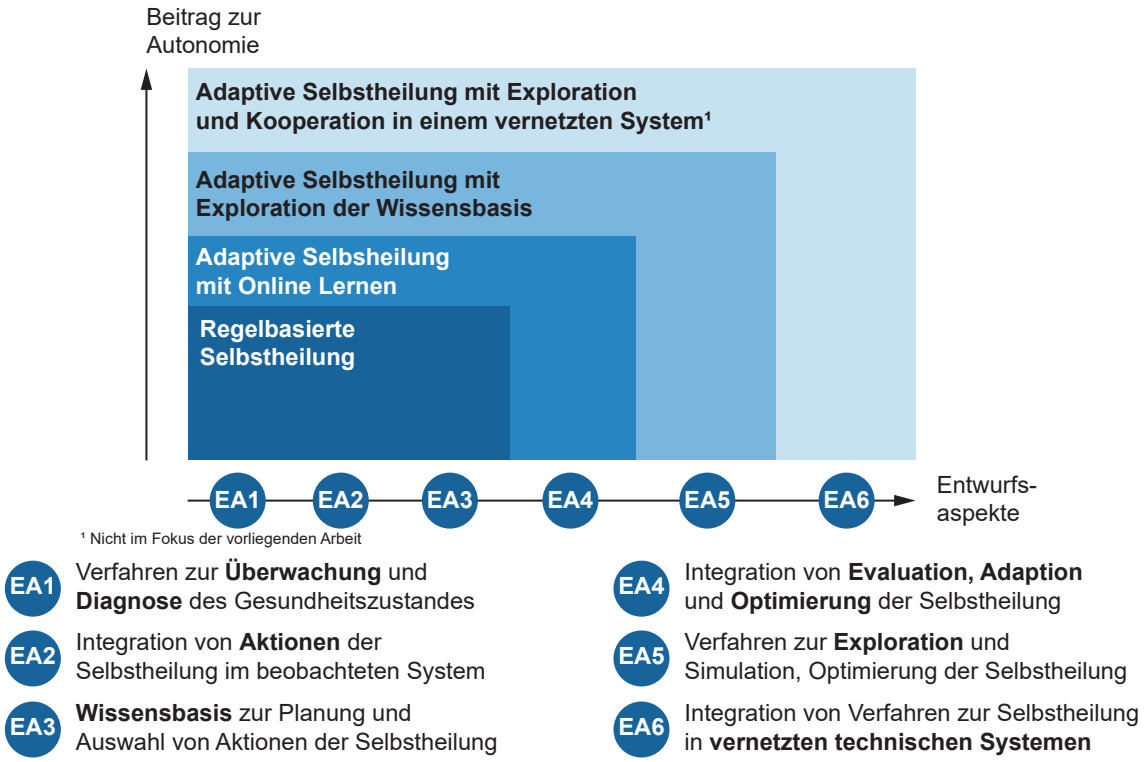


Bild 4-4: Entwurfsaspekte entlang der Leistungsstufen in Anlehnung an [MT17]

4.2.2 Framework

4.2.2.1 Architekturkonzept

Die Selbstheilungseigenschaften greifen auf verschiedenen Architekturebenen eines Intelligenten Technischen Systems. Die aus der O/C-Architektur resultierende Selbstheilung stellt einen eigenständigen Aspekt in der Systemarchitektur dar. Somit sind insgesamt drei Aspekte charakterisierend für selbstheilende Systeme: Das **Mechatronische Grundsystem** repräsentiert das Zusammenwirken von Software und Hardware. Dies umfasst Aktorik, Sensorik, Informationsverarbeitung auf der physischen Ebene. Die **Kognition** ergänzt das mechatronische Grundsystem um die selbstregulierende Informationsverarbeitung. In Anlehnung an dem Schichtenmodell von DUMITRESCU, besteht die selbstregulierende Informationsverarbeitung aus den Ebenen: Kognitive Regulierung, Assoziative Regulierung und Nicht-kognitive Regulierung [Dum10, S. 112]. Die **Selbstheilung** wird durch die Ergänzung von verteilten O/C-Einheiten umgesetzt. Analog zum biologischen Vorbild des Immunsystems werden so Selbstheilungsfunktionen integriert. Das Bild 4-5 zeigt eine

mögliche Integration am Beispiel eines vernetzten Systems (vgl. [GT18, S. 180]). Der Systemverbund besteht aus zwei Personenshuttle und einem Verkehrssystem auf übergeordneter Ebene. Das Verkehrssystem operiert auf der Planungsebene (z.B. Optimierung des

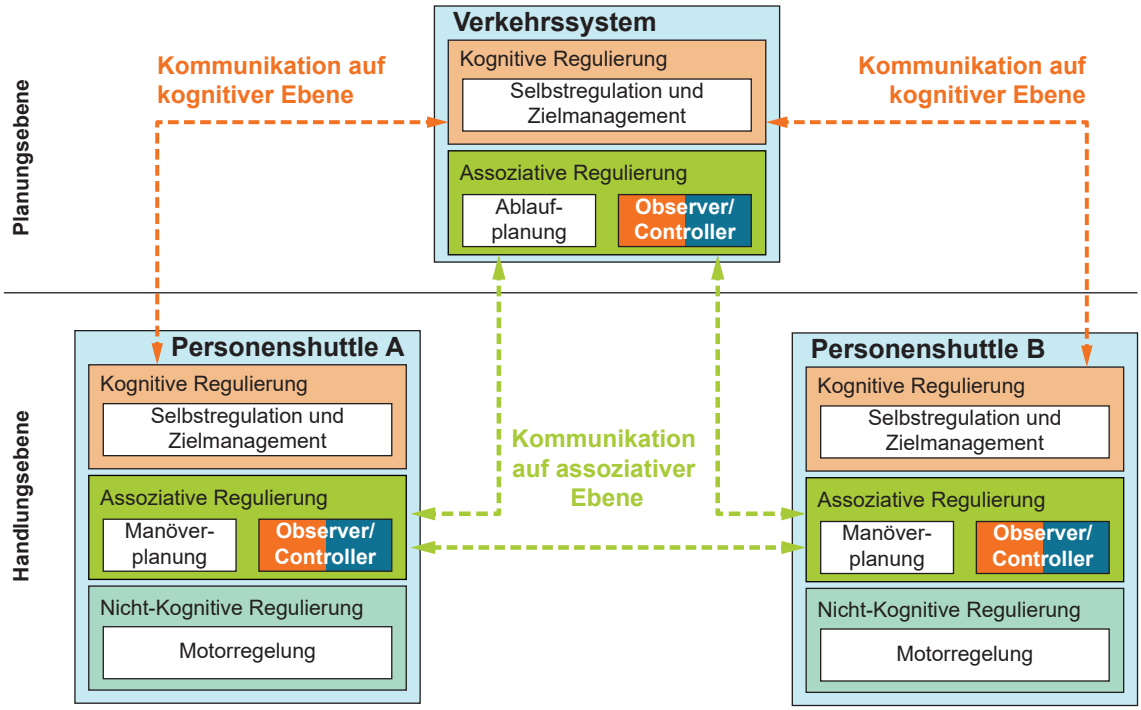


Bild 4-5: Integration der Selbstheilung in Intelligente Technische Systeme nach [GT18, S. 180]

Verkehrsflusses). Die anderen beiden Systeme auf der Handlungsebene (z.B. Navigation und Fahrdienstleistungen). In einem solchen vernetzten System werden so unterschiedliche Sebstheilungsstrategien realisiert. Auf der Handlungsebene etwa die Rekonfiguration von Hardware oder Software, auf der Planungsebene die Reorganisation oder die Zielanpassung. Das Management der Selbstheilung erfolgt über die assoziative Regulierung: Es operiert unter Echtzeit-Anforderungen und bietet eine Schnittstelle zur kognitiven Regulierung, z.B. Zielanpassung.

Im Folgenden wird das Framework selbstheilender Systeme erläutert. Das Framework ist modular, es berücksichtigt die Leistungsstufen und verortet die Entwurfsaspekte. Das Framework wurde im BMBF Projekt KI4AS entwickelt.

4.2.2.2 Modulares Framework

Das Bild 4-6 visualisiert das Framework. Es orientiert sich an der Observer/ Controller (O/C)-Architektur [SNS+11, S. 6f.] und [MT17, S. 178ff.]. Basis bildet das **SuOC**. Dies kann ein funktionales Teilsystemen, oder ein vernetztes System sein. In diesem SuOC sind Aktionen der Selbstheilung integriert. Der **Observer** und der **Controller** regeln den Selbstheilungsprozess. Über ein Kommunikationslayer kommunizieren die Einheiten untereinander und stellen Dienste bereit (**Dienst-API**). Über eine **Modell-API** werden die Wis-

sensbasen wie etwa Modellwissen (z.B. Modellparameter) konfiguriert und parametriert. Zudem greifen die O/C-Einheiten auf historische Daten zu. Über eine Benutzungsschnittstelle (**HMI**) hat der Bediener Zugriff auf die Daten und ihre Visualisierung und kann eine Rückmeldung an die O/C-Einheiten geben. Der Controller nutzt diese Rückmeldung und die Veränderung des Gesundheitszustandes zum verstärkenden Lernen und passt seine Selbstheilungsstrategie an.

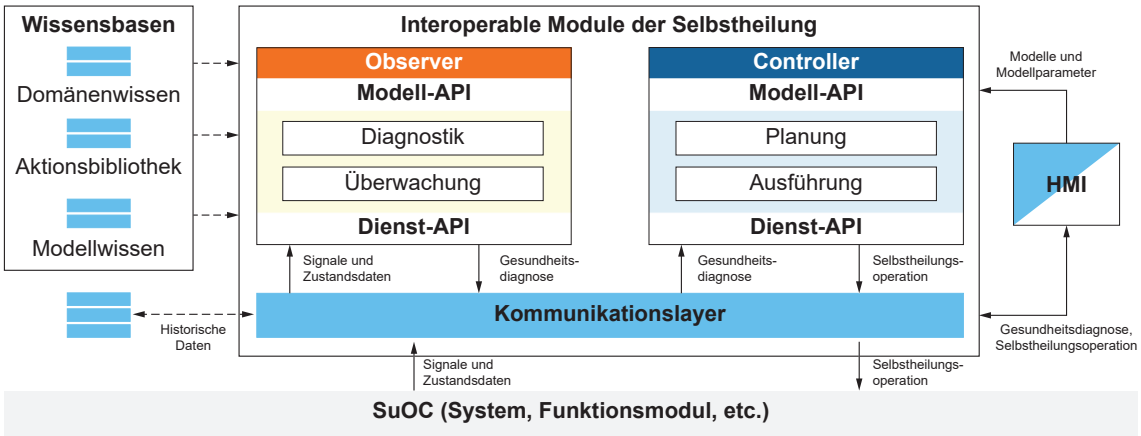


Bild 4-6: Übersicht des Frameworks

Die Interaktion von O/C-Einheiten untereinander lässt sich flexibel gestalten. So ist eine hierarchische Betrachtung möglich, um beispielsweise den Gesundheitszustand von Teilsystemen für ein übergeordnetes System zusammenzufassen. Dies erlaubt eine verbesserte Lokalisierung von Beeinträchtigungen im System. Gleichzeitig können Observer auch parallel verschiedene Sensoren überwachen (z.B. Kamera, Scanner).

Architekturvarianten:

Intelligente Technische Systeme können in einer Systemhierarchie strukturiert sein. Dies sind zum Beispiel: Komponente, Teilsystem, System und vernetztes System. Die Hierarchisierung erfolgt nach funktionalen oder physischen Kriterien [FG13, S. 473f.], [VDI2206, S. 17]. In Hinblick auf die Integration des modularen Frameworks in dieses Ordnungssystem von Intelligenten Technischen Systemen existieren verschiedene Varianten. Diese Varianten sind in Bild 4-7 zusammengefasst. In der **Zentralen Architektur** überwacht ein O/C-Modul das gesamte System. Die Selbstheilung erfolgt domänenübergreifend, zentral und global. Diese Variante eignet sich für Systeme mit Integralbauweise [FG13, S. 717ff.]. Bei einem Ausfall des O/C-Moduls ist die Selbstheilung jedoch nicht mehr gegeben. In der **Dezentralen Architektur** wird jedes Teilsystem durch je ein O/C-Modul überwacht. Die Selbstheilung erfolgt domänenspezifisch, dezentral und lokal. Diese Variante eignet sich für modulare Architekturen. Die **Hybride Architektur** repräsentiert eine Mischform. Die Selbstheilung erfolgt lokal und global. Aktionen der Selbstheilung werden global auf der Planungsebene (z.b. Reorganisation) und lokal auf der Handlungsebene (z.B. Adaptive Regelung) durchgeführt. Durch den dezentralen Ansatz wird die Verfügbarkeit und die Resilienz durch das gegenseitige Management der O/C-Module gesteigert. In der **Teil-**

Hybriden Architektur ist der Observer lokal, die Aktionen der Selbstheilung werden global durch einen Controller koordiniert. Die O/C-Module können ein Bestandteil des Intelligenten Technischen Systems sein oder kontrollieren das SuOC über ein Kommunikationssystem von außen. In den Folgenden Abschnitten werden die einzelnen Module im Detail erläutert.

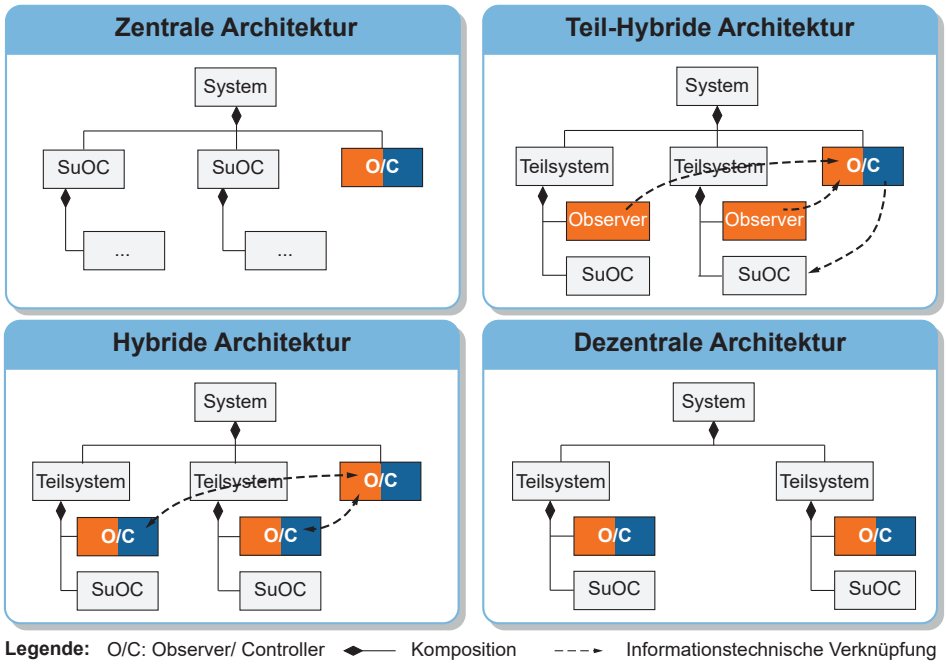


Bild 4-7: O/C-Architekturen in Anlehnung an [SNS+11, S. 29f.]

4.2.2.3 Modul: Observer

Die Modulspezifikation des Observer visualisiert Bild 4-8. Das Bild verdeutlicht sowohl den Informationsfluss, als auch den Lernfluss im Modul.

Merkmalsextraktion: Die Komponente erfasst die System- und Zustandsdaten des SuOC durch eine geeignete Signalverarbeitung. Die Komponente segmentiert das Signal, extrahiert relevante Merkmale oder dekomponiert Zeitserien in ihre Signalanteile. Mögliche Verfahren können sein: Fast-Fourier-Transformation, Wavelet-Verfahren, Spektrogrammverfahren.

Betriebszustandsanalyse: Diese Komponente ermittelt den aktuellen Betriebszustand. Sie ermöglicht eine zustandsabhängige und -sensitive Detektion von Anomalien:

- **Zustandsabhängigkeit:** Das Normalverhalten des Systems ist abhängig vom Betriebszustand. Dies erfordert eine Anpassung der Anomalie Detektion an den jeweiligen Betriebszustand.
- **Zustandssensitivität:** Die Anomalie Detektion lernt das zustandsspezifische Systemverhalten. Durch Wissen über den aktuellen Betriebszustand wird eine Anomalie mit dem Zustand assoziiert, d.h. es wird gelernt, ob eine Anomalie in einem Betriebs-

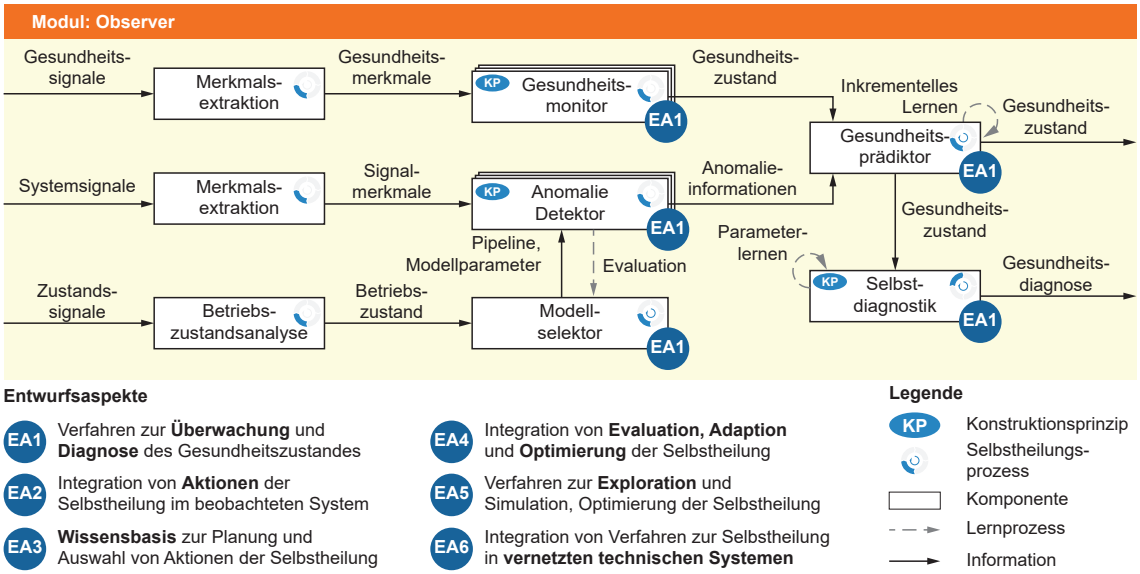


Bild 4-8: Modulspezifikation: Observer

zustand auftritt.

Modellselektor: Der Modellselektor dient zur Auswahl der Anomalie Detektion entsprechend des beobachteten Betriebszustandes. Der Modellselektor lernt aufgrund der Rückmeldung des Gesundheitsprädiktors, um bessere Pipelines für die gegebenen Betriebszustände zu generieren.

Anomalie Detektor: Der Anomalie Detektor überwacht die Signalmerkmale und detektiert signifikante Abweichungen im Normalverhalten der intendierten Funktion im SuOC. Der Detektor kann zustandsabhängig angepasst werden, um der Charakteristik der verschiedenen Signale in verschiedenen Betriebszuständen gerecht zu werden. Über die Merkmalsextraktion werden dem Anomalie Detektor charakteristische Signalmerkmale zugänglich gemacht.

Gesundheitsmonitor: Der Gesundheitsmonitor liefert eine Aussage zu dem Gesundheitszustand des Systems. Der Gesundheitszustand wird über Gesundheitssignale ermittelt. Diese Gesundheitssignale werden über Signalgeneratoren erzeugt. Der Zustandsübergang erfolgt graduell. Daher bieten sich graduelle Zugehörigkeitsfunktionen an. Für jedes Gesundheitssignal ist die Zugehörigkeit zu einem Gesundheitszustand über Fuzzy-Schwellwerte festzulegen:

- **Normal** ($x < 0.25$)**:** Das System operiert in einem gesunden, d.h. spezifizierten Zustand und verfolgt die Ziele und Aufgaben entsprechend der Zielvorgaben.
- **Warnung** ($0.25 \leq x < 0.5$)**:** Im System werden Anomalien detektiert. Der Trend geht in Richtung eines ungesunden Systemzustandes.
- **Kritisch** ($0.5 \leq x < 0.75$)**:** Im System wird eine Beeinträchtigung der Gesundheit beobachtet oder Anomalien detektiert, die in der Vergangenheit zu einer Beeinträchtigung geführt haben.

- **Fatal** ($x \geq 0.75$)**:** Das System steht vor einem kritischen Ausfall. Schwere Schäden am System, dem Bediener oder der Umgebung können die Folge sein.

Die Gesundheitssignale werden über Fuzzy-Pattern-Trees verarbeitet [Sen14]. Durch das Verfahren wird die Systemgesundheit in einem hierarchischen System durch verschiedene, unscharfe logische Operatoren wie der Konjunktion, der Disjunktion oder dem Durchschnitt bestimmt (vgl. Anhang A2.2.1). Auf diese Weise werden komplexe Zusammenhänge von Gesundheitssignalen abgebildet. Ein Beispiel zeigt Bild 4-9.

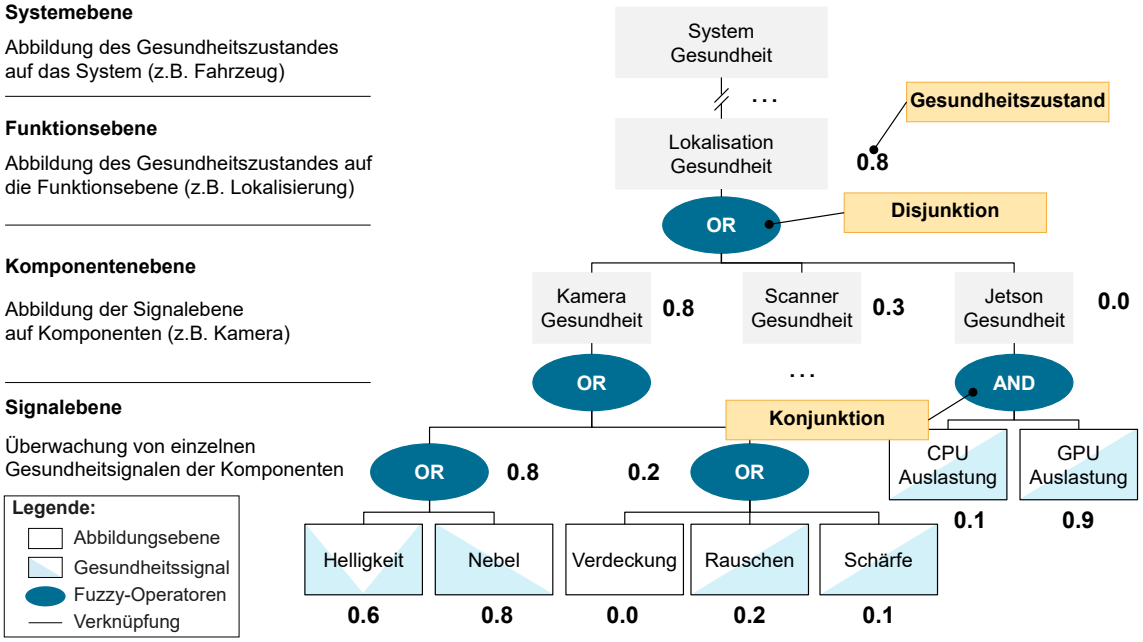


Bild 4-9: Abbildung von Gesundheitssignalen mit Hilfe von Fuzzy-Pattern-Trees

Gesundheitsprädiktor: Die Komponente verknüpft Anomalie Detektor und Gesundheitsmonitor. Der Gesundheitsprädiktor bewertet die Meldung der Anomalie Detektion, im Kontext des Gesundheitszustandes des Systems. Eine Anomalie wird, entsprechend der Zustände des Gesundheitsmonitors bewertet. Hierzu ist eine entsprechende Zeitspanne zu definieren, zu der die Anomalie mit der Veränderung des Gesundheitszustandes assoziiert werden kann.

Selbstdiagnostik: Ziel der Selbstdiagnostik ist die Ermittlung der Ursache anhand des beobachteten Gesundheitszustandes. Das Diagnosemodell beinhaltet eine Beschreibung möglicher Ursachen, Gesundheitszustände und dessen Beziehungswissen. Dabei existieren verschiedene Verfahren, die zur Abbildung der Zusammenhänge dienen. Bei einer undeutlichen Ursache, ist weitere Evidenz erforderlich. Die DIN ISO 17359 beschreibt verschiedene Verfahren, wie etwa die Wiederholung einer Messung oder das Einholen weiterer Expertise bezüglich einer Schadensart z.B. durch den Benutzer [DIN17359].

4.2.2.4 Modul: Controller

Die Spezifikation vom Controller zeigt Bild 4-10. Der Controller setzt die Planung und die Ausführung der Selbstheilung um. Der Controller setzt zwei Lernparadigmen um: Online, durch die Anpassung von Wahrscheinlichkeiten von Zustandsübergängen, und Offline durch die Exploration von Selbstheilungsstrategien auf der Wissensbasis [SH18, S. 22ff.]. Die Komponenten sind modular aufgebaut. Die Exploration kann je nach anzustrebender Leistungsstufe ergänzt werden. Den Kern des Moduls bildet das verstärkende Lernen mit

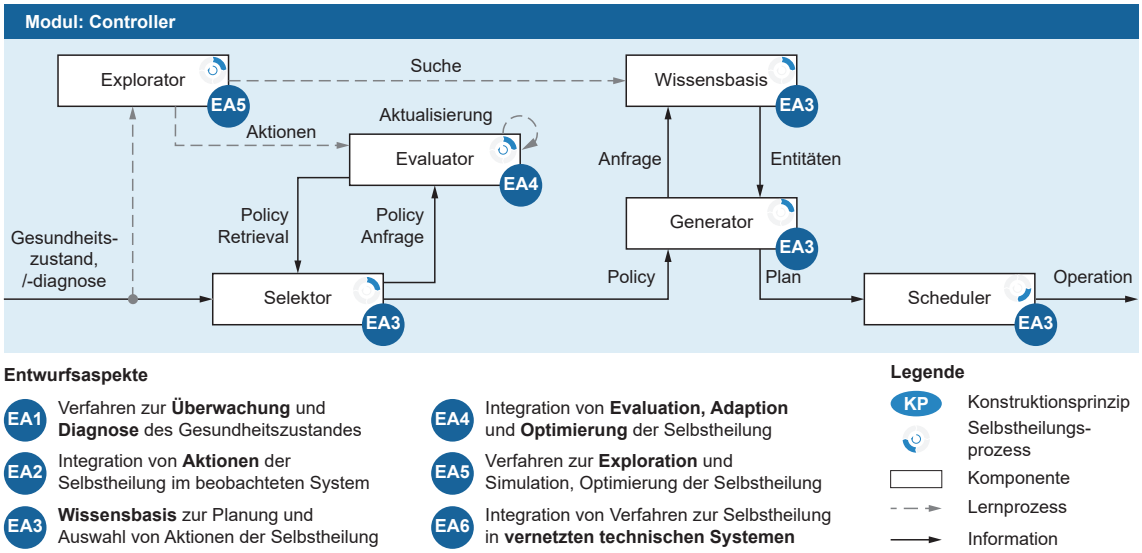


Bild 4-10: Modul: Controller

Hilfe eines Markov-Entscheidungsprozesses. Dieser Prozess dient dazu, eine optimale Selbstheilungsstrategie (policy) unter Berücksichtigung des Gesundheitszustandes und der Diagnose aus dem Observer zu explorieren und anzuwenden.

Evaluator: Kern dieser Komponente ist der Entscheidungsprozess als Modell der Selbstheilung. Die Gesundheitszustände $\mathsf{HS} \rightarrow \mathbb{N}_0$ des SuOC sind wie folgt bewertet:

$$\text{val(hs)} = \begin{cases} 0, & \text{hs = healthy} \\ 1, & \text{hs = warning} \\ 2, & \text{hs = critical} \\ 3, & \text{hs = fatal} \end{cases}$$

Gleichung 4-4: Äquivalenzwerte der Gesundheitszustände

Der Observer sendet dem Controller eine Diagnose $\mathsf{d_i} \in \mathsf{D}$. Diese beschreibt die Art und den Ort des Fehlers im SuOC. Darüber hinaus existieren eine Menge an Aktionen $\mathsf{A} = \{\mathsf{a_1}, ..., \mathsf{a_k}\}$ die zur Selbstheilung im SuOC durchgeführt werden. Der Zustandsraum ist definiert als:

$$S = \bigcup_{d \in D, a \in A} HS_{d,a}$$

Gleichung 4-5: Zustandsraum

Das Belohnungssystem ist mit $R = (r_{d,a,hs_{t+1},hs_t})$, $r \in \mathbb{R}_0^+$ und $\beta, \delta, \epsilon \in \mathbb{R}_0^+$ definiert (Formel 4-3).

$$r_{d,a,hs_t,hs_{t+1}} = \begin{cases} \beta \cdot (val(hs_t) - val(hs_{t+1}), & val(hs_t) - val(hs_{t+1}) > 0 \\ -\delta, & val(hs_t) - val(hs_{t+1}) = 0 \\ \epsilon \cdot (val(hs_t) - val(hs_{t+1}), & val(hs_t) - val(hs_{t+1}) < 0 \end{cases}$$

Gleichung 4-6: Belohnungsfunktion

Für die Wahrscheinlichkeiten im Entscheidungsprozess gilt:

$$p(hs_{t+1} \mid hs_t, b, d_i, a_i) = dir(x_i, \alpha_i)$$

Gleichung 4-7: Wahrscheinlichkeit

Das Lernen erfolgt durch die Anpassung der Wahrscheinlichkeiten mit Hilfe von Thompson-Sampling [RRK+17] und der Dirichlet-Verteilung kter Ordnung mit $\alpha_i = 1 \; \forall \; 1 \leq i \leq k$ (Formel 4-5). B ist die Beta-Funktion [SH18].

$$dir(x_1, ..., x_k; \alpha_1, ..., \alpha_k) = \frac{1}{B(\alpha_1, ..., \alpha_k)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$$

Gleichung 4-8: Dirichlet-Verteilung [SH18, S. 15]

Der α-Parameter ist ein Produkt aus der apriori-Übergangswahrscheinlichkeit und der Konfidenz k mit $k \in \mathbb{N}^+$. Der α-Parameter repräsentiert das apriori-Domänenwissen über den Entscheidungsprozess. Das Lernen erfolgt durch eine Anpassung des Konfidenzniveaus auf Basis der Beobachtungen aus dem SuOC.

Selektor: Der Selektor wählt eine Policy π der Selbstheilung für die betreffende Diagnose und den zugehörigen Gesundheitszustand im SuOC aus. Bei der Policy handelt es sich um eine Menge von Selbstheilungsaktionen, die das Verhalten des Controllers beeinflussen.

Formal ist die Policy eine Wahrscheinlichtkeitsverteilung der Aktionen für alle möglichen Gesundheitszustände [Lap20]. Der Selektor wählt eine Policy, welche die zu erwartende Belohnung maximiert. Für die optimale Policy π^* gilt nach Formel 4-6:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a_t \sim \pi(\cdot|hs_t)} \left[\sum_{t=0}^{T} \gamma^t r(hs_t, b, a_t) \right]$$

γ ist der Diskontierungsfaktor zwischen $0.9 \leq \gamma \leq 0.99$ [Lap20, S.39].

Explorator: Diese Komponente generiert neue Aktionen der Selbstheilung aus der Wissensbasis. Dabei werden bestehende Aktionen auf den Gesundheitszustand angewandt und bewertet. Diese Komponente setzt die dritte Stufe der Selbstheilung im System um.

Wissensbasis: Die Wissensbasis umfasst ein semantisches Beschreibung des SuOC. Insbesondere der Software und Hardware sowie das Beziehungswissen zwischen diesen Entitäten [SH18, S. 28]. Die Wissensbasis wird im Betrieb laufend aktualisiert. Beispielsweise mit dem Gesundheitszustand der Entitäten. Die Struktur der Ontologie zeigt Bild 4-11.

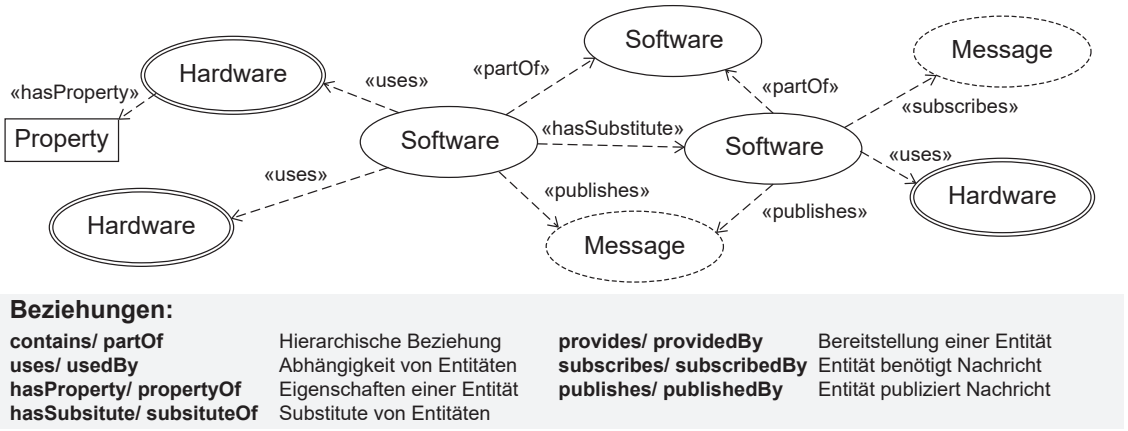


Bild 4-11: Repräsentation der Wissensbasis als Ontologie

Generator: Der Generator übersetzt die Policy in einen Plan. Dieser Plan beschreibt die Reihenfolge mit der Aktionen der Selbstheilung im SuOC durchgeführt werden. Beispielsweise welche Entititäten aktiviert oder isoliert und wie die Informationsflüsse im SuOC rekonfiguriert werden müssen.

Scheduler: Das Scheduling realisiert die Ablaufsteuerung. Die Ablaufsteuerung ruft die Aktionen der Selbstheilung im SuOC über Operationen sequentiell entsprechend der Planvorgaben des Generators aus. Der Datenlogger zeichnet die Aktivitäten und Ereignisse des Schedulers auf und aktualisiert die Wissensbasis. Diese Daten werden vom Evaluator genutzt, um die Wahrscheinlichkeiten im Entscheidungsprozess zu aktualisieren. Dies ermöglicht die Rückkopplung und somit ein Lernen erfolgreicher Aktionen.

4.2.3 Ausgewählte Konstruktionsprinzipien

In diesem Kapitel werden ausgewählte Konstruktionsprinzipien des Frameworks vorgestellt und mit Hilfe eines Ordnungsrahmens strukturiert. Konstruktionsprinzipien repräsentieren generalisiertes Lösungswissen für den Entwurf selbstheilender Systeme entlang der Leistungsstufen. Sie setzen dabei Funktionsbausteine im Observer und Controller um. Die Kombinierbarkeit ist über das entwickelte Framework und dessen Ein- und Ausgangsgrößen sichergestellt.

4.2.3.1 Konstruktionsprinzipien: Gesundheitsmonitor

Der Gesundheitsmonitor überwacht den Gesundheitszustand im SuOC. Dies kann durch eine Zustandsüberwachung oder durch eine Prognose zukünftiger Gesundheitszustände auf der Basis von historischen Zustandsdaten erfolgen. Der Gesundheitsmonitor wird für bekannte Ausfall- und Sicherheitsrisiken entwickelt. Diese leiten sich aus der systemischen Risikoanalyse ab (vgl. Kap. 4.4.2). Für jedes Ereignis werden die zu überwachenden *Gesundheitssignale* und *Signalquellen* identifiziert und die *Gesundheitsbereiche* festgelegt. Die Abbildung des Gesundheitszustandes erfolgt über einen Fuzzy-Pattern-Tree. Ein Gesundheitsmonitor kann verschiedene Aspekte überwachen:

- **Zustand:** Überwachung der Gesundheitssignale einer Komponenten im SuOC. Der Gesundheitsmonitor ist auf die Überwachung der Verfügbarkeit einer Komponente oder seinen Zustand (z.B. Vibration) gerichtet.
- **Leistung:** Überwachung wie gut das SuOC seine intendierte Funktion oder die vorgegebenen Ziele erfüllt. Die Überwachung erfordert die Definition einer Zielfunktion und -metrik. Systemziele werden aus dem Zielsystem abgeleitet.
- **Ergebnis:** Überwachung einer Ergebnisgröße einer Aktivität oder eines Prozesses im SuOC. Ergebnisgrößen leiten sich aus den Anwendungsfällen ab und sind durch eine Qualitätsmetrik spezifiziert (z.B. Prozessergebnis).

Das Bild 4-12 gibt eine Übersicht zu ausgewählten Konstruktionsprinzipien für den Gesundheitsmonitor. Das Ordnungssystem erfolgt nach Aspekt und nach Art des Konstruktionsprinzips in deskriptive und prognostische Verfahren. Weitere Verfahren sind in Anhang A2.1.2 erläutert.

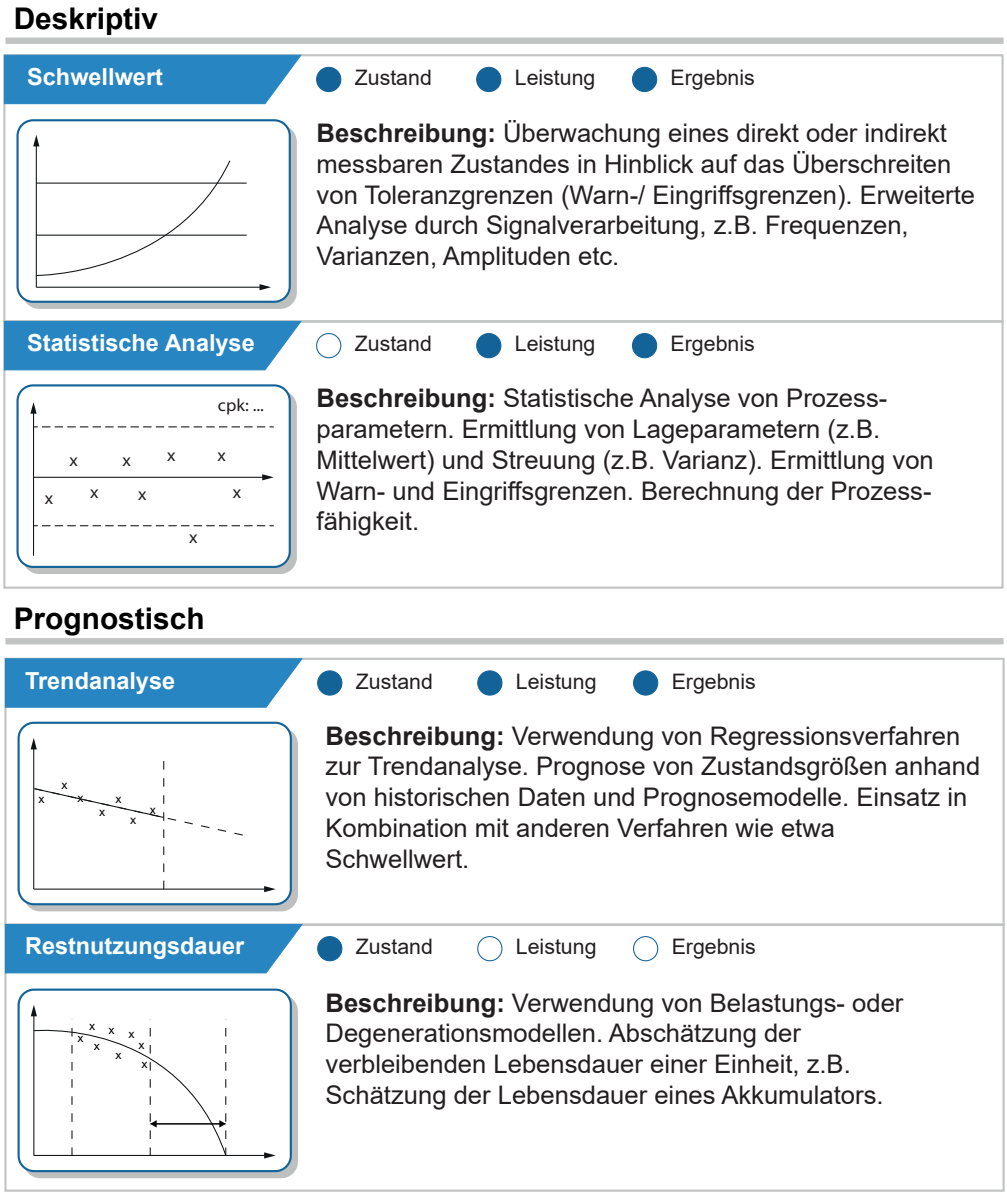


Bild 4-12: Ausgewählte Konstruktionsprinzipien für die Gesundheitsüberwachung

4.2.3.2 Konstruktionsprinzipien: Anomalie Detektion

Ziel der Anomalie Detektion ist die Detektion von signifikanten Abweichungen im Normalverhalten im SuOC. Die Abweichungen können als Folge von Funktionsstörungen, Verhaltensanpassungen oder durch Umgebungseinflüsse auftreten. Dabei ist die Art der Anomalie, die Größe und der Ort einer Anomalie sowie das zeitliche Auftreten der Anomalie zu erkennen. Die Anomalie Detektion ist auf das Verhalten eines Systems gerichtet. Das Verhalten wird durch Aktivität, Zustand oder Sequenzen repräsentiert:

- **Aktivität:** Detektion von Anomalien in einer Aktivität wie z.B. Drehzahl erzeugen. Beobachtung von Größen wie Strom, Drehzahl und Temperatur in einer Aktivität.
- **Zustand:** Detektion von Anomalien in einem Systemzustand. Beobachtung von multivariaten Signalen, z.B. Zustand: Fahren.

- **Sequenz:** Detektion von Anomalien in einem Ablauf von Ereignissen und Zuständen, z.B. sequentielle Systemaufrufe bei der Ausführung von Tasks.

Ein Ordnungsrahmen unterstützt bei der frühzeitigen Auswahl geeigneter Konstruktionsprinzipien für die Anomalie Detektion (vgl. Bild 4-13). Der **Eingang** beschreibt die Eingangssignale und deren Eigenschaften. So z.B. kategorische Daten, quasi-lineare Signale oder stochastische Signale wie etwa Sensordaten. Der Aspekt **Verfahren** legt fest, welche Art des Verhaltens - *Aktivität, Zustand oder Sequenz* - überwacht werden soll und wie eine Anomalie in den Signalen repräsentiert ist. Dies sind nach CHANDOLA *Punktanomalien, kontextuelle Anomalien* oder *kollektive Anomalien* [CBK09, S. 7ff.]. Der **Ausgang** definiert das Ergebnis der Anomalie Detektion als *Label* oder *Score*. Die **Datenverfügbarkeit** berücksichtigt das Lernverfahren. Gibt es Trainingsdaten zum Normalverhalten, eignen sich z.B. *semi-überwachte Verfahren*. Eine Anomalie Detektion kann weiteren **Randbedingungen** unterliegen, z.B. Zielplattform.

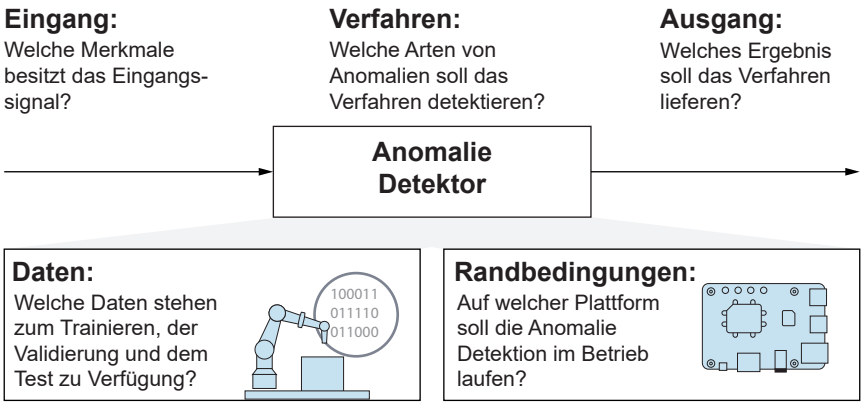


Bild 4-13: Ordnungsrahmen zur Auswahl von Anomalie Detektoren

Entsprechend des Ordnungsrahmens sind die Konstruktionsprinzipien nach Aktion, Zustand oder Sequenzen strukturiert. Das Bild 4-14 zeigt ausgewählte Konstruktionsprinzipien für die Anomalie Detektion. Weitere Konstruktionsprinzipien sind in Anhang A2.1.1 zusammengefasst. Die Detektoren wurden im Forschungsprojekt KI4AS in einer Software-Bibliothek umgesetzt. Ein Bewertungskatalog entlang des Ordnungsrahmens unterstützt bei der qualitativen Vorauswahl (vgl. Kap. A2.2.2).

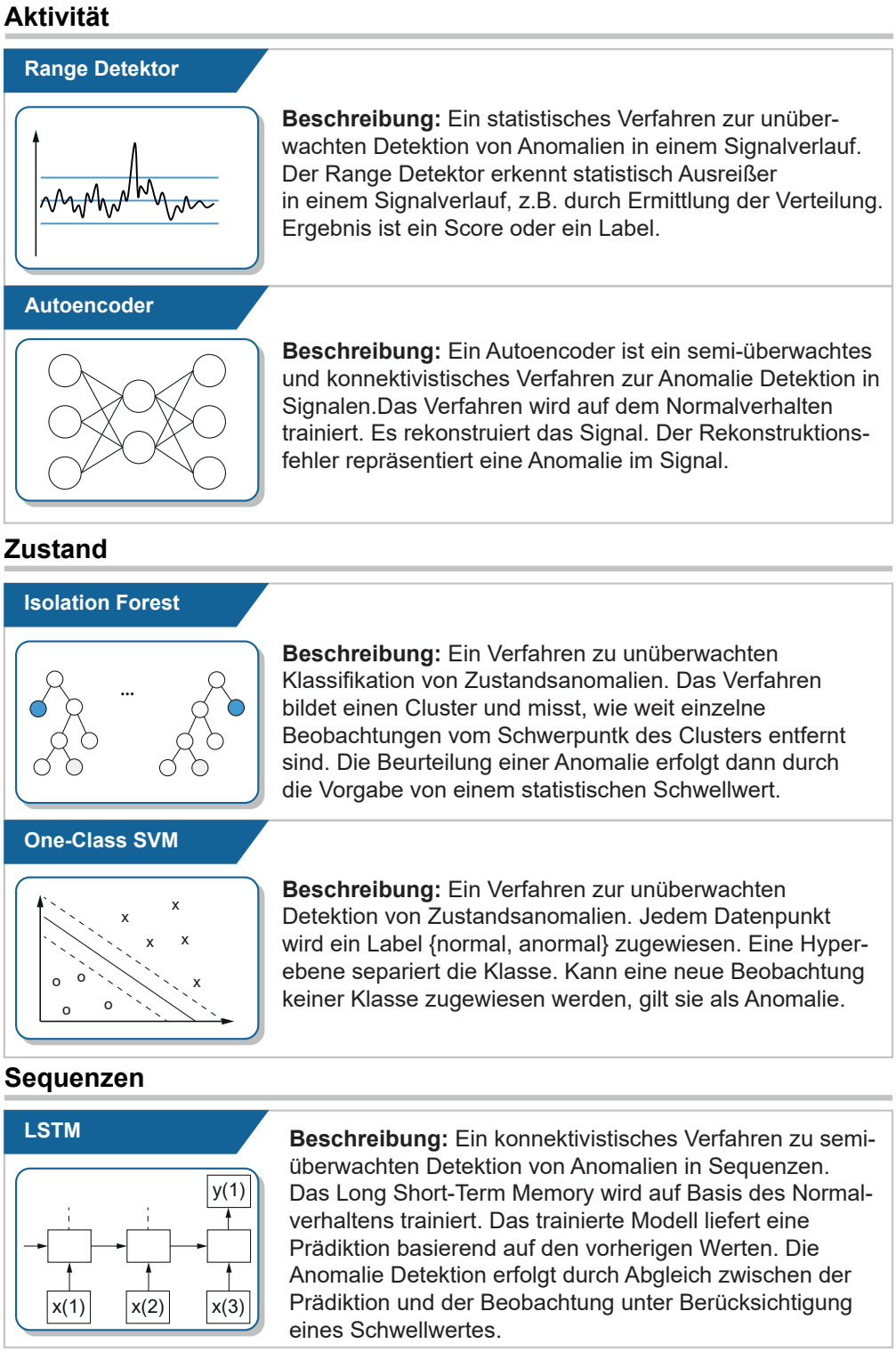


Bild 4-14: Ausgewählte Verfahren der Anomalie Detektion (Auswahl)

4.2.3.3 Konstruktionsprinzipien: Selbstdiagnostik

Die Aufgabe der Selbstdiagnostik besteht in der Bestimmung der Art eines Fehlers und der Lokalisierung im System auf Grundlage der Beobachtungen aus der Überwachung. Die Konstruktionsprinzipien sind in drei Bereich eingeordnet: Die **Inferenzmethoden**

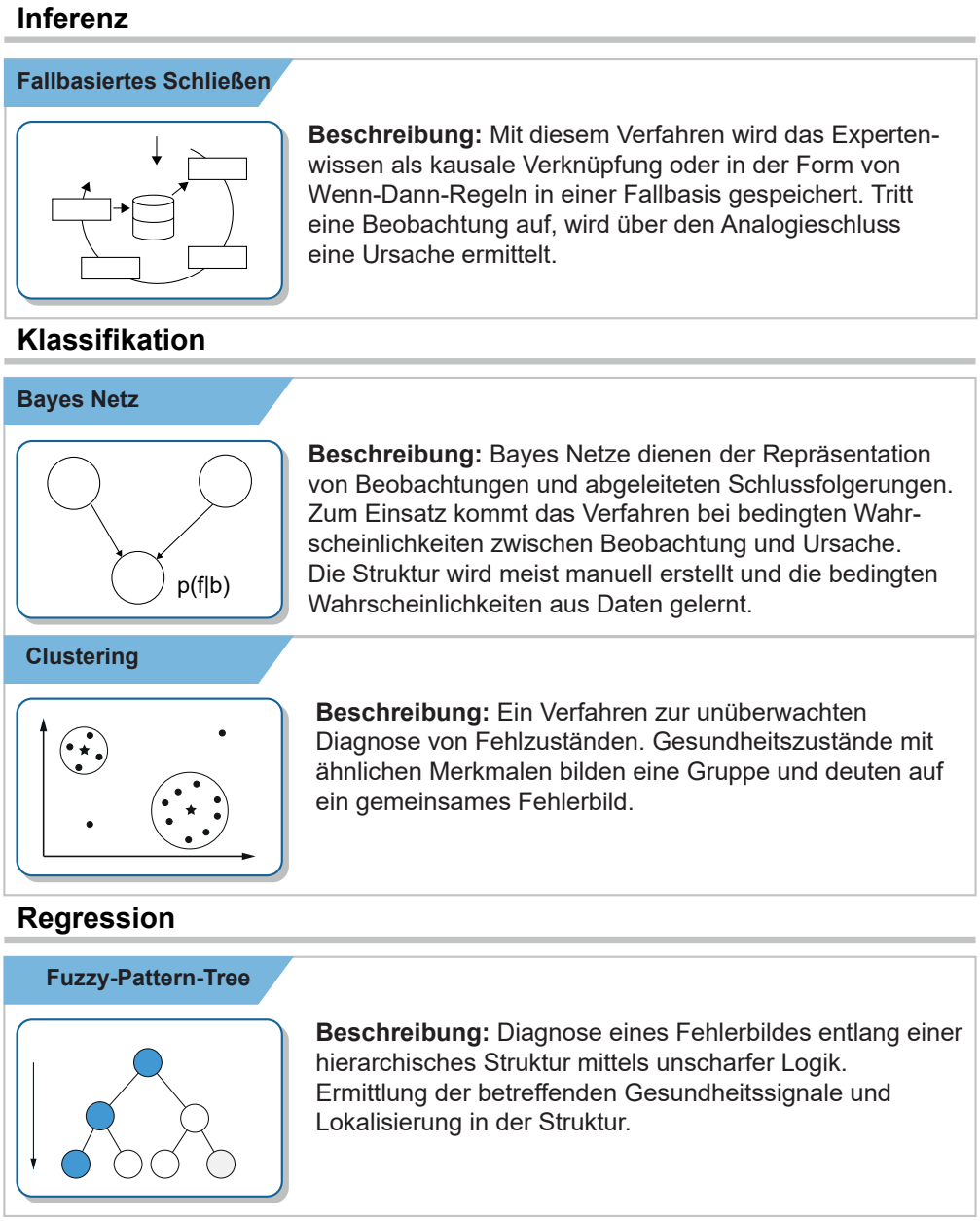


Bild 4-15: Ausgewählte Konstruktionsprinzipien der Diagnostik

bilden eine Kausalität von Gesundheitszustand, Fehler und Ereignis ab. Die **Klassifikationsmethoden** klassifizieren ein Fehlerbild anhand des Gesundheitszustandes und die **Regressionsmethoden** bilden Beziehungen zwischen einer abhängigen und einer oder mehreren unabhängigen Variablen ab [Ise08, S. 39f.]. Das Bild 4-15 beschreibt ausgewählte Konstruktionsprinzipien der Diagnostik. Weitere Konstruktionsprinzipien sind in Anhang A2.1.3 dargestellt.

4.2.3.4 Konstruktionsprinzipien: Selbstheilungsaktion

Die Selbstheilungsaktionen dienen zur Wiederherstellung eines normalen Gesundheitszustandes. Eine Strukturierung der Selbstheilungsaktionen erfolgt nach Systemebene (Level

0: Komponente, Level 1: Teilsystem, Level 2: System und Level 3: Vernetztes System) sowie nach der Ebene der selbstregulierenden Informationsverarbeitung. Die Konstruktionsprinzipien sind in drei Kategorien gegliedert: Parameteranpassung, Strukturanpassung und weitere Aktionen (Bild 4-16).

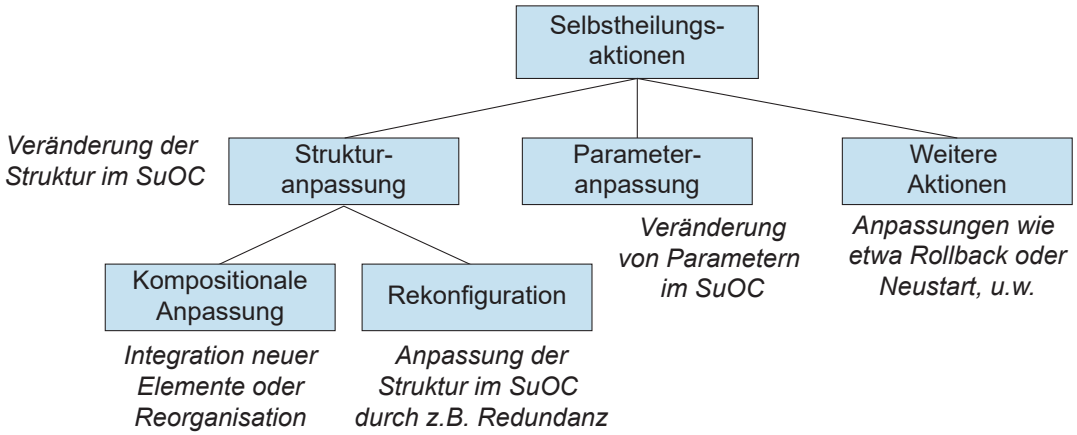


Bild 4-16: Strukturierung der Selbstheilungsaktionen [GRS14, S. 7]

Das Bild 4-17 visualisiert eine Auswahl von Selbstheilungsaktionen. Weitere Prinzipien sind in Anhang A2.1.4 zusammengefasst. Diese Konstruktionsprinzipien repräsentieren Lösungswissen. Im Rahmen der systemischen Risikoanalyse sind für die einzelnen Beeinträchtigungen mögliche Selbstheilungsaktionen mit den Domänenexperten zu erarbeiten und in der Prinziplösung selbstheilender Systeme zu modellieren (vgl. Kap. 4.4.2). Fällt zum Beispiel ein Sensor aus, wäre die Rekonfiguration zur Aufrechterhaltung der Funktion ein möglicher Lösungsansatz. Daraus können weitere Anforderungen resultieren.

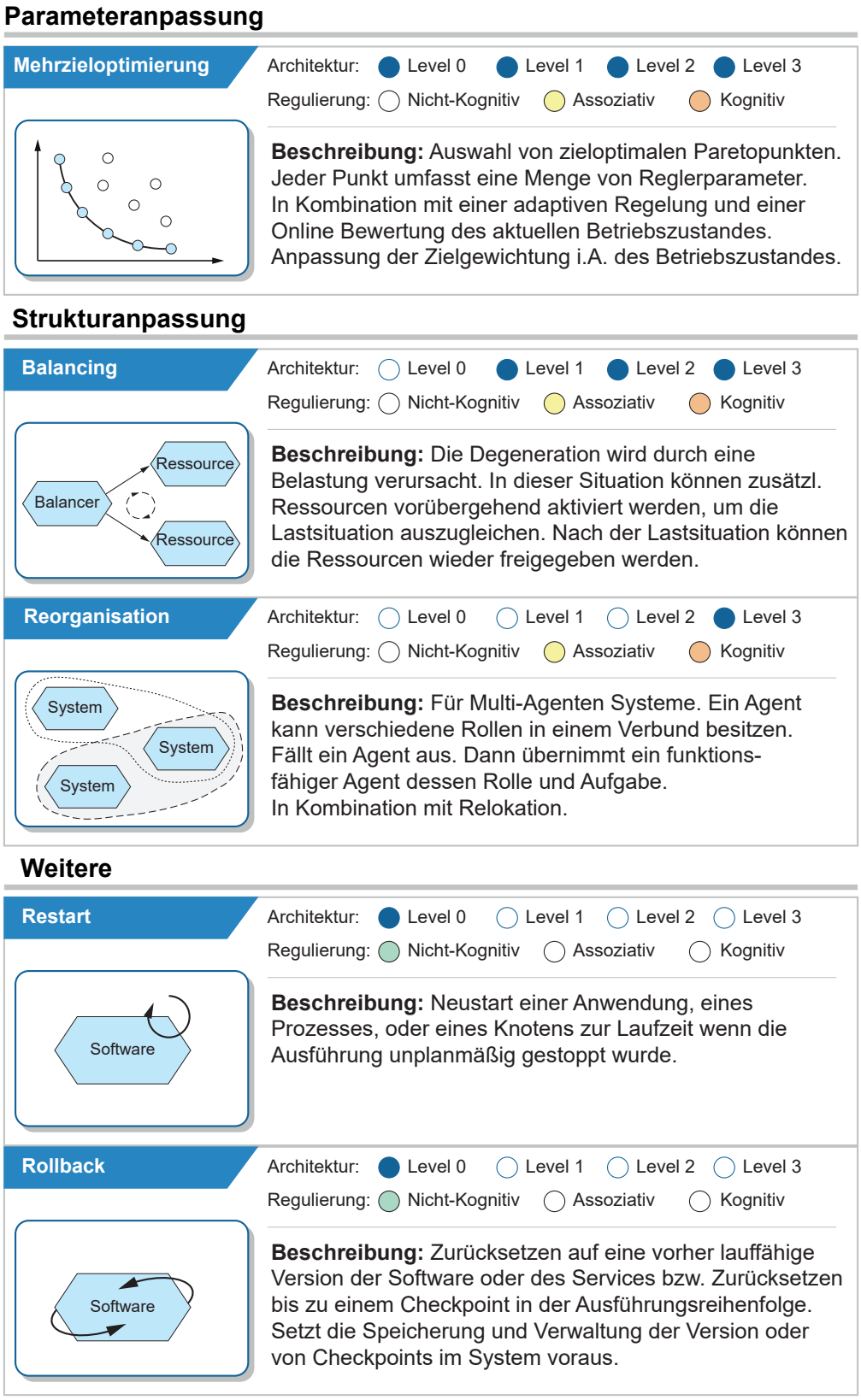


Bild 4-17: Ausgewählte Konstruktionsprinzipien der Selbstheilung

4.3 Entwurfstechnik für selbstheilende Systeme

Die Integration der Selbstheilung erfolgt im Systementwurf durch eine Potenzialanalyse. Die erforderlichen Aspekte sind in dem Sprachkonzept zu berücksichtigen. CONSENS (vgl. Kap. 3.3.1.2) liefert eine interdisziplinäre Beschreibungssprache komplexer technischer Systeme und orientiert sich am Systementwurf. Die Sprache wurde zur Modellierung von sicherheitskritischer Aspekte sowie um ein modellbasiertes Absicherungskonzept erweitert. Die Erweiterungen werden in den jeweiligen Partialmodellen hervorgehoben.

4.3.1 Systemkonzeption

Ergebnis der Systemkonzeption ist die Prinziplösung selbstheilender autonomer Systeme. Zur Modellierung dieser Systeme wurde die Sprache CONSENS erweitert (vgl. Bild 4-18). Die Erweiterungen betreffen die Modellierung von Anwendungsszenarien, Umfeld, Anforderungen, Funktionen, Wirkstruktur und das Zielsystem.

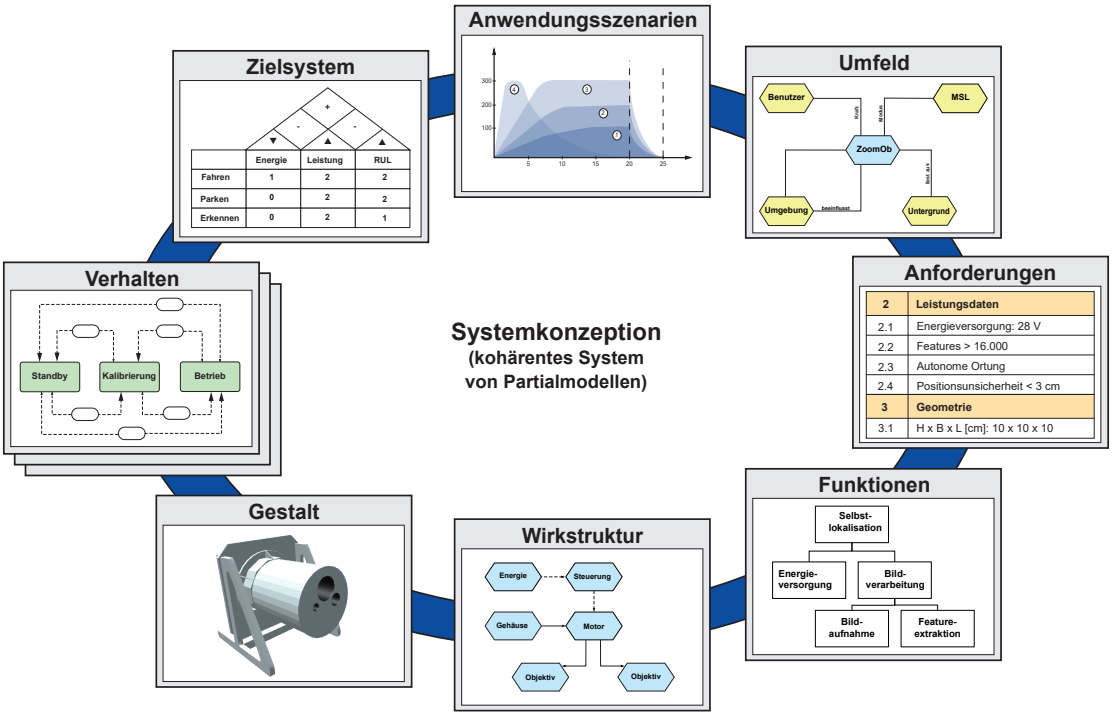


Bild 4-18: Erweiterte Partialmodelle der Systemkonzeption in Anlehnung an [GTS14, S. 83]

Umfeldmodell: Das Umfeldmodell legt die Systemgrenze fest. In diesem Aspekt wird das System als *„Black Box“* abgebildet und die Interaktion sowie die Einflüsse zwischen dem System mit seinem sozio-technischen Umfeld modelliert. Das Umfeld wird über *Umgebungselemente* und deren Wechselwirkung mit dem System modelliert. Eine geeignete Klassifizierung von Elementen zur Modellierung von Strukturmodellen liefert [Kai14, S. 74ff.]. Die Elemente werden weiter verfeinert in *Agent*, *Ambiente* und *Objekt*. Diese Unterscheidung ermöglicht eine Vernüpfung mit dem Szenariobaum (s. Testkonzeption).

Das Profil zur Modellierung des Umfeldmodells ist in Anhang A2.4 Bild A-22 dargestellt.

Anwendungsszenario: Ein Anwendungsszenario beschreibt eine Betriebssituation und das gewünschte Systemverhalten. Das Systemverhalten kann textuell oder durch Verhaltensmodelle konkretisiert werden. Neben den Anwendungsszenarien werden *Fehlanwendungsfälle*, *Gefährdungsszenario* und *Unfallszenario* sowie verschiedene Beziehungen in diesem Partialmodell berücksichtigt (vgl. Bild 4-20). *Fehlanwendungen* konkretisieren die Störflüsse und beschreiben einen missbräuchlichen Anwendungsfall. Ein Fehlanwendungsfall beschreibt eine ungewollte Interaktion zwischen einem Stakeholder oder einem externen System mit dem System, die das System nicht zulassen bzw. verhindern soll. SINDRE ET AL. haben das Konzept der Fehlanwendungsfälle eingeführt [SO19]. Die Modellierung erfolgt in CONSENS durch eine Erweiterungsbeziehung vom Anwendungsszenario. Die ISO/PAS 21488 definiert mögliche Fehlanwendungen und Leitwörter zur Identifikation möglicher Fehlanwendungen. Diese sind in Bild 4-19 dargestellt [ISO 21448].

Prozess	Leitwort	Beispiel
Wahrnehmen	Nicht verstehen	Fehlbedienung aufgrund einer komplizierte Bedienung
	Falsch erkennen	Kognitive Überlastung/ Überladung
Bewerten	Fehleinschätzen	Falsche Reaktion aufgrund einer Fehleinschätzung
Agieren	Irren/ Versehen	Ablenkung, Konzentrationsverlust
	Beabsichtigen	Beabsichtige Fehlanwendung (Verstoß gegen Sicherheitsregeln)
	Unfähig	Nicht Beherrschbarkeit einer Betriebssituation

Bild 4-19: Leitwörter zur Identifikation von Fehlanwendungen (vgl. [ISO 21448, S. 47])

Eine *Gefährdungsszenario* beschreibt einen Umstand bei der eine Person, das System oder die Umwelt mindestens einer Schadensquelle ausgesetzt ist [IEC61508b]. Eine Schadensquelle kann durch eine Fehlanwendung des Systems begründet sein. Eine Gefährdungsszenario ist mit der Eintrittswahrscheinlichkeit verknüpft. Fehlanwendungen und Gefährdungsszenario können zu einer Unfallsituation führen. Die *Unfallsituation* wird mit der Schwere ihrer Folgen bewertet. Durch Kombination von Eintrittwahrscheinlichkeit und Schwere ergibt sich das Risiko der Gefährdungsszenario. In Anhang A2.4 Bild A-19 ist das erweiterte Profil der Anwendungsszenarien abgebildet.

Konkrete Syntax	Beschreibung
«Misuse»	**Fehlanwendungsfall:** Zu verhindernde Funktion
«Hazard»	**Gefährdungsszenario:** Mögliche Gefährdung in einem Anwendungsszenario
«Accident»	**Unfallszenario:** Ereignis, bei dem ein Schaden entsteht
«attacks»	**attackiert:** Fehlanwendung attackiert Anwendung
«prevents»	**verhindert:** Anwendung verhindert Fehlanwendung
«causes»	**verursacht:** Gefährdungsszenario verursacht Unfall
«Misactor»	**Misakteur:** Akteur des Fehlanwendungsfalls

Bild 4-20: Erweiterte Modellelemente für das Partialmodell Anwendungsszenario

Anforderungen: Dieser Aspekt umfasst eine strukturierte Sammlung aller Anforderungen an das System. Jede Anforderung wird lösungsneutral, textuell und möglichst quantifizierbar beschrieben [GTS14, S. 85]. Durch Ableitungs- und Erweiterungsbeziehungen werden Anforderungen in einen Zusammenhang gebracht. Anforderungen leiten sich aus dem Umfeld und den Anwendungsszenarien ab. Zur Modellierung sicherheitsrelevanter Anforderungen, wurde das Profil um Sicherheitsanforderungen erweitert. Anforderungen an die Sicherheit leiten sich aus den Fehlanwendungsfällen und Gefährdungsszenario ab. Für jede Sicherheitsanforderung ist ein Integritätslevel nach der IEC 61508 anzugeben [IEC61508b]. Das Profil zur Modellierung von Anforderungen ist in Anhang A2.4 Bild A-20 dargestellt.

Funktionen: Funktionen resultieren aus den Anwendungsszenarien des Systems. Ausgehend von der Hauptfunktion wird die Funktionalität des Systems hierarchisch gegliedert [GTS14, S. 85]. Das Profil unterscheidet verschiedene Funktionen. Dabei erfolgt eine Partitionierung der lösungsneutralen Funktion in die Domänen *Mechanik*, *Software*, *Sensorik*, *Aktorik* und *Kommunikation*. Jeder Funktion kann ihr Sicherheitslevel zugewiesen werden[1]. Die Software-bezogene Funktionen besitzen weitere Attribute wie ihre Einordnung in die Schichtenarchitektur Intelligenter Technischer Systeme und ihre Einordnung in Sensieren, Planen und Agieren. Handelt es sich um eine Funktion des maschinellen Lernen können die erforderlichen Daten angegeben werden. Das Profil zur Modellierung von Funktionen ist in Anhang A2.4 Bild A-21 dargestellt.

Wirkstruktur: Die Wirkstruktur beschreibt die disziplinübergreifende Struktur des Systems. Die Wirkstruktur umfasst *Systemelemente*, *Ports* und *Flussbeziehungen*. Das Profil

[1] Dieses leitet sich aus der Risikobewertung der Gefährdungsszenario ab. Eine Funktion ist über das Anwendungsszenario mit der Gefährdungsszenario verknüpft.

unterscheidet verschiedene Arten von Systemelementen: Sensorik, Aktorik, Mechanik, Kommunikation, Software und Elektronik. Die Elemente orientieren sich an der Grundstruktur fortgeschrittener mechatronischer Systeme. Zudem können die Systemelemente mit einem Domänen-Tag ergänzt werden. Der Tag gibt an, welcher Domäne das Systemelement primär zuzuordnen ist (z.B. Software). Der Informationsfluss kann nach *Nachricht*, *Service*, *Aktion* weiter konkretisiert werden. Ein Informationsfluss kann mindestens ein Datum beinhalten. Ein *Datum* wird durch Daten- und Werttyp, Auflösung und Wertebereich konkretisiert. Das Profil zur Modellierung der Wirkstruktur ist in Anhang A2.4 in Bild A-22 sowie der Signale Bild A-23 visualisiert. Bei der Anwendung der Sprache sind die Modellierungsrichtlinien nach KAISER zur Erstellung plausibler, vollständiger und richtiger Systemstrukturen zu berücksichtigen [Kai14, S. 92ff.].

Zielsystem: Das Zielsystem legt die externe, internen und inhärenten Ziele des Systems fest. *Externe Ziele* sind durch den Benutzer vorgegeben. Die *inhärenten Ziele* dienen dem Systemzweck. Externe und inhärente Ziele, die in einer Situation durch das System verfolgt werden, sind die *internen Ziele* des Systems [Poo11]. Die Modellierung des Zielsystems erfolgt nach Bild 4-21. Zunächst werden die Ziele des Systems oder einer spezifischen Funktion ermittelt und die *Zielrichtung* für jedes Ziel festgelegt. Anschließend wird die *Zielkorrelation* bewertet. Die Zielkorrelation beschreibt, ob Ziele zueinander im Konflikt stehen, neutral sind oder sich verstärken. Für jedes Ziel wird anschließend ein *Zielparameter* festgelegt. Aus den Anwendungsszenarien werden, für das Zielsystem relevante, *Situationen* abgeleitet. Zuletzt wird für jede Situation die *Zielrelevanz* ermittelt. Die Zielrelevanz beschreibt die Wichtigkeit eines Ziels in einer spezifischen Situation. Sie dient zur Gewichtung der Ziele. Das Profil zur Modellierung des Zielsystems zeigt Anhang A2.4 Bild A-24.

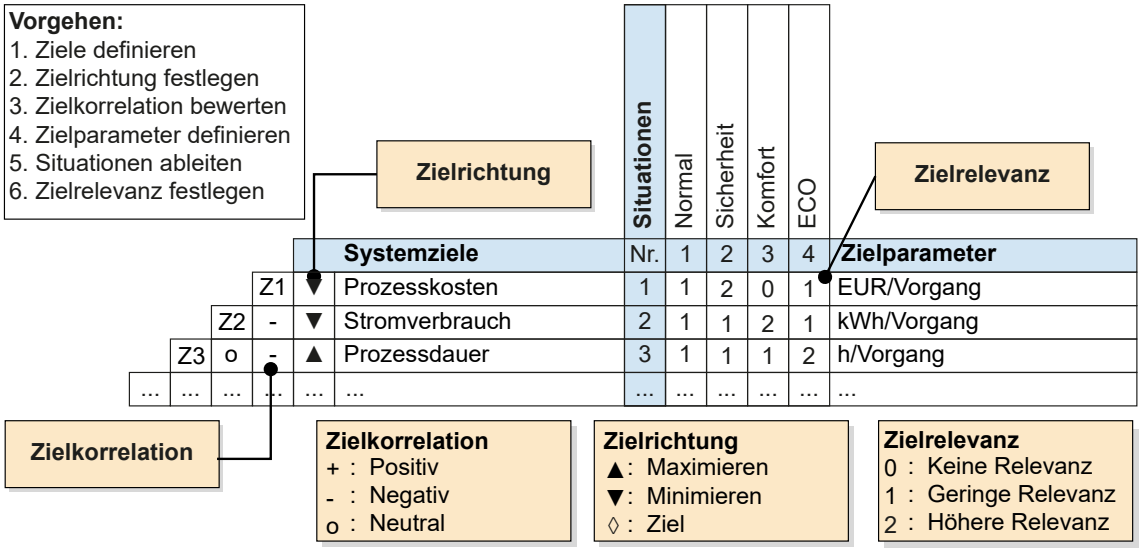


Bild 4-21: Zielsystem

4.3.2 Absicherungskonzeption

Das modellbasierte Absicherungskonzept ist integraler Bestandteil der Spezifikation und dient als Basis für die fortlaufende Integration und Absicherung im Entwicklungsprozess. Das Absicherungskonzept umfasst eine kohärente Menge an Partialmodellen die über die Anforderungen mit dem Systemkonzept verknüpft sind (vgl. Bild 4-22) Das Profil der Absicherungskonzept ist in Anhang A2.4 Bild A-25 zusammengefasst.

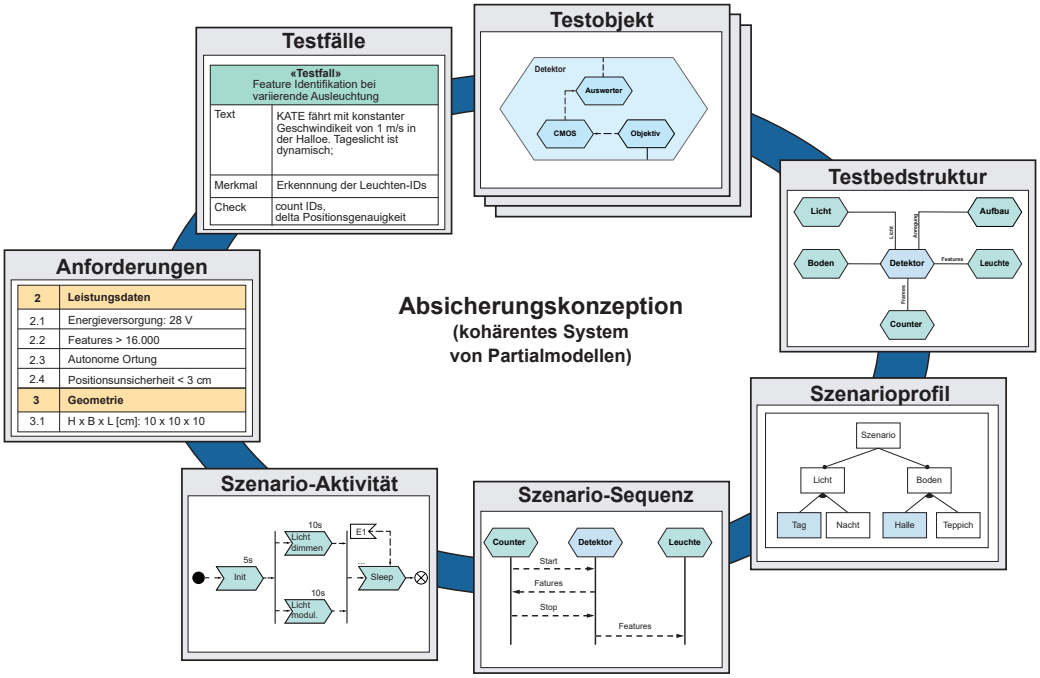


Bild 4-22: Partialmodelle des Absicherungskonzeptes

Testfälle: Der Testfall dient zur Überprüfung einer in den Anforderung zugesicherten Eigenschaft oder Funktion eines Testobjektes. Ein Testfall besitzt eine eindeutige Nummer und einen Namen, sowie eine textuelle Beschreibung. Ein Testfall beinhaltet mindestens ein überprüfbares Merkmal. Darüber hinaus kann der Ablauf des Testfalls textuell oder durch Aktivitäten modelliert werden. Ebenso ist genau eine Anforderung zu referenzieren. Durch diese Referenzierung wird im Modell geprüft, ob jede Anforderung durch mindestens einen Testfall abgesichert ist. Testfälle können zudem auf einzelne Teststufen allokiert werden, z.B. Komponente, System. Über die Verbindung Testfall zu Anforderung zu Systemelement ist das Testobjekt implizit im Modell definiert.

Konkrete Syntax	Beschreibung
«Testcase»	**Testfall:** Beschreibt elementaren und funktionalen Test
«Requirement»	**Anforderung:** Geforderte Eigenschaft eines Systems
«verify»	**Verifiziert:** Verknüpfung von Testfall und Anforderung
	Assoziiert: Verknüpfung von Akteur zu Testfall
«trace»	**Verknüpft:** Verknüpft Testfall mit Virtuellen Szenario
	Akteur: Verantwortliche Rolle zur Durchführung des Tests
	Allokation: Zuordnung von Testfällen zu einem Virtuellen Testbed

Bild 4-23: Modellelemente für das Partialmodell Testfall

Testobjekt: Das Testobjekt ist ein Teil des Systemkonzeptes, das mit dem Testfall untersucht werden soll. Grundlage zur Auswahl des Testobjektes ist auch eine mögliche Konfiguration oder verschiedene Varianten. Dies ist abhängig vom Stand der Systemkonzeption und den ggf. noch abzuwägenden Varianten. Grundlage sind die in der Systemkonzeption erarbeiteten *«implements»* und *«excludes»*-Beziehungen zwischen Systemelementen.

Testbedstruktur: Dieser Aspekt spezifiziert das Testbed und seine Wechselwirkungen mit dem Testobjekt. Hierzu wird das Partialmodell Wirkstruktur um das *«Testelement»* erweitert. Das Testelement definiert spezifische Elemente des Testbeds und wird genutzt, um mit dem Testobjekt zu kommunizieren (Material-, Energie-, Informationsfluss oder einer mechanische Verbindung). Dies sind zum Beispiel Elemente eines Prüfstandes. Die Testbedstruktur dient als Basis für die Konkretisierung von virtuellen oder realen Prüfständen und legt relevante Schnittstellen am Testobjekt im Systementwurf fest.

Szenarioprofil: Das Szenarioprofil ist eine Repräsentation aller relevanter parametrierbaren Szenarioelemente und deren statistische Verteilung in der Betrieb des Systems. Das Szenariomodell bildet eine konsistente Kombination von Szenarioelementen aus diesem Szenarioprofil. Das Szenarioprofil ist ein hierarchisches Diagramm, welches die Bestandteile eines Szenarios in Gruppen mit zunehmender Konkretisierung beschreibt. Das Bild 4-24 zeigt die Elemente zur Modellierung eines Szenarioprofils. Ein Szenarioelement kann sich aus mehreren Elementen zusammensetzen. Diese sind erforderlich oder optional. Die Variablität wird über logische Verknüpfungen realisiert. Bei einer UND-Beziehung sind alle Elemente relevant. Bei einer XODER-Beziehung ist genau ein Szenarioelement auszuwählen. Die ODER-Beziehung legt eine minimale und eine maximale Anzahl von Szenarioelementen fest. Szenarioelemente können sich gegenseitig ausschließen (exkludiert) oder gegenseitig erfordern (erfordert). Jedes Szenarioelement umfasst Parameter, eine Beschreibung und eine Angabe zur statistischen Häufigkeit im Betrieb des Systems.

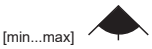
Konkrete Syntax	Beschreibung
	**Szenarioelement:** Objekte oder Merkmale im Betrieb mit Häufigkeit
	**Teilelement:** Szenarioelement e_1 ist Konkretisierung von e
	**Parameter:** Parameter eines Element mit min/max-Wertebereich
	**Beschreibung:** Textuelle Beschreibung eines Szenarioelements
	Optional: Optionales Szenarioelement im Szenario
	Erforderlich: Erforderliches Szenarioelement im Szenario
	**UND-Verknüpfung:** Alle Teilelemente erforderlich
	**XODER-Verknüpfung:** Auswahl von max. einem Szenarioelement
	**ODER-Verknüpfung:** Auwahl von min. bzw. max. Anzahl von Teilelemente
	erfordert: Szenarioelement erfordert eine anderes Element
	exkludiert: Element schließt ein anderes Element aus

Bild 4-24: Syntax und Semantik von Szenarioprofilen

Szenario–Sequenz: Das Diagramm ist Teil der Ablaufbeschreibung eines Szenarios. Der Aspekt beschreibt die Interaktion zwischen den aktiven Szenarioelementen in einem Szenario. Die Interaktion wird in Form von Nachrichten entlang einer Lebenslinie modelliert. Das Modell orientiert sich an der Notation der Sequenzdiagramm [GTS14, S. 118].

Szenario–Aktivität: Das Diagramm Die Szenario–Aktivität ist Teil der Ablaufbeschreibung. Es beschreibt den Szenarioablauf durch Aktivitäten. Die Aktivitäten sind durch Kontroll- und Objektflüsse miteinander verknüpft. Das Partialmodell ordnet die Aktivitäten in eine zeitliche Abfolge. Aktivitäten sind einem aktiven Szenarioelement zugewiesen bzw. werden durch dieses ausgeführt, z.B. Licht modulieren durch eine Leuchte. Modelliert werden ebenso Ereignisse, die auf eine Aktivität einwirken - diese zum Beispiel auslösen oder beenden oder eine Entscheidung herbeiführen. Das Partialmodell orientiert sich an der Notation der Aktivitätsdiagramme [GTS14, S. 118].

Sprachprofil zur Werkzeugunterstützung

Das nachfolgende Bild 4-25 zeigt die abstrakte Syntax als Sprachprofil zur Modellierung des Absicherungskonzeptes. Die Übersicht über die konkrete Syntax des Testfall Diagramms ist in Bild 4-23 dargestellt.

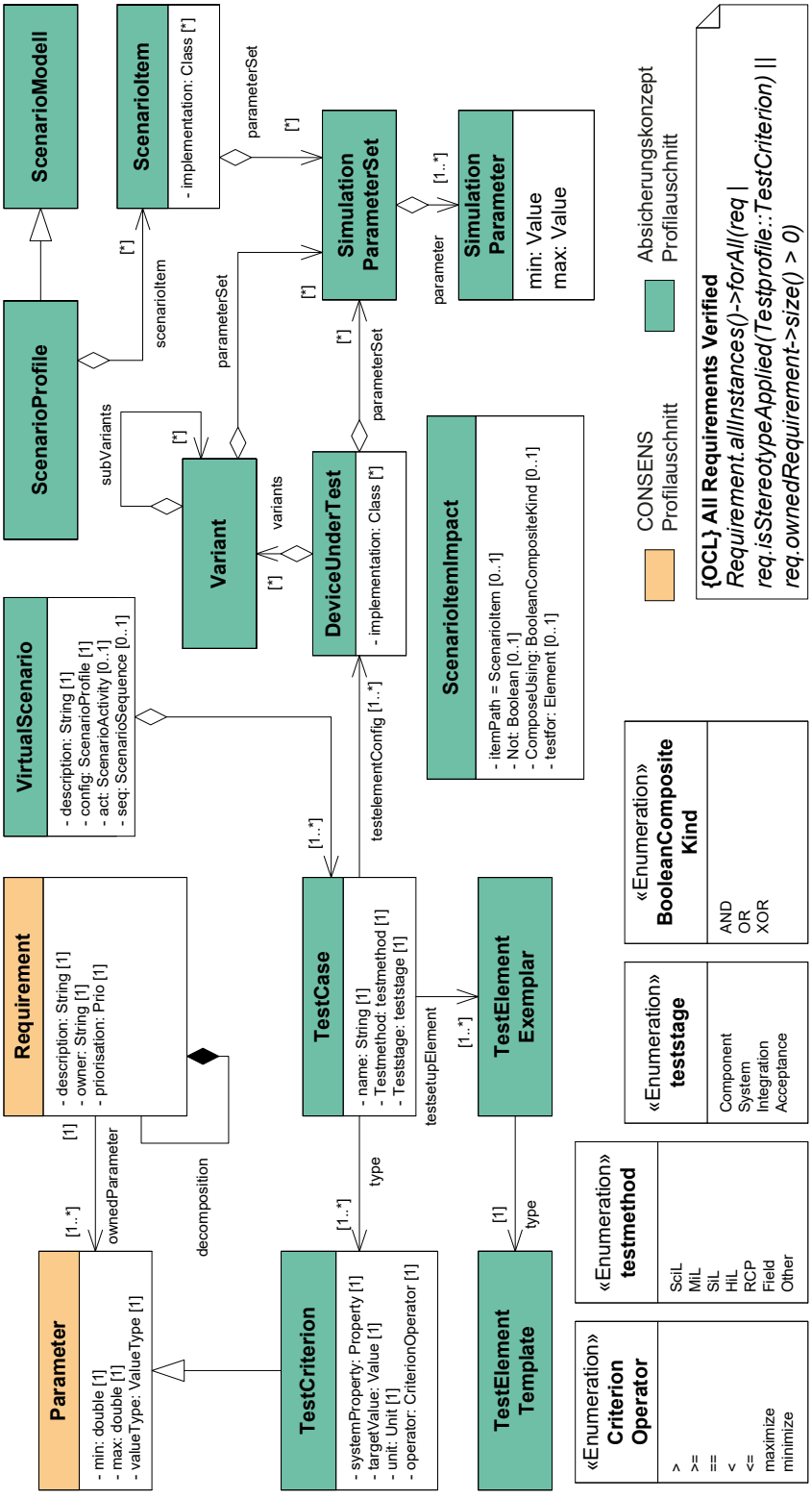


Bild 4-25: Sprachprofil zur Modellierung des Absicherungskonzeptes

Ein Testfall besitzt Attribute wie *Name, Methode* und *Teststufe* und verweist auf mindestens ein überprüfbares Kriterium bzw. Eigenschaft *«TestCriterion»*. Jedes Testkriterium enthält einen Verweis auf eine Eigenschaft des Systems (Attribute *«SystemProperty»*). Dem Testfall ist das Constraint *All Requirements Verified* zugeordnet. Dieses Constraint stellt sicher, dass zu jeder Anforderung mindestens ein Testfall existiert. Die Zuweisung von Anforderung und Testfall geschieht über eine *«verifies»*-Beziehung.

Die Testbedstruktur wird mit Hilfe der Stereotypen *«TestelementTemplate»* und *«TestelementExemplare»* spezifiziert. Die Zuordnung erfolgt über Assoziationen. Des Weiteren zeigt ein Testfall auf eine Menge von Testobjekten *«DeviceUnderTest»*. Die Verbindung zwischen Testobjekt und dem repräsentierten Element des Systemkonzeptes wird durch das Attribut *«implementation»* festgehalten. Die Parametrierung eines Testobjektes für einen konkreten Testfall wird durch *«SimulationParameter»* festgelegt und durch *«SimulationParameterSet»* strukturiert. Um alternative Konfigurationen oder Varianten abzubilden, stellt das Testprofil den Stereotypen *«Variant»* bereit.

Ein Szenario *«Virtual Scenario»* kann mehrere Testfälle kombinieren. Dies trifft zu, wenn durch das Szenario mehrere Testfälle untersucht werden. Ein Testfall zeigt daher über eine *«trace»*-Beziehung auf mindestens ein Szenario. Mit dem Attribut *«config»* erfolgt eine Zuweisung zum Szenarioprofil *«ScenarioProfile»*.

Die Umsetzung im Cameo Systems Modeller zeigt Kap. A2.4.

4.3.3 Bündelung von Testfällen

Entsprechend des Absicherungskonzeptes ist jede Anforderung mit mindestens einem Testfall verknüpft. Die Überprüfung der Vollständigkeit kann über eine **Domain Mapping Matrix (DMM)** erfolgen [LMB09, S. 54]. Für die Absicherung des Systems ist zu ermitteln, welche Testfälle in einem Virtuellen Testbed gebündelt werden. Als Grundlage für die Entscheidungsfindung dient die **Design Structure Matrix (DSM)**. In dieser sind die Testfälle in Zeilen und Spalten aufgetragen. Analysiert wird die Kombinierbarkeit von Testfällen. Beispielsweise werden Testfälle zur Absicherung der Lokalisierung zusammengefasst. Die Bündelung der Testfälle erfolgt über eine **Cluster-Analyse**. Das Bild 4-26 zeigt drei Cluster. Eine Cluster-Analyse bündelt Testfälle zu homogenen Gruppen, die eine hohe Abhängigkeit zueinander und nur eine begrenzte Anzahl an Abhängigkeiten zu den anderen Testfällen aufweisen [LMB09, S. 227]. Entsprechend der Cluster können anschließend die Virtuellen Szenarien umgesetzt werden.

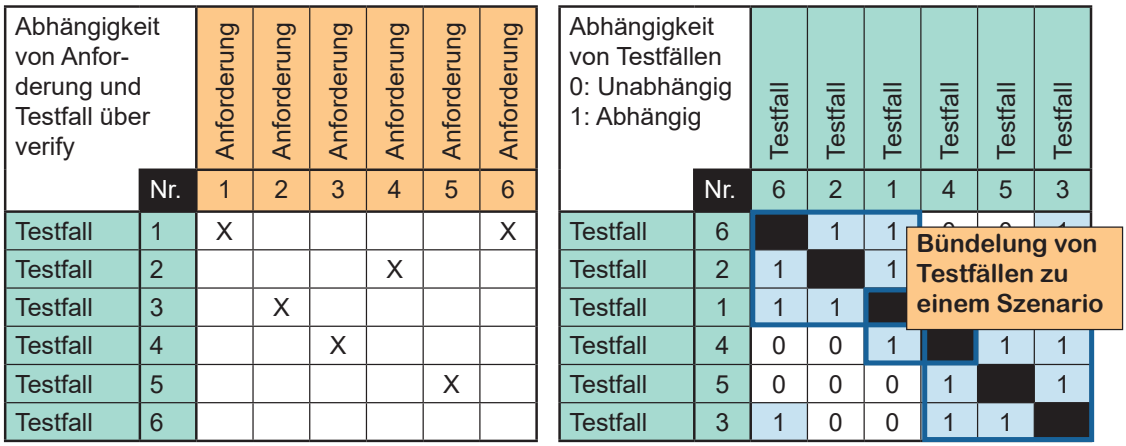


Bild 4-26: Bündelung homogener Testfälle zu einem Szenario

4.4 Potenzialanalyse

Die Selbstheilung ist für missionskritische Funktionen des Systems vorgesehen. Grundlage bildet eine systemische Risikoanalyse im Systementwurf. Die Funktionale Risikoanalyse besteht aus zwei Phasen: Der **Analyse von Potenzialen für Selbstheilung** und der **systemischen Risikoanalyse**.

4.4.1 Analyse von Potenzialen

Die Potenzialanalyse dient der e der Leistungsstufe eines selbstheilenden Systems. Die Grundlage bildet das Potenzialportfolio (vgl. Bild 4-27). Jedes Systemelement ist in drei Dimensionen bewertet:

- **Vernetzungsgrad:** Der Vernetzungsgrad ist eine Kennzahl über die Vernetzung des Systemelements im System. Die Kennzahl bildet sich aus der Aktiv- und Passivsumme der Flussbeziehungen eines Systemelements. Der Vernetzungsgrad ist ein Maß für die Komplexität und die Wichtigkeit. Ein hoher Vernetzungsgrad deutet auf eine starke Vernetzung des Systemelements im System. Ein Ausfall des Systemelements hat eine hohe Auswirkung auf verbundene Systemelemente.

- **Wissensgrad:** Der Wissensgrad ist eine Kennzahl über die Antizipationsfähigkeit von Betriebsszenarien zur Entwicklungszeit. Die Kennzahl bildet sich als Ergebnis einer Anwendungsanalyse. Eine geringer Wissensgrad bedeutet, dass es eine Vielzahl unbekannter Betriebsszenarien und unsichere Betriebszustände geben kann. Das System muss daher lernfähig und anpassungsfähig sein.

- **Integritätslevel:** Das Integritätslevel ist eine Kennzahl über das Sicherheits- und Ausfallrisiko des Systemelements im SuOC [ISO26262]. Die Kennzahl bildet sich aus dem Schadensausmaß, der Eintrittswahrscheinlichkeit, der Begrenzung und der Häufigkeit einer Gefährdung.

Anhand dieser Bewertungsdimensionen lässt sie Leistungsstufe eines selbstheilenden Systems festlegen (s. Kap. 4.2.1.2).

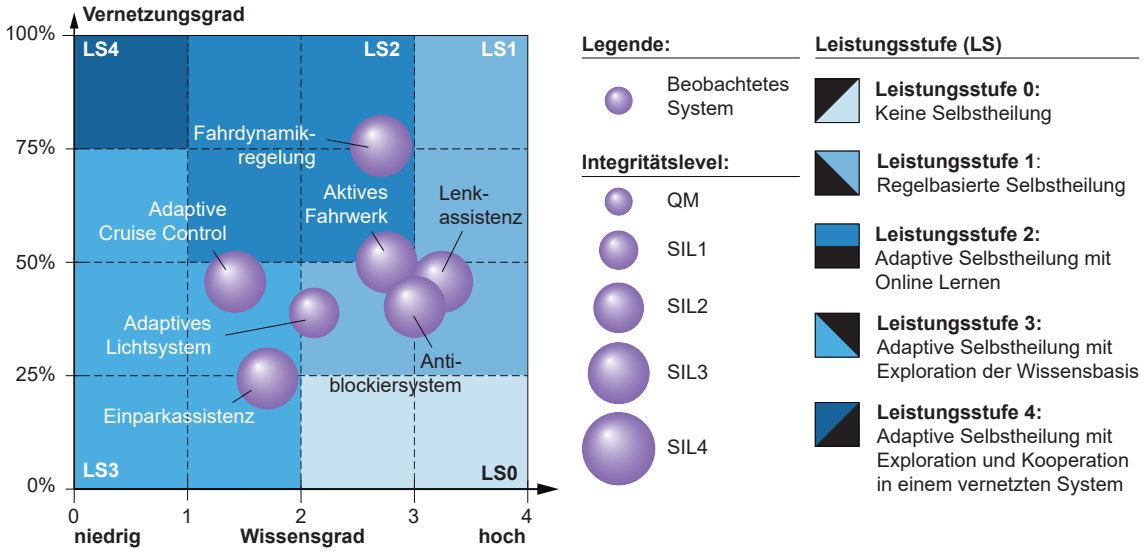


Bild 4-27: Potenzialportfolio zur Bewertung der Leistungsstufe

Vernetzungsgrad

Der Vernetzungsgrad basiert auf der *Design Structure Matrix* (DSM). Die DSM ist eine quadratische binäre Matrix. In der Matrix werden die Verbindungen zwischen zwei Systemelementen durch Stoff-, Material- und Informationsfluss oder logischen Beziehungen (vgl. Bild 4-28) erfasst [LMB09, S. 50]. Die DSM lässt sich aus dem Partialmodell Wirkstruktur überführen.

Anhand der DSM werden verschiedene Kennzahlen ermittelt: Die Aktivsumme, die Passivsumme, die Kritikalität und Aktivität sowie der Vernetzungsgrad. Die **Aktivsumme** ist die Anzahl der ausgehenden Flüssen von einem Systemelement. Sie ist eine Kennzahl über den Einfluss eines Systemelements. Die **Passivsumme** ist die Anzahl der eingehenden Flüsse. Sie ist eine Kennzahl über die Beeinflussung eines Systemelements. Die **Kritikalität** ergibt sich aus der Multiplikation von Aktiv- und Passivsumme. Es ist eine Kennzahl der Sensitivität. Die **Aktivität** ist die Division von Aktiv- und Passivsumme [LMB09, S. 202]. Der **Vernetzungsgrad** ist eine Kennzahl der Vernetzung eines Systemelements im System. Er gibt sich aus der Summe von Aktiv- und Passivsumme im Verhältnis zur maximalen Vernetzung entsprechend der Gleichung 4-7.

$$\text{Vernetzungsgrad}_{SE_i} = \frac{\text{Aktivsumme}_{SE_i} + \text{Passivsumme}_{SE_i}}{2(n-1)}$$

Gleichung 4-9: Vernetzungsgrad von einem Systemelement (SE)

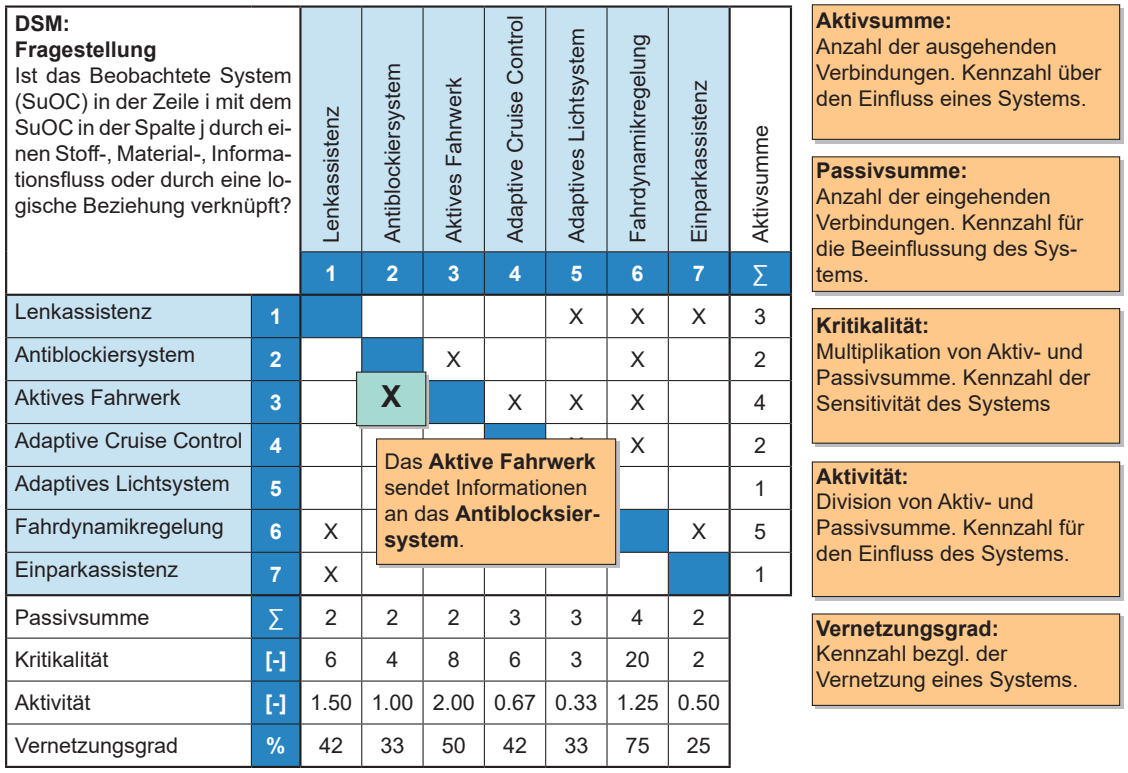


Bild 4-28: Design Structure Matrix am Beispiel der Fahrzeugfunktionen

Wissensgrad

Ziel der Anwendungsanalyse ist die Bewertung des Wissensgrades in Bezug auf ein Systemelement. Der Wissensgrad repräsentiert, wie gut Betriebsszenarien zur Entwicklungszeit durch das Entwicklungsteam antizipiert werden können. Die Bewertung erfolgt durch eine **Scoring-Modell**. Der Vorteil liegt in einer frühzeitigen Bewertung. Die Anwendungsanalyse umfasst eine Menge an Kriterien k mit $k = (1, .., K)$ und eine Menge an logischen Systemelementen SE_i mit $i = (1, ..., n)$. Jedes Kriterium besitzt eine Gewichtung w_k. Die Gewichtung ist zusammen mit den Domänenexperten festzulegen. Der Wissensgrad ergibt sich aus dem Summenprodukt von Gewichtung und Bewertung $SE_{i,k}$ (vgl. Gleichung 4-8). Das Bild 4-29 zeigt die Anwendungsanalyse an einem Beispiel.

Anwendungsanalyse: 1. Definieren Sie das Beobachtete System (SuOC) 2. Legen Sie die Gewichtung der Kriterien fest. 3. Bewerten Sie das SuOC an den Kriterien 4. Ermitteln Sie den Wissensgrad (WG) 5. Diskutieren Sie den Wissensgrad 6. Legen Sie Maßnahmen zur Reduktion fest. Bewertungskriterien:		**Gewicht** %	Lenkassistenz 1	Antiblockiersystem 2	Aktives Fahrwerk 3	Control
Quantifizierbare Betriebseinflüsse	1	10	3	3	2	
Vollständige Anwendungsszenarien	2	10	2	3	3	2
Kommunikation und Vernetzung	3	25	4	3	3	1
Selbstregulierendes Verhalten	4	20				
Technologischer Reifegrad (TRL)	5	15				
Antizipation des Ausfallverhaltens	6	10				
Hinreichende Absicherung	7	10				
Wissensgrad (WG):		**100%**	**3.30**	**3.00**	**2.70**	**1**

Bewertung
0: Unbekannt
1: Überwiegend Unbekannt
2: Zum Teil Bekannt
3: Überwiegend Bekannt
4: Bekannt

Bild 4-29: Anwendungsanalyse am Beispiel der Fahrerassistenz

Die Analyse nutzt sieben Bewertungskriterien. Die Bewertungsbereich liegt zwischen **Bekannt** (Bewertung = 4) und **Unbekannt** (Bewertung = 0). Die Kriterien basieren auf einer Literaturrecherche zum Thema *Betriebskonzept/ ConOps*[2] [IEEE1362], *Operational Design Domain* [KF19], [NHSTA16] und *Maschinensicherheit* [Lev04], [ISO12100] und *Autonome Systeme* [DGS+18], [FAS17], [RKG+18].

$$\text{Wissensgrad}_{SE_i} = \sum_{k=1}^{K} w_k \cdot SE_{i,k};\ \text{wobei} \sum_{k=1}^{K} w_k = 1 \text{ und } w_k > 0\ (k=1,...,K)$$

Gleichung 4-10: Wissensgrad (WG)

Systemische Risikoanalyse

Die Risikoanalyse basiert auf der System-Theoretic Process Analysis (STPA) nach LEVESON [Lev04, S. 179ff.] (vgl. Kap. 3.3.2.4). Die STPA basiert auf sogenannten Sicherheitskontrollstrukturen in einem System. Die Methode analysiert sicherheitskritische Regelkreise und identifiziert unsichere Kontrollstrukturen [SH20, S. 26]. Das Bild 4-30 zeigt das entwickelte Modell, ordnet die Begriffe ein und verdeutlicht die Zusammenhänge der Risikoanalyse. Ausgangspunkt bildet ein **Unfall**. Dies ist ein schädigendes Ereignis für System, Mensch oder Umwelt (z.B. *Kollision des Fahrzeugs mit einem Objekt*). Das Ereignis tritt als Konsequenz einer Gefährdung und der Interaktion zwischen dem System und seiner Umgebung auf (z.B. *Blockade der Lenkung bei* $v = 50\,km/h$). Eine **Gefährdung**

[2] Concept of Operations

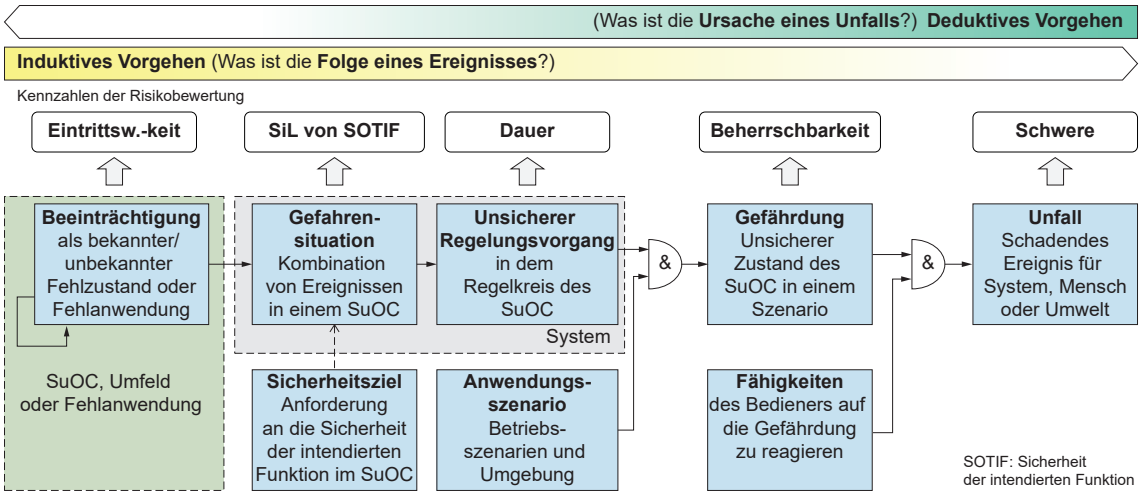


Bild 4-30: Modell und Zusammenhänge der Risikoanalyse

repräsentiert einen unsicheren Zustand des SuOC in einem Anwendungsszenario. Eine Gefährdung ergibt sich aus der Kombination von einem **unsicheren Regelungsvorgang** (z.B. *Keine Unterstützung bei einem Lenkmanöver*) in einem Anwendungsszenario (z.B. *Überholmanöver in der Stadt*). Eine Regelungsvorgang beeinflusst gezielt das Systemverhalten in einem Regelkreis. Dies kann zum Beispiel die Veränderung einer Stellgröße oder ein Schaltvorgang sein. Die STPA unterscheidet vier Arten von unsicheren Regelungsvorgängen:

- Eine Regelungsvorgang ist erforderlich, wird aber nicht ausgeführt
- Eine Regelungsvorgang wird ausgeführt, ist aber nicht erforderlich
- Eine Regelungsvorgang wird zu früh bzw. zu spät ausgeführt
- Eine Regelungsvorgang wird zu kurz bzw. zu lang ausgeführt

Ein **Gefahrensituation** bildet das Wurzelelement der dynamischen Fehlerbaumanalyse (z.B. *Keine Unterstützung aufgrund einer unbeabsichtigten Blockade*). Es repräsentiert eine Kombination von Beeinträchtigungen in einem SuOC die zu einem unsicheren Regelungsvorgang im Regelkreis führen können. Für die Gefahrensituation sind Sicherheitsanforderungen zu formulieren. **Beeinträchtigungen** ergeben sich aus dem SuOC (z.B. *Defekter Sensor*), dem Umfeld (z.B. *Schlupf bei glatter Fahrbahn*) oder einer Fehlanwendung (z.B. *Abschaltung durch Benutzer*). Beeinträchtigungen sind mit ihrer Eintrittswahrscheinlichkeit zu bewerten. Die Risikoanalyse kann deduktiv oder induktiv erfolgen.

Das Bild 4-31 zeigt den Ablauf der Risikoanalyse. Das Vorgehen umfasst vier Phasen mit definierten Aufgaben und konkreten Resultaten. Der Ablauf erfolgt iterativ und inkrementell. Die Analyse ist beendet, wenn alle Sicherheitskontrollstrukturen im SuOC untersucht sind. Zunächst werden Anwendungsszenarien und mögliche Fehlanwendungen identifiziert. Aus diesen werden Unfälle und Gefahrensituationen abgeleitet und eine Risikobewertung nach Schwere und Beherrschbarkeit durchgeführt. Ergebnis dieser ers-

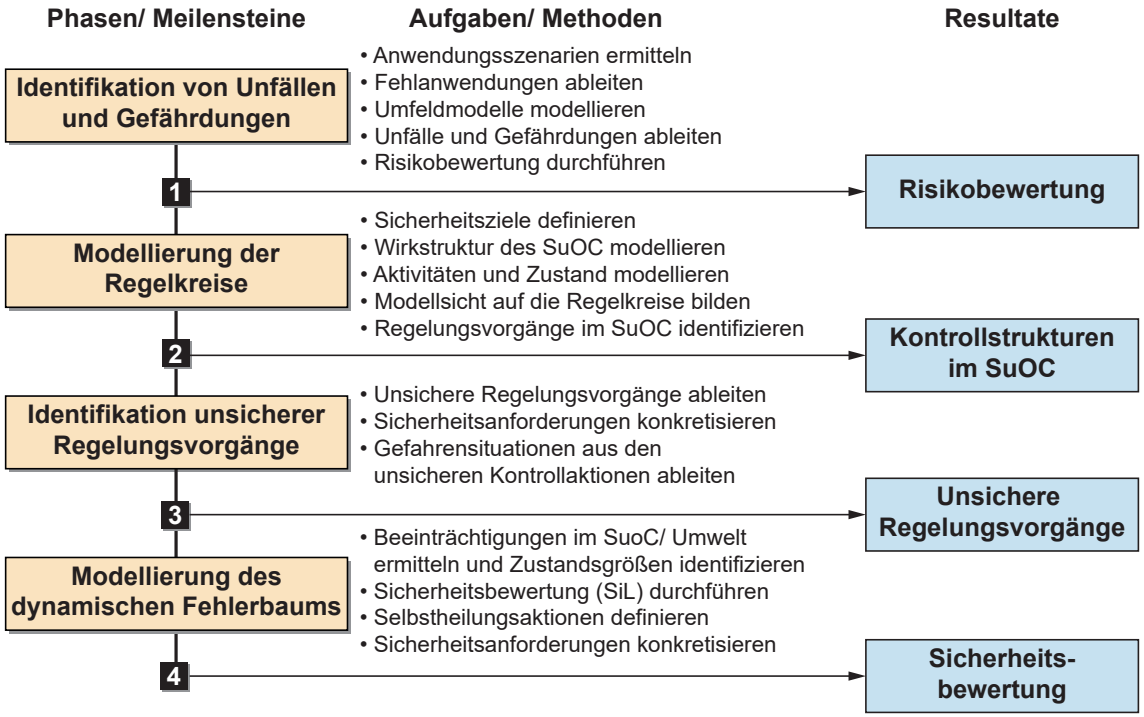


Bild 4-31: Vorgehen der Risikoanalyse

ten Phase sind bewertete Risiken. Im zweiten Schritt sind die Regelkreise im System zu modellieren. Diese bilden die konkrete Sicht auf die Kontrollstruktur des SuOC. Dabei werden die Regelungsvorgänge im System identifiziert. Ergebnis dieses Schritts sind die Regelkreise im System. Mit diesem Ergebnis werden dann unsichere Regelungsvorgänge identifiziert und mit den Risiken verknüpft. Dabei werden die Gefahrensituationen ermittelt, die zu einem unsicheren Regelungsvorgang in einem Regelkreis führen und auf eine mangelnde Sicherheitskontrollstruktur hindeuten. Ergebnis dieser Phase ist eine Liste unsicherer Regelungsvorgänge und Gefahrensituationen. Im letzten Schritt werden Beeinträchtigungen identifiziert und weitere Anforderungen an die Sicherheit der intendierten Funktion formuliert. Ebenso werden die Zustandsgrößen für die Überwachung und mögliche Selbstheilungsaktionen ermittelt. Das Ergebnis ist ein Sicherheitsintegritätslevel (SIL) des SuOC.

Sprachprofil zur Werkzeugunterstützung

In der vorliegenden Arbeit wird die STPA um eine qualitative Risikobewertung nach der IEC 61508 (Anhang Kap. A2.3.1) [IEC61508a] erweitert und in das CONSENS Sprachprofil integriert. Dies ermöglicht eine integrative Anwendung im Systementwurf. Das nachfolgende Bild 4-32 zeigt die abstrakte Syntax als Sprachprofil zur integrativen Modellierung von Risiken. Ein Fehlanwendungsfall *«Misuse»* besitzt die Attribute *description* sowie *recognition «ReType», judgment «JuType»* oder *action «AcType»* zur Beschreibung der Art einer Fehlanwendung durch einen Akteur (vgl. Bild 4-19). Der Fehlanwendungsfall ist eine Spezialisierung vom Anwendungsszenarion *«ApplicationScenario»* und ist mit einem Anwendungsszenario verknüpft.

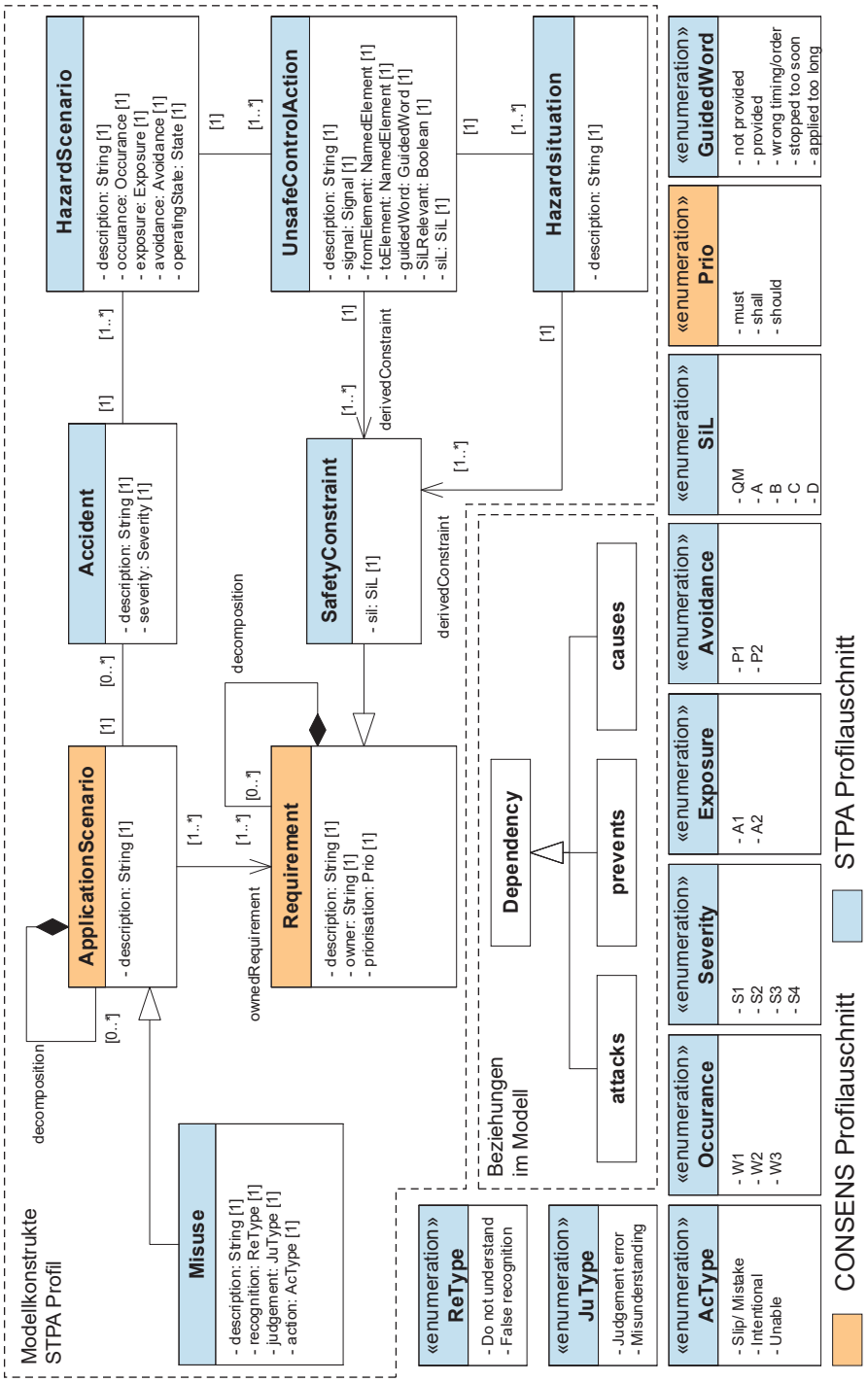


Bild 4-32: Erweitertes Profil der modellbasierten Risikoanalyse

Ein Unfall *«Accident»* besitzt die Attribute *description* und *severity* und ist ebenfalls mit dem Anwendungsszenario verknüpft. Die Verknüpfung der Modellelemente unterscheidet die Abhängigkeiten *«attacks», «prevents», «causes»*. Gefährdungsszenario *«Hazard-Scenario»*, unsicherer Regelungsvorgang *«UnsafeControlAction»* und Gefahrensituation *«HazardSituation»* sind über die Assoziation verknüpft. Ein unsicherer Regelungsvorgang beinhaltet weitere Attribute wie etwa das Leitwort *«Guided Word»* und zeigt auf ein Signal sowie auf die beiden Systemelemente zwischen denen die Regelungsvorgänge stattfindet. Aus dem unsicheren Regelungsvorgang leitet sich mindestens eine Sicherheitsanforderungen *«SafetyConstraints»* ab. Dies stellt eine Spezialisierung der Anforderung *«Requirement»* dar.

Die Schwere *«Severity»*, die Eintrittswahrscheinlichkeit *«Occurance»*, die Häufigkeit *«Exposure»* sowie die Beherrschbarkeit *«Avoidance»* sind als Enumeration umgesetzt. Über die Verknüpfung im Profil ergibt sich das SIL. Die Bewertung folgt dem Risikographen in Kapitel A2.3.1 [ISO26262]. Das Sprachprofil zeigt Anhang A2.4.

4.4.2 Erweiterte Fehlerbaumanalyse

Für jede Gefahrensituation wird ein erweiterter dynamischer Fehlerbaum modelliert. Die Modellierung erfolgt deduktiv von der Gefahrensituation zur Beeinträchtigung. Der Baum beschreibt Beeinträchtigungen aus der Hardware, Software oder der Umgebung des SuOC. Über logische und dynamische Gatter erfolgt die Verknüpfung dieser Ereignisse.

Die Modellierung des dynamischen Fehlerbaums erweitert die Notation nach IWANEK um dynamische Gatter sowie um weitere Modellelemente. Das Konzept dynamischer Gatter wurde von DUGAN ET AL. eingeführt [DBB92b]. Diese Gatter bilden Reihenfolgen von Ereignissen (PAND), Redundanzen (SPARE) sowie funktionale Abhängigkeiten (FDEP) in einem Fehlerbaum ab. Das Bild 4-33 zeigt das Vorgehen an einem Anwendungsbeispiel. Zunächst wird eine Gefahrensituation ausgewählt. Anschließend werden die Beeinträchtigungen deduktiv über logische und dynamische Gatter verknüpft. Jeder Beeinträchtigung wird das betreffende Systemelement aus dem SuOC sowie eine Zustandsgröße zur Überwachung zugewiesen. Die Zustandsgröße dient der Entwicklung von Signalgeneratoren für den Gesundheitsmonitor. Zusammen mit den Domänenexperten werden Selbstheilungsaktionen geplant und mit der betreffenden Beeinträchtigung verknüpft.

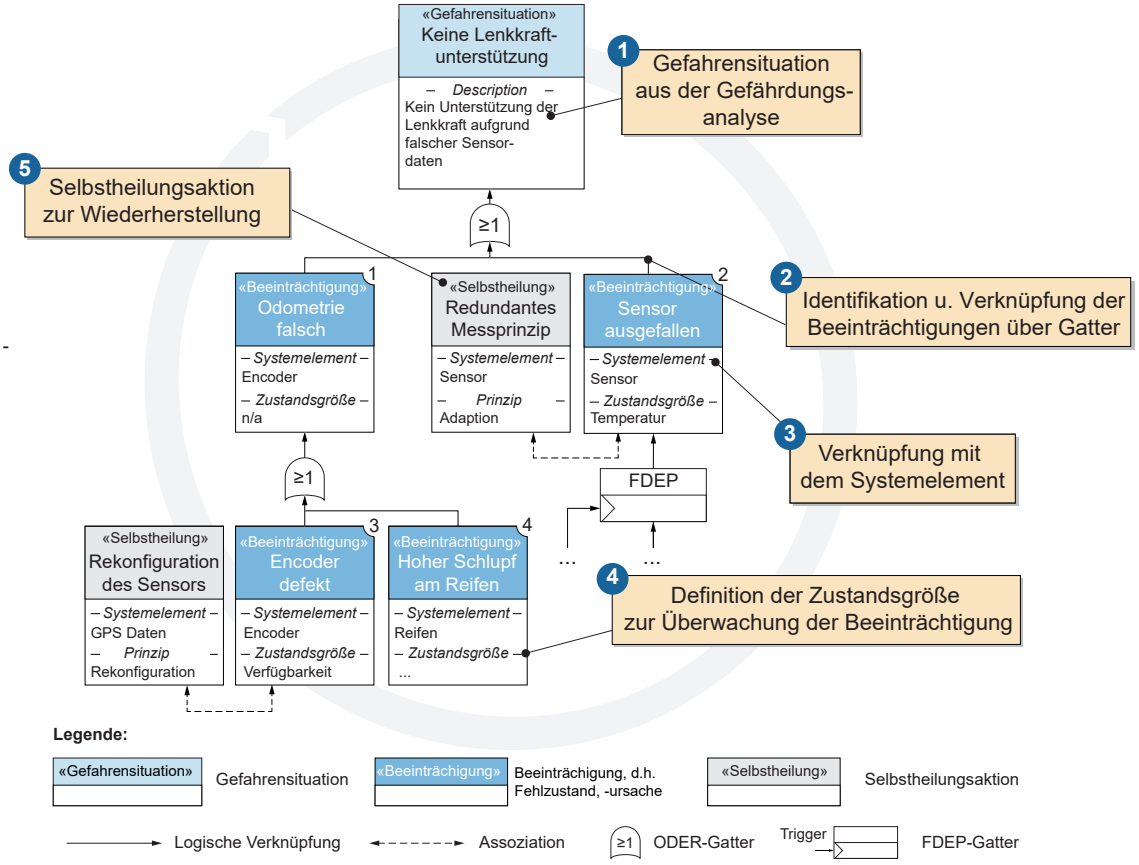


Bild 4-33: Vorgehen zur Modellierung erweiterter dynamischer Fehlerbäume

4.5 Werkzeugkonzept

In diesem Kapitel wird ein Virtuelles Testbed entwickelt. Es dient als Validierungsumgebung für die Analyse und Bewertung der Eigenschaften selbstheilender Systeme. Hierzu wird das *Scenario-in-the-Loop*-Testing eingeführt und die Anforderungen zur Umsetzung in einem Virtuelles Testbed formuliert. Darauf aufbauend wird das Framework des Virtuellen Testbeds erläutert. Das Kapitel schließt mit einem Kennzahlensystem zur Bewertung der der Selbstheilungseigenschaften.

4.5.1 Scenario-in-the-Loop-Testing

Zur Absicherung der Eigenschaften wird das *Scenario-in-the-Loop*-Testing in dieser Arbeit eingeführt. Beim *Scenario-in-the-Loop* werden wahrscheinliche Szenarien definiert, modelliert und durch automatisierte Tests simuliert. Szenarien sind Nachbildungen möglicher Situationen und Ereignisse, die in der realen Welt auftreten. Zudem bilden sie typischerweise externe Einflüssen und Interaktionen mit des Systems mit seiner Umgebung ab. Ziel dieses Verfahrens ist es, möglichst viele Betriebsszenarien virtuell abzusichern und das Verhalten des Systems virtuell zu untersuchen. Die Absicherungsstrategie orientiert sich an WANG ET AL. und ist in Bild 4-34 visualisiert.

Der Testraum ist durch virtuelle Szenarien geprägt. Die Definition eines Szenarios orientiert

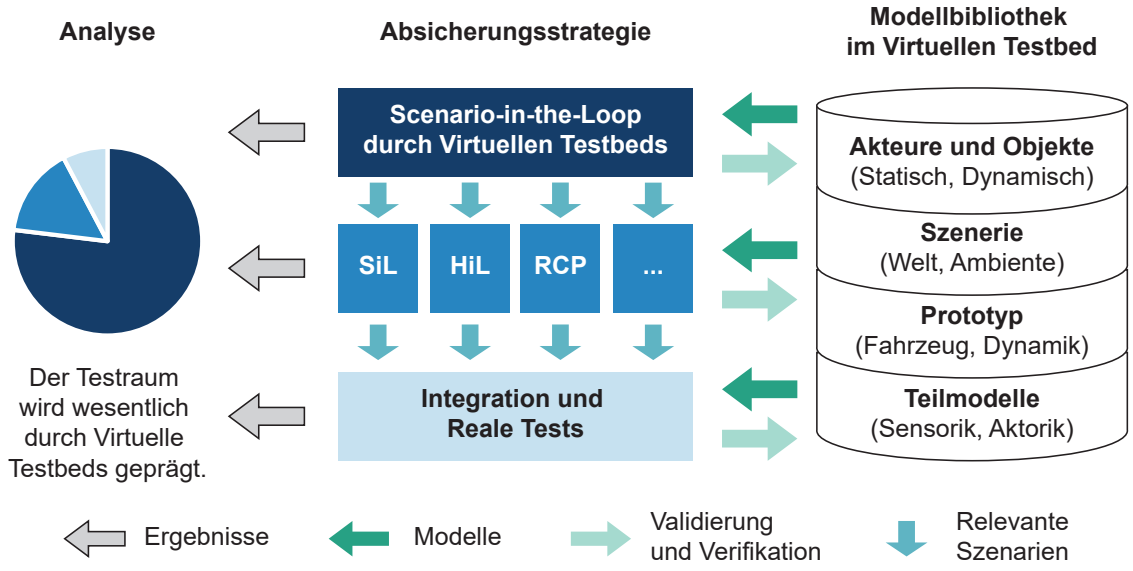


Bild 4-34: Strategie der Absicherung, in Anlehnung an [WFJ+17, S. 6]

sich an ULBRICH ET AL. [UMR+15, S. 986]. Den Kern dieser Strategie bilden Virtuelle Testbeds und Modellbibliotheken. Eine Modellbibliothek umfasst die relevanten und validierte Teilmodelle der Akteure und Objekte, der Szenerie sowie des Systems. Eine konsistente Kombination dieser Teilmodelle wird durch das Szenarioprofil ermöglicht (vgl. Kapitel 4.3.2). Die Absicherung erfolgt auf verschiedenen Stufen. Die Simulation im Virtuellen Testbed durch *Scenario-in-the-Loop* ermöglicht eine frühzeitige Analyse und Simulation sowie eine Automatisierung von Testfällen. Mit fortschreitender Modelltiefe bilden XiL-Techniken die nächste Stufe. Die Feldtests werden dann für ausgewählte Szenarien durchgeführt [WFJ+17, S. 6]. Die Inbetriebnahme des Systems erfolgt somit schrittweise. Mit jedem Schritt werden synthetische und reale Daten zum Trainieren und Testen, etwa für die Anomalie Detektion erzeugt.

4.5.2 Anforderungen an das virtuelle Testbed

Das Virtuelle Testbed führt die Analysetechniken und Domänenmodelle zu einer leistungsfähigen Validierungsumgebung zusammen. An das Virtuelle Testbed werden entsprechend dem *Scenario-in-the-Loop* verschiedene Anforderungen gestellt:

- **Einfache Auswertung:** Unterstützung von *Online* und *Offline* Analysen durch das Virtuelle Testbed. Online bedeutet, dass die Verfahren der Selbstheilung während der Simulation ausgewertet werden. Für die offline Analyse muss das Virtuelle Testbed Daten bereitstellen, z.B. Szenario Replay.
- **Automatisierte Datenannotation:** Das Virtuelle Testbed muss eine einfache Annotation der Trainings- und Testdaten ermöglichen.
- **Einfache Interaktion:** Das Virtuelle Testbed muss benutzerfreundlich sein, d.h. einfach zu bedienen und verständlich.

- **Integrierte Analysetechniken:** Das Virtuelle Testbed muss eine Validierungsumgebung bereitstellen, in denen Analysetechniken sicher, reproduzierbar und automatisiert durchgeführt werden können.
- **Realistische Visualisierung:** Das Virtuelle Testbed muss eine leistungsfähige 3D-Visualisierung und Multiphysik bereitstellen.
- **Multidomänen Simulation:** Kopplung von Teilmodellen zu einer Systemsimulation. Simulation von diskreten und dynamischen Verhalten des Systems, sowie der Fähigkeiten autonomer Systeme wie Sensieren, Planen, Agieren.
- **Virtuelle Inbetriebnahme:** Unterstützung der kontinuierlichen Integration. Kopplung von Modellen mit unterschiedlicher Modellierungstiefen. Unterstützung von Virtuellen Szenarien, XiL-Techniken und Inbetriebnahme.
- **Verteilte Simulation:** Unterstützung einer verteilten Umgebung. Das Virtuelle Testbed kann aus einer Vielzahl an Knoten bestehen, die auf mehreren Maschinen verteilt sind und untereinander Nachrichten austauschen.
- **Einfache Erweiterung:** Einfache Anbindung weiterer Entwicklungswerkzeuge wie zum Beispiel Matlab/ Simulink oder Datenbanken zur Durchführung von Co-Simulationen und zum Datenaustausch.

4.5.3 Framework des Virtuellen Testbeds

Entsprechend der Anforderungen an das Virtuelle Testbed ist ein Framework auf Basis des Robot Operating Systems (ROS) und dessen Ökosystem umgesetzt. ROS ist eine Open Source Anwendung (BSD Lizenz) zur Entwicklung von Anwendungen in der Robotik. Das ROS wird von der Open Source Robotics Foundation betreut und von einer internationalen Community unterstützt. ROS ist mehrsprachig und unterstützt C++ und Python [OSRF19-olb], [QGS15]. Das entwickelte Framework des Virtuellen Testbeds zeigt Bild 4-35. Das Framework dient der frühzeitigen Absicherung der intendierten Funktion und der Selbstheilungseigenschaften. Grundlage des Frameworks bilden die **Partialmodelle** der Entwurfstechnik (vgl. Kapitel 4.3). Auf Basis der Wirkstruktur und der Verhaltensmodelle wird der Virtuelle Prototyp entwickelt. Die Struktur des Virtuellen Prototypen entspricht der Wirkstruktur. Die Testfälle aus dem Absicherungskonzept sind Grundlage für den **Simulationsmanager**. Dieser führt die Testfälle auf der Plattform automatisiert aus. In einem Szenario werden Testfälle als Cluster untersucht (vgl. Kapitel 4.3.3).

Die selbstregulierende Informationsverarbeitung wird in ROS umgesetzt und in das **Virtuelle Szenario** eingebunden. Das Streckenmodell umfasst u.a. die Kinematik und die Dynamik. Die Modelle werden über XARCO in das Szenario eingebunden. Die Modellierung des Szenarios erfolgt auf Basis des Szenariobaums und des Szenarioablaufs aus dem Absicherungskonzept. Die Simulation kombiniert den Virtuellen Prototypen und die Virtuelle Umgebung zu einem Virtuellen Szenario. Dabei werden u.a. Techniken der

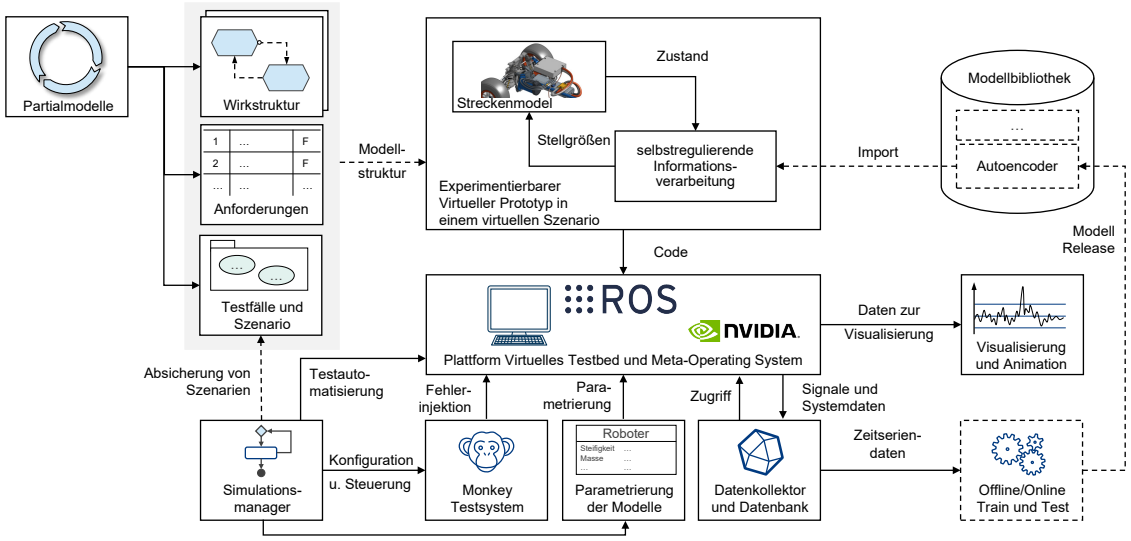


Bild 4-35: Framework des Virtuellen Testbeds in der Übersicht

3D-Visualisierung, der Multiphysik zur Simulation physikalischer Prozesse sowie virtuelle Sensoren und Aktoren in der Simulationsumgebung zusammengeführt. Die Modelle können über die **Parametrierung** eingestellt werden (z.B. Messdistanz und Auflösung des Laserscanners im Szenario). Die **Modellbibliothek** umfasst Detailmodelle zur Sensorik, Aktorik und der Informationsverarbeitung des Virtuellen Prototypens (z.B. Lokalisierung) sowie der Virtuellen Umgebung (z.B. Fußgänger und ihr Verhalten). Die Bibliothek ist erweiterbar. Die **Plattform** ermöglicht die Ausführung des Szenarios und realisiert die Kommunikations- und Datenschicht im Virtuellen Testbed. Über die **Visualisierung** erfolgt die Animation und graphische Darstellung der Signalverläufe.

Das vorgestellte Grobkonzept und dessen prototypische Implementierung stellt eine Validierungsumgebung zur Analyse von Eigenschaften der Selbstheilung bereit. Mit der modularen Architektur, kann die Umgebung erweitert werden. Zudem wird ein sukzessiver Übergang in die reale Anwendung unterstützt. In den folgenden Kapitel werden spezifische Module des Virtuellen Testbeds im Detail erläutert.

4.5.4 Qualitätsbewertung der Selbstheilung

Das Bild 4-35 visualisiert die Module zur Analyse und Bewertung der Eigenschaften selbstheilender Systeme. Basis bildet eine **Datenbank** in der die Signale und Systemdaten aus der Simulationsplattform geschrieben werden. Diese Zeitserien-Daten werden zum **Offline bzw. Online Training und Test** der Modelle verwendet. Die Modelle werden anhand eines Kennzahlensystems bewertet. So ergibt sich das **Qualitätsprofil**. In der Anforderungsanalyse sind die Zielwerte für diese Kennzahlen festzulegen und mit Hilfe des Virtuellen Testbeds zu überprüfen. Erfüllt ein Modell das vorgegebene Qualitätsprofil kann es über ein Release der Modellbibliothek hinzugefügt werden. In der nächsten Iteration wird es als Verfahren in die selbstregulierende Informationsverarbeitung des virtuellen Prototypen eingebunden.

Die Kennzahlen dienen der Bewertung und zur Auswahl von Verfahren. Die Kennzahlen werden in drei Klassen eingeteilt: Leistung, Zeit und Ressourcen (vgl. Bild 4-36). Ihre Kombination repräsentiert das Qualitätsprofil [BW14]. Das Qualitätsprofil dient zur Auswahl maschineller Lernverfahren. Die **Leistung** evaluiert die Güte der Verfahren. Die **Zeit** bewertet das zeitliche Verhalten eines Verfahrens. Die **Ressourcen** repräsentiert den Ressourcenbedarf eines Verfahrens. Entsprechend dieser Klassen werden verschiedene Kennzahlen verwendet. Das Kennzahlensystem kann entsprechend erweitert und angepasst werden.

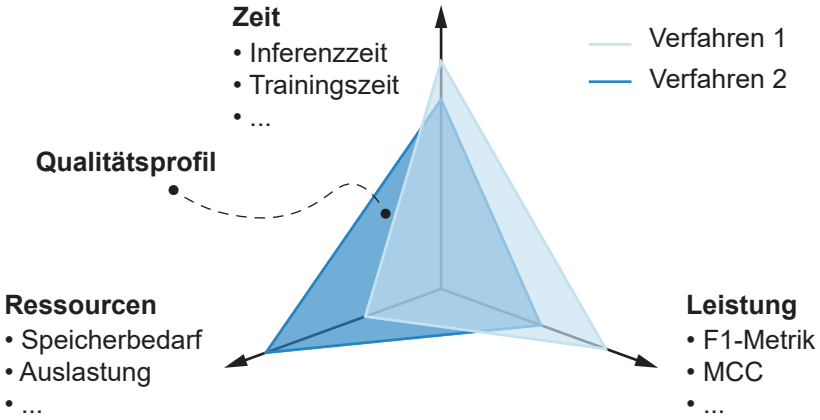


Bild 4-36: Leistung, Zeit und Ressourcen als Qualitätsprofil

Leistung: Die Leistung eines Verfahrens wird anhand verschiedener Kennzahlen bewertet. Zur Detektion von Anomalien und Diagnose von Ursachen werden Klassifikationsverfahren eingesetzt. Zur formalen Definition repräsentieren y und $\hat{y}$ den wahren Wert und den vorhergesagten Wert einer Ergebnisgröße. Die vier Werte der Konfusionsmatrix sind richtig positiv, richtig negativ, falsch positiv und falsch negativ, kurz $\widehat{tp}$, $\widehat{tn}$, $\widehat{fp}$, and $\widehat{fn}$ mit n als Anzahl der Werte in der Datensatz [Ger17], [KNR+15]:

$$\widehat{tp}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 1, \hat{y}_i = 1]\!], \widehat{tn}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 0, \hat{y}_i = 0]\!],$$
$$\widehat{fp}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 0, \hat{y}_i = 1]\!], \widehat{fn}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 1, \hat{y}_i = 0]\!],$$

Gleichung 4-11: Konfusionsmatrix

- **Trefferquote (Recall):** Anteil der als richtig klassifizierten Anomalien an der Gesamtheit der tatsächlichen Anomalien. [Pow11].

$$d_{RE}(y, \hat{y}) = \frac{\widehat{tp}}{\widehat{tp} + \widehat{fn}}$$

Gleichung 4-12: Recall

- **Genauigkeit (Precision):** Gibt den Anteil der korrekt als positiv klassifizierten Anomalien an der Gesamtheit der als positiv klassifizierten Anomalien an [Pow11].

$$d_{PR}(y, \hat{y}) = \frac{\widehat{tp}}{\widehat{tp} + \widehat{fp}}$$

Gleichung 4-13: Precision

- **F1-Maß:** Das F-Maß ist ein kombiniertes statistische Maß von Genauigkeit und Trefferquote (Formel 4-11) [Pow11], [KNR+15].

$$d_{F_1}(y, \hat{y}) = \frac{2 \cdot d_{PR}(y, \hat{y}) \cdot d_{RE}(y, \hat{y})}{d_{PR}(y, \hat{y}) + d_{RE}(y, \hat{y})} = \frac{2\widehat{tp}}{2\widehat{tp} + \widehat{fn} + \widehat{fp}}$$

Gleichung 4-14: F1-Metrik

- **Matthews-Korrelationskoeffizient (MCC):** Wird als Qualitätsmaß für binäre Klassifikatoren verwendet. Die Anwendung des Koeffizienten eignet sich für Klassen mit unterschiedlicher Größe. Der Koeffizient ist 0, wenn der Klassifikator eine zufällige Vorhersage liefert [Pow11], [BBC+00]. Der MCC ermittelt sich nach Formel 4-12.

$$d_{MCC}(y, \hat{y}) = \frac{\widehat{tp} \times \widehat{tn} - \widehat{fp} \times \widehat{fn}}{\sqrt{(\widehat{tp} + \widehat{fp})(\widehat{tp} + \widehat{fn})(\widehat{tn} + \widehat{fp})(\widehat{tn} + \widehat{fn})}}$$

Gleichung 4-15: Matthews-Korrelationskoeffizient

- **Konvergenz:** Die Konvergenz beschreibt, ob das Verfahren im Verlauf der Iterationen einem bestimmten Wert näher kommt und sich so eine dominante Selbstheilungsstrategie herausbildet.
- **Widerstandsfähigkeit:** Eine Kennzahl des Systems einer Beeinträchtigung dauerhaft widerstehen zu können. Sie gibt an, bei welcher Beeinträchtigung das System die Warn- und Eingriffsgrenzen überschreitet.

Zeit: Die zeitliche Bewertung von Verfahren erfolgt durch die folgende Kennzahlen:

- **Inferenzzeit:** Die Inferenzzeit repräsentiert die Schnelligkeit, mit der eine Klassifizierung oder ein Vorhersage zur Laufzeit erfolgt.
- **Trainingszeit:** Die Trainingszeit beschreibt die Trainingsdauer, die ein Verfahren zum Lernen auf den Trainingsdaten benötigt.
- **Mean-Time-To-Recovery:** Beschreibt die mittlere Zeitspanne zwischen Beeinträchtigung und Wiederherstellung einer intendierten Funktion.

Ressourcen: Die Bewertung des Ressourcenbedarfs eines Verfahrens erfolgt durch die beiden Kennzahlen:

- **Auslastung:** Kennzahl der Prozessorauslastung. Sie verdeutlicht die Rechenzeit, die ein Verfahren für die Inferenz zur Laufzeit benötigt.
- **Speicherallokation:** Kennzahl über den Speicherbedarf und die Platzkomplexität des gewählten Verfahrens.

Das Bild 4-37 zeigt ein Kennzahlensystem zur Bewertung der Eigenschaften von selbstheilenden Systemen. Die Kennzahlen ermöglichen eine Analyse der Verfahren und Optimierung der Architektur. Das Kennzahlensystem verdeutlicht das Optimierungspotenzial hinsichtlich der Leistung, Zeit oder Ressourcenbedarf. Ebenso wird ein quantitativer Vergleich der Verfahren miteinander im virtuellen Testbed ermöglicht. Darüber hinaus werden Trade-Off-Analysen unterstützt, beispielsweise die Abwägung zwischen Trefferquote und Inferenzzeit. Hierdurch werden Entwurfsentscheidungen fundiert.

Empfohlene Qualitätskennzahlen:		Module selbstheilender Systeme				
		Anomalie Detektor	Gesundheits-monitor	Selbst-diagnostik	Controller	O/C-Architektur
Leistung	Trefferquote	++	+	+	+	+
	Genauigkeit	++	+	+	+	o
	F1-Maß	++	o	++	+	o
	MCC	++	o	++	o	o
	Konvergenz	o	o	o	++	o
	Widerstands-fähigkeit	o	o	o	o	++
Zeit	Trainingszeit	+	+	+	+	o
	Inferenzzeit	++	++	++	++	o
	MTTR	o	o	o	o	++
Resource	Auslastung	++	++	++	++	o
	Allokation	++	++	++	++	o

Legende: o: Keine Empfehlung für oder wider | +: Empfehlenswert | ++: Sehr empfehlenswert

Bild 4-37: Kennzahlensystem zur Analyse und Bewertung selbstheilender Systeme

4.6 Analysetechnik zur virtuellen Absicherung

Die Analyse und Bewertung der Eigenschaften selbstheilender Systeme erfordert eine leistungsfähige Validierungsumgebung. Die Umgebung stellt Methoden zur Analyse und Auswertung bereit. In diesem Kapitel werden die Analyseverfahren entwickelt. Abschließend wird die Realisierung der Validierungsumgebung erläutert.

4.6.1 Suchfeldanalyse

Das Simian Army Testing (vgl. Kap. 3.4.2) ist ein geeigneter Ansatz zur automatisierten Fehlerinjektion. Die Anwendung des Verfahrens wird in dieser Arbeit auf selbstheilende Systeme übertragen. Es dient der Absicherung und zur Bewertung sowie zur Generierung von Daten für das maschinelle Lernen. Das Monkey Testing kann kontinuierlich im Entwicklungsprozess eingesetzt werden.

Für die Verfahrensentwicklung ist das **Suchfeld** festzulegen. Ein Suchfeld wird durch die Dimensionen Art und Ort der Beeinträchtigung aufgespannt. Der **Ort der Beeinträchtigung** ergibt sich aus dem Aufbau eines autonomen Systems, d.h. Sensorik, Aktorik, Selbstregulation, Kommunikation und Netzwerk, Daten und Informationen sowie Umgebung. Das Bild 2-9 in Kap. 2.3.2 zeigt eine Auswahl an Beeinträchtigungen. Diese werden anhand von Leitwörtern systematisiert:

- **Nicht:** Eine Funktion wird nicht ausgeführt, z.B. ein Service der Informationsverarbeitung ist nicht verfügbar oder die Sensorik bzw. Aktorik ist ausgefallen.
- **Unklar:** Eine Funktion wird nicht eindeutig ausgeführt, z.B. Rauschen eines Sensors aufgrund von Interferenz.
- **Mehr:** Eine Funktion wird zu viel, zu lang oder zu spät ausgeführt, z.B. Überlastung des Netzwerks oder aufgrund von zu verzerrten Daten.
- **Weniger:** Eine Funktion wird zu wenig, zu kurz oder zu früh ausgeführt, z.B. Leistungsabfall einer Funktion.
- **Falsch:** Eine Funktion wird falsch ausgeführt, z.B. Daten sind korrumpiert oder ein Objekt wird falsch klassifiziert.

Das Bild 4-38 zeigt das Suchfeld. Anhand des Suchfeldes werden geeignete Monkeys definiert. Sie imitieren und simulieren die Art und den Ort der Beeinträchtigung.

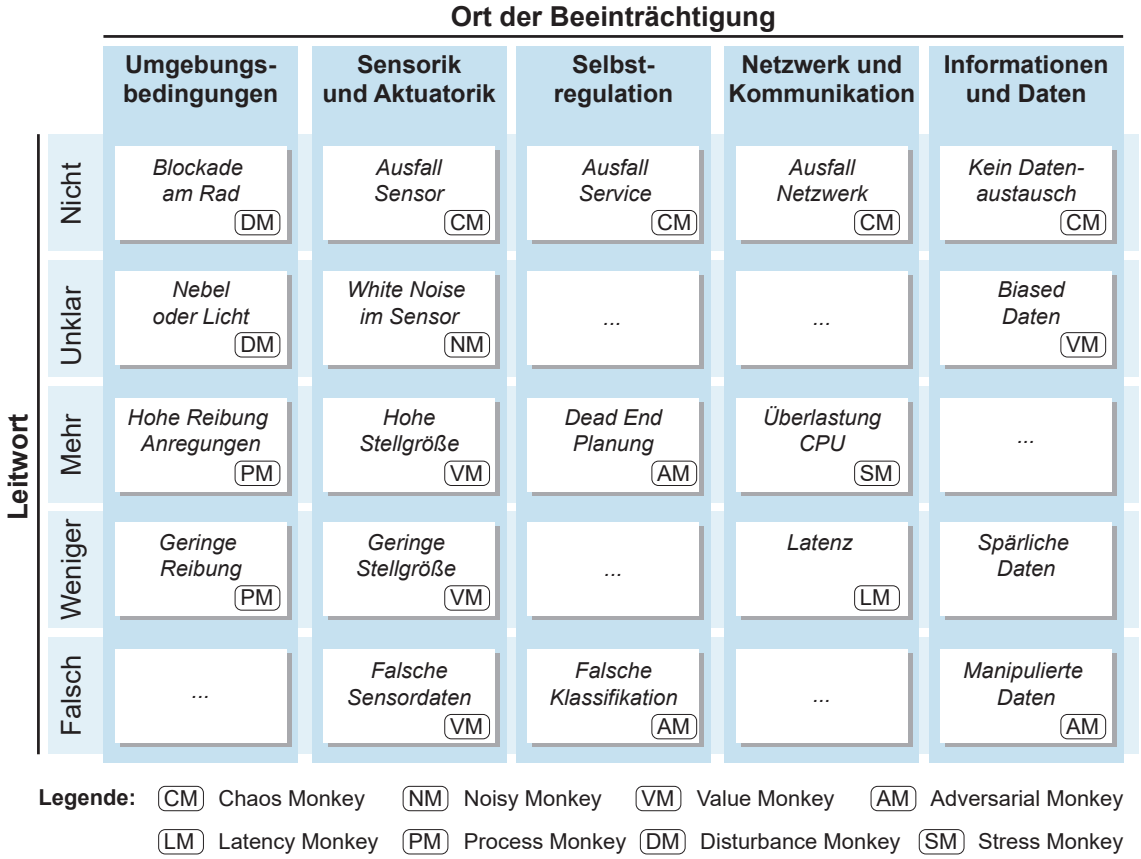


Bild 4-38: Exemplarisch Störgrößen auf Basis einer Suchfeldanalyse

4.6.2 Fortgeschrittenes Monkey Testing

Das Fortgeschrittene Monkey Testing ist ein agentenbasiertes und modulares Framework zur automatisierten Fehlerinjektion. Der Einsatz eignet sich als Erweiterung der XiL-Techniken und kann bereits im Systementwurf eingesetzt werden. Das Verfahren dient

zur Bewertung der Resilienz in einem selbstheilenden System. Ein Monkey besitzt eine spezifische Rollenbeschreibung und ein zugewiesenes Aufgabenprofil [TH18, S. 19ff.]. Die oben beschriebene Suchfeldanalyse dient zur Definition von Rollen und Aufgaben. Es werden folgende Monkeys vorgeschlagen:

- **Chaos Monkey:** Dieser Monkey simuliert komplette Ausfälle von Services oder Knoten im Netzwerk, wie z.B. durch einen Abriss einer Datenverbindung oder durch einen Ausfall eines Sensors, Prozessors oder einer Steuereinheit.
- **Noisy Monkey:** Addiert Rauschen auf das Nutzsignal eines Sensors. Zum Einsatz kommen verschiedene Rauschmodelle bei denen die Intensität variiert werden kann. Die Intensität wird über den Mittelwert und Standardabweichung parametriert. Ebenso kann die Dauer vorgegeben werden.
- **Value Monkey:** Dieser Monkey operiert auf den Sensorwerten und generiert inkorrekte oder zufällige Sensorwerte. Dabei können Muster wie zum Beispiel eine Abweichung, ein Drift, ein Trend oder das Einfrieren von Sensorwerten simuliert werden.
- **Adversarial Monkey:** Erzeugen gezielte Eingaben, um die Modelle der Selbstregulation zu täuschen. Sie ermöglichen sogenannte Blackbox-Angriffe auf das System. Adversarial Monkeys versuchen einen Service zu korrumpieren. Der Monkey dient zur Untersuchung von Robustheit der erlernten Modelle.
- **Latency Monkey:** Simuliert zeitliche Verzögerungen und Abbrüche der Nachrichten in der Kommunikation zwischen den Einheiten wie z.B. Sensor und Steuereinheit. Dies simuliert zum Beispiel Überlastungen im Kommunikationsnetzwerk oder Abbrüche in der Verbindung.
- **Process Monkey:** Simuliert Störungen in der Kinematik und Dynamik des Grundsystems und der Strecke. Dies erfolgt z.B. durch Veränderung der Dämpfungs- und Federwerten oder Gelenken oder Anregungen.
- **Disturbance Monkey:** Simuliert Umgebungsbedingungen mit negativen Einfluss auf das Systemverhalten. Zum Beispiel Variation der Helligkeit, Generierung von Nebel oder von Reibungskoeffizienten.
- **Stress Monkey:** Nimmt die Verarbeitungseinheiten unter Last. Simuliert werden ebenso hohe Zugriffsraten auf Verzeichnisse und Dateien oder die Auslastung des Arbeitsspeichers.

Diese Verfahren sind als Lösungswissen aufbereitet. Zwei exemplarische Steckbriefe sind in Bild 4-39 und Bild 4-40 dargestellt. Eine Auswahl der genannten Monkeys wurden prototypisch als agentenbasierte Testverfahren in einem Virtuellen Testbed umgesetzt.

Der Steckbrief umfasst drei Felder: Die **Rollenbeschreibung** beschreibt das Verfahren sowie die Art und den Ort der Beeinträchtigung. In der **Aufgabenbeschreibung** werden

die Aktivitäten zusammengefasst. Eine Strategie repräsentiert eine konsistente Kombination dieser Aktivitäten, z.B. betreffende Einheit, parametriertes Rauschmodell, Dauer der Aktivitäten. Jeder Strategie wird ein Seed-Wert zugewiesen. Der Seed-Wert ist eine Zufallszahl, mit der die Strategie durch einen Benutzer reproduzierbar ausgeführt werden kann. Der **Ablauf** wird in Form eines Aktivätsdiagramms dargestellt und beschreibt die Aktivitäten des Monkeys. Den Aktivitäten sind **Logging und Parameter**, z.B. Zeitstempel, Status, zugewiesen. Dies ermöglicht eine automatisierte Datenannotation.

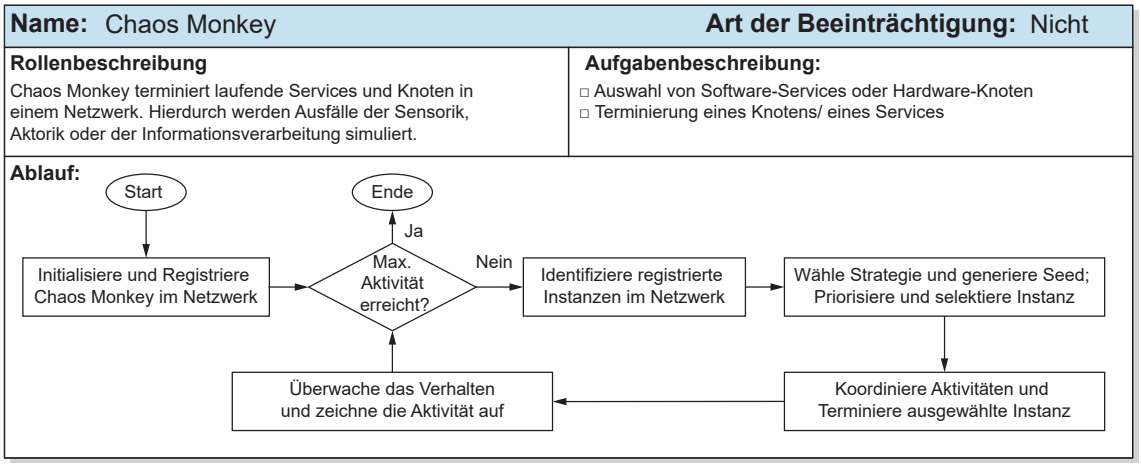


Bild 4-39: Steckbrief: Chaos Monkey

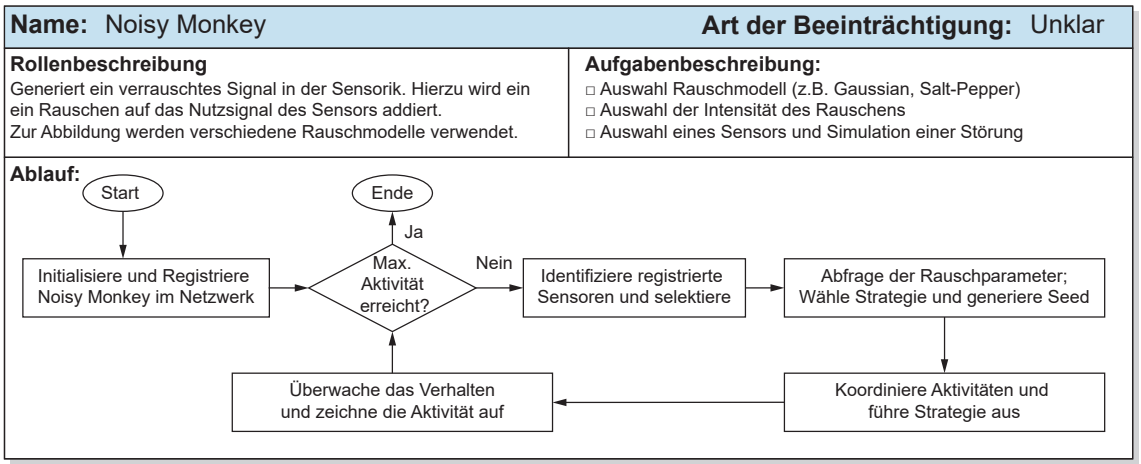


Bild 4-40: Steckbrief: Noisy Monkey

4.6.3 Formale Beschreibung

In der Arbeit wird die Informationsverarbeitung des selbstheilenden Systems als dynamischer Graph repräsentiert. $G(t) = (V, E(t))$ mit $V = \{v_0, ..., v_n\}$ eine Menge an Knoten und $E(t) \subseteq V(t) \times V(t)$ die Menge an Kanten im Graph zu einem Zeitpunkt t. Die Kanten sind dynamisch und sind zu einem Zeitpunkt t *aktiv* oder *inaktiv*. Als Annahme gilt: Der Graph ist beobachtbar, d.h. zu jedem Zeitpunkt t sind die Knoten und aktiven Kanten im System bekannt. Ein Knoten in dem Graph repräsentiert ein Software- oder Hardwareelement in

dem System, z.B. Sensorknoten. Direkte Kanten repräsentieren Daten- bzw. Informationsflüsse zwischen zwei Knoten. Ein vereinfachtes Modell von einem Graphen ist in Bild 4-41 visualisiert.

Aus dem Graphen sind **Komplexitätsmaße** ableitbar. Diese Komplexitätsmaße dienen dem Testverfahren für eine informierte Suche nach und Auswahl von kritischen Einheiten im System. Als Maß dienen *Aktivität, Kritikalität, Artikulationsknoten* und *Rückkopplungen* [LMB09, S. 201ff.]: Die **Aktivität** bildet sich aus der Division von Aktiv- und Passivsumme[3]. Die relative Kennzahl beschreibt die Tendenz zu aktiven oder passiven Verhalten der Einheit im System. Die **Kritikalität** eines Knotens ergibt sich aus der Multiplikation von Aktiv- und Passivsumme. Die Kennzahl beschreibt die relative Sensitivität gegenüber Änderungen in dem Netzwerk. Ein **Artikulationsknoten** repräsentiert die einzige Verbindung zwischen zwei Cluster in einem Netzwerk. Eine **Rückkopplung** besteht aus zwei oder mehr Knoten, die sich wechselseitig beeinflussen.

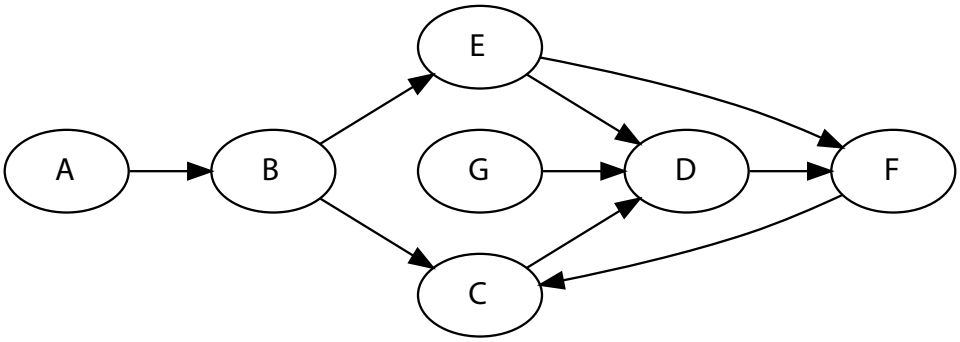


Bild 4-41: Repräsentation der Informationsflüsse eines Systems als Graph

4.6.4 Realisierung

Die Softwarearchitektur des Monkey Testsystem zeigt Bild 4-42. Sie Architektur umfasst vier Hauptmodule. Der **Simulationsmanager** interagiert mit dem Anwender des Virtuellen Testbeds und steuert den Simulationsablauf. Die Anwenderschnittstelle ermöglicht den Start/Stopp der Simulation und dient zur Testautomatisierung. Das Modul besteht aus einem Konfigurator und einen Datenlogger. Der Konfigurator selektiert einen Monkey und setzt dessen Parameter. Ebenso ermöglicht es die dynamische Konfiguration von Simulationsparametern zur Simulationszeit. Der Datenlogger verfolgt die Aktivitäten des Monkeys auf dem System und zeichnet diese mit einem Zeitstempel auf.

Das **Monkey Testsystem** bildet den Kern der Implementierung. Es repräsentiert eine Modellbibliothek in der die Monkeys gespeichert sind. Über den Simulationsmanager werden die Monkeys instanziiert. Der Scheduler überwacht, steuert und koordiniert die Aktivitäten. Der Instanzselektor wählt eine Instanz aus. Der Nachrichtenfänger manipuliert

[3] Aktivsumme: Anzahl ausgehender Verbindungen, Passivsumme: Anzahl eingehender Verbindungen

die Signale, z.B. Sensorsignale oder Abschaltung von einem Knoten. Das Kontextmodul analysiert den Systemzustand. Er dient dazu kontextabhängig Aktivitäten auszulösen. Das **Meta-Operating System** umfasst den Server und das Messaging im System.

Das Modul **Datenkollektor und Datenbank** stellt einen Datenkollektor, eine Zeitserien-Datenbank und eine Server-Infrastruktur bereit. Der Kollektor hat Zugriff auf die Sensor- und Systemsignale aus der Plattform und sammelt die Daten aus verschiedenen Datenquellen. Hierdurch ist eine automatisierte Datenannotation realisiert. Das Modul stellt eine API für das Training und das Testing der Modelle des maschinellen Lernen bereit. Die Kommunikation der Bausteine erfolgt über TCP/IP. Ebenso wird eine Anbindung an die reale Hardware ermöglicht.

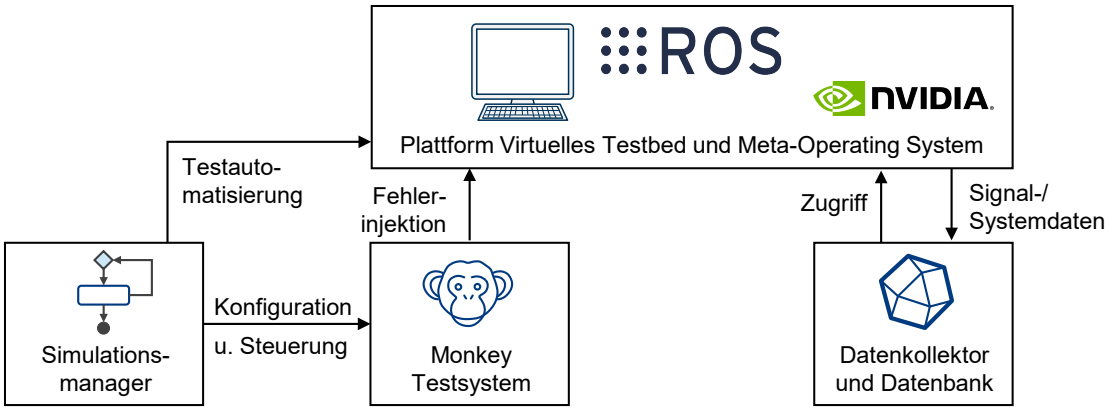


Bild 4-42: Realisierung des Monkey Testsystems

4.7 Vorgehensmodell

Dieses Kapitel fasst die Methoden und Hilfsmittel zu einer Vorgehenssystematik zusammen. Sie ist Teil des modellbasierten Systementwurfs. Das Kap. 4.7.1 erläutert das Vorgehen durch wiederkehrende Prozessbausteine und definiert konkrete Ergebnisse. Das Kap. 4.7.2 definiert die Rollen und Verantwortlichkeiten in den Prozessbausteinen.

4.7.1 Makrozyklus

Das Vorgehen zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme ist inkrementell und iterativ. Das entwickelte Vorgehen orientiert sich auf Makroebene an der VDI 2206 [VDI2206, S. 29] und an der ISO 21448 [ISO 21448, S. 10]. Ergebnis des Vorgehens ist ein validierter Prototyp des selbstheilenden Systems. Das Bild 4-43 zeigt das Vorgehen idealtypisch als Makrozyklus. Die Phasen werden mehrmals durchlaufen. Mit jedem Durchlauf steigt der Reifegrad des Prototyps. Zur Vereinfachung und Lesbarkeit ist in dem Bild nur eine Iteration dargestellt.

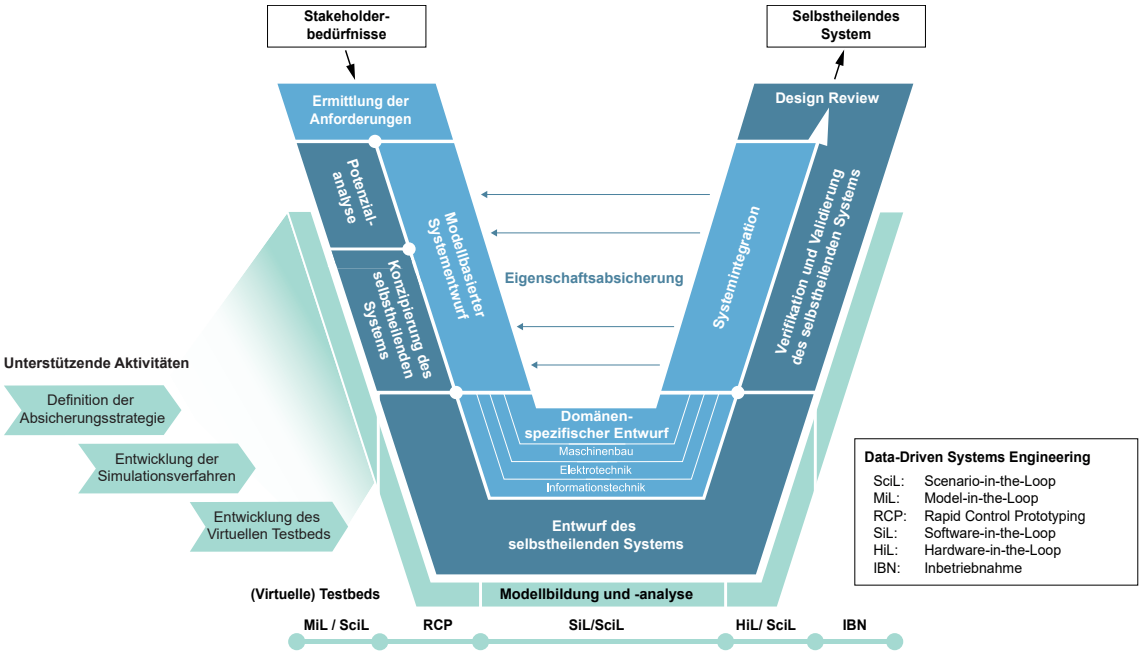


Bild 4-43: Vorgehensmodell auf Makrozyklus

Anforderungsphase

Ermittlung der Anforderungen: In dieser Phase werden die Stakeholder und deren Anforderungen an das zu entwickelnde System ermittelt und analysiert. Anforderungen repräsentieren die Entwicklungsziele. Grundlage bilden die Partialmodelle Kap. 4.3.1. Zunächst wird das Systemumfeld und die Systemgrenze definiert und Anwendungsszenarien identifiziert. Durch die Interaktion des Systems mit seinem Umfeld ergeben sich potenzielle Fehlanwendungen. Iterativ werden die Anforderungen an das System erarbeitet und die Funktionen ermittelt.

Ergebnis: Priorisierte Anforderungen an das zu entwickelnde System.

Konzeptphase

Die **Konzeptphase** umfasst den Modellbasierten Systementwurf die Potenzialanalyse sowie die Konzipierung des selbstheilenden Systems:

Modellbasierter Systementwurf: Mit der **Synthese** wird die Prinziplösung erarbeitet. Aus den Anforderungen und Anwendungsfällen werden die Funktionen ermittelt und mögliche Lösungselemente identifiziert [Dum10]. Die Systemelemente werden anschließend zu einer Wirkstruktur zusammengefügt. Parallel werden Verhalten und Grobgestalt des Systems definiert. Ebenso wird das Zielsystem erarbeitet.

Ergebnis: Prinziplösung des intelligenten technischen Systems

Potenzialanalyse: Ziel dieser Phase ist die Analyse von Potenzialen zur Integration von Eigenschaften der Selbstheilung. Hierzu werden kritische Funktionen und Strukturen im System identifiziert und als SuOC definiert. Als Hilfsmittel dient die in Kap. 4.4 entwi-

ckelte Potenzialanalyse. Die Bewertung erfolgt anhand der Dimensionen: Wissensgrad, Komplexität und Sicherheitsrisiko. Für das SuOC wird der anzustrebende Ziel-Reifegrad als Normstrategie festgelegt. Darüber hinaus werden Sicherheitsziele des Systems festgelegt. Für das SuOC werden Beeinträchtigungen ermittelt, die zu einer Gefahrensituation führen können. Dabei werden Zustandsgrößen identifiziert, mit denen diese Beeinträchtigungen durch das System überwacht werden können. Ebenso werden Operationen zur Selbstheilung definiert.

Ergebnis: Potenziale zur Integration von Selbstheilung.

Konzipierung des selbstheilenden Systems: Das Ziel dieser Phase ist die Integration der Selbstheilungseigenschaften in die Prinziplösung des Intelligenten Technischen Systems. Grundlage bildet die Architektukonzepte und das entwickelte Framework (vgl. Kap. 4.2.2). Die Partialmodelle sind um die Bausteine des Frameworks zu ergänzen und die Signale und Daten zu beschreiben. Darüber hinaus sind die Operationen der Selbstheilung in die Prinziplösung zu integrieren (z.B. Redundanz). Das Ergebnis umfasst ein Konzept zur Umsetzung selbstheilender Systeme entsprechend des festgelegten Ziel-Reifegrades und der zugehörigen Entwurfsaspekte.

Ergebnis: Konzept und Architektur des selbstheilenden Systems.

Entwurfsphase

In dieser Phase erfolgt der domänenspezifische Entwurf des selbstheilenden Systems und der Entwurf der einzelnen Module der Selbstheilung.

Domänenspezifischer Entwurf: Auf Basis der Prinziplösung erfolgt der domänenspezifische Entwurf des intelligenten technischen Systems. Dies umfasst die Mechanik, die Elektrik/Elektronik sowie die Software und Regelungstechnik. Es werden die Sensorik, Aktuatorik, die selbstregulierende Informationsverarbeitung sowie die Kommunikation und die Benutzungsschnittstelle umgesetzt. Die domänenspezifischen Teilmodelle werden fortlaufend in einem virtuellen Prototypen zusammengeführt.

Ergebnis: Experimentierbarer virtueller Prototyp

Entwurf des selbstheilenden Systems: In diesem Schritt werden die Entwurfsaspekte und zugehörigen Konstruktionsprinzipien für das ausgewählte SuOC ausgewählt, entwickelt und integriert. Das Framework wird umgesetzt und in den experimentierbaren virtuellen Prototyp integriert. Die Modelle zur Überwachung, Diagnose, Planung und Ausführung sowie die zugehörigen Wissensbasen werden entwickelt und trainiert. Das Training erfolgt auf Basis von Scenario-in-the-Loop-Testing (vgl. Kap. 4.5.1).

Ergebnis: Modelle und Wissensbasen zur Realisierung des selbstheilenden Systems.

Integrationsphase

In dieser Phase erfolgt die schrittweise Integration der Module zum selbstheilenden System. Teil der Integration ist die kontinuierliche Verifikation und Validierung. Die Phase schließt

mit einem Design Review und der Freigabe.

Verifikation des selbstheilenden Systems: Ziel ist die Verifikation von Eigenschaften der Selbstheilung anhand der Anforderungen. Dabei wird überprüft, ob die Verfahren in dem Framework korrekt funktionieren. In diesem Schritt werden die Komponenten und die Schnittstellen des Frameworks getestet und hinsichtlich ihres Qualitätsprofils evaluiert (vgl. Kap. 4.5.4). Dies erfolgt zum Beispiel durch anforderungsbasiertes Testen sowie durch Komponenten- und Schnittstellentests in dem virtuellen Testbed.

Ergebnis: Verifikation auf Basis bekannter Szenarien.

Validierung des selbstheilenden Systems: Ziel dieser Phase ist die Validierung von Eigenschaften der Selbstheilung anhand von virtuellen Szenarien, XiL-Techniken und Feldtests mit physischen Prototypen. Dies erfolgt durch szenario-basiertes Testen. Dabei wird die Resilienz des Systems, sowie die Kennzahlen zur Bewertung der Eigenschaften der Selbstheilung untersucht. Monkey Testverfahren operieren auf dem Prototypen. Untersucht wird das Sicherheits- und Ausfallrisiko des Systems. Die Qualitätsbewertung kann Online und Offline erfolgen.

Ergebnis: Validierung auf Basis unbekannter Szenarien.

Design Review: Ziel dieser Phase ist die Freigabe des selbstheilenden Systems in der vorgesehenen Betriebsdomäne. Domänen- und Sicherheitsexperten analysieren und bewerten die Ergebnisse aus der Verifikation und Validierung und schätzen das Rest- und Ausfallrisiko ein. Dabei ist zu entscheiden, ob das Risiko des selbstheilenden Systems innerhalb der Akzeptanz liegt und ob das selbstheilende Systeme in den Anwendungsszenarien und den Betriebsprofile resilient funktioniert. Das Review erfolgt z.B. durch ein gemeinsames Walktrough.

Ergebnis: Review und Freigabe des selbstheilenden Systems.

Unterstützende Aktivitäten

Unterstützende Aktivitäten dienen der Entwicklung selbstheilender Systeme und stellen die Werkzeugumgebung zur Simulation, Analyse und Absicherung bereit. Dieser Bereich umfasst die Definition einer Absicherungsstrategie, die Entwicklung und Anpassung von Simulations- und Analysetechniken sowie die Validierung des Virtuellen Testbeds.

Definition der Absicherungsstrategie: Ziel dieser Phase ist die Absicherungsstrategie selbstheilender Systeme und das zugehörige Absicherungskonzept auf Systemebene (vgl. Kap. 4.3.2). Die Absicherungsstrategie definiert auf welche Teststufe welche Testfälle untersucht werden. Die Testfälle werden somit den XiL-Teststufen zugewiesen. Zuerst werden die Testfälle aus den Anforderungen und Anwendungsszenarien abgeleitet und mit diesen verknüpft. Anschließend wird ein statistisches Betriebsprofil mit den optionalen und erforderlichen Bestandteile definiert. Aus dem statistischen Betriebsprofil werden konsistente Szenarien abgeleitet. In der Absicherungsstrategie sind zudem die Kennzahlen und Zielbereiche für die Eigenschaftsbewertung eines selbstheilenden Systems zu definieren.

Ergebnis: Absicherungsstrategie und Absicherungskonzept des selbstheilenden Systems.

Entwicklung der Simulationsverfahren: Ziel dieser Phase sind wiederverwendbare Simulationsverfahren und Analysetechniken zur Absicherung im Virtuellen Testbed. Auf Basis des entwickelten Monkey Testings (vgl. Kap. 4.6.2) werden die Monkeys auf das zu entwickelnde System angepasst und die erforderlichen Eingangs- und Aufgabenparameter für die Aufgaben ermittelt. Anschließend werden die Aufgaben (z.B. Drift in einem Sensor erzeugen) modelliert und die erforderlichen Komponenten umgesetzt (z.B. Dynamische Rekonfiguration). Das entwickelte Verfahren wird parametriert (z.B. Dauer und Anzahl einer Aktivität). Die Verifikation überprüft, ob der Monkey korrekt operiert. Die Validierung stellt sicher, dass Monkey eine Beeinträchtigung korrekt approximiert. Nach der Verifikation und Validierung wird das Verfahren in der Bibliothek angelegt und kann in der Simulation verwendet werden. Das Ergebnis dieses Prozesses ist das Monkey Testing.

Ergebnis: Angepasstes Monkey Testsystem

Entwicklung des Virtuellen Testbeds Ziel dieser Phase ist Sicherstellung aussagekräftiger Modelle für das maschinelle Lernen und für das Scenario-in-the-Loop Testing. Das Vorgehen orientiert sich an RAST [Ras15, S. 13ff.]. Zunächst werden die Anforderungen an das Virtuelle Testbed aus der Absicherungsstrategie abgeleitet. Sie legen das Ziel und den Zweck der Simulation fest (z.B. Simulation der Navigation). Anschließend werden die erforderlichen Modelle mit der entsprechenden Modellierungstiefe und Genauigkeit erarbeitet sowie deren Eingangs- und Ausgangsgrößen festgelegt. Davon ausgehend werden reale Referenzexperimente aufgesetzt und das virtuelle Modell entwickelt. Das Virtuelle Modell wird verifiziert und parametriert. Die Modellierung kann durch ein mathematisch-physikalisch korrektes Modell oder durch eine datengetriebene Modellierung approximiert werden. Das virtuelle Modell wird mit dem Referenzexperiment verglichen. Mittels dem Vergleich zwischen Realität und virtuellem Abbild wird das virtuelle Modell kalibriert. Dieses Modell wird abschließend in der Modellbibliothek mit einer Beschreibung der Funktion, Eingabe und Ausgabe, der Parameter sowie der Randbedingung zur Wiederverwendung angelegt. Das Ergebnis dieses Prozesses sind validierte und verifizierte Modelle.

Ergebnis: Virtuelles Testbed mit validierten Modellen.

Zusammenspiel der Phasen

Das Zusammenspiel der Phasen und die Entscheidungsknoten zeigt Bild 4-44. Für die Schritte sind Steckbriefe ausgearbeitet und in Anhang A2.5 dokumentiert. Diese Steckbriefe definieren wiederkehrende Prozessbausteine und weisen diesen Prozessbausteine Rollen, Verantwortlichkeiten und ihre Interaktion zu.

4.7.2 Rollenprofile

Die Entwicklung selbstheilender Systeme erfolgt durch das synergetische Zusammenspiel verschiedener Disziplinen und beteiligter Rollen. Die jeweiligen Rollen besitzen

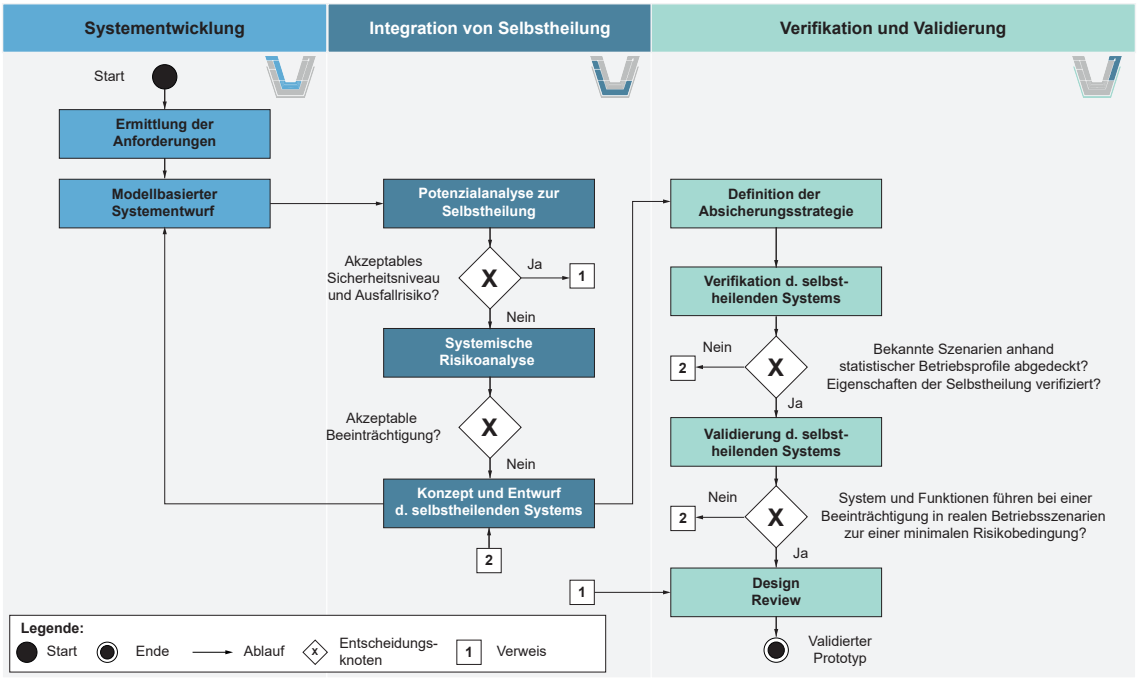


Bild 4-44: Prozessbausteine

konkrete Aufgaben in der Entwicklung. Entsprechend dieser Tätigkeiten nehmen sie einen Standpunkt ein und haben eine spezifische Sicht auf die Modelle. Diese Sichten werden mit Hilfe des entwickelten Sprachprofils gebildet und können in den Autorenwerkzeugen erzeugt werden. Das Rollenprofil orientiert sich an den 12 Systems Engineering Rollen von SHEARD [She96, S. 2] sowie relevante Domänenexperten. Für die Entwicklungssystematik ergeben sich so 10 Rollenprofile, die in der Entwicklungssystematik interdisziplinär zusammenwirken.

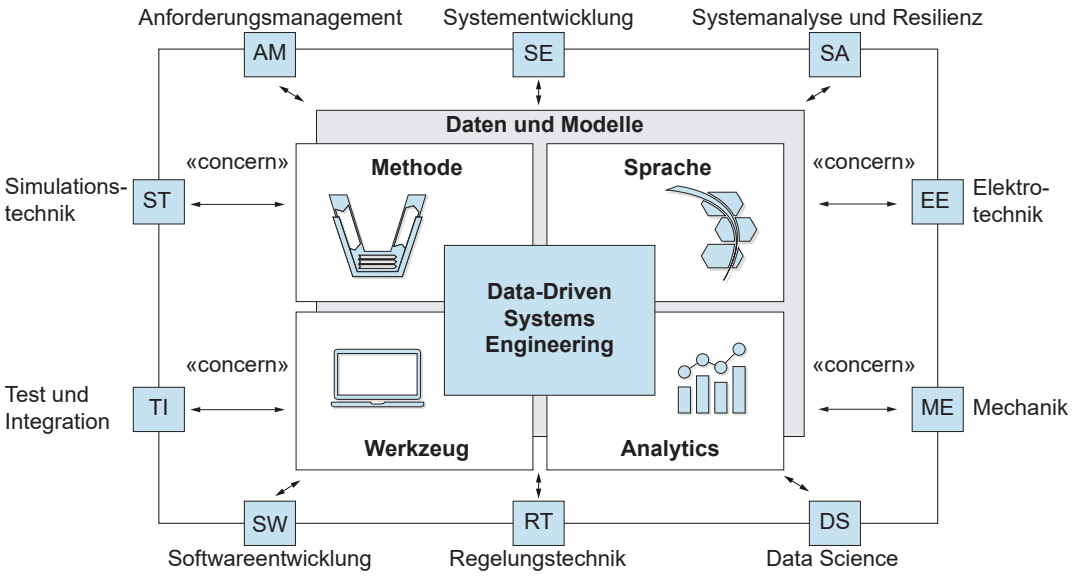


Bild 4-45: Ausgewählte Rollen

Anforderungsmanagement (AM): Das Anforderungsmanagement ermittelt die Stake-

holder und identifiziert die Anwendungsszenarien des Systems. Daraus leitet die Rolle Anforderungen ab, verwaltet und priorisiert diese.

Systementwicklung (SE): Die Systementwicklung erarbeitet die Systemarchitektur, d.h. Funktionen und Struktur auf Basis der Anwendungsszenarien, Umfeldmodell und Anforderungen. Die Sicht fokussiert die Funktionen, die Wirkstruktur und Verhalten sowie das konsistente Zusammenspiel dieser Aspekte.

Data Science (DS): Die Rolle verantwortet die datengetriebene Modellierung selbstheilender Systeme und die Verarbeitungspipeline. Dazu benötigt die Rolle eine Sicht auf die Struktur und das Verhalten des Systems. Insbesondere auf die Daten, Signale, Datenquellen und Informationsflüsse. Die Anforderungen und Anwendungsszenarien sind Teil der Domänenbeschreibung.

Elektrotechnik (EE): Die Rolle verantwortet die Entwicklung elektrischer und elektronischer Funktionen, z.B. Mikrocontroller, Energieversorgung oder Schaltungstechnik. Dazu benötigt die Rolle eine Sicht auf die Struktur, insbesondere auf die Energie- und Informationsflüsse sowie elektronischen Systemelemente im System.

Mechanik (ME): Die Rolle definiert die Gestalt und die Konstruktion des Systems. Dazu sind gestaltbeeinflussende Größen in der Prinziplösung darzustellen. Dies erfolgt durch die Sicht auf die Struktur und mechanischen Kräfte sowie gestaltbehafteten Systemelemente. Die Grobgestalt zeigt u.a. die Hüllflächen, Wirkprinzipien und die Geometrie des Systems.

Regelungstechnik (RT): Die Rolle entwickelt ein Reglerkonzept und setzt die Mehrzieloptimierung um. Dazu sind u.a. die zu relevanten Größen (z.B. Stell-, Regel- und Störgrößen) und zugehörigen Parameter (z.B. PID, Trägheit) zu identifizieren, sowie die Zielfunktionen zu definieren. Dazu sind Sichten auf die Anforderungen, Struktur, Verhalten sowie das Zielsystem erforderlich.

Simulationstechnik (ST): Entwicklung und Validierung des Virtuellen Testbeds. Sicht auf die Testszenarien und Testfälle sowie die Testdaten. Erhebung der Anforderungen an die Simulation für das Scenario-in-the-Loop-Testing.

Systemanalyse und Resilienz (SA): Umsetzung des Frameworks selbstheilender Systeme. Sicht auf die Anforderungen, Potenziale zur Integration von Eigenschaften der Selbstheilung und Definition des SuOC. Die Rolle definiert den anzustrebenden Ziel-Reifegrad und legt die Entwurfsaspekte des selbstheilenden Systems fest. Er stellt das entsprechende Domänenwissen auf Systemebene (z.B. Fuzzy-Pattern-Tree, Wissensbasen) bereit und definiert die Operationen zur Selbstheilung im System.

Softwareentwicklung (SW): Entwicklung der Informationsverarbeitung (z.B. Betriebssystem, Kommunikation, Datenverarbeitung) des selbstheilenden Systems. Sicht auf das diskrete intendierte Verhalten der Software, wie etwa Zustände, Aktivitäten und Sequenzen. Sicht auf die Daten- und Signalflüsse in der Struktur.

Test und Integration (TI): Diese Rolle entwickelt die Absicherungsstrategie und ist

verantwortlich für die fortlaufende Verifikation und Validierung selbstheilender Systeme mittels XiL-Techniken. Sicht auf die Testfälle mit messbaren Testkriterien zur Überprüfung der Anforderungen und Anwendungsszenarien sowie eine Sicht auf die Testbedstruktur und den Szenarioablauf.

Im Anhang A2.5.1 sind die Rollen zusammengefasst und ihr Anliegen konkretisiert. Die relevanten Partialmodelle sind hervorgehoben.

Verantwortlichkeitsprofil

Die Responsibility Assignment Matrix (RAM)-Matrix definiert die Rollen und die Verantwortlichkeiten im Prozess. Die Zuweisung erfolgt auf Basis der RASCI-Nomenklatur [Tsc16, S. 131]:

- **Responsible (R):** Die Rolle verantwortet den Prozessbaustein und führt den Prozess maßgeblich aus.
- **Accountable (A):** Die Rolle ist zuständig für die Freigabe und für die Überprüfung der Ergebnisse.
- **Supportive (S):** Die Rolle unterstützt die Durchführung der Tätigkeit. Es handelt sich um Experten, die aktiv in den Prozess und die Tätigkeit eingebunden werden.
- **Consulted (C):** Die Rolle berät bei der Durchführung der Tätigkeit und steht bei Bedarf zur Verfügung.
- **Informed (I):** Rolle wird über das Ergebnis informiert. Es handelt sich dabei um die Übertragung von Informationen.

Das Bild 4-46 zeigt die RAM-Matrix für die Entwicklungssystematik.

Prozessschritt	DS	EE	ME	RT	AM	SE	SI	SA	SW	TI
Ermittlung der Anforderungen	C	C	C	C	R/A	S	-	C	C	I
Modellbasierter Systementwurf	S	S	S	S	S	R/A	I	S	S	I
Potenzialanalyse zur Selbstheilung	S	S	S	S	C	S	I	R	S	I
Systemische Risikoanalyse	S	S	S	S	C	S	I	R	S	I
Konzept und Entwurf des selbstheilenden Systems	S	S	S	S	C	S	I	R/A	S	I
Definition der Absicherungsstrategie	C	C	C	C	C	C	S	S	C	R/A
Verifikation des selbstheilenden Systems	C	C	C	C	C	C	S	S	C	R/A
Validierung des selbstheilenden Systems	C	C	C	C	C	C	S	S	C	R/A
Design Review des selbstheilenden Systems	S	S	S	S	S	S	C	R/A	S	S

R: Responsible	AM: Anforderungsmgmnt.	SA: Systemanalyse
A: Accountable:	DS: Data Science	SE: Systementwicklung
S: Supportive	EE: Elektrotechnik	ST: Simulationstechnik
C: Consulted	ME: Mechanik	SW: Softwareentwicklung
I: Informed	RT: Regelungstechnik	TI: Test und Integration

Bild 4-46: RAM-Matrix

5 Anwendung der Entwicklungssystematik

Dieses Kapitel zeigt die Anwendung der Systematik. Das Vorgehen wird in Kap. 5.1 vollständig von der Anwendungsidee bis zum physischen Prototypen durchlaufen. Das Kapitel schließt mit einer Bewertung der Systematik anhand der Anforderungen (Kap. 5.2). Die Anwendung der Systematik basiert auf den Inhalten und Ergebnissen der vom BMBF geförderten Forschungsprojekte Lichtsensor-basierte Ortungs- und Navigationsdienste für autonome Systeme (LIONS)[1] und KI4AS[2] sowie den Vorarbeiten aus dem Verbundprojekt INVIRTES[3].

5.1 Anwendungsbeispiel: Mobiler Roboter

In der intelligenten Fabrik kollaborieren diverse Maschinen und Roboter in einem vernetzten Systemverbund. Die Kommunikation zwischen den Maschinen und ein autonomer Warentransport sind essentiell für die Aufrechterhaltung des Betriebs. Die Verfügbarkeit und Sicherheit dieses komplexen Systems ist stets zu gewährleisten. Fahrerlose Transportsysteme wie etwa mobile Roboter setzen einen autonomen Warentransport um. Als Anwendungsbeispiel der Systematik wird daher ein Szenario zum intralogistischen Warentransport mit autonomen Robotern realisiert.

5.1.1 Anforderungsphase

Ermittlung der Anforderungen

In der intelligenten Fabrik operieren mobilen Roboter in einer unstrukturierten und dynamischen Umgebung. Dabei haben sie verschiedene Stationspunkte wie etwa das Lager, Arbeitsplatz oder die Basis- und Ladestation. Entsprechend der Transportaufgabe werden diese Stationspunkte angefahren. Mobile Roboter müssen sicher operieren und verfügbar sein. Dabei müssen die Mobilen Roboter ihre Umgebung wahrnehmen und ihren eigenen Betriebszustand überwachen. Bei Ausfall eines Transportsystems (z.B. aufgrund eines niedrigen Ladezustandes) kann die Aufgabe von einem anderen System übernommen werden. Dieses Anwendungsszenario ist in Bild 5-1 dargestellt.

---

[1] Förderkennzeichen: 16ES0373K, Laufzeit von 2015 bis 2018

[2] Förderkennzeichen: 03VP02551, Laufzeit: 2017 bis 2020

[3] Förderkennzeichen: 50RA1309, Laufzeit: 2014 bis 2018

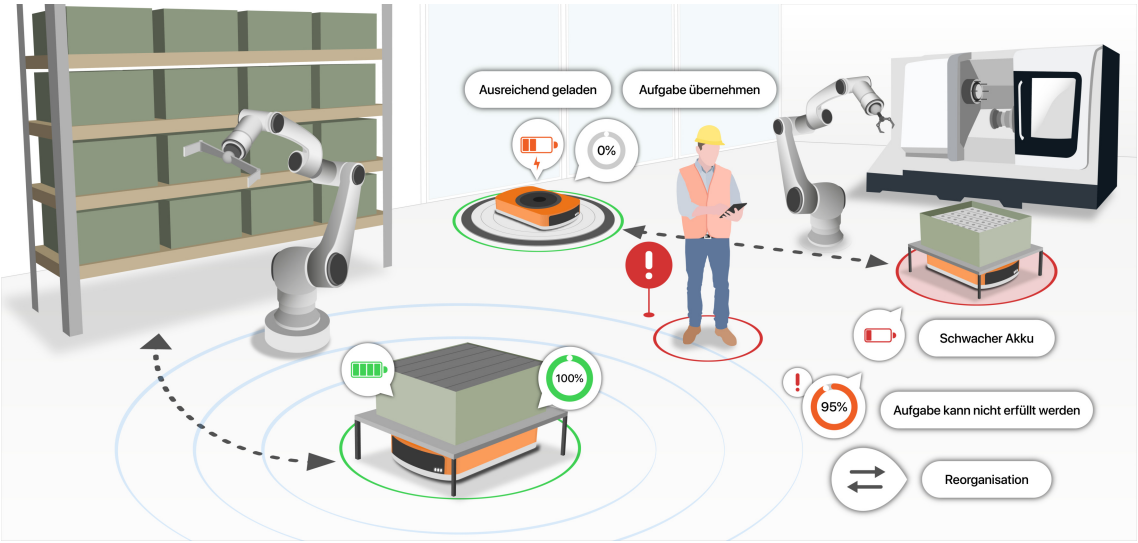


Bild 5-1: Anwendungsszenario in der intelligenten Fabrik

Das Anwendungsszenarios wird durch das Umfeldmodell in Bild 5-2 konkretisiert. Durch die Interaktion des Systems mit seiner Umgebung leiten sich weitere Anwendungsfälle ab. Entlang des Produktlebenszyklus existieren verschiedene Umfeldmodelle, z.B. Produktion, Betrieb, Wartung. Ebenso kann ein Umfeldmodell für spezifische Betriebsszenarien ermittelt werden (z.B. System laden). In Summe wurden 35 Anwendungsszenarien identifiziert.

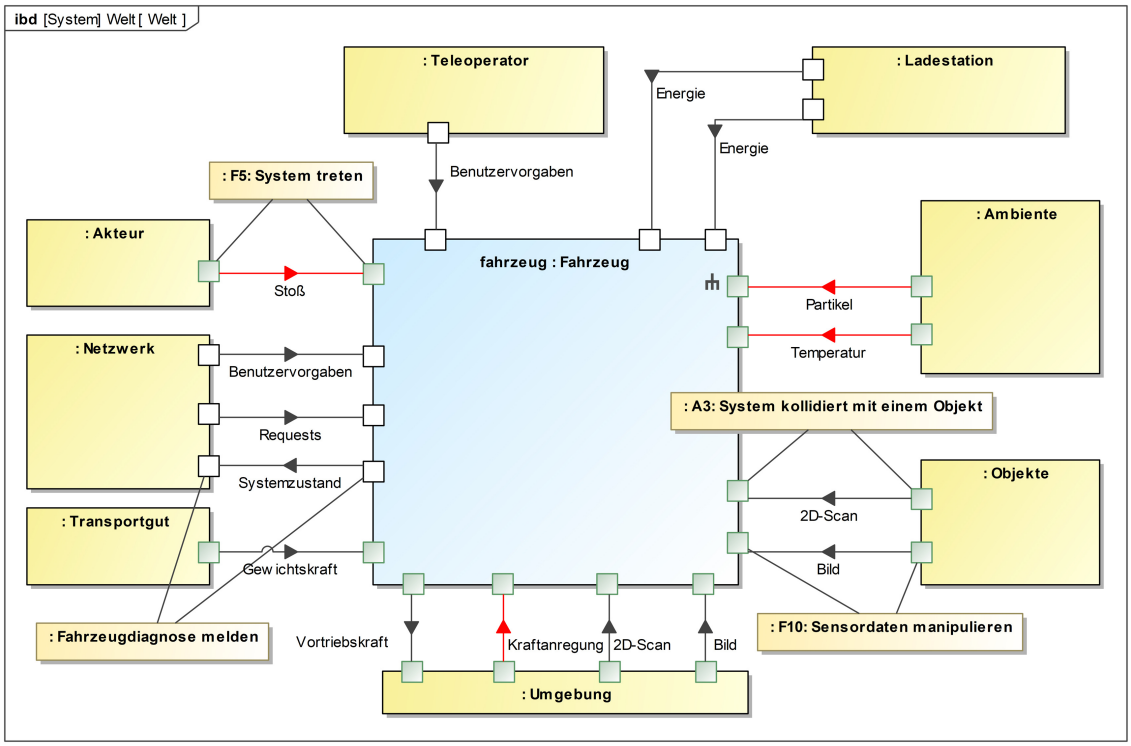


Bild 5-2: Umfeldmodell des Mobilen Roboters

Die Inhalte der Partialmodellen Umfeld und Anwendungsfälle dienen als Grundlage für die Anforderungsanalyse. Das Ergebnis ist das Partialmodell Anforderungen wie in Bild 5-3

dargestellt. Dieses Partialmodell beschreibt die Anforderungen, verknüpft diese mit den Anwendungsfällen und mit den Testfällen des Systems. Ebenso werden die Anforderungen mit den Funktionen und den realisierenden Systemelementen verknüpft.

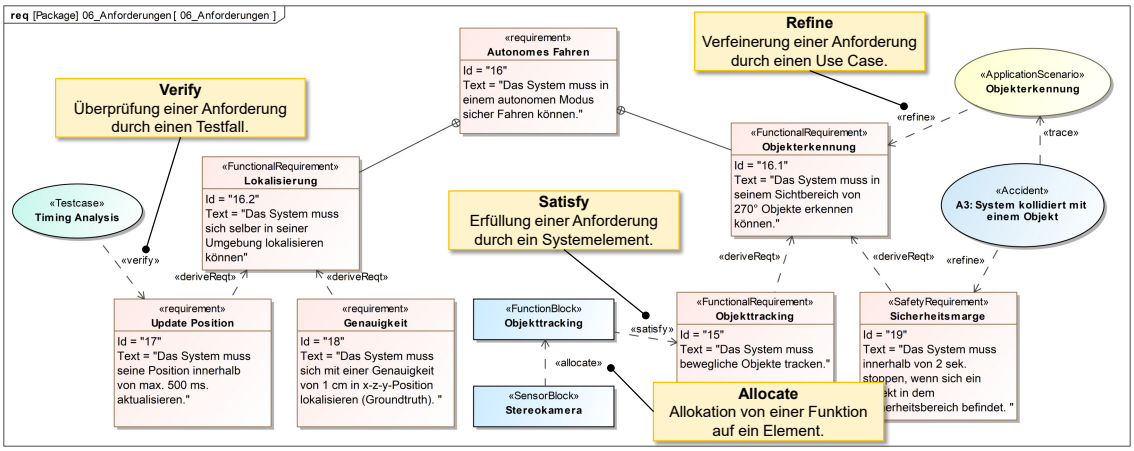


Bild 5-3: Modell der Anforderungen vom Mobilen Roboter

5.1.2 Konzeptphase

5.1.2.1 Modellbasierter Systementwurf

Das Bild 5-4 zeigt das Vorgehen im modellbasierten Systementwurf. Basierend auf den Anforderungen erfolgt die Synthese. Das Ergebnis umfasst Wirkstruktur, Zielsystem und Verhalten. Diese Partialmodelle werden durch Hilfsmittel wir der Risiko- und Potenzialanalyse sowie der Absicherung ergänzt. Zeitanalysen dienen der Machbarkeitsanalyse und verfeinern weitere Anforderungen.

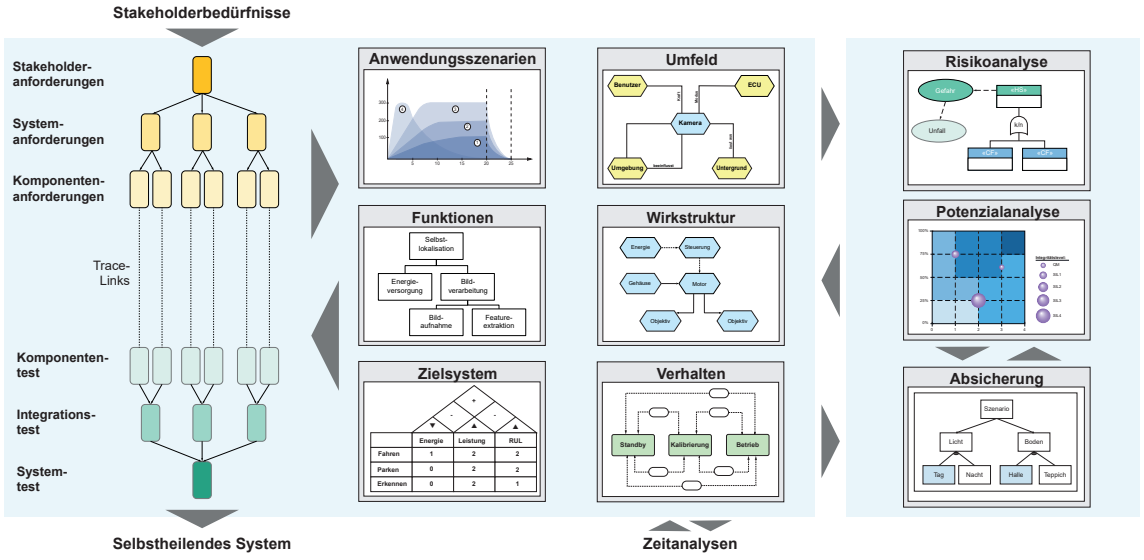


Bild 5-4: Modellbasierter Systementwurf

Als Referenzmodell der Informationsverarbeitung dient die funktionale Schichtenarchi-

tektur nach DUMITRESCU [Dum10]. Die funktionale Architektur visualisiert Bild 5-5. Die Hauptfunktionen gliedern sich in *Sensieren*, *Planen* und *Agieren*. Die Architektur hierarchisiert die Informationsverarbeitung in die Schichten *Nicht-Kognitive Regulierung*, *Assoziative Regulierung* und *Kognitive Regulierung*.

Teil der Nicht-kognitiven Regulierung sind die Selbstlokalisierung und die Motorregelung. Diese erfolgt unter Echtzeit-Bedingungen. Auf der Assoziativen Regulierung ist die Objekterkennung und die Situationsanalyse sowie die Manöverplanung und das Bewegungsmanagement integriert. Die Kognitive Regulierung umfasst die Exploration und Kartierung sowie das Zielmanagement und die Routenplanung. Die Routenplanung erfolgt auf globaler Ebene zwischen den Stationen durch Planungsalgorithmen die auf der Karte operieren. Die Manöverplanung setzt diese globale Pfadplanung in eine Manöverplanung um. Die Manöverplanung berücksichtigt dabei die Kinematik des Systems und ermittelt einen lokalen Pfad. Dieser Pfad wird über das Bewegungsmanagement in Sollwert-Vorgaben an die Lenkung und den Antrieb übersetzt und an die Motorregelung weitergegeben.

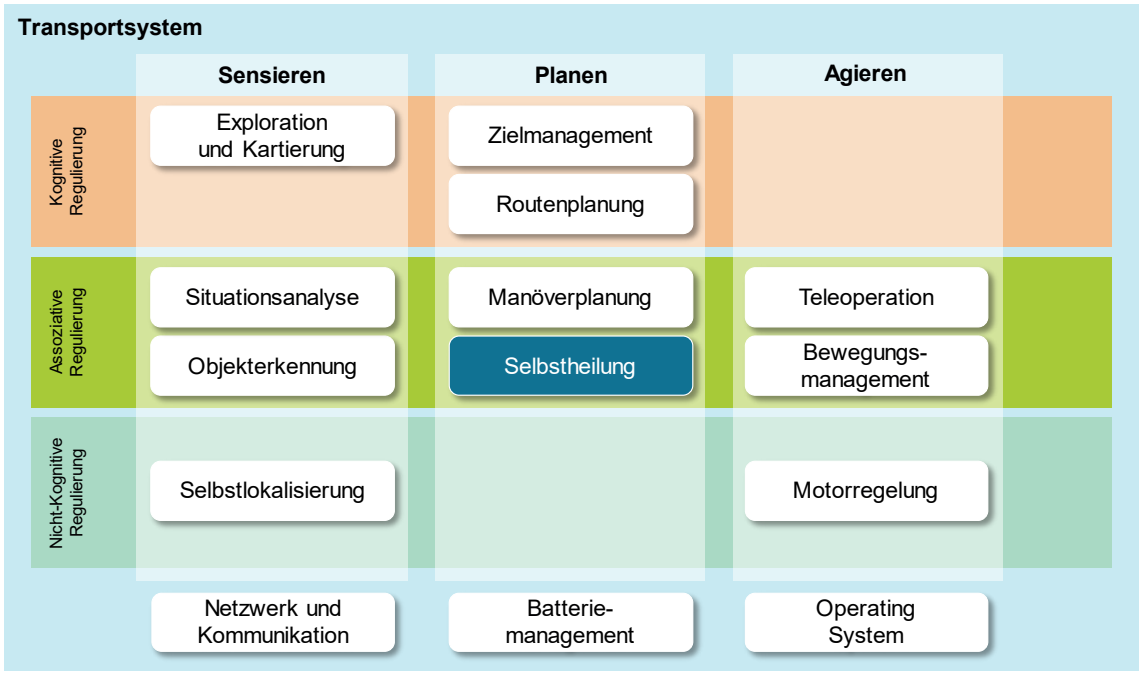


Bild 5-5: Funktionen der selbstregulierenden Informationsverarbeitung

Die *Routenplanung* erfolgt situationsabhängig und unter Berücksichtigung von vorgegebenen Zielen. Das resultierende Zielsystem wird durch die **Zielmatrix** konkretisiert (vgl. Bild 5-6). Für jedes Ziel wird ein Zielparameter festgelegt. Das Ziel *„Z1: Fahrzeit“* besitzt den Parameter *Fahrtdauer*. Diese ist zu minimieren (Zielrichtung). Die Zielkorrelation von *Fahrzeit* und *Energieverbrauch* ist negativ. Für das System sind vier Situationen definiert. In der Situation *Normal* sind alle Ziele gleich wichtig. In der Situation *Priorität* hat das Ziel *Fahrzeit* eine höhere Priorität als die anderen Ziele.

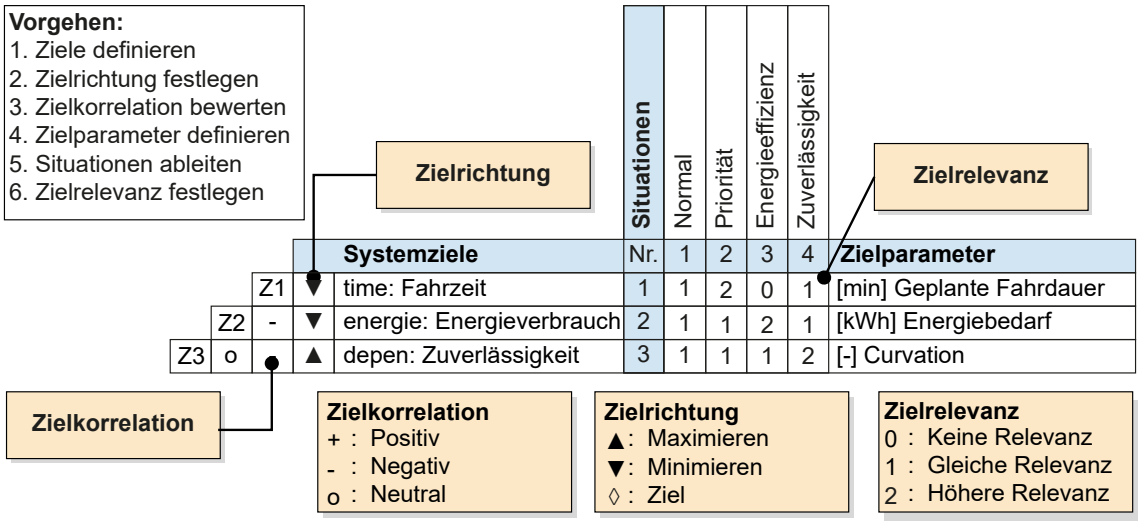


Bild 5-6: Zielmatrix der Routenplanung

Im nächsten Schritt werden die funktionsumsetzenden Systemelemente mit Hilfe eines Morphologischen Kasten [FG13] identifiziert. Das Resultat ist die Wirkstruktur (Bild 5-7). Die Wirkstruktur beschreibt die Systemelemente, Schnittstellen und die Flussbeziehungen. In der **Kommunikationsmatrix** werden die Signale auf die jeweiligen Ports der Systemelemente allokiert.

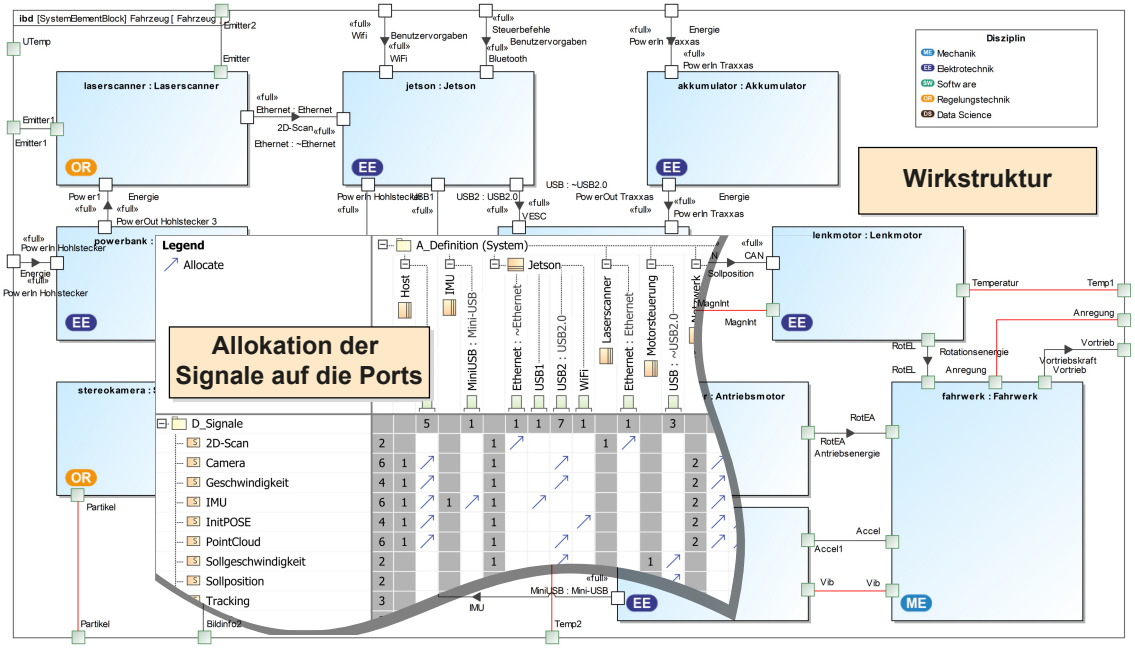


Bild 5-7: Wirkstruktur des Mobilen Roboters mit Zuweisung der Disziplin

Die Wirkstruktur zeigt eine Zuweisung der Systemelemente auf die Domänen durch ein Domain-Tags. Dadurch kann eine disziplinspezifische Sicht auf die Wirkstruktur erzeugt werden. Mit Hilfe dieser Sicht werden die Schnittstellen, z.B. Eingang- und Ausgangsgrößen oder Parameter der Systemelemente sowie die jeweiligen verknüpften

Modellartefakte wie etwa Anforderungen als Basis für den disziplinspezifischen Entwurf aufbereitet. Dies reduziert die Modellkomplexität und erhöht das Verständnis über die Wirkzusammenhänge.

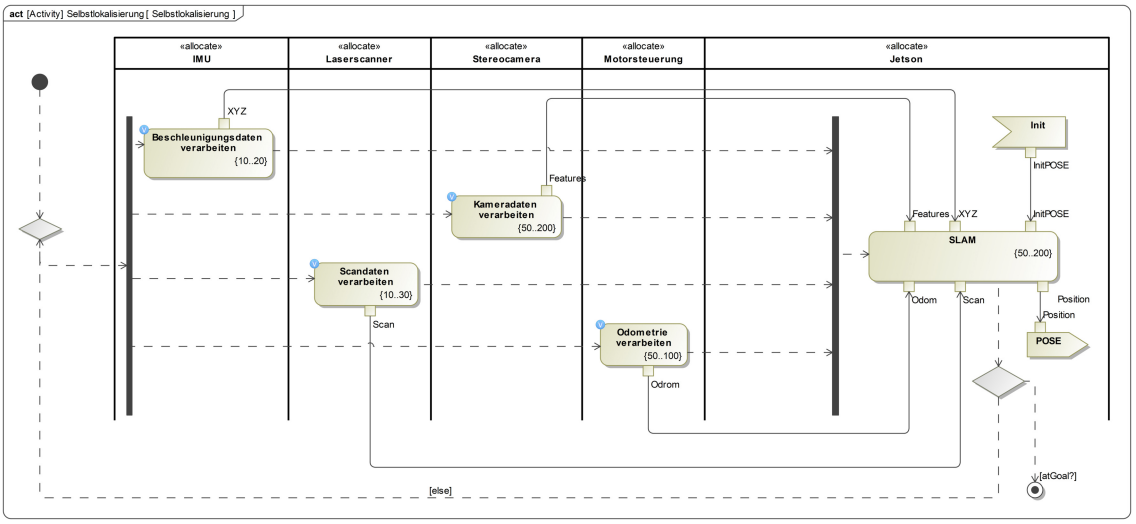


Bild 5-8: Aktivitätsdiagramm

Das Partialmodell **Verhalten - Aktivität** definiert den Ablauf und mögliche Ereignisse. Dabei werden die Objekt- und Kontrollflüsse modelliert. Die einzelnen Aktivitäten werden auf die Systemelemente der Wirkstruktur allokiert. Das Bild 5-8 zeigt das Aktivitätsdiagramm am Beispiel der Selbstlokalisierung.

Mit Hilfe des Partialmodells Verhalten - Aktivität werden **Zeitanalysen** durchgeführt. Diese unterstützen bei einer ersten Machbarkeitsanalyse der Lösung und der Anforderung. Bild 5-9 zeigt das Ergebnis einer Zeitanalyse für die Selbstlokalisierung. Die Grafik zeigt die maximale Ausführungszeit der einzelnen Aktivitäten.

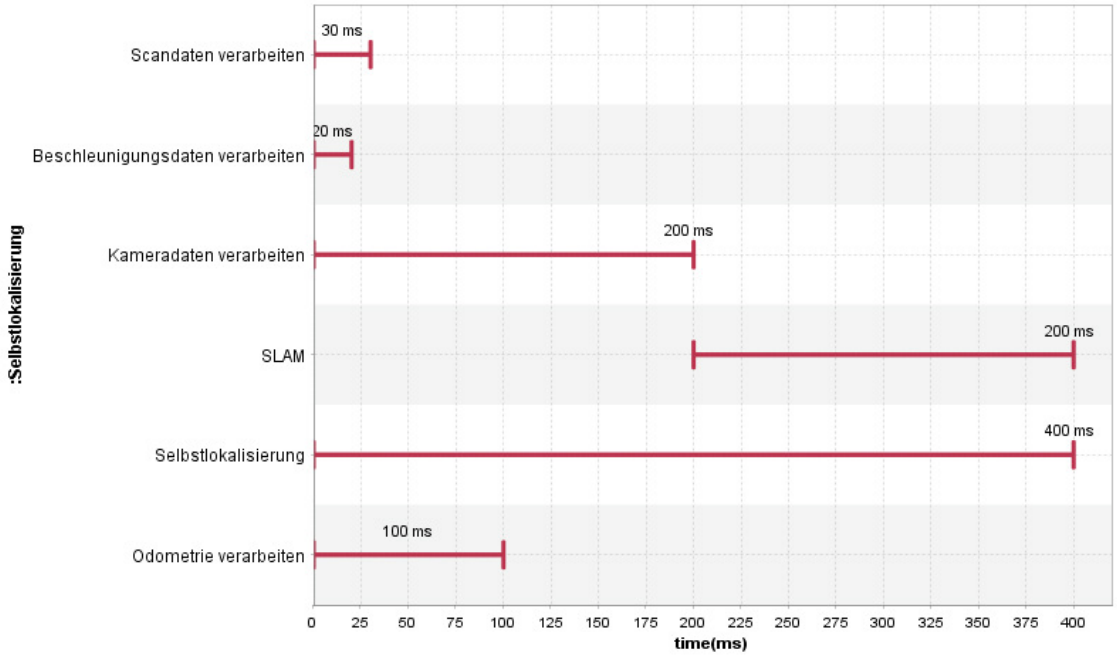


Bild 5-9: Zeitanalyse: Maximale Zeit (2.5 Hz)

5.1.2.2 Potenzialanalyse zur Selbstheilung

Die Artefakte der vorherigen Phasen bilden die Grundlage der Potenzialanalyse zur Integration von Eigenschaften der Selbstheilung in ein Intelligentes Technisches System. Das Vorgehen ist in Bild 5-10 zusammengefasst.

Der erste Schritt ist die **Gefährdungsanalyse**. Anhand der Anwendungsszenarien und der Umfeldmodelle wurden mögliche Unfälle und Gefährdungen in einem Workshop ermittelt und anhand ihrer Eintrittswahrscheinlichkeit, Schwere und Häufigkeit bewertet. Das Bild 5-10 zeigt die Ableitung von Unfällen aus den Anwendungsszenarien. Auf Basis der Wirkstruktur wurde die **Komplexitätsanalyse** durchgeführt. Die Analyse ermittelt Kennzahlen mit den der Vernetzungsgrad eines Systemelements ermittelt wird. Schließlich wird der **Wissensgrad** der Systemelemente in Bezug auf die Betriebsdomäne ermittelt. Diese Analysen werden in der **Potenzialanalyse** in einem Portfolio zusammengeführt. Dieses Portfolio ermittelt die Potenziale zur Selbstheilung im System und legt die Leistungsstufe als anzustrebende Normstrategie fest. Die Erkenntnisse werden anschließend in der **systemischen Risikoanalyse** detailliert.

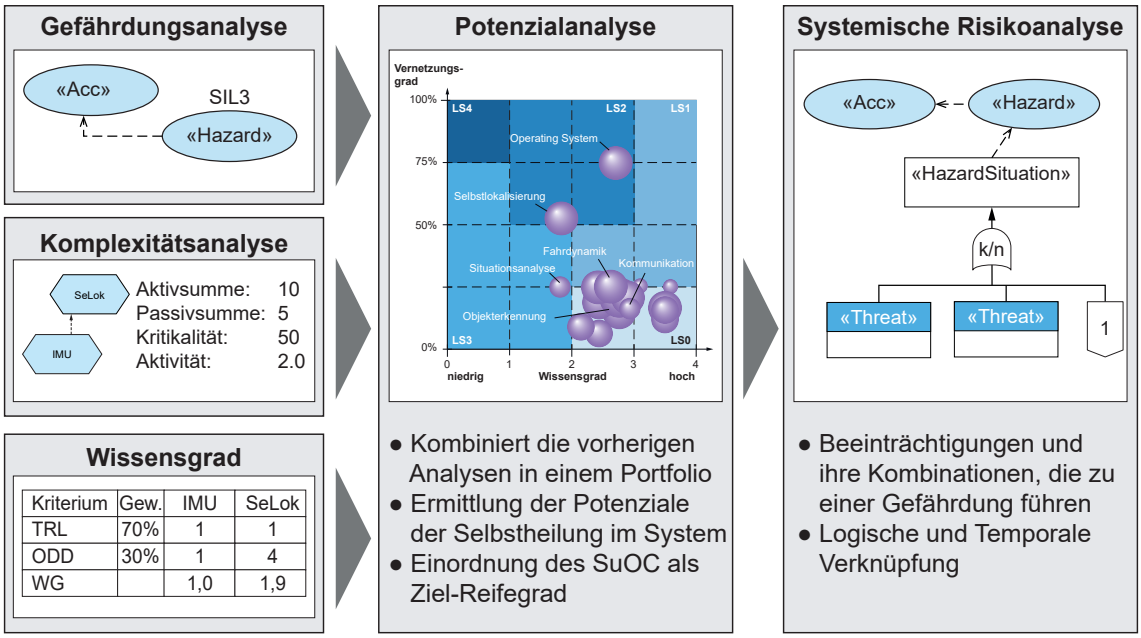


Bild 5-10: Vorgehen der Potenzial- und Risikoanalyse

Aus der Potenzialanalyse ergeben sich Anforderungen an die Sicherheit des Systems. Mit dem Sprachprofil können diese Modellelemente miteinander verknüpft werden. Auf diese Weise ergibt sich eine Durchgängigkeit in den Modelle. Diese Durchgängigkeit von Gefährdungsanalyse, Anforderungen bis zu der Absicherung zeigt Bild 5-11. Die Anforderungen sind testbar formuliert und mit einem Testfall verknüpft.

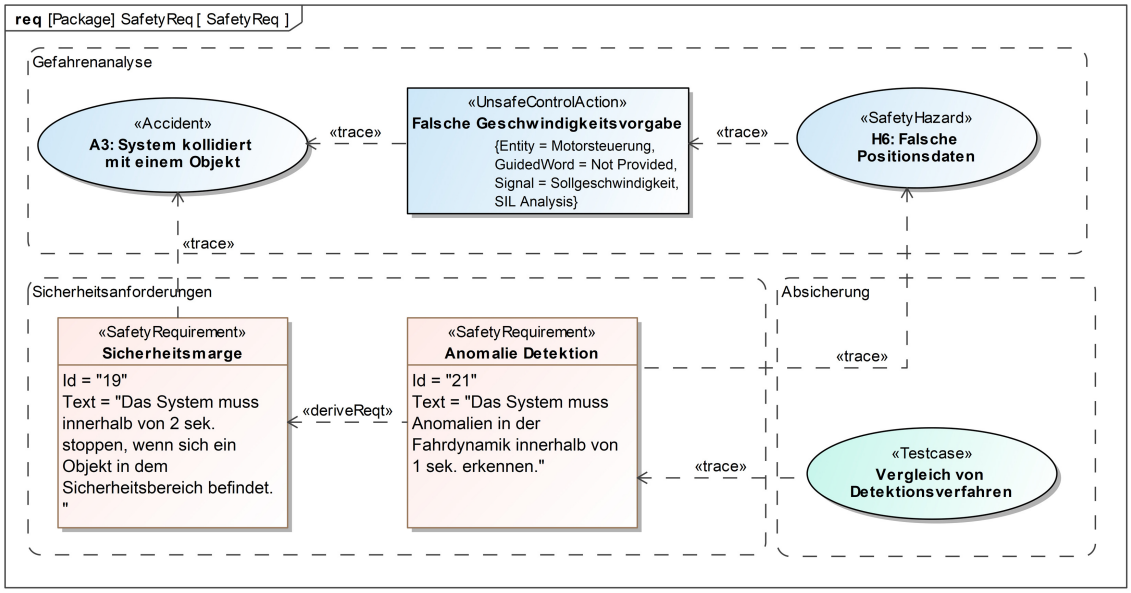


Bild 5-11: Gefahren- und Anforderungsanalyse

Das Bild 5-12 zeigt das Ergebnis der Potenzialanalyse als Portfolio. Ziel ist es, den Wissensgrad zu erhöhen und die Konnektivität sowie das Sicherheits- und Ausfallrisiko im System zu reduzieren. Das Ergebnis zeigt zum Beispiel, dass für das *Operating System*

und für die *Selbstlokalisierung* die Leistungsstufe 2: *Adaptive Selbstheilung mit Online Lernen* anzustreben ist. Im Folgenden werden Selbstheilungseigenschaften für die Selbstlokalisierung umgesetzt.

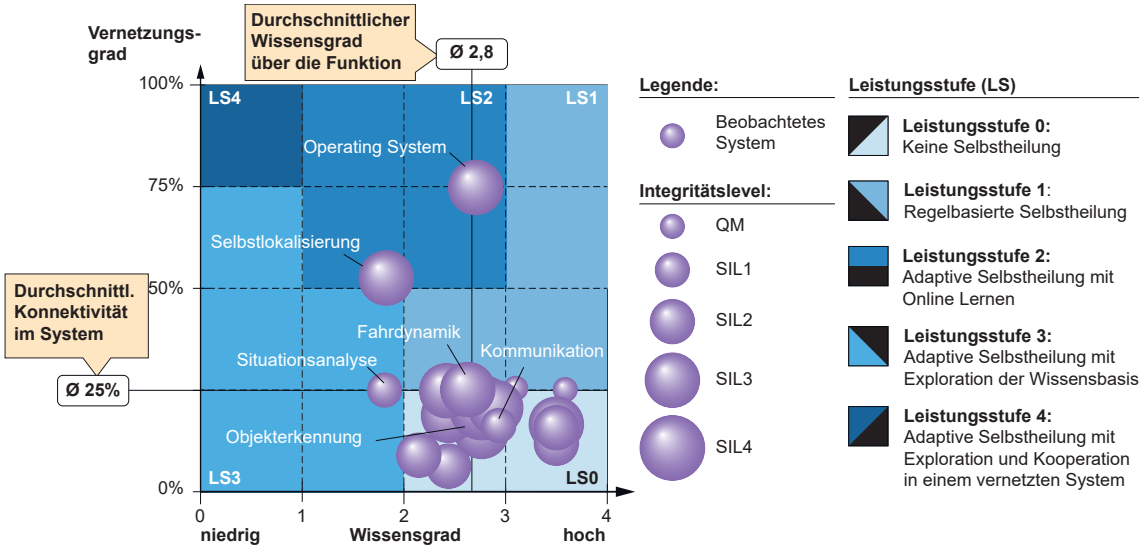


Bild 5-12: Potenziale zur Selbstheilung

5.1.2.3 Systemische Risikoanalyse

Auf Basis der Ergebnisse der Potenzialanalyse werden die systemischen Risiken ermittelt. Der erweiterte dynamische Fehlerbaum beschreibt eine Kombination von Beeinträchtigungen, die zu einem unsicheren Regelungsvorgang im Regelkreis des SuOC und zu einer Gefährdung führen. Das Vorgehen erfolgt deduktiv von der Gefahrensituation bis zu den Beeinträchtigungen.

Zuerst wird das SuOC und dessen Regelkreise definiert. Das Bild 5-13 zeigt im oberen Bereich die Wirkstruktur. Das SuOC zeigt eine Sicht auf den Regelkreis in der Wirkstruktur. Das Beispiel zeigt die Sicht auf die Wirkstruktur der Selbstlokalisierung als SuOC. Zunächst werden die unsicheren Regelungsvorgänge im System identifiziert und anschließend die Gefahrensituationen ermittelt. Die Modellierung des dynamischen Fehlerbaums erfolgt dann integrativ auf Basis der Wirkstruktur und erleichtert so die Identifikation von Beeinträchtigungen. Durch die Sicht können die relevanten Flussbeziehungen und Systemelemente nacheinander analysiert und mögliche auslösende Ereignisse ermittelt werden. Die Analyse wird mit den im SuOC beteiligten Disziplinen durchgeführt. Das Resultat ist ein erweiterter dynamischer Fehlerbaum.

Mit Hilfe des Fehlerbaums werden Verfahren zur *Überwachung* der Beeinträchtigungen und ihre Zustandsgrößen sowie mögliche Aktionen der *Selbstheilung* erarbeitet. Das Lösungswissen aus Kapitel 4.2.3 unterstützt bei der Lösungsfindung. Als weiteres Hilfsmittel wurde ein Kartenset für die Modellierung des Fehlerbaums in einem Workshop entwickelt.

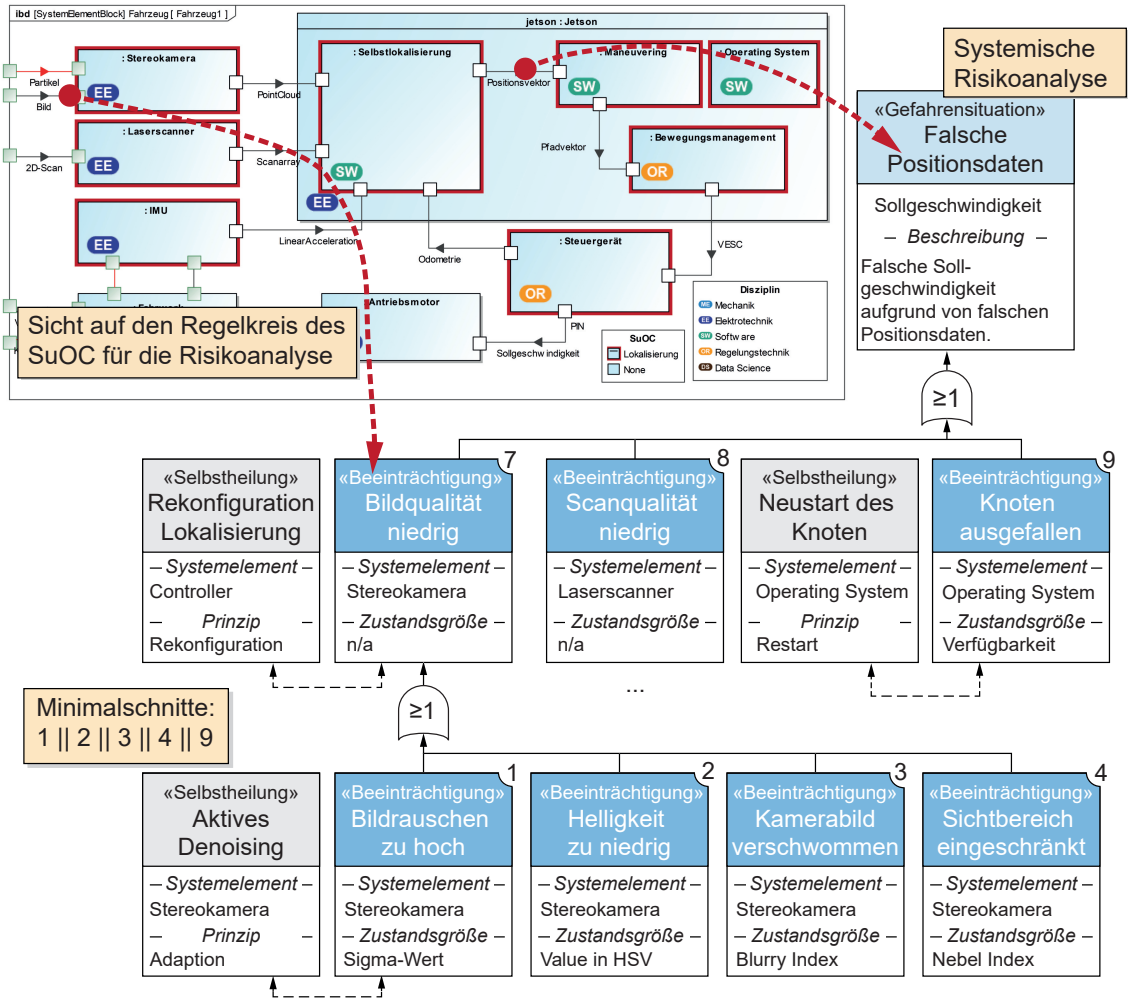


Bild 5-13: Erweiterter Dynamischer Fehlerbaum (Ausschnitt)

Der Dynamische Fehlerbaum bildet einen Ausschnitt der Gefahrensituation *„Falsche Positionsdaten“* ab. Diese können dazu führen, dass das Fahrzeug eine falsche Sollgeschwindigkeit an die Motoren sendet, was zu einer Kollision mit einem Objekt führen kann. Das dargestellte Kausalszenario umfasst sieben bekannte Beeinträchtigungen. Die Beeinträchtigung *„Bildqualität niedrig“* ist durch vier Basisereignisse und ein logisches ODER aggregiert. Für jedes Basisereignis wird eine Zustandsgröße ermittelt, mit dem die Beeinträchtigung überwacht wird. Ebenso werden die Beeinträchtigungen mit einer *Selbstheilung* verknüpft. Beispielsweise wird das Bildrauschen über den Sigma-Wert überwacht. Durch ein *„Aktives Denoising“* als *Selbstheilung* wird das Signal-Noise-Ratio (SNR) im Bild optimiert. Relevant für die Konzipierung des Gesundheitsmonitors sind die Minimalschnitte im Fehlerbaum. Für die Zustandsgrößen werden Signalgeneratoren umgesetzt.

Stehen quantitative Daten wie die Ausfallrate einer Einheit oder die Häufigkeit einer Beeinträchtigung zur Verfügung, kann die Wahrscheinlichkeit für die Gefahrensituation ermittelt werden. Dabei kann der semi-formale dynamischer Fehlerbaum in ein formales Modell überführt werden. Ein offenes Werkzeug zur Modellierung und Analyse dynamischer Fehlerbäume ist DFTCALC [ABB+13].

5.1.2.4 Konzipierung des selbstheilenden Systems

Mit der Potenzialanalyse ist für die *Selbstlokalisierung* eine anzustrebende Leistungsstufe der Selbstheilung festgelegt. In diesem Schritt erfolgt die Konzipierung des selbstheilenden Systems. Grundlage bildet das Lösungskonzept aus Kapitel 4.2 und die zugehörigen Entwurfsaspekte. Das Bild 5-14 visualisiert das Konzept für die O/C-Komponenten. Der Observer überwacht das SuOC. Das Subsystem kommuniziert mit dem Controller. Der Controller führt lokale Aktionen der Selbstheilung auf dem SuOC aus. In kritischen Zuständen wird das Notfallmanagement aktiviert.

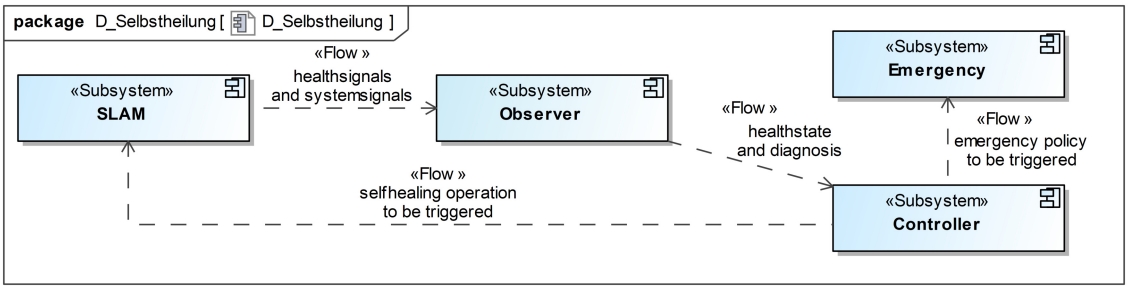


Bild 5-14: Konzipierung der O/C-Komponenten (Hybride Architektur)

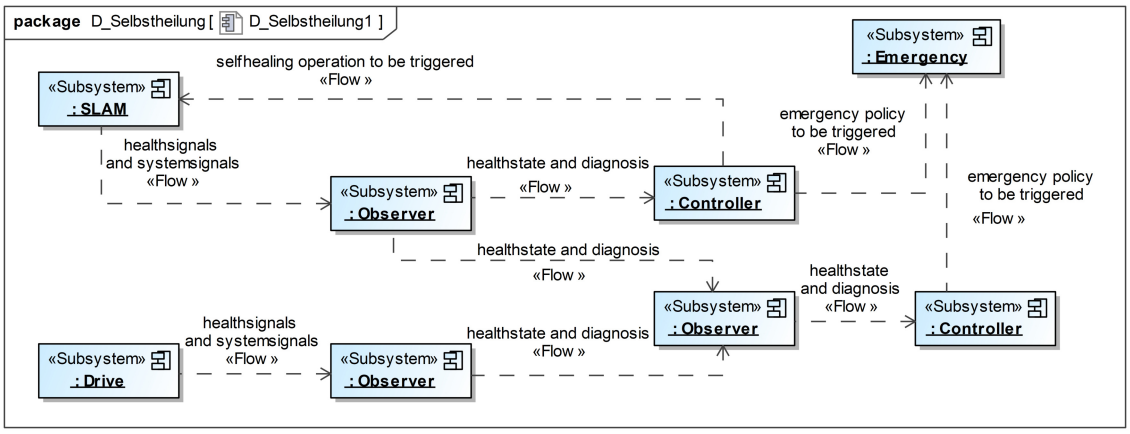


Bild 5-15: Reihenschaltung der O/C-Komponenten (Dezentrale Architektur)

Die O/C-Komponenten sind im selbstheilenden System parallel und in Reihe geschaltet. Dies verdeutlicht Bild 5-15. So überwacht ein weiterer Observer das Antriebssystem. Die Gesundheitszustände der beiden Teilsysteme werden in ein Observer auf Systemebene aggregiert. Die Controller führt dann globale Aktionen aus, z.B. Zielanpassung oder realisiert die Technische Rückfallebene.

Observer

Das Zustandsdiagramm des Observer verdeutlicht sein Verhalten. Nach der *Initalisation* geht das System in den Zustand *Monitoring*. In diesem Zustand erfolgt zunächst ein *Time Check* und anschließend parallel die *Anomaly Detection* und das *Health Monitoring*. Bei einer Veränderung des Gesundheitszustandes erfolgt die Transition in die *Diagnosis*.

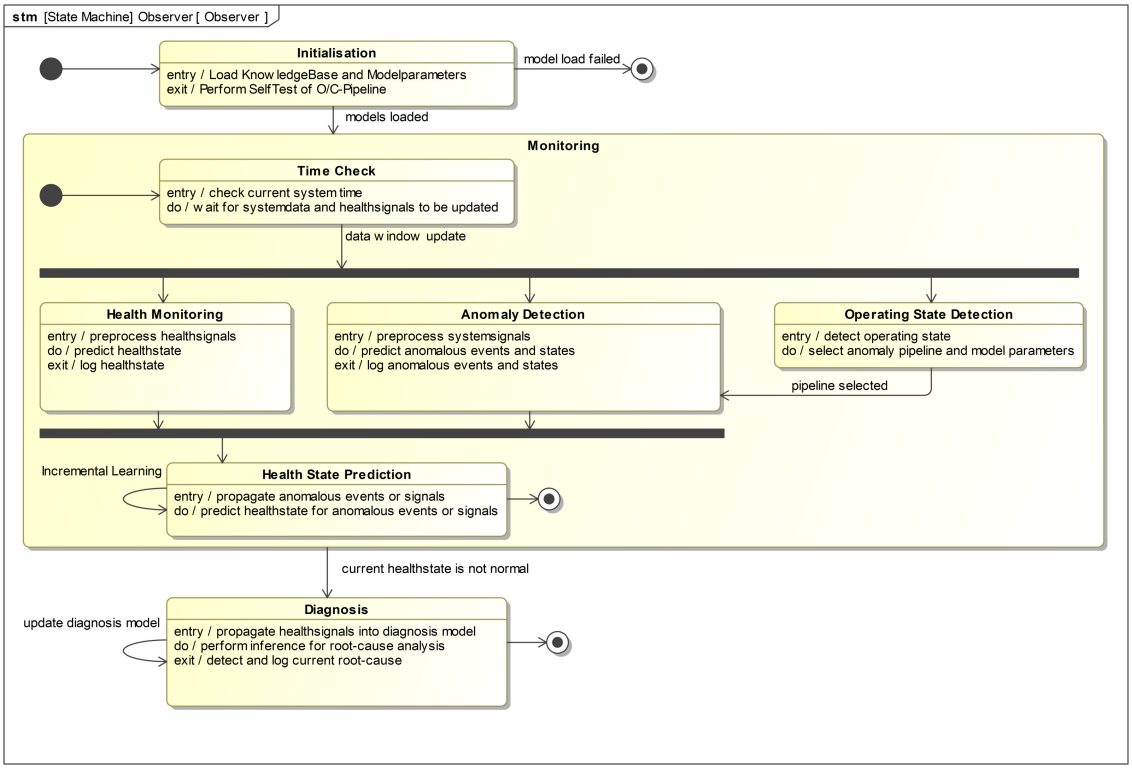


Bild 5-16: Observer Zustandsdiagramm

EA1: Verfahren zur Überwachung und der Diagnose

Gesundheitsmonitoring: Der Gesundheitsmonitor überwacht den Gesundheitszustand und detektiert mögliche Beeinträchtigungen im SuOC: *Selbstlokalisierung*. Ausgangspunkt der Konzipierung ist der dynamische Fehlerbaum und die ermittelten Zustandsgrößen zur Überwachung. Für die einzelnen Zustandsgrößen werden Signalgeneratoren definiert. Diese generieren die Gesundheitssignale für das SuOC. Das Lösungswissen aus Kapitel 4.2.3.1 unterstützen bei der Auswahl von geeigneten Signalgeneratoren. Ein Signalgenerator basiert auf einen Fuzzy-basierten Schwellwertverfahren und strukturiert das Signal in *Normal, Warnung, Kritisch, Fatal*. Die Schwellwerte sind über Zugehörigkeitsfunktionen definiert.

Die Gesundheitssignale werden über den Fuzzy-Pattern-Tree aggregiert. Die Struktur und Hierarchie des Fuzzy-Pattern-Tree wird mit Hilfe des dynamischen Fehlerbaums erarbeitet. Die Aggregation der Gesundheitssignale erfolgt über Fuzzy-Operatoren (siehe Anhang A2.2.1). Der Gesundheitsmonitor in Bild 5-17 umfasst sieben Gesundheitssignale. Die über drei Stufen aggregiert werden. Beispielsweise wird die **Bildqualität** der Stereokamera über die Signalgeneratoren *SNR, Blurry-Index, Helligkeit* und die Verfügbarkeit durch einen *Ping-Test* überwacht.

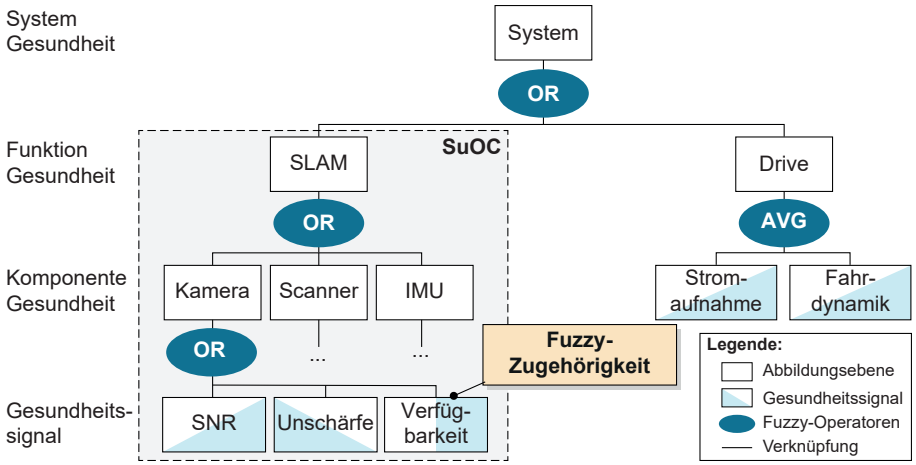
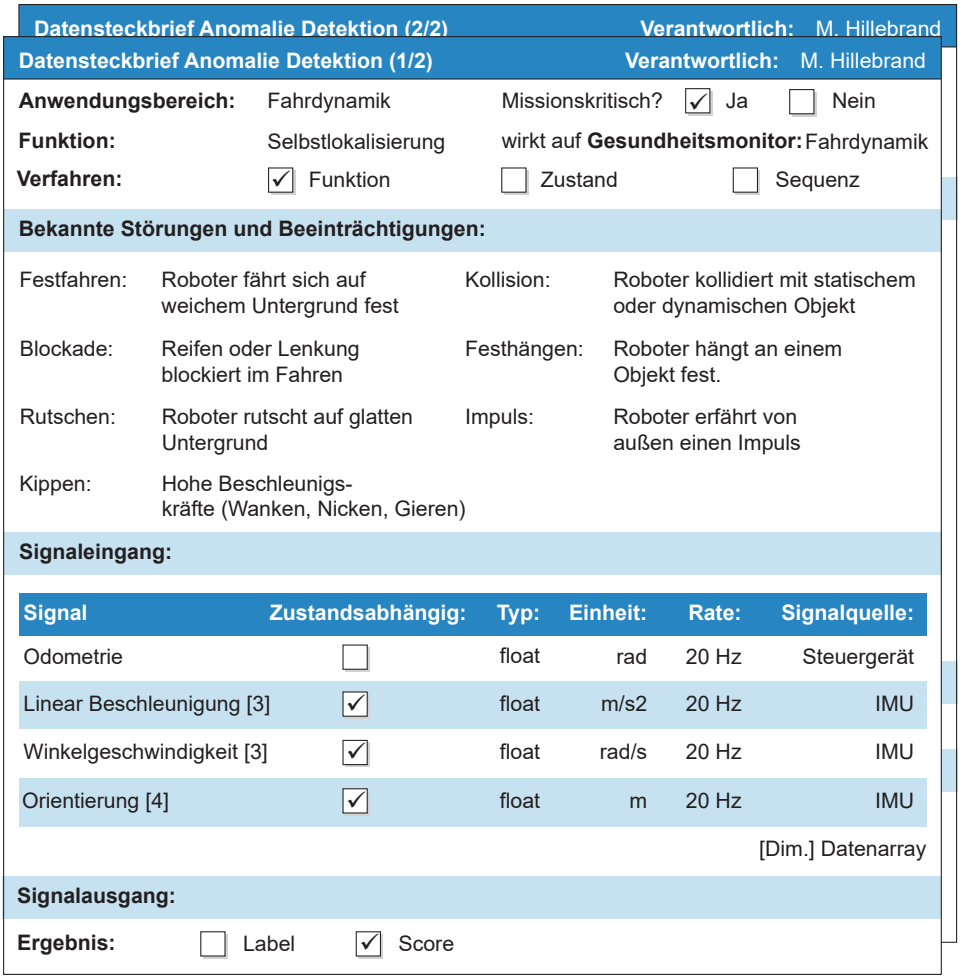


Bild 5-17: Fuzzy-Pattern-Tree (Ausschnitt) für das Anwendungsbeispiel

Anomalie Detektion: Im Fahrzustand des Systems können Bewegungsstörungen auftreten. Zum einen kann sich das System auf dem Untergrund festfahren oder schleudern. Zum anderen kann das System mit einem statischen oder dynamischen Objekt kollidieren, welches sich nicht im Sichtbereich der Umfeldsensorik befindet. Ebenso können sich Kleinteile im Fahrwerk bzw. in der Lenkung befinden oder die Räder blockieren. Zur Auswahl geeigneter Anomalie Detektoren sind die Signaleigenschaften in einem **Datensteckbrief** zusammengefasst (vgl. Bild 5-18). Die relevanten Eingangssignale lassen sich aus der Wirkstruktur des SuOC ableiten. Die Anomalie Detektion soll einen Anomalie Score für ein betrachtetes Zeitfenster liefern. Die *Zielplattform* ist ein Embedded System. Die Anomalie Detektion soll online und semi-überwacht, d.h. durch gelerntes Normalverhalten, erfolgen.



Datensteckbrief Anomalie Detektion (2/2) **Verantwortlich:** M. Hillebrand

Datensteckbrief Anomalie Detektion (1/2) **Verantwortlich:** M. Hillebrand

Anwendungsbereich: Fahrdynamik — Missionskritisch? ☑ Ja ☐ Nein

Funktion: Selbstlokalisierung — wirkt auf **Gesundheitsmonitor:** Fahrdynamik

Verfahren: ☑ Funktion ☐ Zustand ☐ Sequenz

Bekannte Störungen und Beeinträchtigungen:

Festfahren:	Roboter fährt sich auf weichem Untergrund fest	Kollision:	Roboter kollidiert mit statischem oder dynamischen Objekt
Blockade:	Reifen oder Lenkung blockiert im Fahren	Festhängen:	Roboter hängt an einem Objekt fest.
Rutschen:	Roboter rutscht auf glatten Untergrund	Impuls:	Roboter erfährt von außen einen Impuls
Kippen:	Hohe Beschleunigs-kräfte (Wanken, Nicken, Gieren)		

Signaleingang:

Signal	Zustandsabhängig:	Typ:	Einheit:	Rate:	Signalquelle:
Odometrie	☐	float	rad	20 Hz	Steuergerät
Linear Beschleunigung [3]	☑	float	m/s2	20 Hz	IMU
Winkelgeschwindigkeit [3]	☑	float	rad/s	20 Hz	IMU
Orientierung [4]	☑	float	m	20 Hz	IMU

[Dim.] Datenarray

Signalausgang:

Ergebnis: ☐ Label ☑ Score

Bild 5-18: Datensbeschreibung zur Auswahl von Anomalie Detektoren

Betriebszustandsanalyse: Das Normalverhalten des Systems ist abhängig vom Betriebszustand. So unterscheidet sich das normale Signalmuster der Inertiale Messeinheit (IMU) im Zustand *Stehen* vom Zustand *Fahren*. Die Zustandsdetektion erfolgt durch ein Markov-Modell. Der Betriebszustand wird über einen Zustandsvektor repräsentiert.

SuOC
EA2:Selbstheilungsaktionen

In diesem Schritt werden die Aktionen der Selbstheilung konzipiert. Dies sind die Aktionen, die auf dem SuOC durch den *Controller* zustandsabhängig ausgeführt werden. In Kapitel 4.2.3.4 wurde Lösungswissen für die Selbstheilung identifiziert. Diese Prinzipien bilden die Grundlage für die Lösungsfindung. In den Fehlerbaum sind mit den Domänenexperten Aktionen der Selbstheilung erarbeitet und mit den Beeinträchtigungen verknüpft (vgl. Bild 5-13).

Das Bild 5-19 zeigt das Zustandsdiagramm für das betrachtete SuOC. Es zeigt eine redundante Auslegung des SLAM. In dem Zustandsdiagramm sind *Selbsttest* integriert. Ebenso ein *Neustart* in spezifischen Fehlerdiagnosen. Für komplexe Fehlerdiagnosen im Regelkreis des SuOC dient die O/C-Architektur.

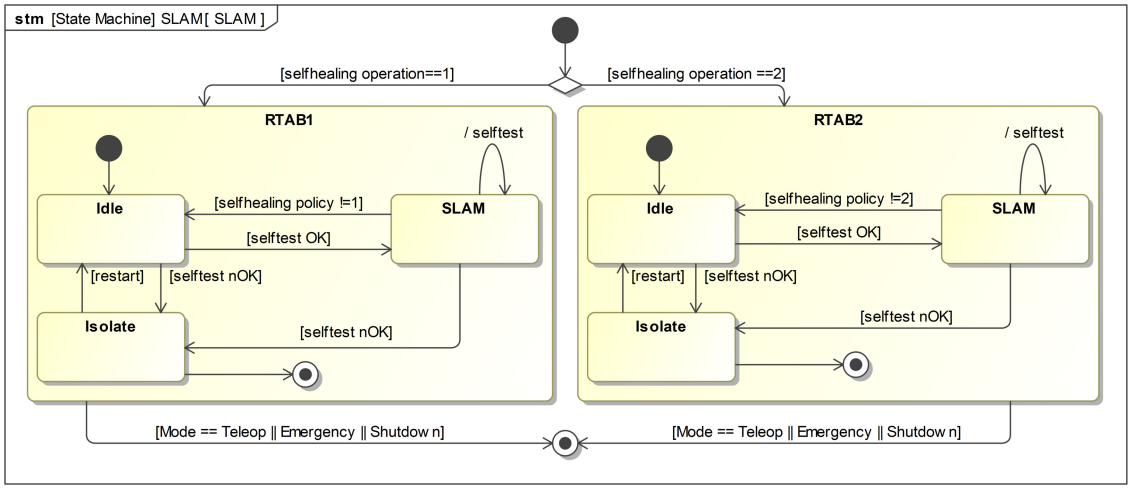


Bild 5-19: Zustandsdiagramm für SuOC SLAM (Ausschnitt)

Aktivitätsdiagramme zeigen die aktiven Konfigurationen. Das Bild 5-20 zeigt das Aktivitätsdiagramm für die RTAB2 Konfiguration. Die Konfiguration zeigt das den Fall für einen beeinträchtigten oder ausgefallenen Laserscanner. Die Aktivität *Scandaten verarbeiten* ist in dieser Konfiguration nicht erforderlich.

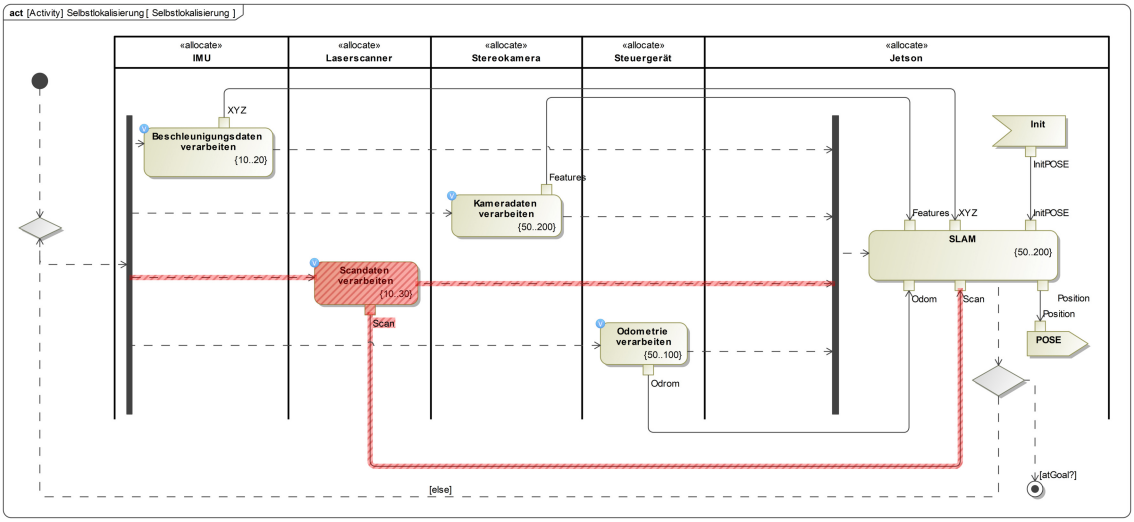


Bild 5-20: Aktivitätsdiagramm für RTAB2 Konfiguration

Controller

Der Controller führt die situationsspezifisch Aktionen der Selbstheilung auf dem SuOC aus. Das Bild 5-21 zeigt das Zustandsdiagramm. Der *Planner* wählt auf Basis des aktuellen Gesundheitszustandes und der Diagnose eine geeignete Policy aus. Bei einer unbekannten oder unsicheren Diagnose sucht die *Exploration* sucht eine geeignete Policy in der Wissensbasis. Der *Scheduler* repräsentiert die Ablaufsteuerung. Die O/C-Architektur lernt inkrementell, d.h. die Strategien werden zur Laufzeit evaluiert.

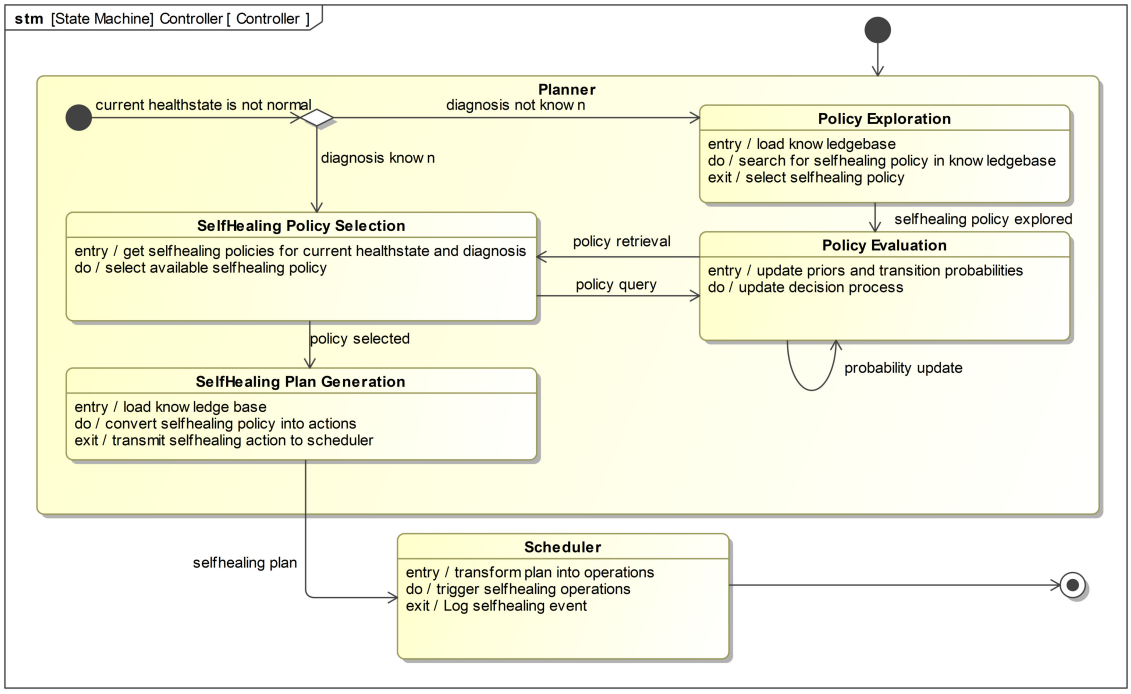


Bild 5-21: Controller Zustandsdiagramm

EA3: Wissensbasis für die Selbstheilung

Das Bild 5-22 zeigt die initiale Wissensbasis. Die Wissensbasis stellt dar, welche Systemelemente für die einzelnen Operation der Selbstheilung erforderlich sind. Der Kontrolltransfer zum Benutzer (Nr. 8) erfordert z.B. die Verfügbarkeit der Teleoperation. Eine Anpassung der Systemziele durch eine geänderte Zielgewichtung ermöglicht eine Optimierung auf die Zuverlässigkeit (Nr. 9) und selektiert Start/ Ziel-Pfade mit wenigen Lenkmanöver (vgl. Situation 4 im Zielsystem Bild 5-6).

#	Name	Kamera : Boolean	2D-Scanner : Boolean	Fake2DScan : Boolean	IMU : Boolean	Odometrie : Boolean	Teleoperation : Boolean	MZO : Boolean	Node : Boolean	Notstop : Boolean
1	EKF-SLAM	false	true	false	true	true	false	false	false	false
2	Emergency	false	false	false	false	false	false	false	false	true
3	Fail-Operational	false	false	false	true	true	false	false	false	false
4	RTAB1	true	true	false	true	true	false	false	false	false
5	RTAB2	true	false	true	true	true	false	false	false	false
6	RTAB3	true	false	false	true	true	false	false	false	false
7	RTAB4	true	true	false	false	false	false	false	false	false
8	Teleop	false	false	false	false	false	true	false	false	false
9	Zieloptimierung	false	false	false	false	false	false	true	false	false

Bild 5-22: Wissensbasis des Systems

EA4: Integration von Evaluation, Adaption und Optimierung

Für die Evaluation und Adaption wird ein Belohnungsfunktion definiert. Die Funktion belohnt Aktionen des Controllers, die zu einer positiven Veränderung des Gesundheitszustandes im SuOC führen. Dabei gilt entsprechend Gleichung 4-3 aus Kap. 4.2.2.4. Selbstheilungsaktionen die zu einer negativen Veränderung des Gesundheitszustandes führen, erhalten eine negativ Belohnung. Ebenso wird ein Prior für die Dirichilet-Verteilung

gewählt, mit dem das apriori-Wissen des Domänenexperten repräsentiert ist.

5.1.3 Entwurfsphase

5.1.3.1 Domänenspezifischer Entwurf

Übergang in den Domänenspezifischen Entwurf
Ausgang für den domänenspezifischen Entwurf bildet der modellbasierte Systementwurf und dessen Prinzipiösung des selbstheilenden Systems. Auf Basis der Wirkstruktur und der Verhaltensdiagramme erfolgt die Detaillierung zum Beispiel in 3D-CAD, Matlab/Simulink oder in dem Robot Operating System (ROS). In der Wirkstruktur sind die Schnittstellen, Ein- und Ausgangsgrößen und die Domänen-Tags spezifiziert. Den Übergang in die Domänen zeigt Bild 5-23.

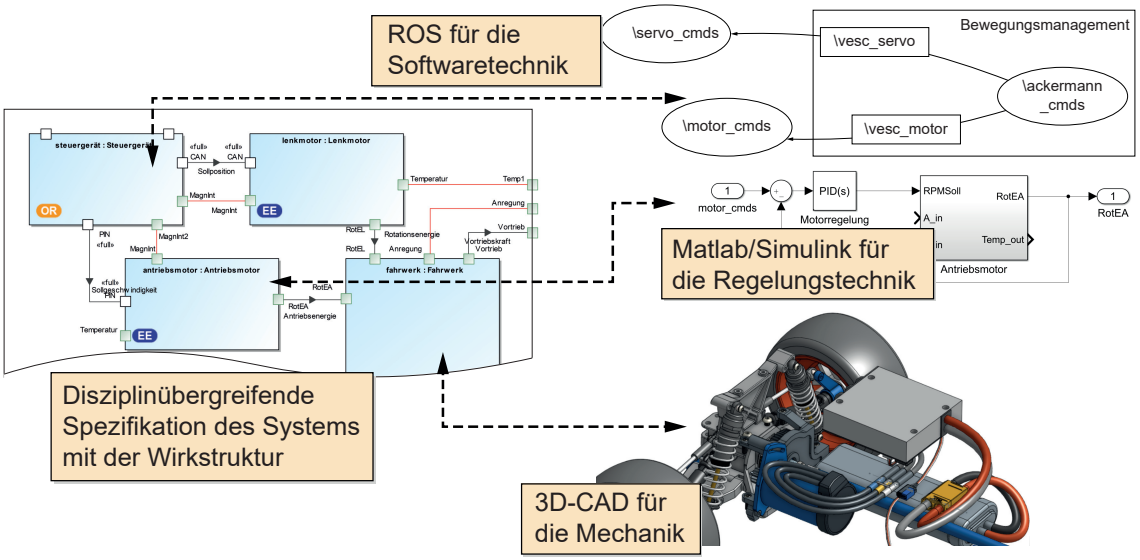


Bild 5-23: Übergang in den domänenspezifischen Entwurf

Im folgenden werden die Struktur, die Gestalt sowie Kommunikation, Datenarchitektur und schließlich die selbstregulierende Informationsverarbeitung erläutert.

Struktur und Gestalt: Das detaillierte Strukturmodell umfasst die Gestalt, die Kinematik und die Dynamik des Systems. Das Strukturmodell wird aus den Partialmodellen Wirkstruktur und Grobgestalt abgeleitet. Das Bild 5-24 visualisiert die Gestalt des mobilen Roboters. Im Modell sind verschiedene Koordinatensysteme festgelegt: Das globale Referenzsystem, ein fahrzeugfestes Koordinatensystem sowie lokale Koordinatensysteme. Die Sensoren sind auf das fahrzeugfeste Koordinatensystem bezogen.

Kommunikation und Daten: Ergebnis dieser Modellierung ist der Kommunikationsgraph (vgl. Bild 5-25): Die Knoten repräsentieren Funktionen, Services oder Aktionen. Die Nachrichten zwischen den Knoten werden über die Kanten repräsentiert. Aus der Datensicht der Wirkstruktur werden die Knoten, Nachrichten und Daten extrahiert. Die Definition einer Nachricht umfasst die Festlegung von Datenart und Datentypen. Die Verknüpfung

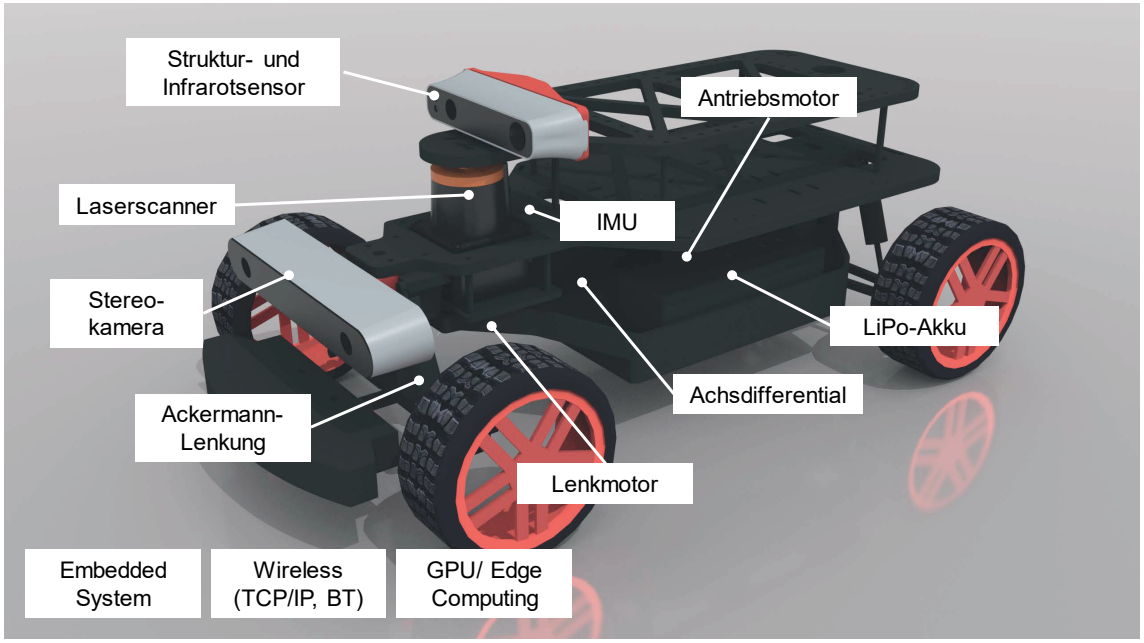


Bild 5-24: Strukturmodell des Mobilen Roboters

von Knoten über die Nachrichten erfolgt über einen Publish/ Subscribe-Mechanismus. Der Graph unterscheidet Hardware-basierte Knoten und Software-basierte Knoten. Ein *Hardware-basierter Knoten* realisiert die Schnittstelle zur (virtuellen) Sensorik und Aktorik. Im Zuge der Virtuellen Inbetriebnahme werden diese Knoten durch die Treiber der realen Sensorik und Aktuatorik substituiert. *Software-basierte Knoten* realisieren Funktionen der selbstregulierenden Informationsverarbeitung. Diese werden auf die Zielplattform wie etwa das Steuergerät migriert.

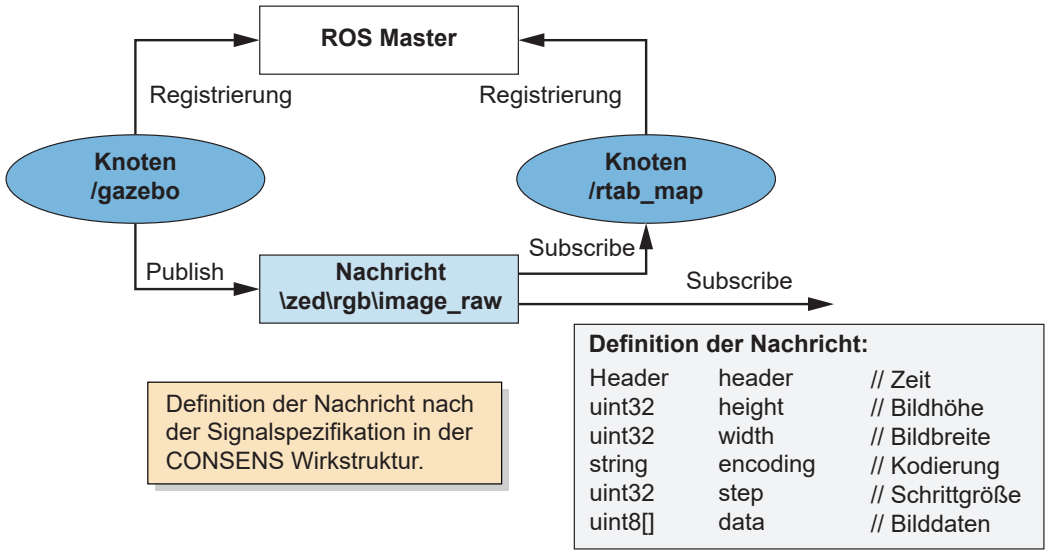


Bild 5-25: Kommunikation und Graph in ROS

Selbstregulierende Informationsverarbeitung: Die Funktionen der selbstregulierenden Informationsverarbeitung sind entsprechend Bild 5-5 realisiert. Die Kartierung ist eine

interne Repräsentation der Umgebung und wird zur Laufzeit auf Basis der Sensordaten erstellt (z.B. Stereokamera und Laserscanner). Diese Repräsentation nutzt das System zur Lokalisierung und zur Navigation in der Umgebung. Umgesetzt wurde die Kartierung durch Real-Time Appearance-Based (RTAB)-Mapping [LM18]. Das Bild 5-26 visualisiert die 3D-Kartierung des Roboters.

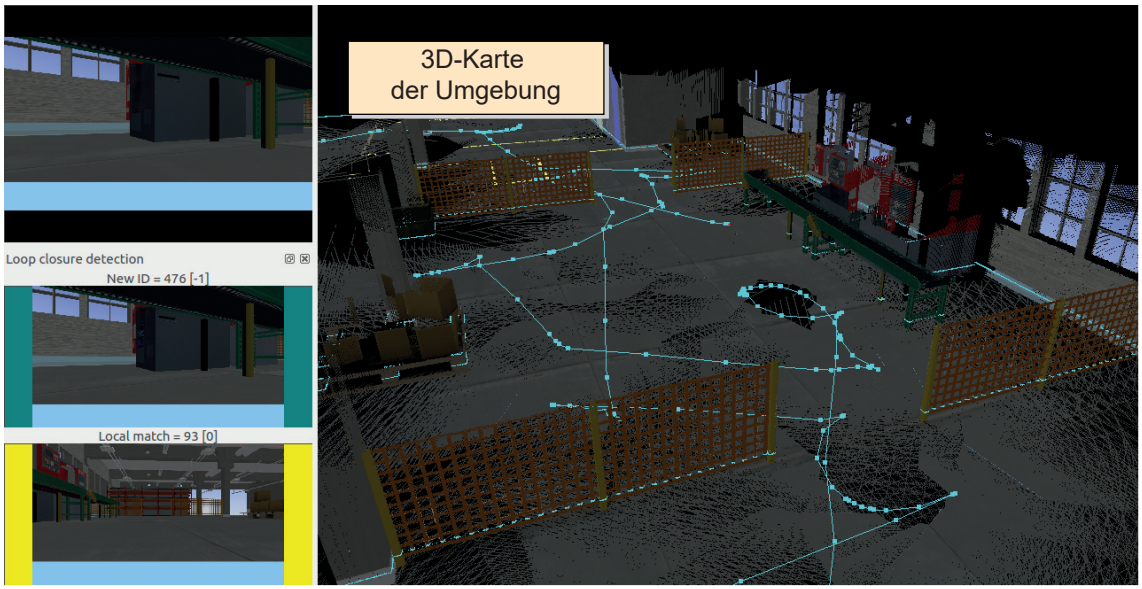


Bild 5-26: 3D-Kartierung der Umgebung mit RTAB-Mapping [LM18]

5.1.3.2 Entwurf des selbstheilenden Systems

Datengetriebene Modellierung: Das Bild 5-27 zeigt das Vorgehen zur datengetriebenen Modellierung selbstheilender Systeme. Die Modellbildung der O/C-Komponenten aus dem Framework erfolgt auf der Basis der KI4AS-Modellbibliothek sowie Daten aus dem Scenario-in-the-Loop (SciL)-Testing. Die Domänenbeschreibung und die Anforderungen der Komponenten sind das Ergebnis der Konzeptphase. Ergebnis der Modellbildung ist das Modell. Dieses wird im Anschluss entsprechend des Frameworks in die O/C-Pipline integriert. Ergebnis dieser Phase ist ein spezifischer Reifegrad des selbstheilenden Systems, z.B. Labormuster, Funktionsmuster. Der KI4AS-Code und die Modellbibliothek sind unter Apache Lizenz veröffentlicht und in Sphinx dokumentiert.

Observer: Anhand des Datenbeschreibung in Bild 5-18 und der Auswahlsystematik erfolgt eine Vorauswahl geeigneter Detektoren aus der KI4AS-Modellbibliothek. Dies sind *K-Nearest Neighbors, Autoencoder, One-Class SVM, Isolation Forest und LSTM-ED* (vgl. Kap. A2.2.2). Diese Detektoren sind in Python und Bibliotheken wie Scikit-Learn [PVG+11], Tensorflow [AAB+15] und Keras [Cho+15] umgesetzt und in die Observer Pipeline eingebunden [KEH20]. Für das Gesundheitsmonitoring sind Signalgeneratoren umgesetzt. Mit Hilfe von Domänenwissen sind die graduellen Gesundheitszustände definiert und anhand von SciL überprüft.

Controller: Das Verstärkende Lernen des Controllers ist mit OpenAI [ABC+18] realisiert

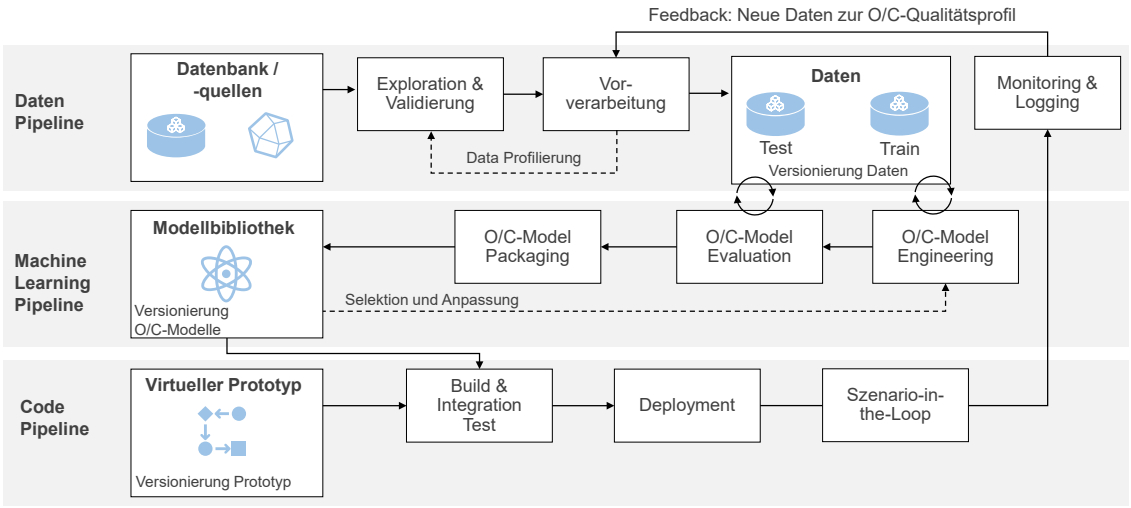


Bild 5-27: Datengetriebene Modellierung selbstheilender Systeme

[SH18]. Die Wissensbasis umfasst eine semantische Beschreibung der Sensorik, Aktorik, der Software und Wissens über die Ein- und Ausgänge, sowie das Beziehungswissen. Die Wissensbasis für das Anwendungsbeispiel visualisiert Bild 5-28. Die Modellierung der Ontologie erfolgt mit Protégé [Mus15] auf Grundlage der Systemspezifikation und der Konzipierung.

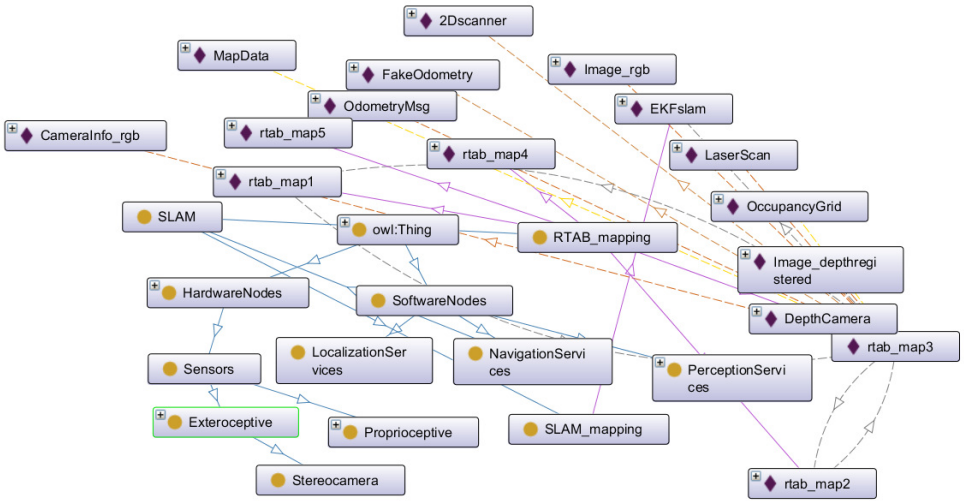


Bild 5-28: Semantische Domänenwissen des mobilen Roboters

O/C-Pipeline: Die O/C-Softwarekomponenten sind in einer Machine-Learning-Pipeline zusammengefügt und werden in den virtuellen Prototypen integriert. Die Inbetriebnahme erfolgt auf einem NVIDIA Jetson TX2 [Nvi19-ol], mit NVIDIA Jetpack SDK und ROS [OSRF19-olb] sowie der InfluxDB [Inf19-ol].

5.1.4 Integrationsphase

Verifikation und Validierung

Für die Szenario-basierte Analyse und die Eigenschaftsabsicherung wird das SciL-Testing mit dem Virtuellen Testbed (vgl. Kap. 4.5) durchgeführt. Das gewählte Betriebszenario ist eine autonome Transportaufgabe in einer offenen Industrieumgebung. Die virtuellen Szenarien leiten sich aus diesem Betriebsszenario ab und sind im Absicherungskonzept definiert (siehe Kap. 5.1.5.1).

Für die Entwicklung der Algorithmen wurden Datensätze mit ca. 41 Features in verschiedene Szenarien generiert. Die Daten sind mit etwa 10 Hz aufgenommen und synchronisiert. In einer Vorverarbeitung erfolgt die Interpolation zwischen Datenpunkten und die Bereinigung von fehlenden Datenpunkten. Der Datensatz zum Training umfasst 56 Min. Simulationszeit mit ca. 367.000 Beobachtungen. Der Datensatz zum Testen umfasst ca. 282.000 Beobachtungen mit einer Simulationszeit von 46 Min.

Observer: Das Bild 5-29 visualisiert die Anomalie Detektion im Observer am Beispiel der Radblockade. Für die Anomalie Detektion wurden verschiedene Verfahren untersucht und anhand der gewählten Kennzahlen bewertet.

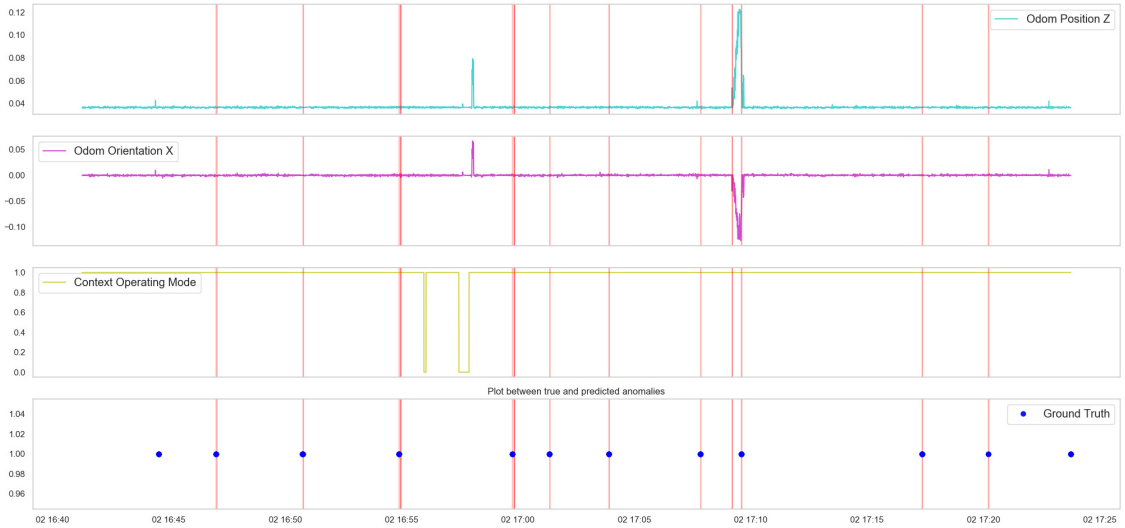


Bild 5-29: Anomalie Detektion am Beispiel Radblockade

Die Tabelle 5-1 zeigt die Ergebnisse in der Übersicht. Der Autoencoder besitzt in den gewählten Testdaten den höchsten F1-Wert und MCC. Hinsichtlich der Inferenzzeit ist die One-Class-SVM am schnellsten.

Tabelle 5-1: Vergleich der Detektoren am Beispiel der Radblockade

Detektor	Training	Inferenz	Precision	Recall	F1-Wert	MCC
Autoencoder	7.100s	0.257s	**0.91**	0.83	**0.87**	**0.86**
K-Nearest Neighbour	**0.005s**	37.000s	0.76	0.82	0.79	0.78
One-Class SVM	1.270s	**0.120s**	0.80	0.83	0.80	0.79
Isolation Forest	2.010s	1.260s	0.32	0.84	0.46	0.06
LSTM-ED	n/a	3.800s	0.565	**1.00**	0.722	0.75

Die Lokalisierung und Diagnose erfolgt über den Fuzzy-Pattern-Tree. Dieser Ansatz wurde im Projekt KI4AS entwickelt. Ausgehend von dem Gesundheitszustand des Systems wird die beeinträchtigte Komponente im System ermittelt und die Diagnose durchgeführt. Neben dem Gesundheitssignal wird eine Konfidenz für die Diagnose ermittelt.

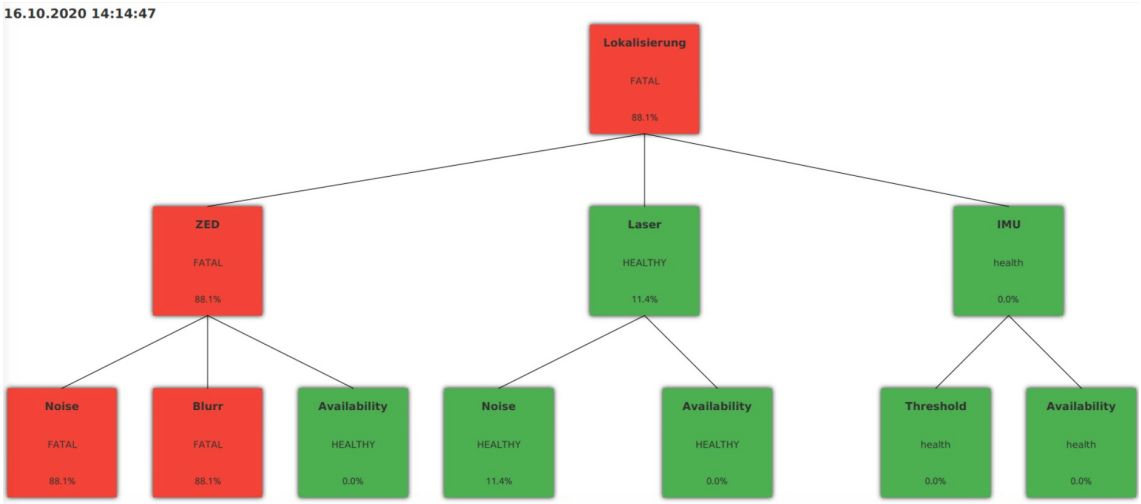
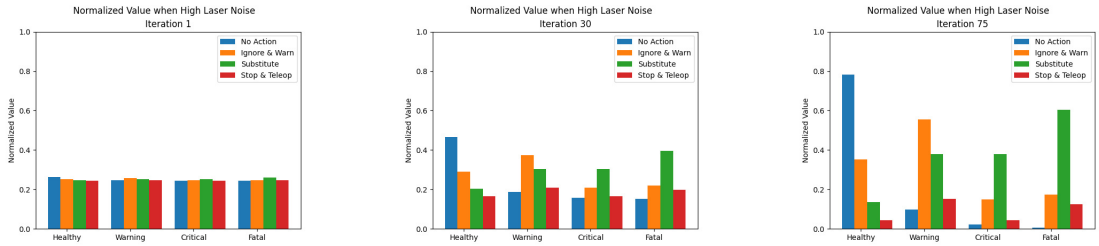


Bild 5-30: FPT-basierte Diagnose

Auf Basis der Überwachung und Diagnose wählt der Controller eine geeignete Reaktionsstrategie. Mit dem gewählten Ansatz lernt der Controller iterativ und inkrementell.

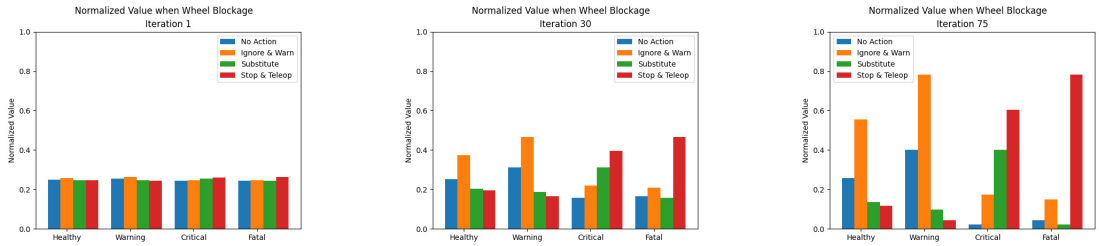
Controller: Die nachfolgenden Bilder zeigen das inkrementelle Lernen des Controllers für verschiedene Fehlerdiagnosen. Das Bild 5-31 zeigt das Fehlerszenario für SNR im Laserscanner. Für die Gesundheitszustand *„Kritisch“* lernt das System die Rekonfiguration und Substitution der Laserscanner, durch zum Beispiel die Stereokamera. Dadurch wird die Selbstlokalisierung trotz Ausfall oder Beeinträchtigung des Sensors aufrechterhalten.

Das Bild 5-32 zeigt das Lernen einer dominanten Selbstheilungsstrategie bei einer Blockade der Räder. In diesem Fall lernt der Controller zu stoppen oder die Kontrolle an den Nutzer zu übergeben.



(a) Iteration 1 (b) Iteration 30 (c) Iteration 75

Bild 5-31: Diagnose: SNR im Laserscanner



(a) Iteration 1 (b) Iteration 30 (c) Iteration 75

Bild 5-32: Diagnose: Blockade der Räder

Der Vergleich verdeutlicht: Das System lernt unterschiedliche Selbstheilungsstrategien für unterschiedliche Gesundheitszustände und Fehlerdiagnosen.

5.1.5 Unterstützende Aktivitäten

5.1.5.1 Definition der Absicherungsstrategie

Das Absicherungskonzept umfasst die Strategie der Absicherung sowie die Testkonzeption entsprechend der Partialmodelle aus Kap. 4.3.2. Das Absicherungskonzept definiert die Testfälle und ordnet diese den Stufen der Absicherung zu. Das Bild 5-33 zeigt eine Reihe von Testfällen für das Scenario-in-the-Loop-Testing im Virtuellen Testbed. Die Testfälle definieren den Funktionsumfang des Virtuellen Testbeds.

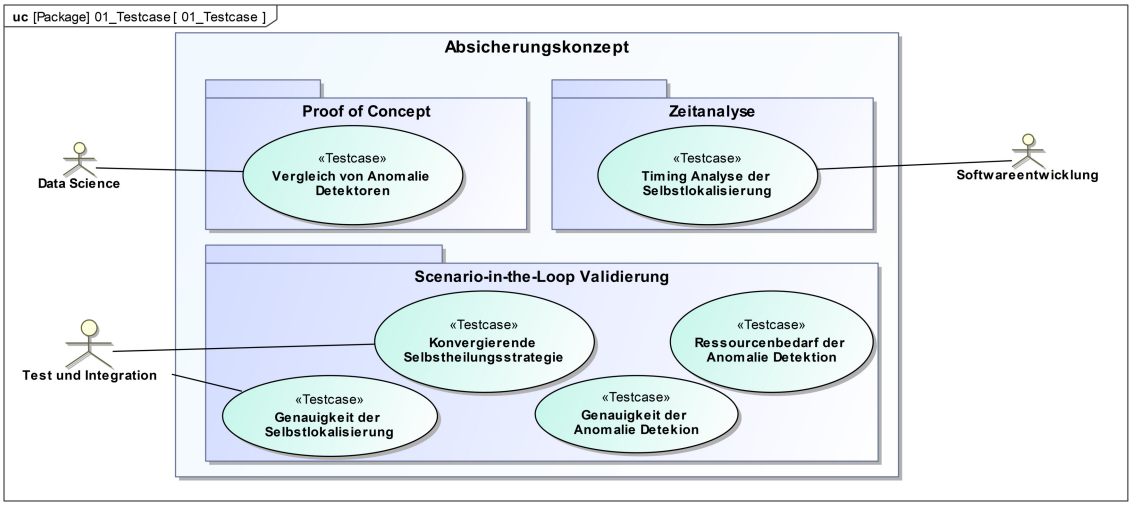


Bild 5-33: Auswahl von Testfällen für das Scenario-in-the-Loop im Virtuellen Testbed

Die Überprüfung eines Testfalls erfolgt in einem Szenario. Das Szenario wird über ein Szenariomodell beschrieben. Das Szenariomodell repräsentiert eine konsistente Kombination von Szenarioelementen aus dem statistischen Szenarioprofil. Das Bild 5-34 visualisiert einen Ausschnitt aus dem Szenarioprofil. In Summe wurden 71 Szenarioelemente in 20 Szenariogruppen identifiziert. Aus diesem Szenarioprofil werden konsistente Szenarien für die Absicherung abgeleitet und mit dem Testfall verknüpft.

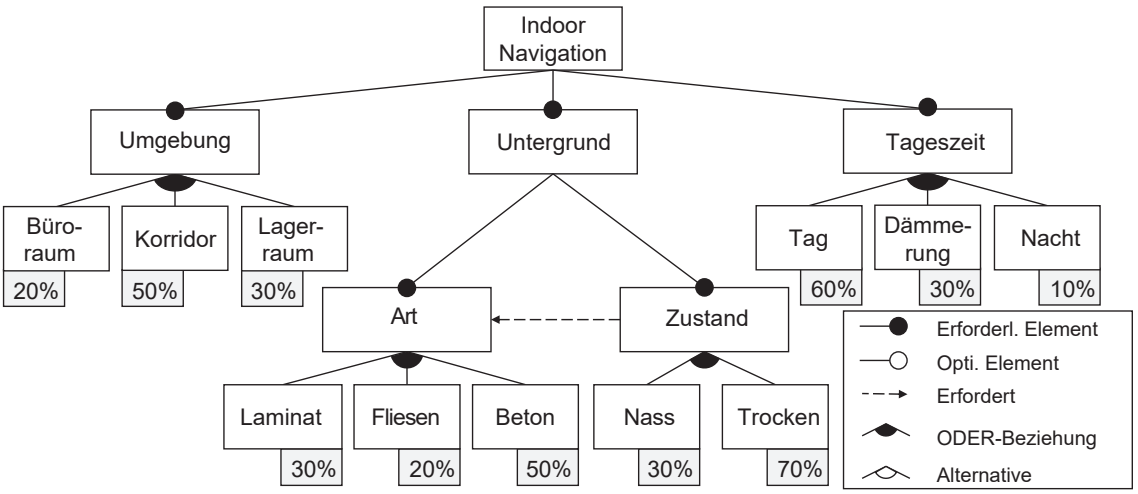


Bild 5-34: Statistisches Szenarioprofil für Scenario-in-the-Loop

Die statistische Verteilung gibt Auskunft über die Betriebsbedingungen des mobilen Roboters. Das Profil zeigt z.B. dass der Roboter überwiegend auf trockenem Beton und unter Tageslicht fährt. Hinsichtlich der Testabdeckung sind diese Profile repräsentativ, z.B. für das Training maschineller Lernverfahren.

5.1.5.2 Entwicklung der Simulationsverfahren

In der entwickelten Umgebung werden Beeinträchtigungen manuell oder automatisiert durch das Monkey Testsystem generiert. Der manuelle Ansatz ermöglicht die Auswahl und Manipulation spezifischer Parameter. Der automatisierte Ansatz dient der Testautomatisierung.

Über die dynamische Rekonfiguration können Modellparameter zur Simulationszeit manuell an der Sensorik, am Grundsystem, der Umgebung oder der Aktorik verändert werden. Das Bild 5-35 zeigt eine Übersicht der Komponenten und ihrer Parameter sowie den Parameterraum für den Rollwiderstand und für das Fahrwerk. Der Rollwiderstand kann für jedes Rad verändert werden. Auf diese Weise werden Beeinträchtigungen durch eine Veränderung der Störgrößen in der Regelstrecke des Systems simuliert.

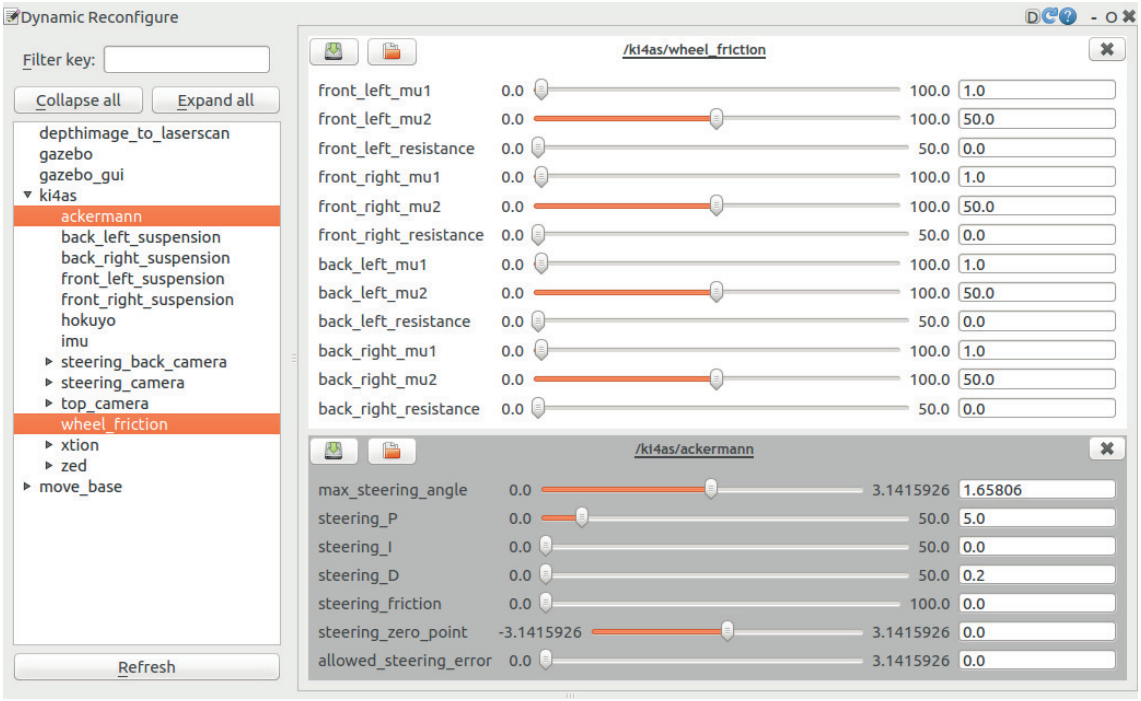


Bild 5-35: Manuelle Simulationssteuerung

Das Monkey Testsystem operiert in dem Parameterraum des Virtuellen Testbeds. Das Bild 5-36 zeigt die Anwendung vom Monkey Testsystem am Beispiel des **Value Monkeys**. Die Aktivitäten werden über die Konsole ausgegeben. In dem ersten Schritt wird ein Seed generiert. Anschließend wird eine zufällige Strategie durch den Monkey erstellt und dem Seed zugewiesen. Die Strategie umfasst die betreffende Komponente, die Intensität, die Dauer und die Art der Änderung. In dem Beispiel wird ein aufsteigender Offset auf den Messwerten des Laserscanners addiert. Anschließend wird der Offset zurückgesetzt. Das Logging dient zur Datenannotation für das maschinelle Lernen. Mit dem Seed wird die Strategie reproduzierbar im Virtuellen Testbed ausgeführt.

```
/home/hlb01/catkin_ws/src/Mobile_Robot/racecar/monkey_testing/launch/value_monkey.launch http://localhost:11
hlb01@pc-eng-hp-ki4as:~$ roslaunch monkey_testing value_monkey.launch
[INFO] [1559557201.918610, 0.000000]: initialize Value Monkey
[INFO] [1559557201.920068, 0.000000]: Seed value has not been set. Generating a new seed!
[INFO] [1559557201.922394, 0.000000]: seed value: 756
[INFO] [1559557201.923462, 0.000000]: component selected: laser_offset
[INFO] [1559557201.924113, 0.000000]: intensity selected: medium
[INFO] [1559557201.924893, 0.000000]: duration selected: 2
[INFO] [1559557201.926378, 0.000000]: Type of Offset: Increasing
[INFO] [1559557201.926633, 0.000000]: changing values
[INFO] [1559557202.057702, 874.505000]: changing back
```

Bild 5-36: Umsetzung des Monkey Testsystem

Das Bild 5-37 zeigt die Aktivität von einem Monkey auf den Bilddaten der Stereokamera. Die Aufgabe des Monkeys ist es, Unschärfe in den Bildern zu erzeugen und somit z.B. die Objekterkennung und die Kartierung im System gezielt zu beeinträchtigen.

Bild 5-37: Unschärfe auf Kameradaten

5.1.5.3 Entwicklung des Virtuelles Testbeds

Das Virtuelle Testbed wird auf der Basis des Absicherungskonzeptes umgesetzt und orientiert sich an dem Framework (vgl. Kap. 4.5). Die 3D-Modellierung für das Virtuelle Testbed erfolgt mit dem Gazebo Simulator [OSRF19-ola]. Gazebo stellt die Unified Robotic Description Format (URDF) und XML Macro Language (XARCO) als Beschreibungsformat bereit. Für den Datenbank-Layer wurde ein Datenkollektor als Schnittstelle zum Robot Operating System [OSRF19-olb] implementiert. Der Agent schreibt die Daten in die InfluxDB [Inf19-ol]. Diese Plattform kann ebenfalls im Betrieb des Systems eingesetzt werden.

Die Plattform Virtuelles Testbed ist in Bild 5-38 dargestellt. Die Bestandteile der virtuellen Umgebung werden durch eine konsistente Kombination von Szenarioelementen aus dem Szenariobaum und ihrer statistischen Verteilung im Szenarioprofil definiert.

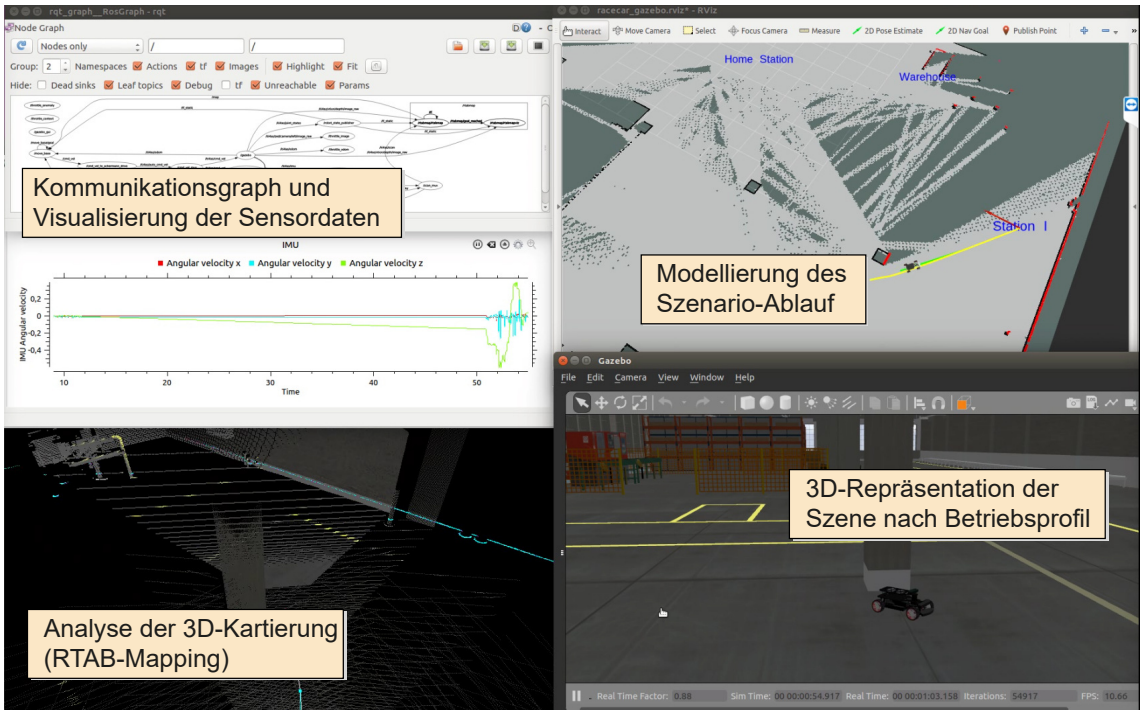


Bild 5-38: Umsetzung des Virtuellen Szenarios im Virtuellen Testbed

Das Virtuelle Testbed dient zum SciL-Testing und zum Hardare-in-the-Loop (HiL)-Testing. In den Szenarien werden Daten für das maschinelle Lernen generiert, mit denen die Verfahren trainiert und getestet werden können. Ebenso ist ein Vergleich verschiedener Verfahren (z.B. zur Anomalie Detektion) möglich. Das Werkzeugkonzept unterstützt den sukzessiven Übergang in die Realität durch X-in-the-Loop (XiL)-Techniken. Das Bild 5-39 zeigt den physischen Aufbau des Roboters und dessen Erprobung in Feldtests. Entsprechende Algorithmen können weiter untersucht und optimiert werden.

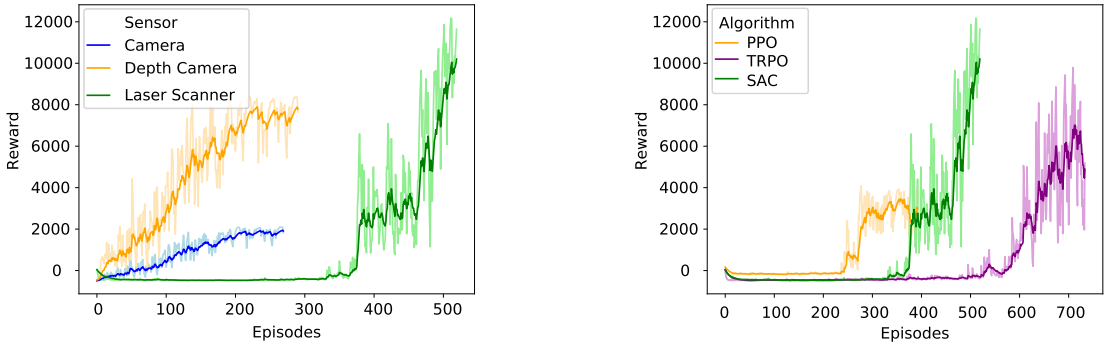
(a) Vorderansicht *(b) Feldtest*

Bild 5-39: Erprobung durch Feldtests

Robustheit durch DeepRL

Die entwickelte Umgebung unterstützt weitere Experimente: Das Virtuelle Testbed kann zum Training und Testen von Ansätzen aus dem Deep Reinforcement Learning (DeepRL)

für die autonome Navigation eingesetzt werden. Das Training erfolgt in der virtuellen Umgebung und unter verschiedenen Betriebsbedingungen wie etwa Sensorrauschen oder Oberflächen mit unterschiedlichen Reibkoeffizienten. So können bereits in der frühen Phase der Entwicklung verschiedene Sensorkonzepte untersucht werden [HLD20]. Das Bild 5-40a zeigt die Belohnungsfunktionen verschiedener Sensorkonzepte wie Stereokamera, Tiefenkamera und Laserscanner für die autonome Navigation. Das Bild 5-40b zeigt die Belohnung der verschiedenen Algorithmen des verstärkenden Lernens. Der Vergleich dieser Ergebnisse unterstützt bei der Auswahl und Auslegung der Sensorik und der Algorithmen.



(a) Sensorauswahl *(b) Algorithmen*

Bild 5-40: DeepRL Training und Test

5.1.6 Erkenntnisse der Anwendung

In diesem Kapitel wurde die Entwicklungssystematik und ihre Bestandteile an einem Anwendungsbeispiel demonstriert. Die Simulation selbstheilender Systeme im virtuellen Testbed und die Erprobung zeigen, dass mit dem gewählten Ansatz die Resilienz des Systems gesteigert werden kann. Insbesondere überwacht das System seinen Gesundheitszustand, führt eine Diagnose möglicher Ursachen durch und rekonfiguriert bei einer Beeinträchtigung eines Sensors oder eine Softwarekomponente selbstständig. Dabei lernt das System für verschiedene Arten von Beeinträchtigungen geeignete Selbstheilungsstrategien. Das Anwendungsbeispiel zeigt somit, dass durch die Integration von Eigenschaften der Selbstheilung die Resilienz auf Systemebene zunimmt. Trotz Beeinträchtigung ist das selbstheilende System funktionsfähig und nimmt nach einem Ausfall wieder eine funktionsfähige Konfiguration an. Dabei lernt das System aus vergangenen Reaktionen und entwickelt sich innerhalb vorgegebener Randbedingungen weiter.

Die Modelle können durch eine virtuelle Inbetriebnahme und XiL-Techniken sukzessiv auf das physische System übertragen und in der physischen Welt erprobt werden. Die Inbetriebnahme zeigt, dass der physische Roboter eine vorgegebene unbekannte Route fahren und den statischen und dynamischen Hindernissen ausweichen kann. Zudem agiert das System resilient gegenüber Sensorrauschen und unterschiedlichen Reibwerten der Oberfläche.

Die Virtuelle Umgebung bietet ein schnelleres Training und Testen der Algorithmen. Anomalien können sicher in der virtuellen Welt erzeugt und das Systemverhalten auf seine Resilienz untersucht werden. Die Parameteroptimierung unterstützt zudem bei der Auslegung des Systems. Das Virtuelle Testbed kann zudem für weitere Experimente und Analysen genutzt werden.

5.2 Bewertung der Arbeit an den Anforderungen

Abschließend wird die erarbeitete *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* anhand der Anforderungen aus Kapitel 2.6 bewertet. Zu diesem Zweck wird für jede Anforderung knapp erläutert, inwiefern sie durch die Systematik erfüllt wird. Bild 5-41 zeigt die Erfüllung der Anforderungen und verortet sie in die Bestandteile der Systematik.

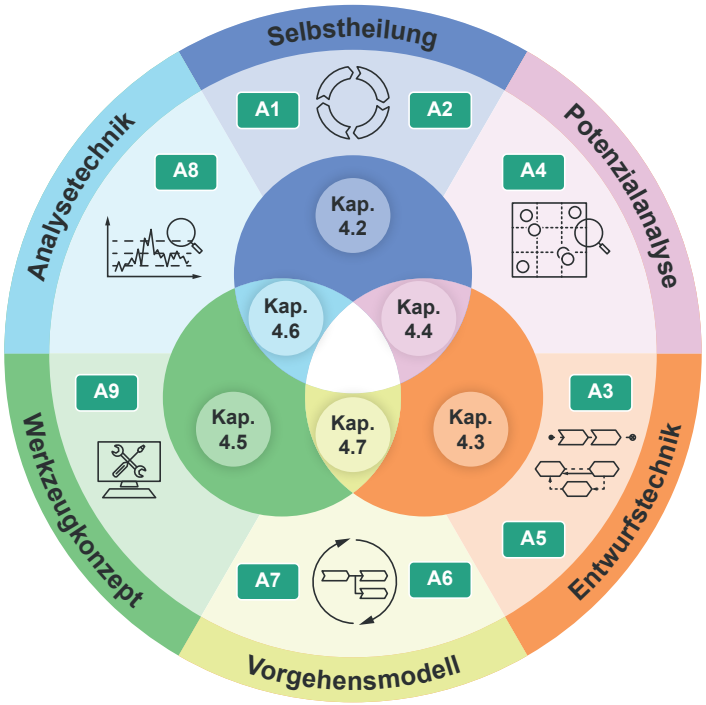


Bild 5-41: Erfüllung der Anforderungen durch die Systematik

A1) Ganzheitliches Selbstheilungskonzept: Das entwickelte *Selbstheilungskonzept* (vgl. Kap. 4.2) ist ein allgemeingültiges Framework. Das Framework umfasst die Eigenschaften der Selbstheilung aus Kap. 2.3.4 und ist um die Anomalie Detektion (vgl. Kap. 2.3.5) erweitert. Die Bausteine der Referenzarchitektur und die Flussbeziehungen untereinander bilden den Aufbau und die prinzipielle Wirkungsweise der Selbstheilung in einem Intelligentem Technischen System ab. Das Konzept berücksichtigt verschiedene Leistungsstufen und begleitende Entwurfsaspekte selbstheilender Systeme. Das Selbstheilungskonzept beschreibt die Integration in die selbstregulierende Informationsverarbeitung *Intelligenter Technischer Systeme*.

A2) Bereitstellung von Konstruktionsprinzipien: Zur Realisierung der Selbstheilung stellt das Kap. 3.3 ausgewähltes *Lösungswissen* zur *Anomalie Detektion*, *Gesundheitsmonitoring*, *Diagnostik* und mögliche *Selbstheilungsaktionen* bereit. Die Systematik unterstützt bei der Auswahl von geeignetem Lösungswissen. Die Kombinationsmöglichkeit der Lösungen ist auf Grund der Schnittstellen der Referenzarchitektur sichergestellt.

A3) Interdisziplinäre Anwendbarkeit: Die Systemkonzeption basiert auf der Spezifikationstechnik CONSENS. In der Arbeit wurde die Sprache um Modellkonstrukte erweitert (vgl. Kapitel 4.3.1). Die Erweiterung betrifft das *Zielsystem*, die *Gefahrenanalyse* sowie die Integration von *Verlässlichkeit* insb. der Sicherheit und Verfügbarkeit. Die Sprache wurde in einem Profil der UML/SysML umgesetzt (vgl. Kap. A2.4).

A4) Identifikation von Potenzialen zur Selbstheilung: Die Potenzialanalyse (vgl. Kap. 4.4) kombiniert die Komplexitätsanalyse, die funktionale Risikoanalyse sowie die Bewertung des Wissensgrades in einem pragmatischen Vorgehen. Die Potenzialanalyse dient der Entscheidungsunterstützung für die Auswahl der Leistungsstufe. Die Systematik unterstützt ferner die Modellierung von *erweiterten dynamischen Fehlerbäumen* (vgl. Kap. 4.4.2). Grundlage bildet die graphische Notation nach IWANEK [Iwa17]. Dabei wurde die Notation um funktionale und temporale Abhängigkeiten zwischen einer Beeinträchtigung und einer Gefahrensituation erweitert. Für die Basisereignisse können Überwachungsverfahren für das *Monitoring* und Selbstheilungsoperationen definiert werden.

A5) Beschreibung der Absicherung: Die Systematik ermöglicht eine integrative Modellierung von System- und Absicherungskonzept. Das Absicherungskonzept besteht aus einer kohärenten Menge an Partialmodellen (vgl. Kap. 4.3.2). Dies sind *Testfall*, *Testobjekt*, *Szenariomodell*, *Testbedstruktur*, *Szenariosequenz* und *Szenarioaktivität*. Die Spezifikation des Absicherungskonzeptes bildet die Basis für die Szenariomodellierung im *Testbed*. Für die Modellierung wurde eine Werkzeugunterstützung umgesetzt. Modellierungsregeln ermöglichen die vollständige Zuweisung von Testfällen auf Anforderungen.

A6) Orientierung am Systementwurf: Ausgehend von den Systemanforderungen werden im Systementwurf die Struktur und das Verhalten des Systems konkretisiert. Ergebnis ist das disziplinübergreifende Lösungskonzept als Basis für den disziplinspezifischen Entwurf. Diese Vorgehensweise greift das *Vorgehensmodell* mit den Phasen *Spezifikation*, *Detailmodellierung* und *integrierte Funktionsabsicherung* explizit auf (vgl. Kap. 4.7). Insbesondere werden Aspekte der Verlässlichkeit und der frühzeitigen Funktions- und Eigenschaftsabsicherung durch das Vorgehensmodell adressiert.

A7) Systematische und interdisziplinäre Vorgehensweise: Das *Vorgehensmodell* der Systematik ermöglicht eine systematische Integration der Selbstheilung. Es ist für den Systementwurf konzipiert (vgl. Kap. 4.7). Das Modell besteht aus neun Phasen die iterativ und inkrementell durchgeführt werden. Für die jeweiligen Phasen sind wiederkehrende Prozessbausteine sowie Rollen und Verantwortlichkeiten definiert (s. Kap. 4.7.2). Es sind konkrete Artefakte als Ergebnis festgelegt und notwendige Hilfsmittel empfohlen. Ferner sind die Phasen des Vorgehensmodells in den etablierten Entwurfsprozess für Intelligente

Technische Systeme integriert. Die Anwendbarkeit des Vorgehensmodells - und somit der Systematik - ist anhand der Entwicklung eines mobilen Roboters gezeigt (vgl. Kapitel 5.1).

A8) Bereitstellung von Analysetechniken: In der Arbeit wurde das *Monkey Testsystem* für Intelligente Technische Systeme entwickelt (vgl. Kap. 4.6.1). Hierzu wurde eine Taxonomie zur Strukturierung von Beeinträchtigungen definiert. Die Taxonomie orientiert sich an spezifischen Leitwörtern und der Grundstruktur dieser Systeme. Auf der Basis dieser Taxonomie sind Monkeys identifiziert, konzipiert und als Lösungswissen dokumentiert. Zudem erfolgte eine prototypische Implementierung für ausgewählte Monkeys in der *Entwicklungsumgebung*. Das fortgeschrittene Monkey Testing unterstützt die Absicherung entlang des Entwicklungsprozesses und ermöglicht eine automatisierte Datenannotation für das maschinelle Lernen.

A9) Werkzeugunterstützung durch ein Virtuelles Testbed: Die Systematik stellt eine *Entwicklungsumgebung* zur Modellierung und Simulation bereit. Die Umgebung kombiniert etablierte Werkzeuge (ROS, Gazebo, RVIZ, InfluxDB) zu einem Virtuellen Testbed (s. Kap. 4.5). Die Umgebung ist erweiterbar und durch die Definition der Schnittstellen sind Werkzeuge austauschbar. Das entwickelte Virtuelle Testbed besteht aus einer Modellbibliothek, Methoden der Interaktion und Visualisierung sowie der Simulationssteuerung. Die Simulationsdaten werden in einer Datenbank gespeichert. Die Daten können für das maschinelle Lernen genutzt werden und ermöglichen eine Scenario-Replay. Ferner unterstützt die Entwicklungsumgebung die kontinuierliche Integration und virtuelle Inbetriebnahme. Die Umgebung wurde an einem Beispiel validiert und kann auf andere Systeme übertragen werden.

Die vorgestellte *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* erfüllt somit die Anforderungen aus Kapitel 2.6. Ihr Anwendung wurde an einem Beispiel *Mobiler Roboter* erprobt und im Forschungsprojekt KI4AS an einem weiteren Anwendungsbeispiel der intelligente Fabrik demonstriert. Die entwickelte Systematik ist geeignet, die Resilienz und somit die Autonomie von Intelligenten Technischen Systemen durch Eigenschaften der Selbstheilung zu steigern.

6 Zusammenfassung und Ausblick

Moderne Erzeugnisse des Maschinenbaus basieren auf dem synergetischen Zusammenwirken von verschiedenen Fachgebieten wie der Mechanik, Elektrotechnik, Software und Regelungstechnik. Der Begriff Mechatronik bringt dies zum Ausdruck. Mit der Digitalisierung und der Künstlichen Intelligenz wandeln sich mechatronische Systeme zu Intelligenten Technischen Systemen. Diese Systeme agieren autonom und kooperieren in einem dynamisch vernetzten Systemverbund. Derartige technische Systeme bieten vielfältige Nutzenpotenziale in der Produktion, Logistik oder im Straßenverkehr [FAS17]. **Autonome Systeme** sind selbstlernend und erreichen ein vorgegebenes Ziel ohne menschlichen Eingriffe und ohne expliziter Programmierung. Sie sind selbstständig und situationsadaptiv. Dazu müssen die Systeme ihre Umgebung *wahrnehmen*, ihre Handlungen zielorientiert *planen* und *ausführen*. Es handelt sich hierbei um komplexe Systeme mit hohen Anforderungen an die **Verlässlichkeit** im Betrieb: Autonome Systeme müssen auch bei unvollständigen Wissen, beeinträchtigter Funktionalität einzelner Teilsysteme oder unsicheren Umgebungsbedingungen ihre Aufgabe erfüllen. Sie müssen im Betrieb in der Lage sein, ihr Verhalten anzupassen, um beispielsweise unzureichende oder ausgefallene Funktionen zu kompensieren. Ein Ansatz zur Umsetzung dieser Fähigkeit ist die **Selbstheilung**. Hierzu muss das System seinen Zustand und sein Verhalten *überwachen*, Störungen und Ursachen *diagnostizieren*, eine Selbstheilungsstrategie *planen* und die Selbstheilungsaktion *ausführen*. Selbstheilende Systeme weisen demnach zusätzliche Merkmale auf: Sie sind *wandlungsfähig*, *robust*, *lernfähig*, *redundant* und *verteilt*.

Die Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme erfordert ein ganzheitliches Lösungskonzept und eine frühzeitige Einbindung von Experten der Künstlichen Intelligenz und der Verlässlichkeit in den Entwicklungsprozess. Derartige selbstheilende Systeme sind hochkomplex. Dies gilt ebenso für die zugehörigen Entwicklungsprozesse. Die wesentlichen **Herausforderungen** sind die zunehmende Interdisziplinarität, die Sicherstellung der Verlässlichkeit, Fehlende Konzepte für die Absicherung und eine fehlende Entwicklungsmethodik (vgl. Kap. 2.5). Um diesen Herausforderungen zu begegnen, muss die Entwicklung selbstheilender Systeme bereits in den frühen Phasen der Entwicklung methodisch unterstützt werden. Es werden ein Selbstheilungskonzept, eine Spezifikationstechnik, sowie dedizierte Hilfsmittel zur interdisziplinären Modellierung und schließlich Techniken für die Szenario-basierte Absicherung selbstheilender Systeme benötigt. Von zentraler Bedeutung ist das Selbstheilungskonzept für Intelligente Technische Systeme, welches ein Framework, relevante Entwurfsaspekte und Lösungswissen zur Realisierung von selbstheilenden Systeme bereitstellt.

Im Rahmen der Arbeit wurden existierende Ansätze zur methodischen Entwicklung Intelligenter Technischer Systeme, Verlässlichkeitskonzepte sowie Verfahren zur Analyse und Bewertung von Systemeigenschaften untersucht. Die untersuchten Ansätze liefern nur eine partielle Unterstützung für die Integration und Realisierung von Selbstheilungseigenschaf-

ten. Eine ganzheitliche Systematik existiert nicht. Von den analysierten Strukturkonzepten erfüllt das Technologiekonzept *Intelligente Technische Systeme* wichtige Anforderungen. Das gleiche gilt für die Spezifikationstechnik CONSENS. Zur ganzheitlichen Betrachtung von Selbstheilung bedarf es jedoch einiger Anpassungen und Erweiterungen der bestehenden Aspekte und Modellkonstrukte. Bei den untersuchten Verlässlichkeitskonzepten bietet der *Observer-Controller* Ansatz in Kombination mit dem *Verlässlichkeitskonzept* einen generischen Ansatz zur Umsetzung von Selbstheilung. Die Architektur bedarf aber einer Erweiterung um das Online Lernen und die Anomalie Detektion. Der *Entwurf von Organic Computing Systemen* bietet einen methodischen Ansatz zur Steigerung der Autonomie durch einen Reifegrad-basierten Ansatz und erforderlichen Entwurfsaspekten. Diese sind generisch und nicht mit einer Spezifikationstechnik gekoppelt. Das *Simian Army Testing* liefert einen guten Ansatz zur Analyse und Absicherung der Selbstheilung. Die Anwendung ist jedoch bisher auf Cloud-Services limitiert. Aus diesen Gründen resultiert ein **Handlungsbedarf** nach einer ganzheitlichen *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme*.

Die entwickelte Systematik greift einige Ansätze aus dem Stand der Technik auf, entwickelt diese weiter und ergänzt sie um neue Werkzeuge und Hilfsmittel. Das **Ergebnis** ist eine Entwicklungssystematik, mit den folgenden Bestandteilen:

- ein **Selbstheilungskonzept** für Intelligente Technische Systeme. Das Selbstheilungskonzept gilt als allgemeingültiges Muster und stellt die Funktionen, Bausteine, Lösungswissen entlang von Leistungsstufen bereit,
- eine **Spezifikationstechnik** für selbstheilende Systeme, die auf der Spezifikationstechnik CONSENS aufbaut, diese um das Absicherungskonzept sowie um Aspekte der Verlässlichkeit erweitert und in einem Modellierungswerkzeug umgesetzt ist.
- **fortgeschrittene Analysetechniken** zur Simulation und Absicherung in einem virtuellen Testbed, mit dem virtuelle Szenarien erzeugt und selbstheilende Systeme und deren Eigenschaften mit einem agentenbasierten Testverfahren trainiert und untersucht werden können, sowie
- ein **iteratives Vorgehensmodell** zur Entwicklung selbstheilender Systeme, das detailliert die durchzuführenden Tätigkeiten, Artefakte und Rollen im Systementwurf beschreibt und die beteiligten Akteure orchestriert.

Die **Anwendung** der Systematik erfolgte am Beispiel eines mobilen Roboters. Hierfür wurde ein intralogistischer Warentransport als Szenario umgesetzt. Im Rahmen der Anwendung wurde das Vorgehensmodell der Entwicklungssystematik vollständig durchlaufen und sowohl die Entwicklungsumgebung als auch die Hilfsmittel zur Spezifikation, Modellierung und Analyse im Systementwurf angewendet. Die Validierung zeigt, dass die entwickelte *Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme* die Anforderungen erfüllt.

In Hinblick auf die Resilienz und die Selbstheilung in technischen Systemen besteht

weiterer Forschungsbedarf: Zukünftige Forschungsarbeiten müssen das Thema Resilienz *in* Selbstregulation und Resilienz *durch* Selbstregulation beleuchten. Hier fehlen z.B. noch Methoden zur Absicherung und zur Vertrauenswürdigkeit. Zudem sind durchgängige Entwicklungs- und Simulationsumgebung anzustreben. Virtuelle Szenarien sind mit realen Daten in Form einer hybriden Modellierung zu kombinieren. Auf diese Weise entstehen aussagekräftige Szenarien für das maschinelle Lernen. Hierbei kann ein Regelwerk dienen, welches die Qualität eines Szenarios und der Modelle sicherstellt. Auf der Basis dieser Szenarien sind zukünftig fortgeschrittene Analyseverfahren anzuwenden. Das entwickelte Monkey Testing für Intelligente Technische Systeme liefert hierfür einen wichtigen Beitrag.

Übergeordnetes Ziel zukünftiger Forschungsaktivitäten ist es, die beteiligten Disziplinen in einer ganzheitliche Produktentstehungsmethodik im Zeitalter der Digitalisierung zu integrieren. Sie muss die Aktivitäten aller beteiligten Akteure berücksichtigen und der zunehmenden Bedeutung der Resilienz und der Künstlichen Intelligenz in Intelligenten Technischen Systemen gerecht werden.

Abkürzungsverzeichnis

AUTOSAR Automotive Open System Architecture

BMBF Bundesministerium für Bildung und Forschung

CONOPS CONcept of OPerationS

CONSENS CONceptual design Specification technique for the ENgineering of mechatronic Systems

CPS Cyber-Physische Systeme

DeepRL Deep Reinforcement Learning

DFG Deutsche Forschungsgemeinschaft

DIN Deutsches Institut für Normung

DLR Deutsches Zentrum für Luft- und Raumfahrt e.V.

FDIR Fault Detection, Isolation and Recovery

HiL Hardare-in-the-Loop

IMU Inertiale Messeinheit

INCOSE International Council on Systems Engineering

INVIRTES Integrierte Entwicklung komplexer Systeme mit Virtuellen Testbeds auf der Basis zentraler Weltmodelle und moderner Konzepte der eRobotik

ISO International Organization for Standardization

ITS Intelligente Technische Systeme

KI Künstliche Intelligenz

KI4AS Validierung Künstlicher Immunsysteme für Autonome Systeme

LIONS Lichtsensor-basierte Ortungs- und Navigationsdienste für autonome Systeme

MBSE Model-Based Systems Engineering

MTBF Mean Time Between Failure

MTTF Mean Time To Failure

ODD Operational Design Domain

OMG Object Management Group

RAM Responsibility Assignment Matrix

ROS Robot Operating System

RTAB Real-Time Appearance-Based

SciL Scenario-in-the-Loop

SFB Sonderforschungsbereich

SIL Sicherheitsintegritätslevel

SNR Signal-Noise-Ratio

SOTIF Safety Of The Intended Functionality

STPA System-Theoretic Process Analysis

SuOC System under Observation and Control

URDF Unified Robotic Description Format

VDI Verein Deutscher Ingenieure

VTB Virtuelles Testbed

XARCO XML Macro Language

XiL X-in-the-Loop

Literaturverzeichnis

Publikationen

[AAA+90] ARLAT, J.; AGUERA, M.; AMAT, L.; CROUZET, Y.; FABRE, J. C.; LAPRIE, J. L.; MARTINS, E.; POWELL, D.: Fault Injection for Dependability Validation: A Methodology and Some Applications. IEEE Transactions on Software Engineering 16 (2), IEEE, New Jersey, 1990, pp. 166–182

[AAB+15] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G.; DAVIS, A.; DEA, J.; DEVIN, M.; GHEMAWA, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANE, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIEGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, Software available from tensorflow.org, 2015, Unter: https://www.tensorflow.org/ (besucht am 02. 11. 2020)

[ABB+13] ARNOLD, F.; BELINFANTE, A.; BERG, F.; GUCK, D.; STOELINGA, M.: DFTCalc: A Tool for Efficient Fault Tree Analysis. In: International Conference on Computer Safety, Reliability, and Security. Sept. 14–27, 2013, Computer Safety, Reliability, and Security, Springer-Verlag, Toulouse, Sept. 2013, pp. 293–301

[ABC+18] ANDRYCHOWICZ, M.; BAKER, B.; CHOCIEJ, M.; JOZEFOWICZ, R.; MCGREW, B.; PACHOCKI, J.; PETRON, A.; PLAPPERT, M.; POWELL, G.; RAY, A.; SCHNEIDER, J.; SIDOR, S.; TOBIN, J.; WELINDER, P.; WENG, L.; ZAREMBA, W.: Learning Dexterous In-Hand Manipulation, arXiv e-prints, 2018

[Abd17] ABDULKHALEQ, A.: A System-Theoretic Safety Engineering Approach for Software-Intensive Systems. Dissertation, Fakultät 5: Informatik, Elektrotechnik und Informationstechnik der Universiät Stuttgart, Universiät Stuttgart, Stuttgart, 2017

[ABK+16] ALBERS, A.; BEHRENDT, M.; KLINGLER, S.; MATROS, K.: Verifikation und Validierung im Produktentstehungsprozess. In: Lindemann, U.: Handbuch Produktentwicklung. Carl Hanser Verlag, München, 2016, S. 541–569

[Aca13] AKADEMIE DER TECHNIKWISSENSCHAFTEN (ACATECH): Deutschlands Zukunft als Produktionsstandort sichern: Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0, Abschlussbericht, Berlin: Akademie der Technikwissenschaften (acatech), 2013

[Agg13] AGGARWAL, C.: Outlier Analysis. Springer-Verlag, New York, 2013

[AKM+09] ABDELWAHED, S.; KARSAI, G.; MAHADEVAN, N.; OFSTHUN, S.: Practical Implementation of Diagnosis Systems Using Timed Failure Propagation Graph Models. IEEE Transactions on Instrumentation and Measurement 58 (2), IEEE, New Jersey, 2009, pp. 240–247

[ALM08] AUF, A.; LITZA, M.; MAEHLE, E.: Distributed Fault-Tolerant Robot Control Architecture Based on Organic Computing Pronciples. Biologically-Inspired Collaborative Computing. International Federation of Information Processing (IFIP) 268, Springer-Verlag, New York, 2008, pp. 115–124

[ALR+04] AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE transaction on dependable and secure computing 1, IEEE, New Jersey, 2004, pp. 11–33

[Alt12] ALT, O.: Modell-Basierte Systementwicklung mit SysML. Carl Hanser Verlag, München, 2012

[AOS+16] AMODEI, D.; OLAH, C.; STEINHARDT, J.; CHRISTIANO, P. F.; SCHULMAN, J.; MANÉ, D.: Concrete Problems in AI Safety. CoRR abs/1606.06565, 2016

[Auf10] AUF, A.-P. E. S.: Eine Organic Computing basierte Steuerung für einen hexapoden Laufroboter unter dem Aspekt reaktiver Zuverlässigkeit und Robustheit. Dissertation, Insitut für Technische Informatik, Universität zu Lübeck, Lübeck, 2010

[AUT17] AUTOSAR: Explanation of Adaptive Platform Design, München, 2017

[AUT20-ol] AUTOSAR (Hrsg.): AUTomotive Open System ARchitecture Adaptive Plattform. Unter: https://www.autosar.org/, 2020

[BBC+00] BALDI, P.; BRUNAK, S.; CHAUVIN, Y.; ANDERSEN, C. A. F.; NIELSEN, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16, 2000, S. 412–424

[BBC+14] BITTNER, B.; BOZZANO, M.; CIMATTI, A.; FERLUC, R. D.; GARIO, M.; GUIOTTO, A.; YUSHTEIN, Y.: An Integrated Process for FDIR Design in Aerospace. In: Ortmeier, F.; Rauzy, A. (Eds.): International Symposium on Model-Based Safety and Assessmemt (IMBSA). Oct. 27–29, 2014, Model-Based Safety and Assessment, Springer-Verlag, New York, 2014, pp. 82–95

[BCR+09] BOZZANO, M.; CIMATTI, A.; ROVERI, M.; KATOEN, J.-P.; NGUYEN, V. Y.; NOLL, T.: Codesign of Dependable Systems: A Component-Based Modeling Language. In: July 13–15, 2009, 2009 IEEE/ACM 7th International Conference on Formal Methods and Models for Co-Design, Cambridge, Massachusetts, USA, 2009, pp. 121–130

[BDD+16] BOJARSKI, M.; DEL TESTA, D.; DWORAKOWSKI, D.; FIRNER, B.; FLEPP, B.; GOYAL, P.; JACKEL, L.; MONFORT, M.; MULLER, U.; ZHANG, J.; ZHANG, X.; ZHAO, J.; ZIEBA, K.: End to End Learning for Self-Driving Cars, arXiv e-prints, 2016

[BGJ+09] BERTSCHE, B.; GÖHNER, P.; JENSEN, U.; SCHINKÖTHE, W.; WUNDERLICH, H.-J.: Zuverlässigkeit mechatronischer Systeme. Grundlagen und Bewertung in frühen Entwicklungsphasen. Springer-Verlag, Heidelberg, 2009

[Bir91] BIROLINI, A.: Qualität und Zuverlässigkeit technischer Systeme: Theorie, Praxis, Management. Springer-Verlag, Heidelberg, 1991

[Bit18] BITKOM: Digitalisierung gestalten mit dem Periodensystem der Künstlichen Intelligenz, 2018

[BL04] BERTSCHE, B.; LECHNER, G.: Zuverlässigkeit im Fahrzeug- und Maschinenbau. Springer-Verlag, Heidelberg, 2004

[BM17] BEHZADAN, V.; MUNIR, A.: Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In: Perner, P. (Ed.). July 15–20, 2017, Springer-Verlag, New York, 2017, pp. 262–275

[BMM+06] BRANKE, J.; MNIK, M.; MÜLLER-SCHLOER, C.; PROTHMANN, H.; RICHTER, U.; ROCHNER, F.; SCHMECK, H.: Organic Computing - Adressing Complexity by Controller Self-organization. In: Nov. 15–19, 2006, Paphos, Cypres, 2006

[Bro85] BROOKS, R.: A robust layered control system for a mobile robot, 1985

[Bue12-ol] BUEST, R.: Netflix: Der Chaos Monkey und die Simian Army - Das Vorbild für eine gute Cloud Systemarchitektur. Unter: https://www.crisp-research.com, 2012

[BW14] BACK, S.; WEIGEL, H.: Design for Six Sigma. Carl Hanser Verlag, München, 2014

[CBK09] CHANDOLA, V.; BANERJEE, A.; KUMAR, V.: Anomaly Detection: A Survey. ACM Computing Surveys 41, ACM, New York, 2009

[CCS16] CRAWEY, E.; CAMERON, B.; SELVA, D.: System Architecture. Strategy and Product Development for Complex Systems. Pearson, Education Limited, Essex, 2016

[Cho+15] CHOLLET, F. et al.: Keras, https://keras.io, 2015

[COM19-ol] COMPASS: FAME Prozess. Unter: http://www.compass-toolset.org/, 2019

[Dai19] DAIMLER: Geschäftsbericht 2019, Stuttgart, 2019

[DBB92a] DUGAN, J.; BAVUSO, S.; BOYD, M.: Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems. IEEE Transactions on Reliability 41, IEEE, New Jersey, 1992, pp. 363–377

[DBB92b] DUGAN, J. B.; BAVUSO, S. J.; BOYD, M. A.: Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability 41 (3), IEEE, New Jersey, 1992, pp. 363–377

[DFG18-ol] DEUTSCHE FORSCHUNGSGEMEINSCHAFT: SPP 1183: Organic Computing. Unter: http://gepris.dfg.de/gepris/projekt/5472210, 2018

[DGH+14] DOGRAMADZI, S.; GIANNACCINI, M. E.; HARPER, C.; SOBHANI, M.; WOODMAN, R.: Environmental Hazard Analysis - a Variant of Preliminary Hazard Analysis for Autonomous Mobile Robots. Journal of Intelligent and Robotic Systems 76 (1), Springer-Verlag, Berlin, 2014, pp. 73–117

[DGS+18] DUMITRESCU, R.; GAUSEMEIER, J.; SLUSALLEK, P.; CIESLIK, S.; DEMME, G.; FALKOWSKI, T.; HOFFMANN, H.; KADNER, S.; REINHART, F.; WESTERMANN, T.; WINTER, J.: Studie „Autonome Systeme“, Studien zum deutschen Innovationssystem 13-2018, 2018

[Die14] DIENST, S.: Analyse von Maschinendaten zur Entscheidungsunterstützung bei der Produktverbesserung durch die Anwendung eines Feedback Assistenz Systems. Dissertation, Naturwissenschaftlich-Technische Fakultät der Universität Siegen, Siegen, 2014

[Dor16] DORI, D.: Model-Based Systems Engineering with OPM and SysML. Springer-Verlag, Heidelberg, 2016

[Dum10] DUMITRESCU, R.: Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Universität Paderborn, Paderborn, 2010

[DWF18] DUMITRESCU, R.; WESTERMANN, T.; FALKOWSKI, T.: Autonome Systeme in der Produktion. Planungssystematik für die Entwicklung autonomer Systeme. Industrie 4.0 Management 34, Gito, Berlin, 2018, S. 17–20

[Ebe15] EBEL, B.: Modellierung von Zielsystemen in der interdisziplinären Produktentstehung. Dissertation, Karlsruher Institut für Technologie (KIT), IPEK Institut für Produktentwicklung, Band 85, Karlsruhe, 2015

[EM17] EHRLENSPIEL, K.; MEERKAMM, H.: Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit. Carl Hanser, München, 7. Auflage, 2017

[FAS17] FACHFORUM AUTONOME SYSTEME: Chancen und Risiken für Wirtschaft, Wissenschaft und Gesellschaft: Abschlussbericht - Langversion, Berlin, 2017

[FG13] FELDHUSEN, J.; GROTE, K.-H.: Konstruktionslehre. Springer-Verlag, Heidelberg, 8. Auflage, 2013

[FGK+04] FRANK, U.; GIESE, H.; KLEIN, F.; OBERSHELP, O.; SCHMIDT, A.; SCHULZ, B.; VÖCKING, H.; WITTING, K.: Selbstoptimierende Systeme des Maschinenbaus. Definition der Selbstoptimierung, 2004

[FMD+13] FREI, R.; MCWILLIAM, R.; DERRICK, B.; PURVIS, A.; TIWARI, A.; SERUGENDO, G.: Self-healing and self-reparing technologies. International Journal of Advanced Manufacturing Technology 69 (5-8), Springer-Link, New York, 2013, pp. 1033–1061

[FMS12] FRIEDENTHAL, S.; MOORE, A.; STEINER, R.: A Practical Guide to SysML. The Systems Modeling Language. Elsevier, Boston, 2012

[Foe13] FOELLINGER, O.: Regelungstechnik: Einführung in die Methoden und ihre Anwendung. VDE Verlag, Berlin, 2013

[Gartner20-ol] GARTNER (Hrsg.): Data Analytics. Unter: https://www.gartner.com/en/information-technology/glossary, 2020

[GDE+19] GAUSEMEIER, J.; DUMITRESCU, R.; ECHTERFELD, J.; PFÄNDER, T.; STEFFEN, D.; THIELEMANN, F.: Innovationen für die Märkte von morgen. Strategische Planung von Produkten, Dienstleistungen und Geschäftsmodellen. Carl Hanser Verlag, München, 2019

[Ger17] GERON, A.: Hands-On Machine Learning with SciKit-Learn and TensorFlow. O'Reilly, Sebastopol, 2017

[GFD+08] GAUSEMEIER, J.; FRANK, U.; DONOTH, J.; KAHL, S.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus. Konstruktion 7-9, VDI-Verlag, Düsseldorf, 2008

[GPH+19-ol] GOODFELLOW, I.; PAPERNOT, N.; HUANG, S.; DUAN, R.; ABBEEL, P.; CLARK, J.: Attacking Machine Learning with Adversarial Examples. Unter: https://openai.com/blog/adversarial-example-research/, 2019

[Gre19-ol] GREMLIN: The Simian Army. Overview and Ressources. Unter: https://www.gremlin.com/chaos-monkey/the-simian-army/, 2019

[GRS+14] GAUSEMEIER, J.; RAMMIG, F.-J.; SCHÄFER, W.; SEXTRO, W. (Hrsg.): Dependability of Self-Optimizing Mechatronic Systems. Springer-Verlag, Heidelberg, 2014

[GRS14] GAUSEMEIER, J.; RAMMIG, F.-J.; SCHÄFER: Design Methodology for Intelligent Technical Systems. Springer-Verlag, Heidelberg, 2014

[GSR+07] GHOSH, D.; SHARMAN, R.; RAO, R.; UPADHYAYA, S.: Self-healing systems - survey and synthesis. Decision Support Systems 42, Elsevier, Amsterdam, 2007, pp. 2164–2185

[GSS14] GOERZ, G.; SCHNEEBERGER, J.; SCHMID, U.: Handbuch der Künstlichen Intelligenz. Oldenburg Verlag, München, 5. Auflage, 2014

[GT18] GAUSEMEIER, J.; TRÄCHTLER, A.: Steigerung der Intelligenz mechatronischer Systeme. Springer-Verlag, Heidelberg, Dez. 2018

[GTS14] GAUSEMEIER, J.; TRÄCHTLER, A.; SCHÄFER, W.: Semantische Technologien im Entwurf mechatronischer Systeme. Carl-Hanser Verlag, München, 2014

[GZB+14] GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A.: A Survey on Concept Drift Adaptation. ACM Computing Surveys 46, ACM, New York, 2014

[HA14] HODGE, V.; AUSTIN, J.: A Survey of Outlier Detection Methodologies. Artificial Intelligence Review 22 (2), Springer-Verlag, New York, 2014, pp. 85–126

[HBP18-ol] HUMAN BRAIN PROJECT: Neurorobotics in the HBP. Unter: http://www.neurorobotics.net, 2018

[HFW+15] HABERFELLNER, R.; FRICKE, E.; WECK, O. de; SIEGFRIED, V.: Systems Engineering. Grundlagen und Anwendung. orell füssli Verlag, Zürich, 13. Auflage, 2015

[HLD20] HILLEBRAND, M.; LAKHANI, M.; DUMITRESCU, R.: A design methodology for deep reinforcement learning for autonomous Systems. In: 5th Conference on Systems Integrated Intelligence (SysInt) 2020, Bremen, Nov. 2020

[HO13] HÖFTBERGER, O.; OBERMAISSER, R.: Ontology-based runtime reconfiguration of distributed embedded real-time systems. In: June 19–21, 2013, 2013 IEEE 16th International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC), IEEE, Paderborn, DE, 2013, pp. 1–9

[HSS16] HELLE, P.; STROBEL, C.; SCHAMAI, W.: Testing of Autonomous Systems - Challenges and Current State-of-the-Art. INCOSE International Symposium 26 (1), Wiley, Hoboken, USA, 2016

[HTR97] HSUEH, M. C.; TSAI, T. K.; RAVISHANKAR, K. I.: Fault Injection Techniques and Tools. Computer 30 (4), IEEE, New Jersey, 1997, pp. 75–82

[IBM05] IBM: An architectural blueprint for autonomic computing, Autonomic Computing White Paper, IBM, 2005

[INC18-ola] INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING: Guide to the Systems Engineering Body of Knowledge (SEBoK). Unter: https://www.sebokwiki.org, 2018

[INC18-olb] INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING: What is Systems Engineering. Unter: https://www.incose.org/systems-engineering, 2018

[Inf19-ol] INFLUX: influxdata. Unter: https://www.influxdata.com/, 2019

[Ise08] ISERMANN, R.: Mechatronische Systeme. Grundlagen. Springer-Verlag, Heidelberg, 2. Auflage, 2008

[Iwa17] IWANEK, P.: Systematik zur Steigerung der Intelligenz mechatronischer Systeme im Maschinen- und Anlagenbau. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 366, 2017

[Kai14] KAISER, L.: Rahmenwerk zur Modellierung einer plausiblen Systemstruktur mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 327, 2014

[KEH20] KUMAR JHA, N.; ENZBERG, S. v.; HILLEBRAND, M.: Using Deep Learning for Anomaly Detection in Autonomous Systems. Solving Engineering Problems with Machine Learning - Introduction to the Special Theme 122, European Research Consortium for Informatics and Mathematics (ERCIM), 2020, pp. 47–48

[KF19] KOOPMAN, P.; FRATRIK, F.: How Many Operational Design Domains, Objects, and Events? In: Jan. 27, 2019, AAAI Workshop on Artificial Intelligence Safety 2019, Honolulu, Hawaii, 2019, pp. 45–48

[KNR+15] KOYEJO, O.; NATARAJAN, N.; RAVIKUMAR, P.; DHILLON, I. S.: Consistent Multilabel Classification. In: NIPS, 2015, S. 3321–3329

[KRA+17] KNOLL, A.; RÖHRBEIN, F.; AKL, M.; KUHN, A.; SCHARMA, K.: Neurorobotics: From computational neuroscience to intelligent robots and back. Informatik Spektrum, Springer-Verlag, Berlin, 2017, pp. 123–128

[Lap20] LAPAN, M.: Deep Reinforcement Learning. mitp, Frechen, 2020

[Lap92] LAPRIE, J. C.: Dependability: Basic Concepts and Terminology. In: Avizienis, A.; Kopetz, H.; Laprie, J. C.: Dependable Computing and Fault-Tolerant Systems. Springer-Verlag, 1992

[Lee10] LEE, J.: Design of Self-Maintenance and Engineering Immune Systems for Smarter Machines and Manufacturing Systems. In: July 1, 2010–July 2, 2019, First IFAC Workshop on Advanced Maintenance Engineering, Services and Technology, Lisbon, Portugal, 2010, pp. 1–11

[Lem16] LEMMER, K.: Neue autoMobilität. Automatisierter Straßenverkehr der Zukunft, 2016

[Lev04] LEVESON, N.: A New Accident Model for Engineering Safer Systems. Safety Science 42, Elsevier, London, 2004

[Lev14] LEVESON, N.: Engineering a Safer World. Systems Thinking applied to Safety. MIT Press, Boston, 2014

[Lin12] LINDLAR, F.: Modellbasierter Evolutionärer Funktionstest. Dissertation, Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, Berlin, 2012

[LKS00] LÜCKEL, J.; KOCH, T.; SCHMITZ, J.: Mechatronik als integrative Basis für innovative Produkte, VDI-Gesellschaft Entwicklung Produktion, V., Düsseldorf, 2000

[LM18] LABBÉ, M.; MICHAUD, F.: RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operationD. Journal of Field Robotics 36, Wiley, Hoboken, USA, 2018

[LMB09] LINDEMANN, U.; MAURER, M.; BRAUN, T.: Structural Complexity Management. An Approach for the Field of Product Design. Springer-Verlag, Heidelberg, 2009

[LWK+08] LEE, J.; WANG, H.; KOBBACY, K.; MURTHY, P.: New Technologies for Maintenance. In: Complex System Maintenance Handbook. Springer-Verlag, London, 2008, S. 49–78

[Maa97] MAASS, W.: Networks of spiking neurons: The third generation of neural network models. Neural Networks 10 (9), Elsevier, London, 1997, pp. 1659–1671

[MHW+16] MICHAEL, J.; HILLEBRAND, M.; WOHLERS, B.; HENKE, C.; DUMITRESCU, R.; TRÄCHTLER, A.: Implementing intelligent technical systems into smart homes by using model based systems engineering and multi-agent systems. Renewable Energy and Power Quality Journal (RE'I&'PQJ) 16 1, 2016, pp. 359–364

[Mod18-ola] MODELICA ASSOCIATION: FMI Version 2.0, FMI for Model Exchange and Co-Simulation. Unter: http://www.fmi-standard.org/, 2018

[Mod18-olb] MODELICA ASSOCIATION: Modelica - A unified Object-Oriented Language for Physical Systems Modeling - Language Specification. Unter: http://www.modelica.org/, 2018

[MSU11] MÜLLER-SCHLÖR, C.; SCHMECK, H.; UNGERER, T.: Organic Computing - A Paradigm Shift for Complex Systems. Birkhäuser Verlag, Basel, 2011

[MT17] MÜLLER-SCHLÖR, C.; TOMFORDE, S.: Organic Computing - Technical Systems for Survival in the Real World. Birkäuser, Basel, 2017

[Mus15] MUSEN, M. A.: The protege project: a look back and a look forward. AI Matters 1, 2015, S. 4–12

[NBS16] NORTH, K.; BRANDNER, A.; STEININGER, T.: Die Wissenstreppe: Information - Wissen - Kompetenz. In: North, K.; Brandner, A.; Steininger, T.: Wissensmanagement für Qualitätsmanager. Springer Gabler, Wiesbaden, 2016, S. 5–8

[Net11-ol] NETFLIX: The Netflix Simian Army. Unter: https://medium.com/netflix-techblog, 2011

[Net19-ol] NETFLIX: SimianArmy. Unter: https://github.com/Netflix/SimianArmy, 2019

[NHSTA16] NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION (NHSTA): Federal Automated Vehicles Policy, Accelerating the Next Revolution In Roadway Safety, Washington D.C.: National Highway Traffic Safety Administration, 2016

[Nvi19-ol] NVIDIA: NVIDIA Jetson TX2. Unter: https://www.nvidia.com, 2019

[OI10] ORZELL, G.; IZRAILEVSKY, Y.: Validating the resiliency of networked applications, US 2012/0072571 A1, 2010

[OSRF19-ola] OPEN SOURCE ROBOTIC FOUNDATION: Gazebo. Unter: http://gazebosim.org/, 2019

[OSRF19-olb] OPEN SOURCE ROBOTIC FOUNDATION: ROS Documentation. Unter: http://wiki.ros.org/, 2019

[OT19] OBERMEIER, B.; TREUGUT, L.: Lernende Systeme in lebensfeindlichen Umgebungen. Potenziale, Herausforderungen und Gestaltungsoptionen, Berlin, 2019

[PC11] PORTINALE, L.; CODETTA-RAITERI, D.: Using Dynamic Decision Networks and Extended Fault Trees for Autonomous FDIR. In: Nov. 7–9, 2011, 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, Washington D.C, USA, 2011, pp. 480–484

[PCG+12] PORTINALE, L.; CODETTA-RAITERI, D.; GUIOTTO, A.; DINOLFO, S.: ARPHA: a software prototype for fault detection, identification and recovery in autonomous spacecrafts ACTA FUTURA. ACTA FUTURA, 2012, pp. 99–110

[PLS18-ol] PLATTFORM LERNENDE SYSTEME (Hrsg.): Glossar. Unter: https://www.plattform-lernende-systeme.de/glossar.html, 2018

[Poo11] POOK, S.: Eine Methode zum Entwurf von Zielsystemen selbstoptimierender mechatronischer Systeme. Dissertation, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 296, 2011

[Pow11] POWERS, D. M.: Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. Journal of Machine Learning Technologies 2, 2011, S. 37–63

[Pri13] PRIESTERJAHN, C.: Analyzing Self-healing Operations in Mechatronic Systems. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Universität Paderborn, Paderborn, 2013

[PS11] PSAIER, H.; SCHAHRAM, D.: A survey on self-healing systems: approaches and systems. Computing 1 (1), Springer-Verlag, New York, 2011, pp. 43–73

[PS15] PETIT, J.; STEVEN, S.: Potential Cyberattacks on Automated Vehicles. IEEE Transactions on Intelligent Transportation Systems 16 (2), IEEE, New Jersey, 2015, pp. 546–556

[PVG+11] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2011, S. 2825–2830

[QGS15] QUIGLEY, M.; GERKLEY, B.; SMART, W.: Programming Robots with ROS. A practical introduction to the Robot Operating System. O'Reilly, Sebastopol, 2015

[Ras15] RAST, M.: Domänenübergreifende Modellierung und Simulation als Grundlage für Virtuelle Testbeds. Dissertation, Institut für Mensch-Maschine-Interaktion, RWTH Aachen, Apprimus Verlag, Aachen, 2015

[Rei10-ol] REINER Jörg; Meyer, J.: Evolutionäres Testen von Steuergeräten. Unter: https://www.elektroniknet.de, 2010

[RHI+17] RATASICH, D.; HÖFTBERGER, O.; ISAKOVIC, H.; SHAFIQUE, M.; GROSU, R.: A Self-Healing Framework for Building Resilient Cyber-Physical Systems. In: May 16–18, 2017, 2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC), Toronto, CA, 2017, pp. 133–140

[Ric09] RICHTER, U. M.: Controller Self-Organisation - Using Learning Classifier Systems. Dissertation, Fakultät für Wirtschaftswissenschaften der Universität Karlsruhe, KIT Scientific Publishing, Karlsruhe, 2009

[RKG+18] RATASICH, D.; KHALID, F.; GEISSLER, F.; GROSU, R.; SHAFIQUE, M.; BARTOCCI, E.: A Roadmap Towards Resilient Internet of Things for Cyber-Physical Systems. CoRR abs/1810.06870, 2018

[RN16] RUSSEL, S.; NORVIG, P.: Artificial Intelligence - A Modern Approach. Addison Wesley, Boston, 3 Edition, 2016

[Rop12] ROPOHL, G.: Allgemeine Systemtheorie. Einführung in transdisziplinäres Denken. Edition Sigma, Berlin, 2012

[RRK+17] RUSSO, D.; ROY, B. V.; KAZEROUNI, A.; OSBAND, I.: A Tutorial on Thompson Sampling. CoRR abs/1707.02038, 2017

[Sch11-ol] SCHAMAI, W.: ModelicaML - Getting Started. Unter: http://www.openmodelica.org/, 2011

[Sen14] SENGE, R.: Machine Learning Methods for Fuzzy Pattern Tree Induction. Dissertation, Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Philipps-Universität Marburg, Marburg, 2014

[SH10] SENGE, R.; HÜLLERMEIER, E.: Pattern trees for regression and fuzzy systems modeling. In: IEEE International Conference on Fuzzy Systems. July 18–23, 2010, 2010 IEEE International Cnoference on Fuzzy Systems, IEEE, New Jersey, 2010, pp. 1–7

[SH18] STANLY, J.; HILLEBRAND, M. (BETREUER): Model-Based Reinforcement Learning for Behavioral Planning in Self-Healing Systems, Universität Paderborn, Heinz Nixdorf Institut, unveröffentlichte Masterarbeit, Paderborn, 2018

[SH20] SCHNIEDER, L.; HOSSE, R.: Leitfaden Safety of the Intended Functionality. Verfeinerung dr Sicherheit der Sollfunktion auf dem Weg zum autonomen Fahren. Springer-Vieweg, Wiesbaden, 2. Auflage, 2020

[She96] SHEARD, S.: Twelve Systems Engineering Roles. INCOSE International Symposium 6, 1996

[SMK+17] SEXTRO, W.; MEYER, T.; KAUL, T.; KIMOTHO, J. K.: Entwicklung verlässlicher, intelligenter Systeme. In: 17.–18. Mai 2017, 28. Tagung Technische Zuverlässigkeit - Entwicklung und Betrieb zuverlässiger Produkte. Verein Deutscher Ingenieure, Leonberg, DE, 2017, S. 17–30

[SNH+10] SEEBACH, H.; NAFZ, F.; HOLTMANN, J.; TICHY, M.; REIF, W.; SCHÄFER, W.: Designing Self-healing in Automotive Systems. In: Xie, B.; Branke, J.; Sadjadi, M.; Zhang, D.; Zhou, X. (Eds.). Oct. 26–29, 2010, International Conference on Autonomic and Trusted Computing. ATC 2010. Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 2010, pp. 194–203

[SNS+10] SEEBACH, H.; NAFZ, F.; STEGHÖFER; REIF, W.: A Software Engineering Guideline for Self-Organizing Resource-Flow Systems. In: Sept. 27–28, 2010, 2010 IEEE 4th International Conference on Self-Adaptive and Self-Organizing Systems, IEEE, Budapest, Ungarn, 2010, pp. 194–203

[SNS+11] SEEBACH, H.; NAFZ, F.; STEGHÖFER, J.-P.; REIF, W.: How to Design and Implement Self-organising Resource-Flow Systems. In: Müller-Schloer, C.; Schmeck, H.; Ungerer, T.: Organic Computing - A Paradigm Shift for Complex Systems. Birkhäuser Verlag, Basel, 2011, pp. 145–161

[SO19] SINDRE, G.; OPDAHL, A.: Capturing security requirements through misuse cases, 2019

[Son15] SONDERMANN-WÖLKE, C.: Entwurf und Anwendung einer erweiterten Zustandsüberwachung zur Verlässlichkeitssteigerung selbstoptimierender Systeme. Dissertation, Lehrstuhl für Mechatronik und Dynamik der Universität Paderborn, Shaker Verlag, Aachen, 2015

[SPA+18] SCHLUSE, M.; PRIGGEMEYER, M.; ATORF, L.; ROSSMANN, J.: Experimentalbe Digital Twins - Streamlining Simulation-Based Systems Engineering for Industry 4.0. IEEE Transactions on Industrial Informatics 14, IEEE, New Jersey, 2018, pp. 1722–1731

[SSL15] STAHL, K.; STÖCKLEIN, J.; LI, S.: Evaluation of Autonomous Approaches using Virtual Environments. In: Shumaker, R.; Lackey, S. (Eds.). Aug. 2–7, 2015, Virtual, Augmented and Mixed Reality VAMR 2015 Held as Part of HCI International 2015, Springer International Publishing, Los Angeles, USA, 2015, pp. 499–512

[SSN11] SIEGWART, R.; SCARAMUZZA, D.; NOURBAKHSH, I.: Introduction to Autonomous Mobile Robots. MIT Press, Cambridge, 2. Auflage, 2011

[Sta15] STAHL, K.: Online Anomaly Detection for Reconfigurable Self-X Real-Time Operating Systems. A Danger Theory-Inspired Approach. Dissertation, Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Paderborn, 2015

[Str98] STRUBE, G.: Modelling Motivation and Action Control in Cognitive Systems. Mind Modelling, 1998, pp. 89–109

[SVE+10] SVENNINGSSON, R.; VINTER, J.; ERIKSSON, H.; TÖRNGREN, M.: MODIFI: A MODel-Implemented Fault Injection Tool. Computer Safety, Reliability, and Security, Springer-Verlag, Berlin, 2010, pp. 210–222

[TH18] TRISTRAM, F.; HILLEBRAND, M. (BETREUER): Simian Army Testing for Autonomous Systems, Universität Paderborn, Heinz Nixdorf Institut, unveröffentlichte Bachelorarbeit, Paderborn, 2018

[THC13] TOMFORD, S.; HÄHNER, J.; CHRISTIAN, M.-S.: Incremental design of organic computing systems - moving design from design-time to runtime. In: July 29–31, 2013, 10th International Conference on Informatics in Control, Automation and Robotics, Rejkjavik, Iceland, 2013, pp. 185–192

[Tho08] THOMPSON, M.: Testing the Intelligence of Unmanned Autnomous Systems. ITEA Journal 29, ITEA, Fairfax, 2008, pp. 380–387

[TM13] TOMFORD, S.; MÜLLER-SCHLOER, C.: Incremental Design of Adaptive Systems. Journal of Ambient Intelligence and Smart Environments 1, IOS Press, Amsterdam, 2013, pp. 179–198

[Trö18] TRÖGER, P.: Unsicherheit und Uneindeutigkeit in Verlässlichkeitsmodellen. Springer Vieweg, Wiesbaden, 2018

[Tsc16] TSCHIRNER, C.: Rahmenwerk zur Integration des modellbasierten Systems Engineering in die Produktentstehung mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 366, 2016

[TSK14] TAN, P.-N.; STEINBACH, M.; KUMAR, V.: Introduction to Data Mining. Pearson International, Essex, 2014

[UMR+15] ULBRICH, S.; MENZEL, T.; RESCHKA, A.; SCHULDT, F.; MAURER, M.: Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In: Sept. 15–18, 2015, 2015 IEEE 18th International Conference on Intelligent Transportation System, IEEE, Las Palmas, Gran Canaria, 2015, pp. 982–988

[UTY13] UMEDA, Y.; TOMIYAMA, T.; YOSHIKAWA, H.: Concepts of Self-Repairing Systems. Measurement and Control 46, Sage, Thousand Oaks, CA, 2013, pp. 176–179

[Vir18-ol] VIRES SIMULATIONSTECHNOLOGIE: OpenSCENARIO Specification. Unter: http://www.openscenario.org, 2018

[Wah17] WAHLSTER, W.: Künstliche Intelligenz als Grundlage autonomer Systeme. Informatik Spektrum 40 (5), Springer-Verlag, Heidelberg, 2017, S. 409–418

[Wei14] WEILKIENS, T.: Systems Engineering mit SysML/UML. d.punkt Verlag, Heidelberg, 3. Auflage, 2014

[Wes17] WESTERMANN, T.: Systematik zur Reifegradmodell-basierten Planung von Cyber-Physical Systems des Maschinen- und Anlagenbaus. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 375, 2017

[WF13] WANDER, A.; FÖRSTNER, R.: Innovative fault detection, isolation and recovery on-board spacecraft: Study and implementation using cognitive automation. In: Oct. 9–11, 2013, 2013 Conference on Control and Fault-Tolerant Systems (SysTol), Nice, France, 2013, pp. 336–341

[WFJ+17] WANG, L.; FAHRENKROG, F.; JUNG, O.; KATES, R.: Prospective safety assessment of highly automated driving functions using stochastic traffic simulation, 2017

[WKG09] WEGENER, J.; KRUSE, P.; GROSS, H.: Evolutionäre Funktionstests für MiL-, SiL-, und HiL-Umgebungen. OBJEKTspektrum Ausgabe Testing, SIGS DATACOM, Troisdorf, 2009

[YTL+01] YAM, R.; TSE, P.; LI, L.; TU, P.: Intelligent predictive decision support system for condition-based maintenance. The International Journal of Advanced Manufacturing Technology 17, Springer-Verlag, New York, 2001, pp. 383–391

[ZAV04] ZIADE, H.; AYOUBI, R.; VELAZCO, R.: A Survey on Fault Injection Techniques. The International Arab Journal of Information Technology (IAJIT) 1 (2), 2004, pp. 171–186

Normen und Richtlinien

[DIN17359] Zustandsüberwachung und -diagnostik von Maschinen - Allgemeine Anleitung. Beiblatt 1: Erläuterung zu Fachbegriffen, Deutsches Institut für Normung (DIN) e.V., Beuth Verlag, Berlin, 2017

[DIN19226] Leittechnik, Regelungstechnik und Steuerungstechnik, Allgemeine Grundbegriffe, Deutsches Institut für Normung (DIN) e.V., Beuth Verlag, Berlin, 1994

[DIN61025] Fehlzustandsbaumanalyse, Deutsches Institut für Normung (DIN) e.V., Beuth Verlag, Berlin, 2007

[DIN9000] Qualitätsmanagement - Grundlagen und Begriffe, Deutsches Institut für Normung (DIN) e.V., 2005

[ECSS-E-ST-10C] Space engineering - System engineering general requirements, European Cooperation for Space Standardization (ECSS), 2009

[ECSS-Q-40-04A] Sneak-Analyse - Teil 1: Methode und Durchführung, European Cooperation for Space Standardization (ECSS), 1997

[ECSS-Q-40-4A-2] Sneak analysis - Part 2: Clue list, European Cooperation for Space Standardization (ECSS), 1997

[ECSS-Q-40A-1] Raumfahrtproduktsicherung, Sicherheit, European Cooperation for Space Standardization (ECSS), 1996

[ECSS-Q-ST-30C] Space product assurance, European Cooperation for Space Standardization (ECSS), 2017

[IEC61508a] Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/ programmierbarer elektronischer Systeme - Begriffe und Abkürzungen, International Electrotechnical Commission, Beuth Verlag, Berlin, 2007

[IEC61508b] Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/ programmierbarer elektronischer Systeme - Überblick über Verfahren und Maßnahmen, International Electrotechnical Commission, Beuth Verlag, Berlin, 2010

[IEE42010] Systems and software engineering. Architecture description, Institute of Electrical und Electronics Engineers (IEEE), New York, 2011

[IEEE1362] IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document, Institute of Electrical und Electronics Engineers (IEEE), New York, 1998

[ISO 21448] Road vehicles - Safety of the intended functionality, International Organization for Standardization (ISO), Geneva, 2019

[ISO12100] Safety of Maschinery - General Principles for Design - Risk Assessment and Risk Reduction, International Standardisation Organisation (ISO), Geneva, 2010

[ISO13381] Zustandsüberwachung und -diagnostik von Maschinen - Prognose. Teil 1: Allgemeine Anleitung, Deutsches Institut für Normung (DIN) e.V., Beuth Verlag, Berlin, 2015

[ISO1471] Recommended practice for architectural description of software-intensive systems, Institute of Electrical und Electronics Engineers (IEEE), New York, 2010

[ISO19450] Automation Systems and Integration, International Standardisation Organisation (ISO), Geneva, 2015

[ISO26262] Road vehicles - Functional Safety - Part 3: Concept Phase, International Organization for Standardization (ISO), Geneva, 2018

[VDI2206] Entwurfsmethodik für mechatronische Systeme, Verein Deutscher Ingenieure (VDI) e.V., Beuth Verlag, Berlin, 2004

[VDI4001] Terminologie der Zuverlässigkeit, Verein Deutscher Ingenieure (VDI) e.V., Beuth Verlag, Berlin, 2006

[VDI4003] Zuverlässigkeitsmanagement, Verein Deutscher Ingenieure (VDI) e.V., Beuth Verlag, Berlin, 2007

[VDI4008] Methoden der Zuverlässigkeit, Zustandsflussgraphen, Verein Deutscher Ingenieure (VDI) e.V., Beuth-Verlag, Berlin, 2014

Anhang

Inhaltsverzeichnis

Seite

A1 Ergänzungen zum Stand der Technik

A1.1 AUTOSAR Adaptive

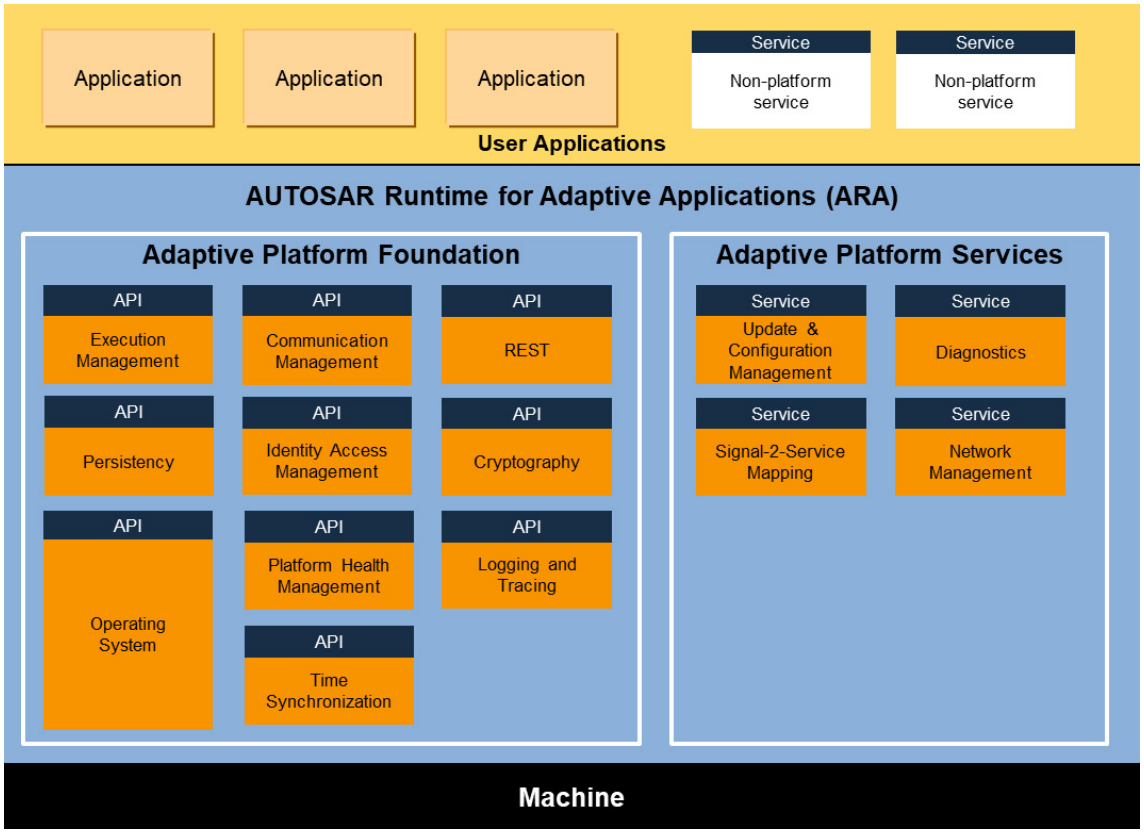


Bild A-1: AUTOSAR

A1.2 Zustandsüberwachung für selbstoptimierende Systeme

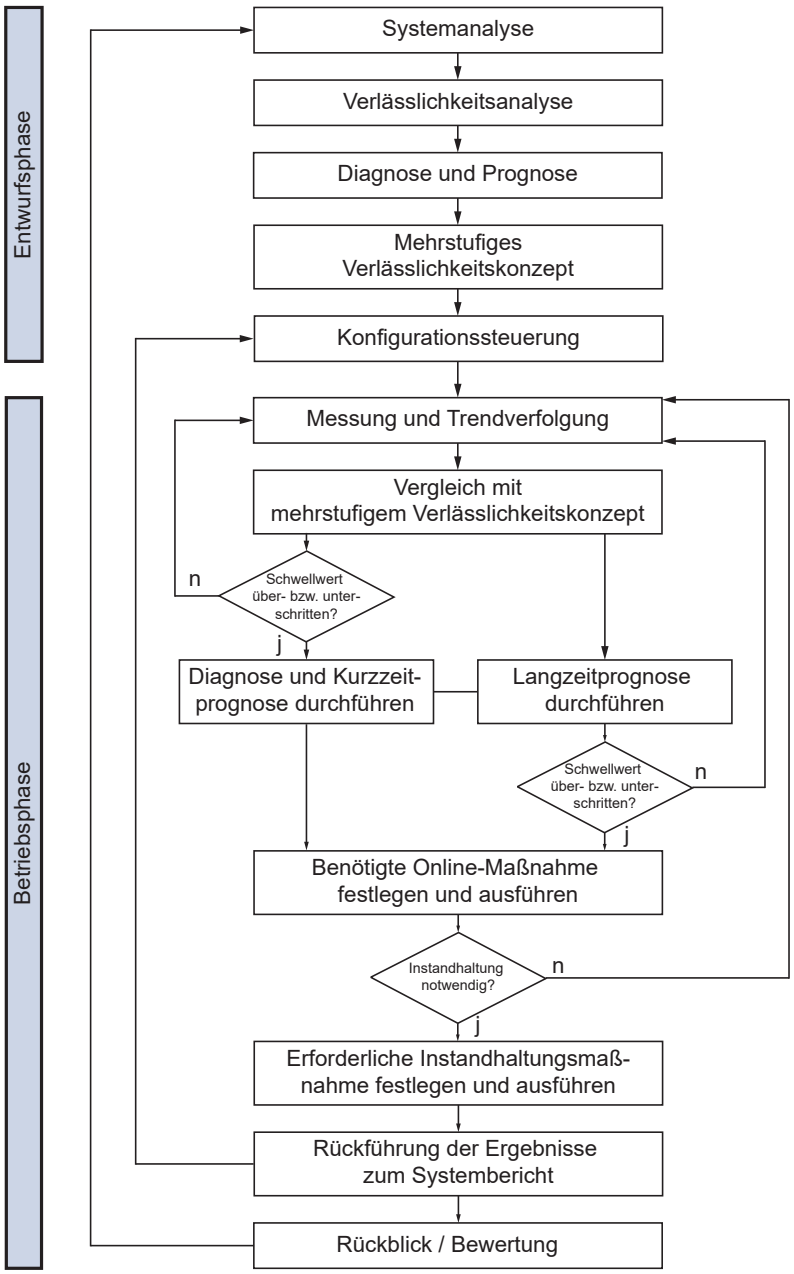


Bild A-2: Zustandsüberwachung für selbstoptimierende Systeme ([Son15], S. 41)

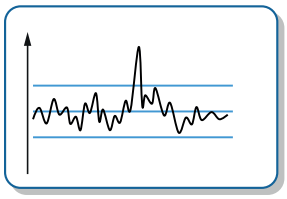
A2 Vertiefende Informationen zu der Systematik

A2.1 Konstruktionsprinzipien

A2.1.1 Konstruktionsprinzipien: Anomalie Detektion

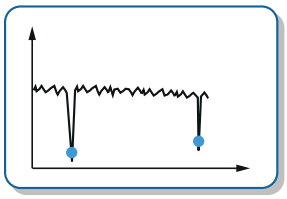
Funktionen

Range Detektor



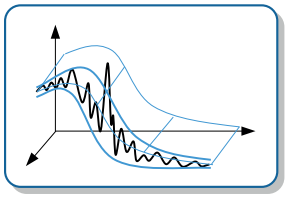
Beschreibung: Ein statistisches Verfahren zur unüberwachten Detektion von Anomalien in einem Signalverlauf. Jedem Datenpunkt wird ein Label {normal, anormal} zugewiesen. Das Verfahren erkennt statistisch Ausreißer in einem Signalverlauf.

Kalman-ARIMA



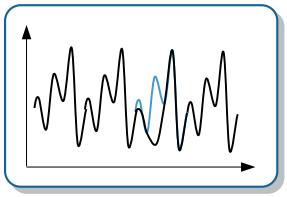
Beschreibung: Ein statistisches Verfahren zur unüberwachten Detektion von Anomalien in einem Signalverlauf Der Kalman-Filter liefert eine Prädiktion. Die Anomalie Detektion erfolgt durch Ermittlung des Residuums zwischen Prädiktion und Signalwert unter Berücksichtigung eines Schwellwertes. Das Verfahren liefert einen Score.

Multivariate Gauß



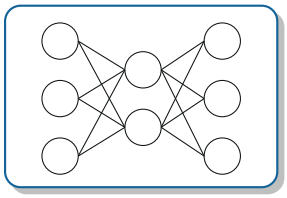
Beschreibung: Ein statistisches und multivariates Verfahren zur unüberwachten Detektion von Anomalien in Signalen. Das Verfahren wird auf dem Normalverhalten trainiert und lernt eine statistische Verteilung der Signale. Statistische Ausreißer gelten als Anomalie. Das Verfahren liefert einen Score.

Matrix Profil



Beschreibung: Das Matrix Profil ist ein unüberwachtes statistisches Verfahren. Eine Zeitserie wird durch Fensterung in Abschnitt zerlegt. Für diesen Abschnitt wird die Distanz zu den anderen Abschnitten ermittelt. Bei einer hohen Distanz ist der Signalabschnitt einzigartig und stellt somit eine Anomalie dar.

Autoencoder

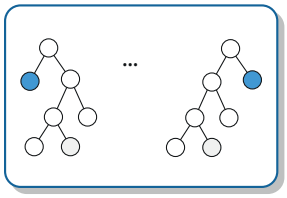


Beschreibung: Ein Autoencoder ist ein semi-überwachtes und konnektivistisches Verfahren zur Anomalie Detektion in Signalen.Das Verfahren wird auf dem Normalverhalten trainiert. Es rekonstruiert das Signal. Der Rekonstruktionsfehler repräsentiert eine Anomalie im Signal.

Bild A-3: Ausgewählte Verfahren zur Anomalie Detektion in Funktionen

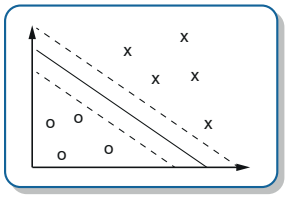
Zustand

Isolation Forest



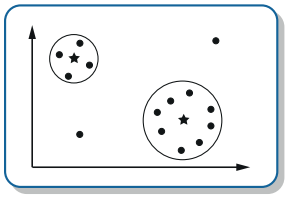
Beschreibung: Ein Verfahren zu unüberwachten Klassifikation von Zustandsanomalien. Das Verfahren bildet einen Cluster und misst, wie weit einzelne Beobachtungen vom Schwerpuntk des Clusters entfernt sind. Die Beurteilung einer Anomalie erfolgt dann durch die Vorgabe von einem statistischen Schwellwert.

One-Class SVM



Beschreibung: Ein Verfahren zur unüberwachten Detektion von Zustandsanomalien. Jedem Datenpunkt wird ein Label {normal, anormal} zugewiesen. Eine Hyperebene separiert die Klasse. Kann eine neue Beobachtung keiner Klasse zugewiesen werden, gilt sie als Anomalie.

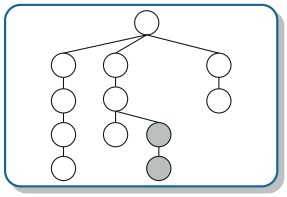
K-Means



Beschreibung: Ein Verfahren zur unüberwachten Detektion von Zustandsanomalien. Beobachtungen mit ähnlichen Merkmalen werden zu einer Gruppe zusammengefasst. Ein Beobachtungswert, der den bestehenden Cluster nicht zugewiesen werden kann, repräsentiert eine Anomalie. Das Verfahren liefert einen Score.

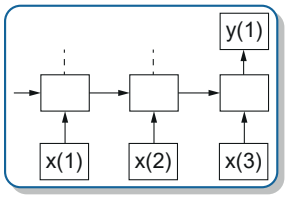
Sequenzen

Suffix Baum



Beschreibung: Ein statistisches Verfahren zur unüberwachten Detektion von Anomalien in Sequenzen (z.B. SysCall Sequenzen). In Kombination mit einem Counter können Zeichenketten identifiziert werden, die bisher nicht aufgetreten sind und als Anomalie gelten.

LSTM



Beschreibung: Ein konnektivistisches Verfahren zu semi-überwachten Detektion von Anomalien in Sequenzen. Das Long Short-Term Memory wird auf Basis des Normalverhaltens trainiert. Das trainierte Modell liefert eine Prädiktion basierend auf den vorherigen Werten. Die Anomalie Detektion erfolgt durch Abgleich zwischen der Prädiktion und der Beobachtung unter Berücksichtigung eines Schwellwertes.

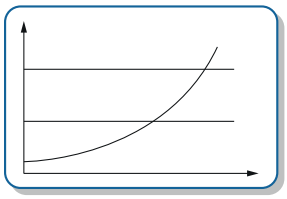
Bild A-4: Ausgewählte Verfahren zur Anomalie Detektion in Zustand und Sequenzen

A2.1.2 Konstruktionsprinzipien: Gesundheitsmonitor

Deskriptiv

Schwellwert

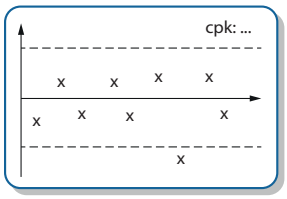
● Zustand ● Leistung ● Ergebnis



Beschreibung: Überwachung eines direkt oder indirekt messbaren Zustandes in Hinblick auf das Überschreiten von Toleranzgrenzen (Warn-/Eingriffsgrenzen). Erweiterte Analyse durch Signalverarbeitung, z.B. Frequenzen, Varianzen, Amplituden etc.

Statistische Analyse

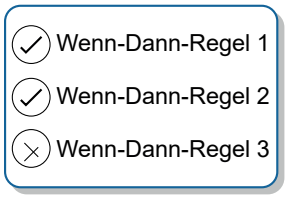
○ Zustand ● Leistung ● Ergebnis



Beschreibung: Statistische Analyse von Prozessparametern. Ermittlung von Parametern der Lage (z.B. Mittelwert) und Streuung (z.B. Varianz). Ermittlung von Warn- und Eingriffsgrenzen. Berechnung der Prozessfähigkeit.

Plausibilität

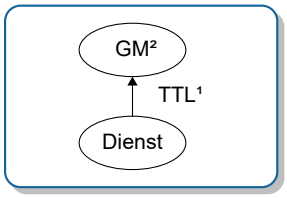
● Zustand ○ Leistung ● Ergebnis



Beschreibung: Regelbasierte Überprüfung eines oder mehrerer Signale auf ihre Plausibilität. Das bedeutet, dass Signale in Hinblick auf ihre Glaubwürdigkeit und deren Kompatibilität untereinander bewertet werden. Dies kann durch logische Regeln erfolgen.

Time-To-Live

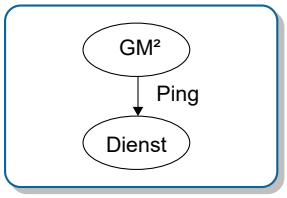
● Zustand ○ Leistung ○ Ergebnis



Beschreibung: Bei diesem Verfahren bestätigt das zu überwachende System, dass es betriebsbereit ist. Dazu meldet es periodisch ein TTL-Signal an den Gesundheitsmonitor. Der Monitor verfolgt den letzten gesendeten Zustand. Aktualisiert sich der Zustand nicht innerhalb einer definierten Periode, geht der Monitor von einem Ausfall aus.

Ping

● Zustand ○ Leistung ○ Ergebnis



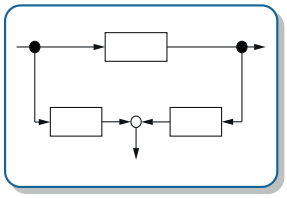
Beschreibung: Das Verfahren überprüft die Verfügbarkeit eines Dienstes. Dazu wird periodisch durch den Gesundheitsmonitor ein Ping an den Dienst geschickt. Wenn der Dienst nicht innerhalb einer definierten Zeitspanne auf den Ping reagiert, wird von einem Ausfall ausgegangen.

Bild A-5: Ausgewählte deskriptive Verfahren zur Überwachung der Gesundheit

Deskriptiv

Modellgetrieben

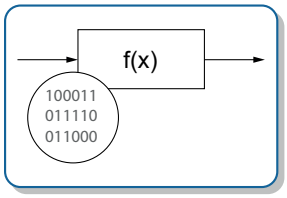
● Zustand ● Leistung ○ Ergebnis



Beschreibung: Modellierung und Verwendung eines mathematisch-physikalisch korrekten Prozessmodells. Das Prozessmodell bildet Zustand und Verhalten (z.B. Dynamik) des Systems ab. Das Monitoring erfolgt in Verbindung mit Schätzverfahren. Kennwerte sind Parameter, Zustandsgrößen oder Residuen.

Datengetrieben

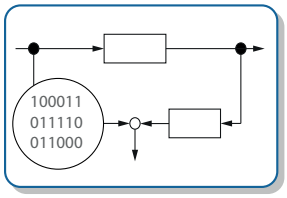
● Zustand ● Leistung ○ Ergebnis



Beschreibung: Trainineren und Verwendung eines datengetriebenen Prozessmodells. Lernen einer Funktion, die das Verhalten oder den Zustand anhand von Eingangs- und Ausgangsdaten abbildet. Kennwerte sind Parameter, Zustandsgrößen oder Paritätsgleichungen (z.B. Cluster-Verfahren zur Fehlererkennung)

Hybride Modellierung

● Zustand ● Leistung ○ Ergebnis

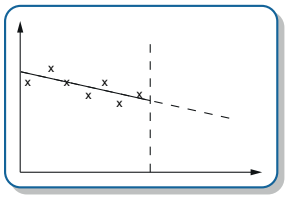


Beschreibung: Kombination von datengetriebenen und modellgetriebenen Modellen.

Prognostisch

Trendanalyse

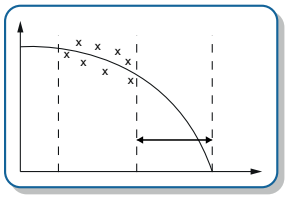
● Zustand ● Leistung ● Ergebnis



Beschreibung: Verwendung von Regressionsverfahren zur Analyse von Trends. Die Regression wird verwendet, um Parameter oder Zustandsgrößen zur prognostizieren. Einsatz in Kombination mit anderen Verfahren wie etwa Schwellwert.

Restnutzungsdauer

● Zustand ○ Leistung ○ Ergebnis



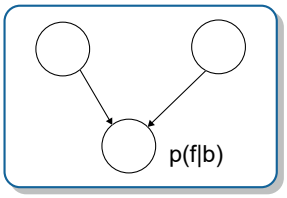
Beschreibung: Verwendung von Belastungs- oder Degenerationsmdoellen zur Abschätzung der verbleibenden Lebensdauer einer Einheit, z.B. Schätzung der Lebensdauer eines Akkumulators.

Bild A-6: Ausgewählte prognostische Verfahren zur Überwachung der Gesundheit

A2.1.3 Konstruktionsprinzipien: Selbstdiagnostik

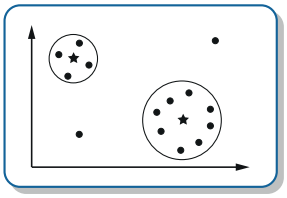
Klassifikation

Bayes Netz



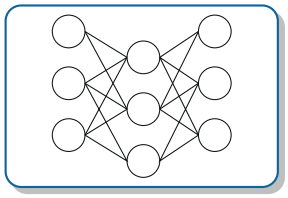
Beschreibung: Bayes Netze dienen der Repräsentation von Beobachtungen und abgeleiteten Schlussfolgerungen. Zum Einsatz kommt das Verfahren bei bedingten Wahrscheinlichkeiten zwischen Beobachtung und Ursache. Die Struktur wird meist manuell erstellt und die bedingten Wahrscheinlichkeiten aus Daten gelernt [Ise08].

Clustering



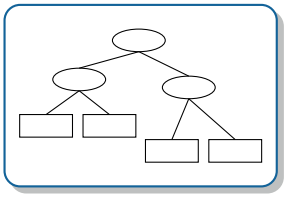
Beschreibung: Ein Verfahren zur unüberwachten Diagnose von Fehlzuständen. Gesundheitszustände mit ähnlichen Merkmalen bilden eine Gruppe und deuten auf ein gemeinsames Fehlerbild [ZCB18].

Neuronales Netz



Beschreibung: Der Zusammenhang zwischen Gesundheitszustand und der Ursache wird durch ein Neuronales Netz trainiert. Zur Laufzeit können die Daten auf bekannte Muster hin analysiert und mögliche Fehlerursachen abgeleitet werden [GBC16].

Entscheidungsbaum

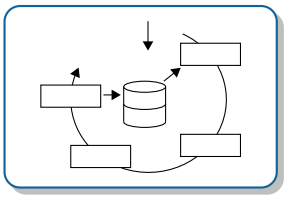


Beschreibung: Geordnete und gerichtete Bäume mit denen Entscheidungsregeln dargestellt werden. Ihre Verwendung eignet sich, wenn Diagnosewissen automatisiert klassifiziert werden kann und aus Erfahrungen formale Reglen hergeleitet werden können. Ein Knoten repräsentiert eine logische Regel [Ise08].

Bild A-7: Ausgewählte Konstruktionsprinzipien: Selbstdiagnostik (I/II)

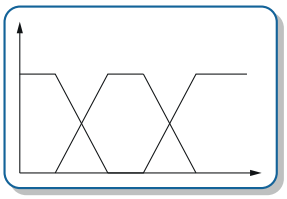
Inferenz

Fallbasiertes Schließen



Beschreibung: Mit diesem Verfahren wird das Expertenwissen als kausale Verknüpfung oder in der Form von Wenn-Dann-Regeln in einer Fallbasis gespeichert. Tritt eine Beobachtung auf, wird über den Analogieschluss eine Ursache ermittelt [Ise08].

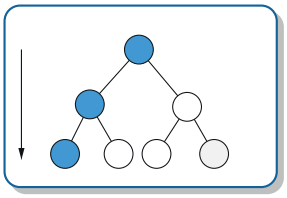
Fuzzy-Logik



Beschreibung: Das Verfahren erweitert das Fallbasierte Schließen um Unschärfe. Beobachtungen sind unscharfe Größen. Mit der Fuzzy-Logik erfolgt ein approximatives Schließen auf eine Ursache [Ise08].

Regression

Fuzzy-Pattern-Tree



Beschreibung: Diagnose eines Fehlerbildes entlang einer hierarchisches Struktur mittels unscharfer Logik. Ermittlung der betreffenden Gesundheitssignale und Lokalisierung in der Struktur [Sen14].

Bild A-8: Ausgewählte Konstruktionsprinzipien: Selbstdiagnostik (II/II)

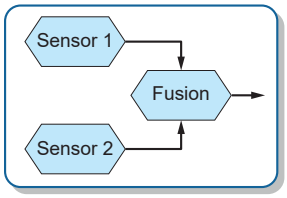
A2.1.4 Konstruktionsprinzipien: Selbstheilungsaktionen

Parameteranpassung

Datenfusion

Architektur: ○ Level 0 ● Level 1 ○ Level 2 ○ Level 3

Regulierung: ● Nicht-Kognitiv ○ Assoziativ ○ Kognitiv

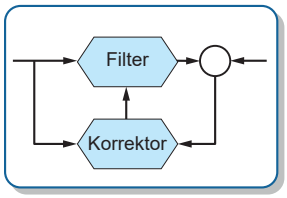


Beschreibung: Verknüpfung der Ausgabedaten von verschiedenen bzw. mehreren Sensoren. Durch die Datenfusion kann ein Ausfall erkannt und kompensiert werden. Unterscheidung nach Datenfusion, Merkmalsfusion oder Entscheidungsfusion.

Adaption

Architektur: ● Level 0 ○ Level 1 ○ Level 2 ○ Level 3

Regulierung: ● Nicht-Kognitiv ○ Assoziativ ○ Kognitiv

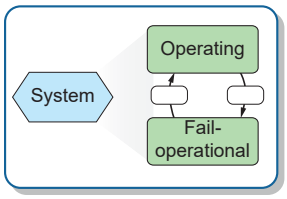


Beschreibung: Ausgleich stochastischer Fehler in den Eingangssignalen durch adaptive Filter. Die Filterparameter werden zur Laufzeit angepasst. Wenn sich z.B. ein Signal verschlechter, vergrößert sich die Kovarianz.

Skalierung

Architektur: ● Level 0 ○ Level 1 ○ Level 2 ○ Level 3

Regulierung: ● Nicht-Kognitiv ○ Assoziativ ○ Kognitiv

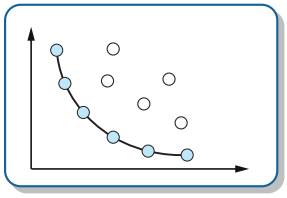


Beschreibung: Veränderung von Leistungsparametern, d.h. Reduzierung bzw. Erhöhung. Wechsel in einen anderen Betriebsmodus. Zum Beispiel niedrigere Auflösung oder Bildrate einer Kamera.

Mehrzieloptimierung

Architektur: ● Level 0 ● Level 1 ● Level 2 ● Level 3

Regulierung: ○ Nicht-Kognitiv ● Assoziativ ● Kognitiv



Beschreibung: Auswahl von zieloptimalen Paretopunkten. Jeder Punkt umfasst eine Menge von Reglerparameter. In Kombination mit einer adaptiven Regelung und einer Online Bewertung des aktuellen Betriebszustandes. Anpassung der Zielgewichtung i.A. des Betriebszustandes.

Bild A-9: Ausgewählte Verfahren der Parameteranpassung

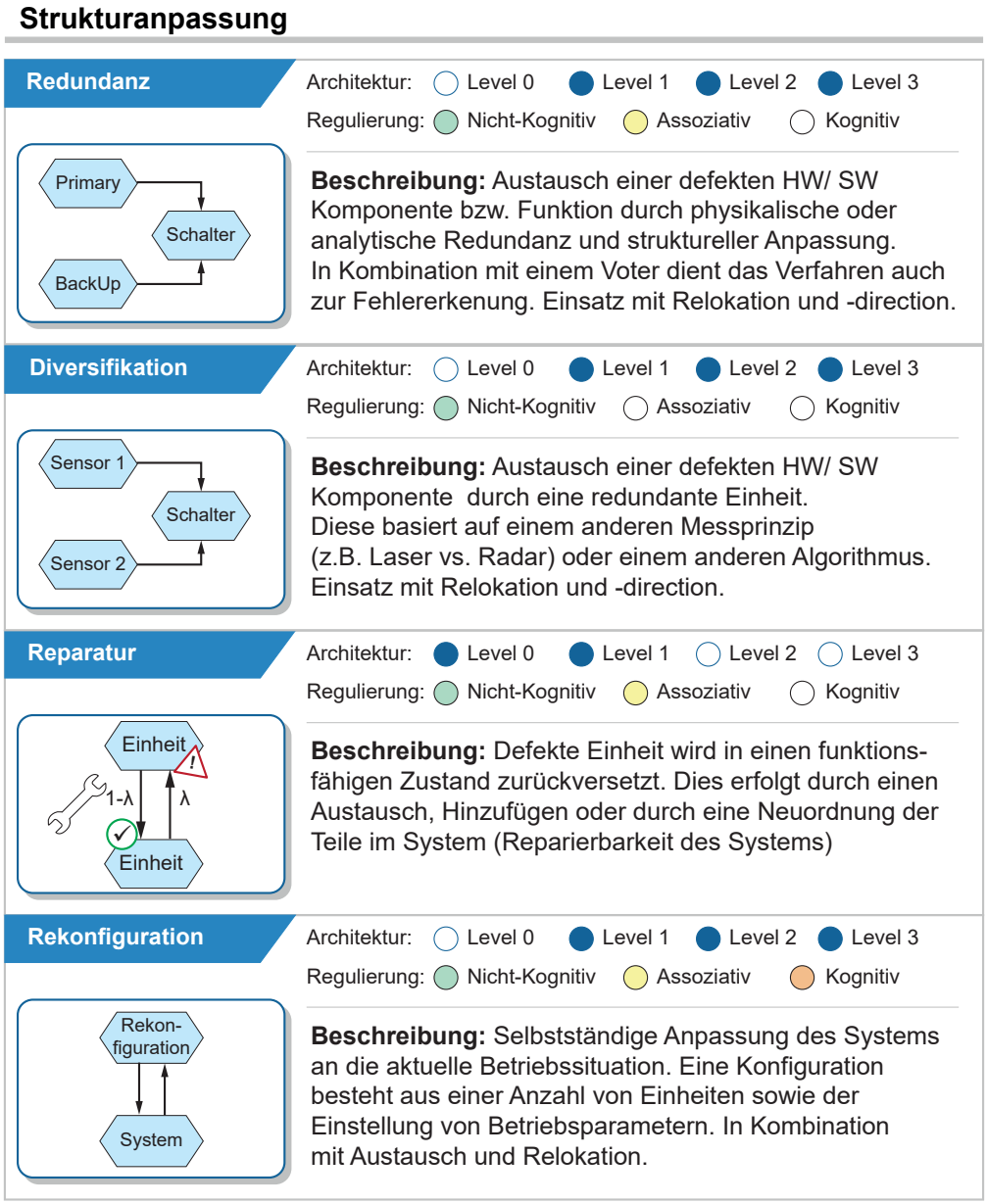


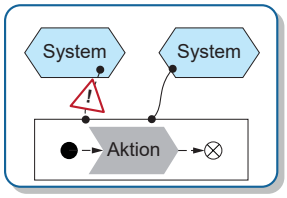
Bild A-10: Ausgewählte Verfahren der strukturellen Anpassung (1/2)

Strukturanpassung

Relokation

Architektur: ○ Level 0 ○ Level 1 ○ Level 2 ● Level 3

Regulierung: ○ Nicht-Kognitiv ● Assoziativ ● Kognitiv

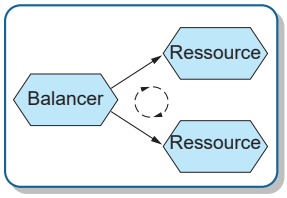


Beschreibung: Verschiebt eine Anwendung oder eine Funktion von einer fehlerhaften Einheit zu einer andere funktionsfähigen Einheit (z.B. Host). Umleitung der Flussbeziheungen (Informations-, und Energieflüsse)

Balancing

Architektur: ○ Level 0 ● Level 1 ● Level 2 ● Level 3

Regulierung: ○ Nicht-Kognitiv ● Assoziativ ● Kognitiv

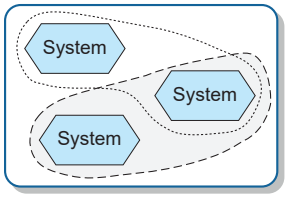


Beschreibung: Die Degeneration wird durch eine Belastung verursacht. In dieser Situation können zusätzl. Ressourcen vorübergehend aktiviert werden, um die Lastsituation auszugleichen. Nach der Lastsituation können die Ressourcen wieder freigegeben werden.

Reorganisation

Architektur: ○ Level 0 ○ Level 1 ○ Level 2 ● Level 3

Regulierung: ○ Nicht-Kognitiv ● Assoziativ ● Kognitiv

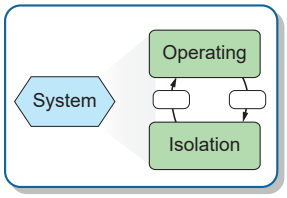


Beschreibung: Für Multi-Agenten Systeme. Ein Agent kann verschiedene Rollen in einem Verbund besitzen. Fällt ein Agent aus. Dann übernimmt ein funktions-fähiger Agent dessen Rolle und Aufgabe.
In Kombination mit Relokation.

Isolation

Architektur: ○ Level 0 ● Level 1 ● Level 2 ● Level 3

Regulierung: ● Nicht-Kognitiv ● Assoziativ ○ Kognitiv



Beschreibung: Das beeinträchtigte Verhalten einer fehlerhaften Einheit wirkt sich nicht auf andere Einheiten des Systems aus. Die fehlerhafte Einheit wird vom System ignoriert und muss seine eigenen Wiederherstellungsroutinen bereitstellen.

Bild A-11: Ausgewählte Verfahren der strukturellen Anpassung (2/2)

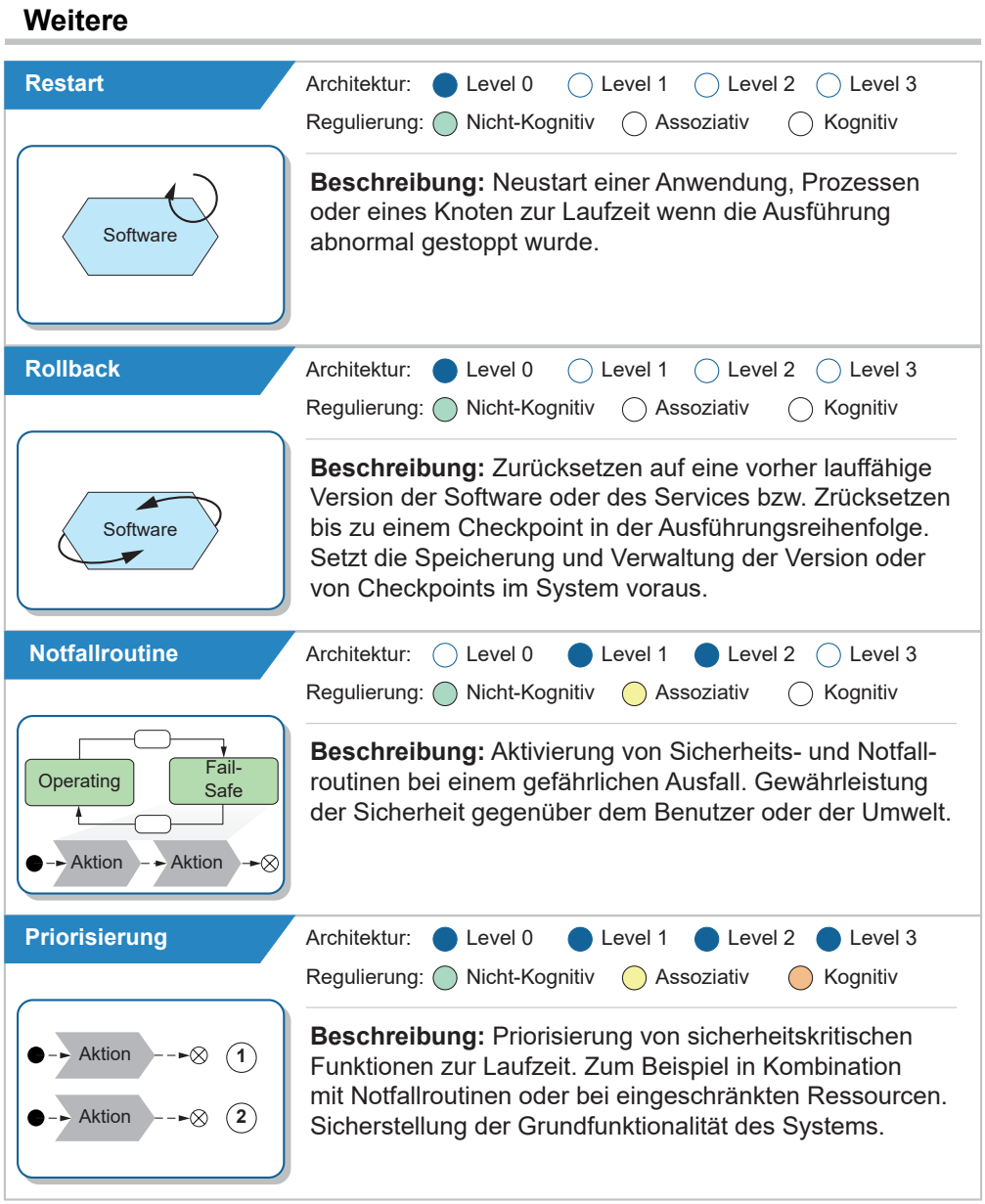


Bild A-12: Weitere Verfahren der Verhaltensanpassung

A2.2 Vertiefende Funktionen der Selbstheilung

A2.2.1 Aggregation im Health Monitor

Die Zuordnung von einem Signalwert auf einen Gesundheitszustand ist unscharf. Daher basiert der Gesundheitsmonitor auf der Fuzzy Logik. Ausgehend von den Signalgeneratoren wird der Gesundheitszustand ermittelt. Hierzu ist die Zugehörigkeitsfunktion mit den Domänenexperten zu definieren. Die Zugehörigkeitsfunktion liegt zwischen 0 und 1 legt die Zugehörigkeit eines Signalwerts für einen Gesundheitszustand fest: *Normal*, *Warning*, *Critical* oder *Fatal*. Zur Modellierung existieren Tools wie die Fuzzy Logic Toolbox in Matlab/Simulink oder die scikit-fuzzy Bibliothek in Python. Das Bild A-13 zeigt die Zugehörigkeitsfunktion für das Bildrauschen.

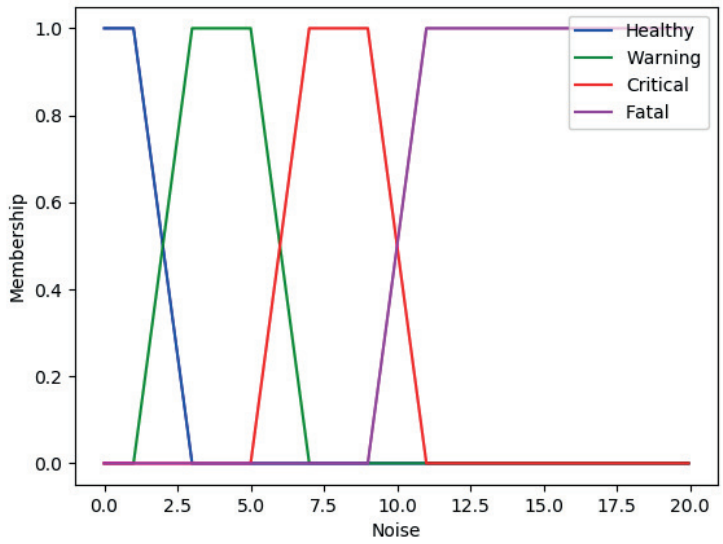


Bild A-13: Zugehörigkeitsfunktion für Bildrauschen

Die Aggregation eines Gesundheitszustand auf die Ebenen Komponente, Funktion und System erfolgt durch Konjunktion, Disjunktion und Averaging. Bei der Konjunktion (AND) müssen alle Kriterien erfüllt sein. Bei der Disjunktion (OR) muss eines der Kriterien erfüllt sein. Beim Averaging wird der Durchschnitt der Kriterien ermittelt. Welche Methode verwendet wird, hängt von der Art des Gesundheitssignals ab. Fällt zum Beispiel eine kritische Komponente aus, kann der Gesundheitszustand der zugehörigen Funktion nicht besser sein, als der Gesundheitszustand der Komponente (ohne Redundanz). Dies entspricht der Disjunktion. Bild A-14 verdeutlicht die Mechanismen der Aggregation.

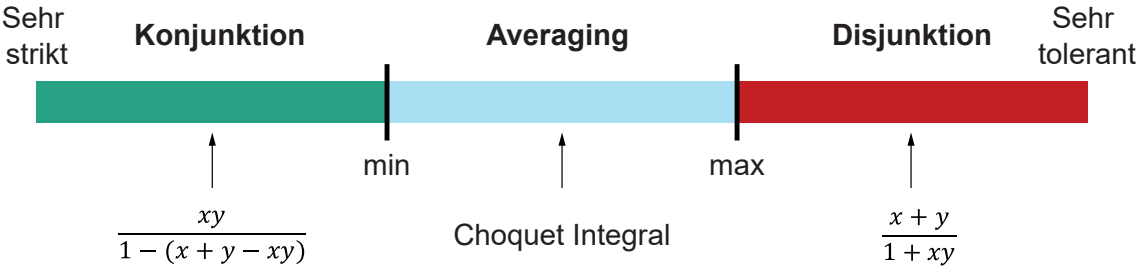


Bild A-14: Aggregation der Gesundheitssignale

Die Mechanismen zur Aggregation sind in den Tabelle A-1 bis A-3 aufgeführt [SH10].

Konjunktion: T-norms		
Name	Definition	Code
Minimum	$\min\{x,y\}$	MIN
Algebraisch	xy	ALG
Lukasiewicz	$\max\{x+y-1,0\}$	LUK
Einstein	$\frac{xy}{2-(x+y-xy)}$	EIN

Tabelle A-1: Mechanismen zur Konjunktion

Disjunktion: T-conorms		
Name	Definition	Code
Minimum	$\max\{x,y\}$	MIN
Algebraisch	$x+y-xy$	COALG
Lukasiewicz	$\max\{x+y,1\}$	COLUK
Einstein	$\frac{x+y}{1+xy}$	COEIN

Tabelle A-2: Mechanismen zur Disjunktion

Konjunktion: T-norms		
Name	Definition	Code
Gew. Durchschnitt	$\alpha x+(1-\alpha)y$	WA
Geord. Gew. Durchschnitt	$\alpha\max\{x,y\}+(1-\alpha)\min\{x,y\}$	OWA

Tabelle A-3: Mechanismen zur Bildung eines Durchschnitts

A2.2.2 Auswahl von Anomalie Detektoren

✓ geeignet - ungeeignet		Range Detektor	Kalman-ARIMA	Multivariate Gauß	Matrix Profil	Auto-encoder
Kriterium	**Ausprägung**	Statistisch	Prädiktion	Statistisch	Statistisch	Konnektion
Eigenschaft des Eingangssignals	Kategorisch	✓	-	✓	-	-
	Periodisch	-	-	-	✓	-
	Quasilinear	-	✓	-	✓	-
	Stochastisch	✓	✓	✓	✓	✓
	Sequenzen	-	-	-	✓	-
Verarbeitung der Anomalie Detekion	Aufzeichnung	-	✓	-	✓	✓
	Batch	✓	-	-	-	✓
	Streaming	✓	✓	✓	-	-
Training und **Daten-verfügbarkeit** für Anomalie Detektion	Überwacht	-	-	✓	-	-
	Semi-Überwacht	✓	-	✓	-	✓
	Unüberwacht	-	✓	✓	✓	✓
	Online	-	✓	-	-	-
Repräsentation der Anomalie in den Daten	Punkt	✓	✓	-	-	✓
	Kontext	-	✓	✓	✓	-
	Kollektiv	-	✓	✓	✓	-
Verhalten der Anomalie Detektion	Funktion	✓	✓	✓	✓	✓
	Zustand	-	-	-	-	-
	Sequenz	-	-	-	-	-
Ergebnis der Anomalie Detektion	Label	✓	-	-	-	-
	Score	-	✓	✓	✓	✓

Bild A-15: Auswahl eines Verfahrens 1/2

✓ geeignet - ungeeignet		Isolation Forest	One-Class SVM	K-Means	LSTM	Suffix Baum
Kriterium	**Ausprägung**	Klassifikation	Klassifikation	Clustering	Konnektion	Statistisch
Eigenschaft des Eingangssignals	Kategorisch	-	-	✓	-	-
	Periodisch	-	-	-	-	-
	Quasilinear	-	-	-	✓	-
	Stochastisch	✓	✓	✓	✓	-
	Sequenzen	-	-	-	✓	✓
Verarbeitung der Anomalie Detekion	Aufzeichnung	✓	✓	-	✓	-
	Batch	✓	✓	✓	✓	-
	Streaming	-	-	-	-	✓
Training und **Daten-verfügbarkeit** für Anomalie Detektion	Überwacht	✓	✓	-	-	-
	Semi-Überwacht	✓	✓	✓	✓	-
	Unüberwacht	✓	✓	-	✓	-
	Online	-	✓	-	-	✓
Repräsentation der Anomalie in den Daten	Punkt	✓	✓	✓	-	-
	Kontext	-	-	-	✓	✓
	Kollektiv	-	-	-	✓	✓
Verhalten der Anomalie Detektion	Funktion	✓	✓	✓	✓	-
	Zustand	✓	✓	✓	-	-
	Sequenz	-	-	-	✓	✓
Ergebnis der Anomalie Detektion	Label	-	✓	-	-	✓
	Score	✓	-	✓	✓	-

Bild A-16: Auswahl eines Verfahrens 2/2

A2.3 Hilfsmittel der Potenzialanalyse

A2.3.1 Risikograph

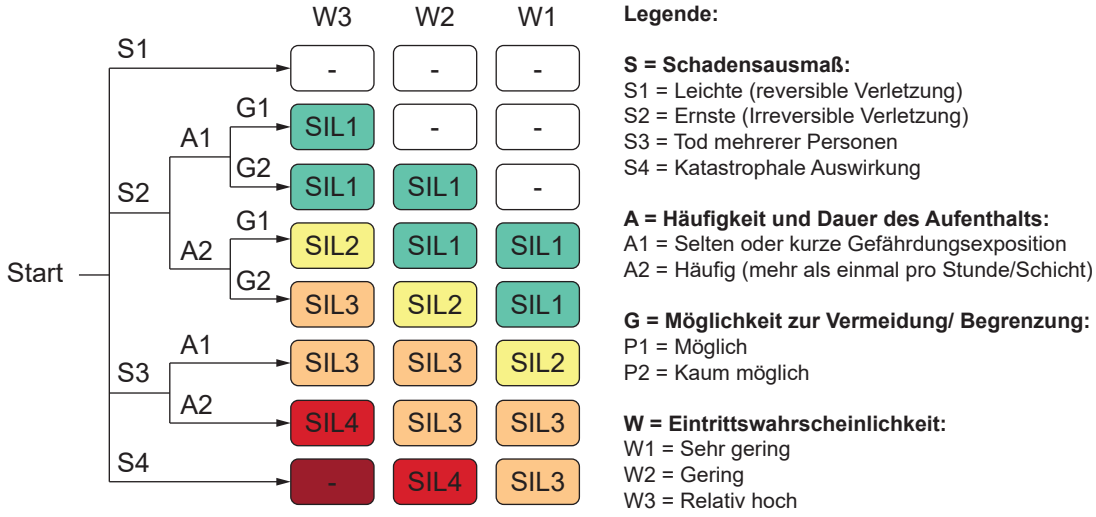


Bild A-17: Ermittlung vom Sicherheitsintegritätslevel (SIL) nach IEC 61508 *[IEC61508a]*

A2.3.2 STPA Profil

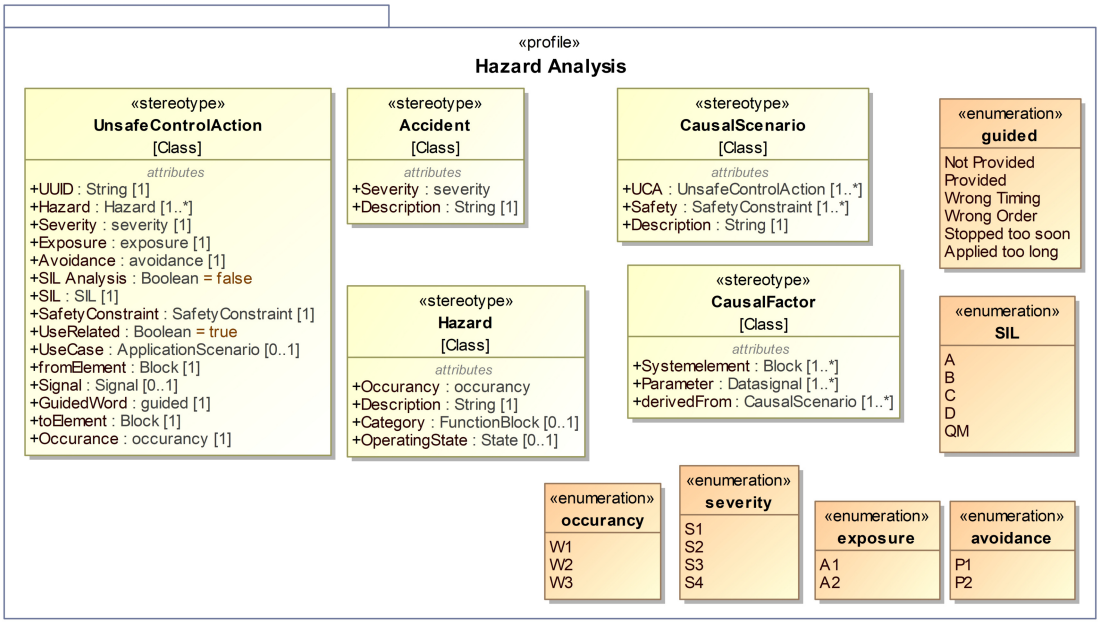


Bild A-18: STPA als Profil der UML/SysML in Cameo Systems Modeller

A2.4 Sprachprofil der SysML/UML

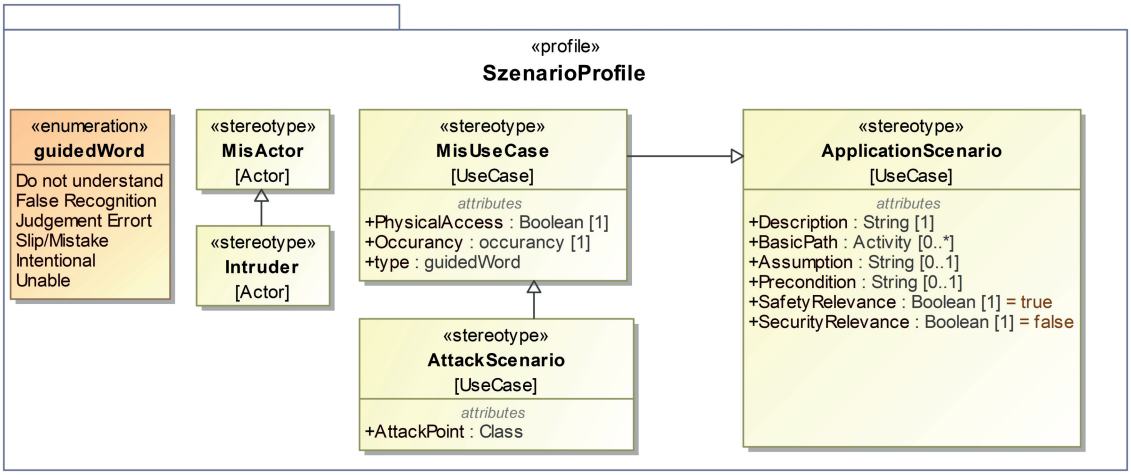


Bild A-19: Profil Szenario

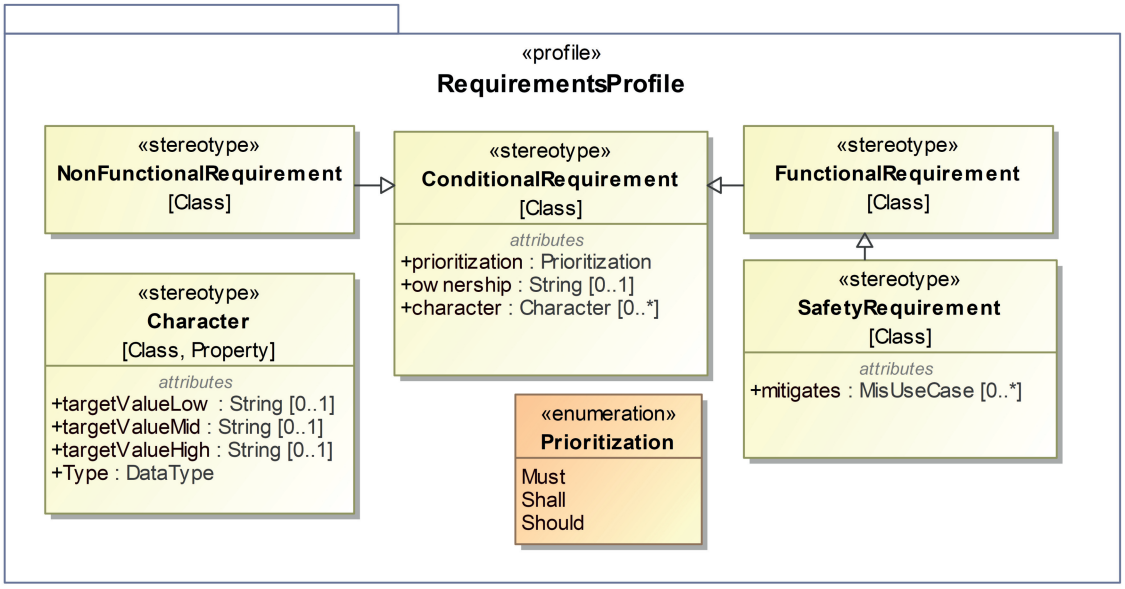


Bild A-20: Profil Anforderungen

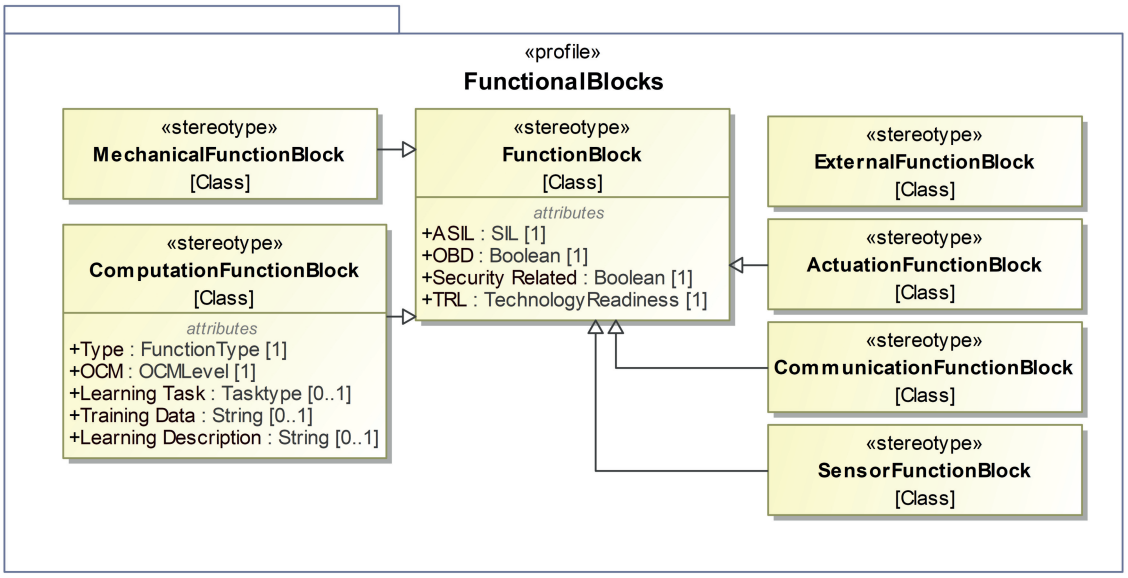


Bild A-21: Profil Funktionen

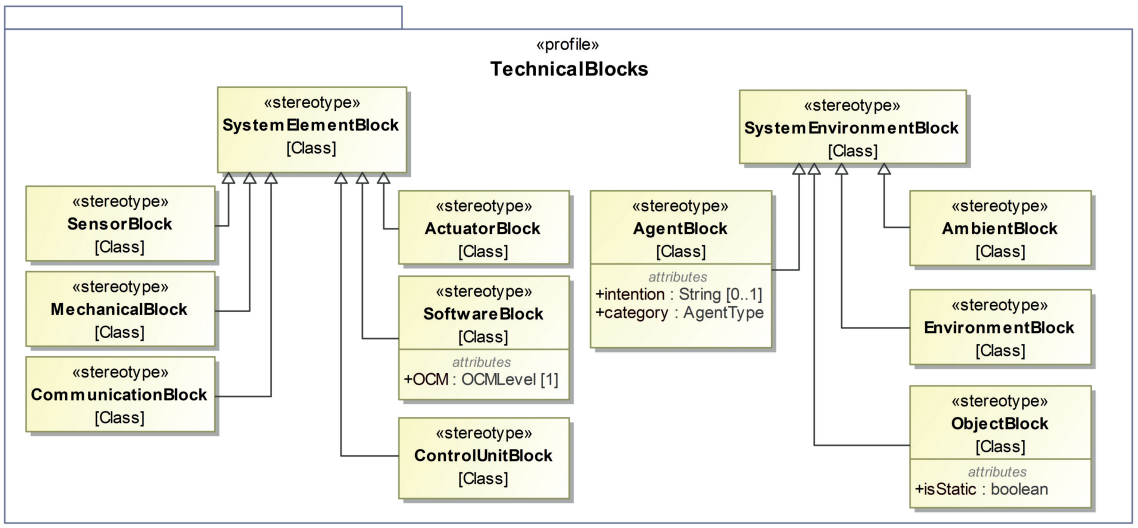


Bild A-22: Profil Systemelemente

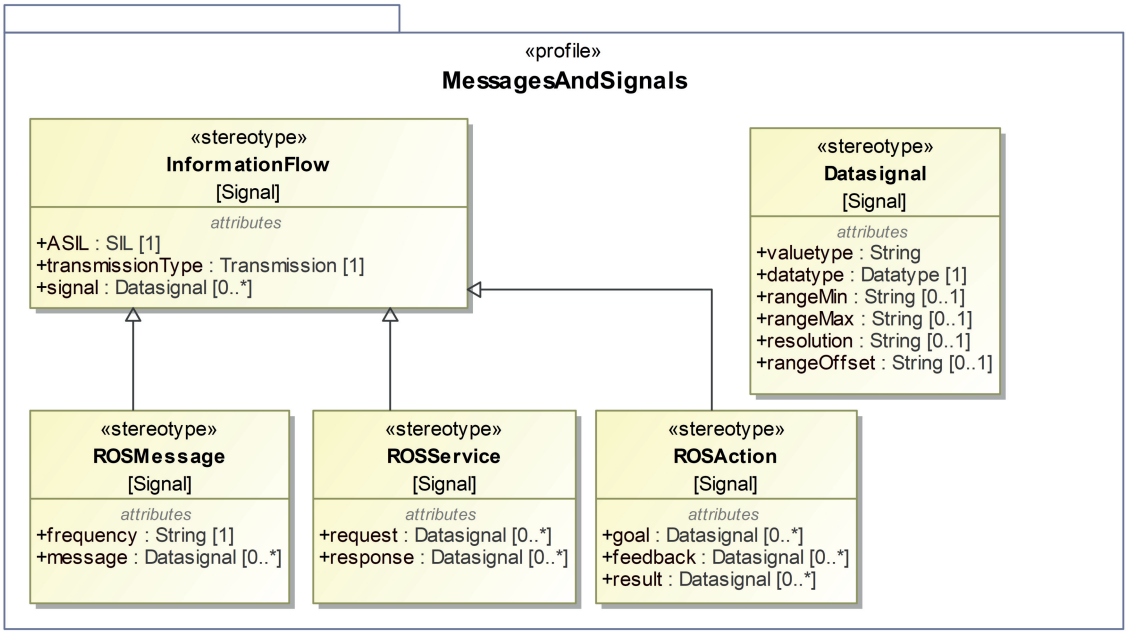


Bild A-23: Signale und ROS Schnittstelle

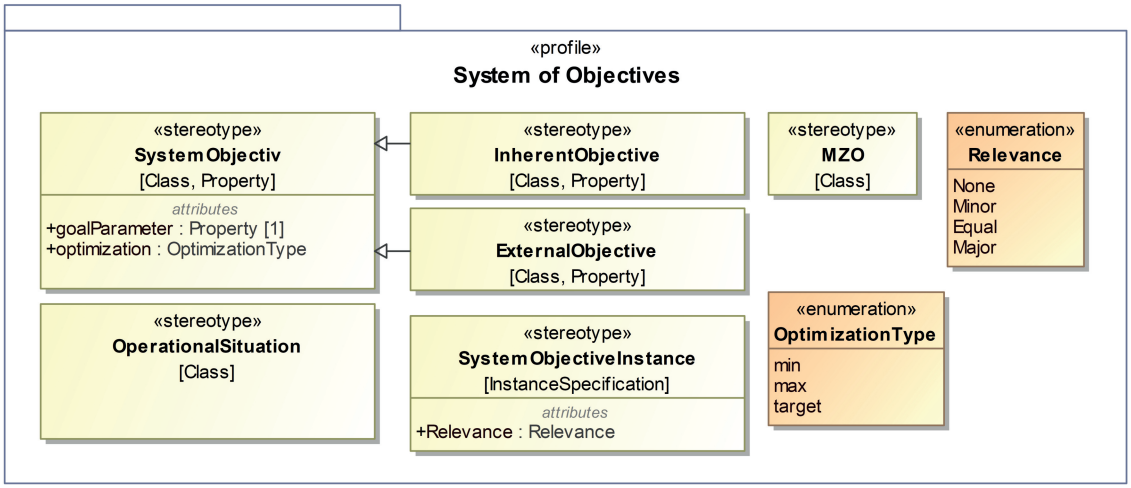


Bild A-24: Profil Zielsystem

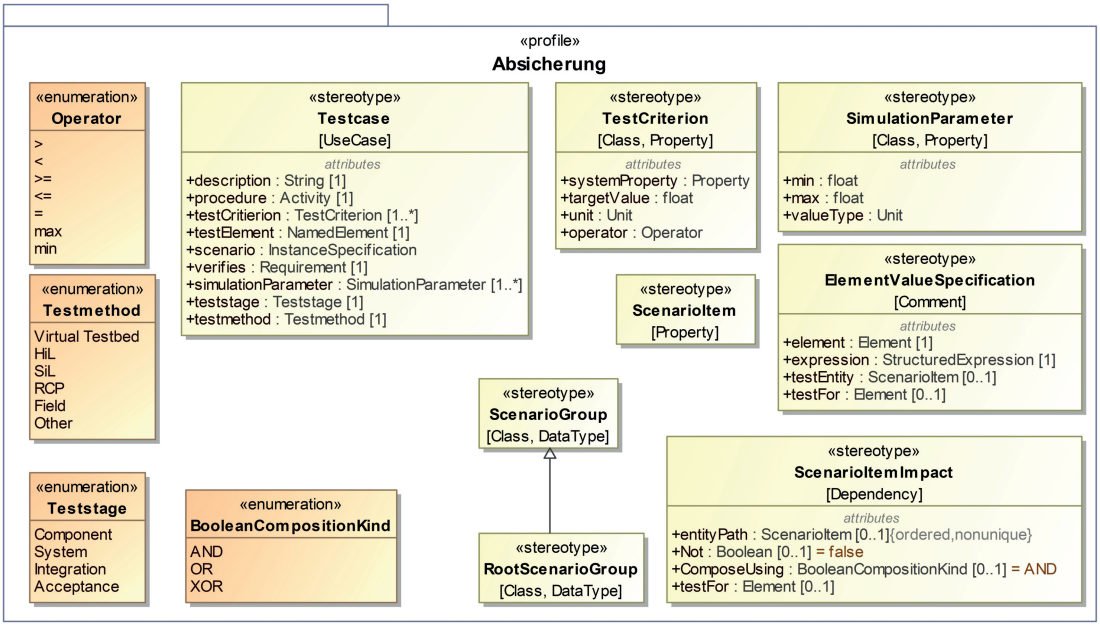


Bild A-25: Profil Testkonzept

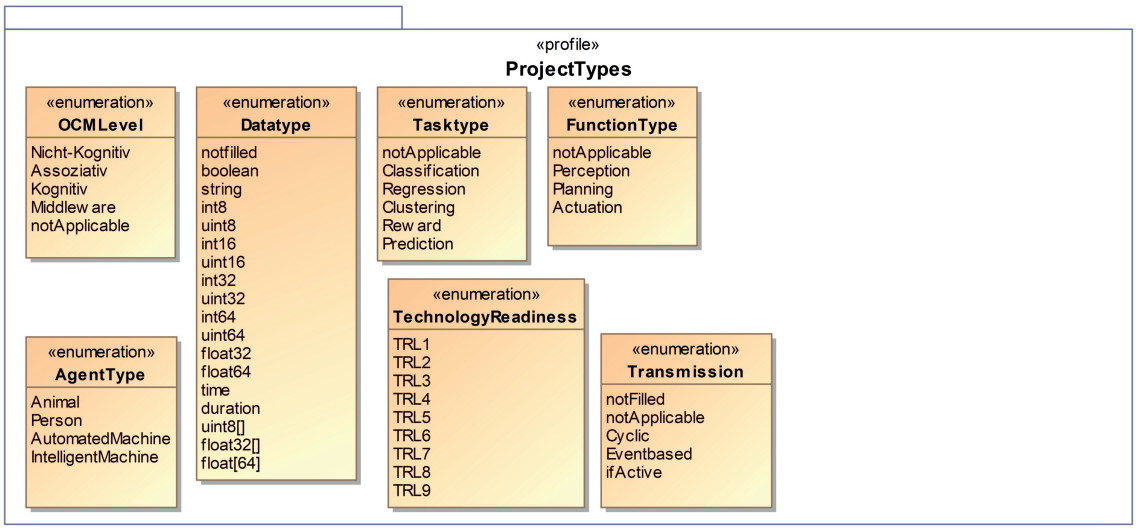


Bild A-26: Projektspezifische Typen

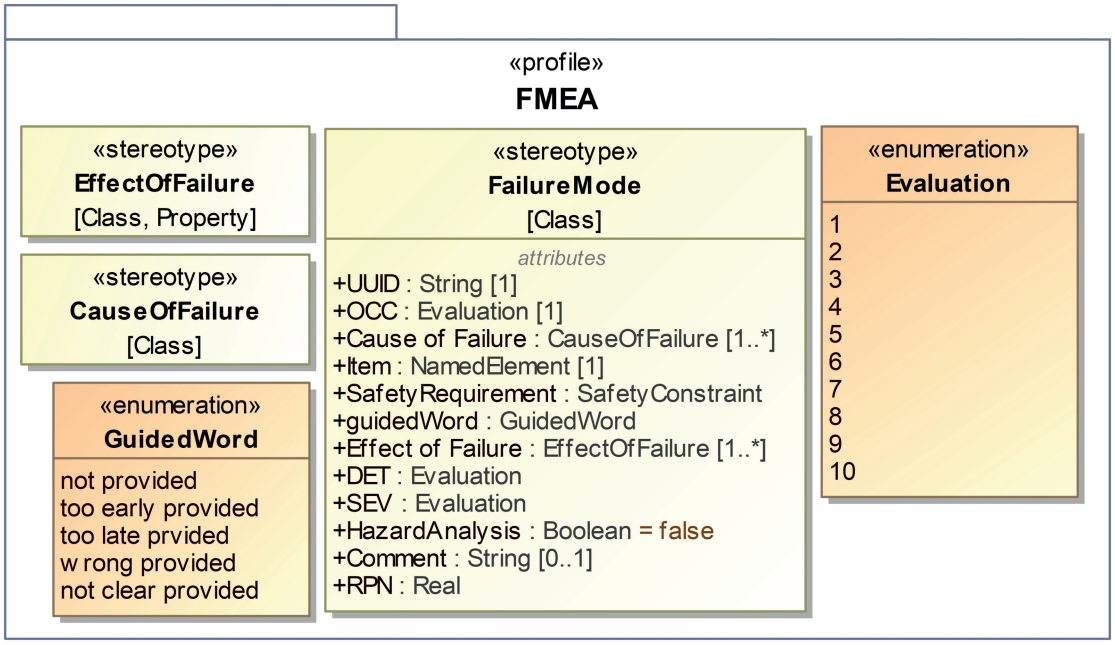


Bild A-27: Profil Risikoprofil

A2.5 Prozessbausteine

A2.5.1 Rollenprofile

Nr.	Bereich/ Rolle	Abk.	Anliegen («concern»)
1	Anforderungs-management	AM	Sicht auf die Stakeholder, **Anforderungen** und **Anwendungsszenarien** des Systems.
2	Data Science	DS	Sicht auf die **Struktur** und das **Verhalten**: Daten, Signale, Datenquellen und Informationsflüsse im System.
3	Elektrotechnik	EE	Sicht auf die **Struktur**, insb. auf die elektrischen Energieflüsse und EE-Funktionen und Systemelemente.
4	Mechanik	ME	Sicht auf die **Struktur**, insb. auf die mechanischen Kräfte und Funktionen sowie auf die gestaltbehafteten Systemelemente.
5	Regelungstechnik	RT	Sicht auf die situationsspezifischen **Systemziele**. Sicht auf die Struktur mit Reglerparameter und **Verhalten**-Zustände.
6	Systementwicklung	SE	Sicht auf die **Funktionen** und die **Struktur** des Systems. Konsistentes Zusammenspiel von **Verhalten** und Struktur.
7	Simulationstechnik	ST	Sicht auf die **Testszenarien** und **Testfälle** sowie Testdaten des Systems. **Anforderungen** an die Simulationen.
8	Systemanalyse und Resilienz	SA	Sicht auf die Anforderungen, Potenziale zur Integration von Eigenschaften d. Selbstheilung und SuOC-Definition.
9	Softwareentwicklung	SW	Sicht auf das diskrete intendierte **Verhalten** der Software. Daten- und Signaleflüsse in der **Strukturansicht**.
10	Test und Integration	TI	Sicht auf die **Testfälle** zur Überprüfung der **Anforderungen**. Sicht auf **Testbedstruktur** und **Szenarioablauf**.

Bild A-28: Rollen und ihr Anliegen

A2.5.2 Anforderungsphase

Ermittlung der Anforderungen

Ermittlung der Anforderungen									
Ziel: Ziel ist die Ermittlung und Bewertung von Anforderungen an das zu entwickelnde System. Ausgehend von der Stakeholder-Analyse werden Anwendungsszenarien, Umfeldmodelle entlang des Systemlebenszyklus erarbeitet. Anschließend werden die Anforderungen ermittelt und priorisiert.									
Nr.	Tätigkeit	AM	SE	DS	EE	ME	RT	SW	SA
1	Stakeholder identifizieren	R	S	-	-	-	-	-	S
2	Anwendungsszenarien erarbeiten	R	S	-	-	-	-	-	S
3	Umfeldmodell erstellen	R	S	-	C	C	C	C	C
4	Anforderungen ableiten	R	S	C	C	C	C	C	C

Bild A-29: Ermittlung der Anforderungen

A2.5.3 Konzeptphase

Modellbasierter Systementwurf

Modellbasierter Systementwurf									
Ziel: Ziel ist die Systemkonzeption. Ausgehend von den Anforderungen und Anwendungsszenarien werden die Funktionen abgeleitet. Darauf aufbauen erfolgt die Synthese des Systems u.a. durch Wirkstruktur, Verhalten, Zielsystem. (Statistische) Zeit-Analysen und erste Dynamikmodelle ermöglichen erste Simulationen und Analysen hinsichtlich der Machbarkeit. Verfeinerung der Anforderungen.									
Nr.	Tätigkeit	AM	SE	DS	EE	ME	RT	SW	SA
1	Funktionshierarchie erarbeiten	S	R	I	I	I	I	I	I
2	Wirkstruktur modellieren	I	R	S	S	S	S	S	I
3	Verhalten-Zustände modellieren	I	R	I	I	-	S	S	I
4	Verhalten-Aktivitäten modellieren	I	R	I	-	-	S	S	I
5	Verhalten-Sequenzen modellieren	I	R	I	-	-	S	S	I
6	Zielsystem definieren	S	R	S	C	-	S	C	S
7	Grobgestalt erarbeiten	I	R	-	S	S	C	C	S
8	Dynamikmodelle erstellen	-	R	C	S	S	S	C	C

Bild A-30: Modellbasierter Systementwurf

Potenzialanalyse zur Selbstheilung

Potenzialanalyse zur Selbstheilung									
Ziel: Ziel ist die Analyse von Potenzialen zur Integration von Eigenschaften der Selbstheilung. Hierzu werden kritische Funktionen und Strukturen als SuOC im System anhand der Dimensionen: Wissensgrad, Komplexität, Risiko bewertet.Für das SuOC wird der anzustrebende Ziel-Reifegrad als Normstrategie festgelegt. Darüber hinaus werden Anforderungen an die Verlässlichkeit ergänzt.									
Nr.	Tätigkeit	AM	SE	DS	EE	ME	RT	SW	SA
1	SuOC (Funktionen/ Struktur) definieren	-	S	-	C	C	C	C	R
2	Fehlanwendungsfälle identifizieren	S	S	S	S	S	-	S	R
3	Gefahrensituationen identifizieren	S	S	S	S	S	S	S	R
4	SuOC-Komplexität analysieren	-	S	-	-	-	-	-	R
5	SuOC-Wissensgrad ermitteln	S	S	S	S	S	S	S	R
6	Sicherheitsrisiko bewerten	S	S	S	S	S	S	S	R
7	Potenziale analysieren und bewerten	I	S	I	I	I	I	I	R
8	Reifegrad der Selbstheilung festlegen	I	S	I	I	I	I	I	R
9	Verlässlichkeitsziele festlegen	R	C	C	C	C	C	C	S

Bild A-31: Potenzialanalyse

Systemische Risikoanalyse

Systemische Risikoanalyse									
Ziel: Ziel ist die Identifikation und Bewertung von systemischen Risiken sowie möglicher Ereignisse und Ereigniskombinationen, die zu einer Gefahrensituation führen. Zu diesen Ereignissen werden mögliche Überwachungsverfahren und Selbstheilungsoperationen ermittelt und mit der Beeinträchtigung im Fehlerbaum verknüpft.									
Nr.	Tätigkeit	AM	SE	DS	EE	ME	RT	SW	SA
1	Gefahrensituation auswählen	S	S	S	S	S	S	S	R
2	Regelkreis im SuOC identifizieren	-	S	C	C	C	S	S	R
3	Unsichere Regelungsvorgänge ermitteln	-	S	S	S	S	S	S	R
4	Dynamischen Fehlerbaum modellieren	-	S	S	S	S	S	S	R
5	Trigger Ereignisse identifizieren	-	S	S	S	S	S	S	R
6	Größe und Signale identifizieren	-	S	S	S	S	S	S	R
7	Zeitliche Fehlerpropagierung ermitteln	-	S	S	S	S	S	S	R
8	Operationen der Selbstheilung ermitteln	-	S	S	S	S	S	S	R
9	Verlässlichkeitsziele verfeinern	R	S	S	S	S	S	S	S

Bild A-32: Systemische Risikoanalyse

A2.5.4 Entwurfsphase

Konzept und Entwurf des selbstheilenden Systems

Konzept und Entwurf des selbstheilenden Systems									
Ziel: In diesem Schritt werden die Entwurfsaspekte und zugehörigen Konstruktionsprinzipien für das ausgewählte SuOC konzipiert und in das intelligente technische System integriert. Es entsteht ein (virtueller) Prototyp des selbstheilenden Systems. An diesem Prototyp werden die Modelle trainiert.									
Nr.	Tätigkeit	AM	DS	EE	ME	RT	SW	SE	SA
1	Funktion, Struktur (SuOC) auswählen	C	S	C	C	S	S	S	R
2	Entwurfsaspekte und Reifegrad auswählen	I	S	C	C	S	S	S	R
3	SuOC Überwachung konzipieren/ entwerfen	I	R	C	C	S	S	C	S
4	SuOC Analyse konzipieren/ entwerfen	I	R	C	C	S	S	C	S
5	SuOC Planung konzipieren/ entwerfen		R	C	C	S	S	C	S
6	SuOC Ausführung konzipieren/ entwerfen	-	S	C	C	R	S	C	S
7	Framework der Selbstheilung umsetzen	-	S	C	C	S	S	S	R
8	Prinziplösung um Entwurfsaspekte ergänzen	S	S	S	S	S	S	R	S
9	Virtuellen Prototypen in HW/SW erarbeiten	C	R	R	R	R	R	S	C
10	Modelle anhand der SciL-Daten trainieren	C	R	C	C	C	C	C	S

Bild A-33: Konzept und Entwurf des selbstheilenden Systems

A2.5.5 Integrationsphase

Verifikation des selbstheilenden Systems

Verifikation des selbstheilenden Systems									
Ziel: Ziel von diesem Schritt ist die Verifikation von Eigenschaften der Selbstheilung. Dabei wird überprüft, ob die Konstruktionsprinzipien in dem Lösungsmodell korrekt funktionieren. In diesem Schritt werden die Komponenten und Schnittstellen der Selbstheilung getestet and evaluiert.									
Nr.	Tätigkeit	AM	SE	SA	TI	ST	DS	-	-
1	Anforderungs-basierte Tests durchführen	S	C	C	R	S	I	-	-
2	Schnittstellen und Komponenten testen	C	C	C	R	S	I	-	-
3	Funktionstests durchführen	C	S	S	R	C	C	-	-
4	Kennzahlensystem überprüfen	I	I	S	R	S	I	-	-

Bild A-34: Verifikation des selbstheilenden Systems

Validierung des selbstheilenden Systems

Validierung des selbstheilenden Systems									
Ziel: Ziel dieser Phase ist die Validierung von Eigenschaften der Selbstheilung anhand von virtuellen Szenarien, Feldtests und realen Prototypen. Dabei wird die Robustheit des Systems sowie Kennzahlen zur Bewertung der Eigenschaften von Selbstheilung untersucht. Monkey Testverfahren werden durchgeführt. Untersucht wird das Sicherheitsniveau und Ausfallrisiko des Systems.									
Nr.	Tätigkeit	AM	SE	SA	TI	ST	DS	-	-
1	Teilmodelle der SW/HW integrieren	-	-	S	S	R	S	-	-
2	Testfälle aus realen Betriebsdaten ableiten	S	-	S	R	C	C	-	-
3	Robustheit des Systems mit SciL testen	-	-	S	R	I	C	-	-
4	Resilienz des Systems mit SciL testen	-	-	S	R	I	C	-	-
5	Monkey Testverfahren mit SciL ausführen	-	-	-	R	-	-	-	-
6	Worst-Case Szenarien testen	C	C	C	R	C	-	-	-
7	Kennzahlensystem bewerten	I	I	I	R	I	I	-	-
8	Szenarien im Feldtest untersuchen	-	S	S	R	I	S	-	-
9	Daten für das ML-Training aufbereiten	-	-	S	R	S	S	-	-
10	Lessons Learned für die Absicherung	S	S	S	R	S	S	-	-

Bild A-35: Validierung des selbstheilenden Systems

Design Review

Design Review									
Ziel: In diesem Schritt wird die Resilienz und Robustheit des selbstheilenden Systems bewertet. Grundlage bildet das Kennzahlensystem aus der Absicherungsstrategie. Für das SuOC und die Konstruktionsprinzipien werden die Kennzahlen und ihr Akzeptanzbereich bewertet. Dabei ist zu entscheiden, ob das Risiko mit den Eigenschaften der Selbstheilung innerhalb der Akzeptanz liegt. Der validierte Prototyp und die Teilmodelle werden für die Domäne freigegeben.									
Nr.	Tätigkeit	AM	DS	EE	TI	RT	SW	SE	SA
1	Kennzahlen und Akzeptanz analysieren	S	S	-	S	S	S	-	R
2	Robustheit und Resilienz bewerten	S	S	-	S	-	-	-	R
3	Walktrough durch die Prinziplösung	S	S	S	R	S	S	S	S
4	Plausibilitätsprüfung durchführen	S	S	S	R	S	S	S	S
5	Prinziplösung und Prototyp freigeben	S	S	S	R	S	S	S	S
6	Release der Modelle und der Pipeline	-	R	-	S	I	I	-	S
7	Funktion in der Betriebsdomäne freigeben	C	S	S	R	S	S	S	R
8	Lessons Learned durchführen	S	S	S	R	S	S	S	S

Bild A-36: Design Review

A2.5.6 Unterstützende Aktivitäten

Definition der Absicherungsstrategie

Definition der Absicherungsstrategie									
Ziel: In diesem Schritt wird die Absicherungsstrategie earbeitet. Die Testfälle werden dokumentiert und mit den Anforderungen verknüpft (*verify*). Die Testfälle werden den XiL-Teststufen zugewiesen. Das Virtuelle Testbed wird für Scenario-in-the-Loop (SciL) konfiguriert und die virtuellen Szenarien modelliert. Zudem sind die Kennzahlen für die Freigabe im Design Review.									
Nr.	Tätigkeit	AM	SE	SA	TI	ST	-	-	-
1	Testfälle ableiten und beschreiben	S	S	S	R	S	-	-	-
2	Kennzahlensystem festlegen	C	S	R	C	C	-	-	-
3	Testobjekt definieren	I	C	C	R	S	-	-	-
4	XiL-Stufen festlegen und Testfälle zuweisen	I	S	C	R	S	-	-	-
5	Testbedstruktur(en) modellieren	I	C	C	R	S	-	-	-
6	Konsistente Virtuelle Szenarien ableiten	C	C	S	S	R	-	-	-
7	Virtuelle Szenarien im Testbed modellieren	I	I	I	I	R	-	-	-
8	Monkey Testing Framework anpassen	I	C	S	S	R	-	-	-
9	Virtuelle Testbeds konfigurieren	I	C	S	S	R	-	-	-

Bild A-37: Definition der Absicherungsstrategie

Erklärung zur Zitation von Inhalten aus studentischen Arbeiten:

In Ergänzung zu meinem Antrag auf Zulassung zur Promotion in der Fakultät für Maschinenbau der Universität Paderborn erkläre ich gemäß der §11 der Promotionsordnung und unter Beachtung der Regelung zur Zitation studentischer Arbeiten:

Die von mit vorgelegte Dissertation habe ich selbstständig verfasst, **und ich habe keine anderen** als die dort angegebenen Quellen und Hilfsmittel benutzt. Es sind **Inhalte** studentischen Ursprungs (studentische Arbeiten) in dieser Dissertation enthalten.

Ich habe die verwendeten Arbeiten entsprechend der Regelung „Zitation aus studentischen Arbeiten in der Dissertation“ zitiert.

Paderborn, 15. Dezember 2021

Das Heinz Nixdorf Institut – Interdisziplinäres Forschungszentrum für Informatik und Technik

Das Heinz Nixdorf Institut ist ein Forschungszentrum der Universität Paderborn. Es entstand 1987 aus der Initiative und mit Förderung von Heinz Nixdorf. Damit wollte er Ingenieurwissenschaften und Informatik zusammenführen, um wesentliche Impulse für neue Produkte und Dienstleistungen zu erzeugen. Dies schließt auch die Wechselwirkungen mit dem gesellschaftlichen Umfeld ein.

Die Forschungsarbeit orientiert sich an dem Programm „Dynamik, Mobilität, Vernetzung: Eine neue Schule des Entwurfs der technischen Systeme von morgen“. In der Lehre engagiert sich das Heinz Nixdorf Institut in Studiengängen der Informatik, der Ingenieurwissenschaften und der Wirtschaftswissenschaften.

Heute wirken am Heinz Nixdorf Institut acht Professoren mit insgesamt 130 Mitarbeiterinnen und Mitarbeitern. Pro Jahr promovieren hier etwa 15 Nachwuchswissenschaftlerinnen und Nachwuchswissenschaftler.

Heinz Nixdorf Institute – Interdisciplinary Research Centre for Computer Science and Technology

The Heinz Nixdorf Institute is a research centre within the Paderborn University. It was founded in 1987 initiated and supported by Heinz Nixdorf. By doing so he wanted to create a symbiosis of computer science and engineering in order to provide critical impetus for new products and services. This includes interactions with the social environment.

Our research is aligned with the program “Dynamics, Mobility, Integration: Enroute to the technical systems of tomorrow.” In training and education the Heinz Nixdorf Institute is involved in many programs of study at the Paderborn University. The superior goal in education and training is to communicate competencies that are critical in tomorrows economy.

Today eight Professors and 130 researchers work at the Heinz Nixdorf Institute. Per year approximately 15 young researchers receive a doctorate.

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 376 JÜRGENHAKE, C.: Systematik für eine prototypenbasierte Entwicklung mechatronischer Systeme in der Technologie MID (Molded Interconnect Devices). Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 376, Paderborn, 2017 – ISBN 978-3-942647-95-3

Bd. 377 WEBER, J.: Modellbasierte Werkstück- und Werkzeugpositionierung zur Reduzierung der Zykluszeit in NC-Programmen. Dissertation, Fakultät für Wirtschaftswissenschaften, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 377, Paderborn, 2018 – ISBN 978-3-942647-96-0

Bd. 378 OESTERSÖTEBIER, F.: Modellbasierter Entwurf intelligenter mechatronischer Systeme mithilfe semantischer Technologien. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 378, Paderborn, 2018 – ISBN 978-3-942647-97-7

Bd. 379 ABELDGAWAD, K.: A System-Level Design Framework for Networked Driving Simulation. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 379, Paderborn, 2018 – ISBN 978-3-942647-98-4

Bd. 380 JUNG, D.: Local Strategies for Swarm Formations on a Grid. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 380, Paderborn, 2018 – ISBN 978-3-942647-99-1

Bd. 381 PLACZEK, M.: Systematik zur geschäftsmodellorientierten Technologiefrühaufklärung. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 381, Paderborn, 2018 – ISBN 978-3-947647-00-2

Bd. 382 KÖCHLING, D.: Systematik zur integrativen Planung des Verhaltens selbstoptimierender Produktionssysteme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 382, Paderborn, 2018 – ISBN 978-3-947647-01-9

Bd. 383 KAGE, M.: Systematik zur Positionierung in technologieinduzierten Wertschöpfungsnetzwerken. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 383, Paderborn, 2018 – ISBN 978-3-947647-02-6

Bd. 384 DÜLME, C.: Systematik zur zukunftsorientierten Konsolidierung variantenreicher Produktprogramme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 384, Paderborn, 2018 – ISBN 978-3-947647-03-3

Bd. 385 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 14. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 8. und 9. November 2018, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 385, Paderborn, 2018 – ISBN 978-3-947647-04-0

Bd. 386 SCHNEIDER, M.: Spezifikationstechnik zur Beschreibung und Analyse von Wertschöpfungssystemen**.** Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 386, Paderborn, 2018 – ISBN 978-3-947647-05-7

Bd. 387 ECHTERHOFF, B.: Methodik zur Einführung innovativer Geschäftsmodelle in etablierten Unternehmen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 387, Paderborn, 2018 – ISBN 978-3-947647-06-4

Bd. 388 KRUSE, D.: Teilautomatisierte Parameteridentifikation für die Validierung von Dynamikmodellen im modellbasierten Entwurf mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 388, Paderborn, 2019 – ISBN 978-3-947647-07-1

Bd. 389 MITTAG, T.: Systematik zur Gestaltung der Wertschöpfung für digitalisierte hybride Marktleistungen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 389, Paderborn, 2019 – ISBN 978-3-947647-08-8

Bezugsadresse:
Heinz Nixdorf Institut
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 390 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 15. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 21. und 22. November 2019, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 390, Paderborn, 2019 – ISBN 978-3-947647-09-5

Bd. 391 SCHIERBAUM, A.: Systematik zur Ableitung bedarfsgerechter Systems Engineering Leitfäden im Maschinenbau. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 391, Paderborn, 2019 – ISBN 978-3-947647-10-1

Bd. 392 PAI, A.: Computationally Efficient Modelling and Precision Position and Force Control of SMA Actuators. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 392, Paderborn, 2019 – ISBN 978-3-947647-11-8

Bd. 393 ECHTERFELD, J.: Systematik zur Digitalisierung von Produktprogrammen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 393, Paderborn, 2020 – ISBN 978-3-947647-12-5

Bd. 394 LOCHBICHLER, M.: Systematische Wahl einer Modellierungstiefe im Entwurfsprozess mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 394, Paderborn, 2020 – ISBN 978-3-947647-13-2

Bd. 395 LUKEI, M.: Systematik zur integrativen Entwicklung von mechatronischen Produkten und deren Prüfmittel. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 395, Paderborn, 2020 – ISBN 978-3-947647-14-9

Bd. 396 KOHLSTEDT, A.: Modellbasierte Synthese einer hybriden Kraft-/Positionsregelung für einen Fahrzeugachsprüfstand mit hydraulischem Hexapod. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 396, Paderborn, 2021 – ISBN 978-3-947647-15-6

Bd. 397 DREWEL, M.: Systematik zum Einstieg in die Plattformökonomie. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 397, Paderborn, 2021 – ISBN 978-3-947647-16-3

Bd. 398 FRANK, M.: Systematik zur Planung des organisationalen Wandels zum Smart Service-Anbieter. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 398, Paderborn, 2021 – ISBN 978-3-947647-17-0

Bd. 399 KOLDEWEY, C.: Systematik zur Entwicklung von Smart Service-Strategien im produzierenden Gewerbe. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 399, Paderborn, 2021 – ISBN 978-3-947647-18-7

Bd. 400 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 16. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 2. und 3. Dezember 2021, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 400, Paderborn, 2021 – ISBN 978-3-947647-19-4

Bd. 401 BRETZ, L.: Rahmenwerk zur Planung und Einführung von Systems Engineering und Model-Based Systems Engineering. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 401, Paderborn, 2021 – ISBN 978-3-947647-20-0

Bd. 402 LIANG, W.: Ultrabreitbandige Sampler in SiGe-BiCMOS-Technologie für Analog-Digital-Wandler mit zeitversetzter Abtastung. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 402, Paderborn, 2021 – ISBN 978-3-947647-21-7

Bezugsadresse:
Heinz Nixdorf Institut
Universität Paderborn
Fürstenallee 11
33102 Paderborn

Intelligente Technische Systeme kombinieren Sensorik, Aktorik und die Selbstregulation zu intelligenten Regelkreisen. Diese Systeme lösen selbstständig komplexe Aufgaben in ihrer Anwendungsdomäne. Dabei agieren sie situativ und zielführend. Derartige Systeme eröffnen neue Möglichkeiten in der Produktion oder in der Mobilität. Mit diesen Nutzenpotenzialen geht die Zunahme der Komplexität einher. Sie stellen neue Anforderungen an die Verlässlichkeit.

Ein Lösungsansatz bietet die Integration von Selbstheilungseigenschaften. Selbstheilende Systeme überwachen ihren Gesundheitszustand, detektieren selbstständig Beeinträchtigungen und können ihre intendierte Funktion aufrechterhalten oder wiederherstellen. Diese Systeme sind wandlungsfähig, robust, lernfähig, redundant und verteilt. Übergeordnetes Ziel ist die Steigerung der Resilienz und Autonomie. Eine durchgehende Systematik zur Entwicklung dieser selbstheilenden Systeme existiert bisher nicht.

Diese Systematik ist Gegenstand der vorliegenden Arbeit. Ihr Einsatz ist für ein interdisziplinäres Entwicklungsteam konzipiert. Die Systematik führt die Bereiche Selbstheilung, Data-Driven Systems Engineering und Virtuelle Testbeds zusammen. Das Ergebnis ist ein Selbstheilungskonzept, eine Entwurfstechnik, sowie eine Werkzeugumgebung zum Szenario-basierten Trainieren und Testen. Ein Vorgehensmodell führt diese Bestandteile zu einer Systematik zusammen und macht sie für Dritte anwendbar. Die Anwendung der Systematik ist anhand eines autonomen Fahrzeugs gezeigt.