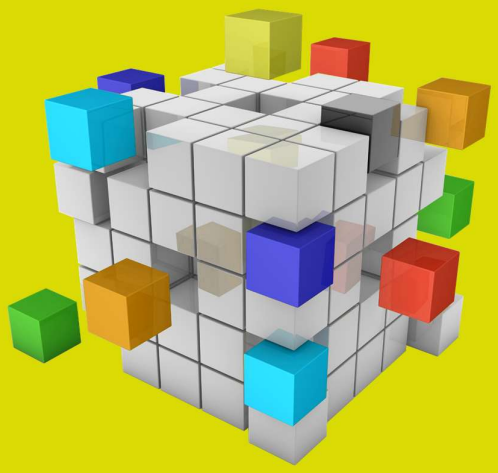
invenio



Whitepaper

INTEGRIERTES SYSTEMS-, SW- UND HW-ENGINEERING ZUR ÜBERWINDUNG DER ENGINEERINGSILOS

VERSION	DATUM	AUTOR	ÄNDERUNG(EN)
1.0	30.09.2024	Enrico Seidel	Erstausgabe

INHALT

Zusammenfassung **3**
1 Einleitung **3**
2 Herausforderungen zur Überwindung der Engineeringsilos **3**
2.1 Mangel an korrekter technischer Kommunikation 3
2.2 Redundante Anstrengungen und daraus resultierende Ineffizienzen 4
2.3 Inkonsistente Daten und Modelle 4
2.4 Verzögerte Problemidentifikation 4
2.5 Begrenzte Wiederverwendbarkeit 5
2.6 Hemmende Innovation durch fehlende Kompetenz 5
3 invenio Systems Engineering Framework **6**
3.1 Die 3 Dimensionen der iSEF-Architekturbeschreibung 6
3.1.1 Die Modellsichten des iSEF 7
3.1.2 Verknüpfen der Elemente zwischen den Modellsichten 7
3.1.3 Die Abstraktionsebenen des iSEF 8
3.1.4 Hierarchy Depth (Hierarchische Modelltiefe) 9
3.2 Methodische Modellierungsschritte zum Systemdesign 9
4 Exzellenz durch angewandtes Systems-Engineering **9**
5 Forschungsbedarf und Ausblick **10**
6 Referenzen **11**

Zusammenfassung

Die wachsende Komplexität moderner Systeme erfordert einen Wandel von separaten Ingenieursilos hin zu einem integrativen Ansatz, der alle Ingenieursdisziplinen einbezieht. Das invenio Systems Engineering Framework (iSEF) der invenio AG schließt mit einem neuen Entwicklungswerkzeug die Lücken zwischen Systems-, Software- und Hardware-Engineering und setzt damit ein neu formuliertes architekturzentriertes modellbasiertes Systems Engineering (ACMBSE) effektiv um. Speziell durch die formale und systematische Verknüpfung aller Modellsichten und Abstraktionsebenen in der Architektur fördert das iSEF die disziplinübergreifende Zusammenarbeit und gewährleistet einen kohärenten Entwicklungsprozess. Weiterhin wird derzeitig ein praxisorientiertes ACMBSE-Ingenieursausbildungsprogramm zur Seite gestellt, welches unterstützend das Problem des Silo-Engineerings lösen soll.

1 EINLEITUNG

Die Konvergenz fortschrittlicher Technologien in der Automobil-, Luft- und Raumfahrt- sowie anderen Hightech-Industrien hat in den vergangenen Jahren zu einer beispiellosen Systemkomplexität geführt [3]. Traditionelle Ingenieuransätze, oft durch dokumentenorientiertes, isoliertes Arbeiten in Disziplinen (Silos) gekennzeichnet, kämpften bisher damit, diese Komplexität effektiv zu bewältigen. Die nahtlose Integration von Systems Engineering (SE), Software-Engineering (SWE) und Hardware-Engineering (HWE) ist daher wichtiger denn je.

Als Antwort auf diese Herausforderung stellt die invenio Systems Engineering GmbH ein Framework (iSEF) vor. Dieses Framework bringt durch die Verknüpfung von Modellsichten über alle Systemabstraktionsebenen hinweg die Stärken des architekturzentrierten modellbasierten Systems Engineering (ACMBSE) in den Fokus und ermöglicht eine umfassende und kohärente Architekturbeschreibung zu [5].

Dieses Whitepaper skizziert in einer Übersicht die Grundprinzipien von iSEF, seine praktische Anwendung beispielhaft in der Automobildomäne [4], und damit die Vorteile des Abbaus von Silos zur Erreichung von Synergien zwischen den Ingenieurbereichen.

2 HERAUSFORDERUNGEN ZUR ÜBERWINDUNG DER ENGINEERINGSILOS

Historisch entwickelte sich das Systems-, Software- und Hardware-Engineering unabhängig voneinander, mit eigenen Methoden, Werkzeugen und Kulturen, insbesondere bei Anforderungen und der Architektur [6,7]. Diese Trennung führte daher oft zu Missinterpretationen, redundanter Dokumentation und Ineffizienzen. In komplexen Projekten, mit erheblichen Abhängigkeiten zwischen Systemen, Software und Hardware, wird der Mangel an der Integrationsfähigkeit damit problematisch. Oft war textuelles Requirements Engineering das einzige verbleibende Bindeglied zur Kommunikation zwischen den Disziplinen [4]. Traditionelle dokument-basierte Ansätze sind daher nicht mehr ausreichend. Ein auf MBSE-Prinzipien basierender architektur-zentrierter, Modellierungsansatz (ACMBSE), unterstützt durch ein robustes Framework wie das iSEF, ist daher unerlässlich, um diese Herausforderungen zu bewältigen. Im Folgenden sollen einige wichtige Aspekte der Herausforderungen und deren mögliche Lösungen aufgezeigt werden [3].

2.1 Mangel an korrekter technischer Kommunikation

Eine der größten Herausforderungen in Engineeringsilos ist der Mangel an effektiver Kommunikation und Koordination zwischen den Ingenieursdisziplinen und daraus resultierende geschaffene Barrieren in der Zusammenarbeit.

Unterschiedliche Methoden, Werkzeuge, Sprachen und Notationen führen zu Missverständnissen und Fehlkommunikation, wodurch die Abstimmung technischer Inhalte zwischen Teams erschwert wird. Das Ergebnis sind fragmentiertes Wissen und unzusammenhängende Entwicklungsartefakte, die die Effizienz und Effektivität von Ingenieurprojekten erheblich beeinträchtigen können.

Wichtig ist daher eine gemeinsame technische Sprache. Die grafische Notation UML/SysML [8] erleichtert, richtig gelernt und angewendet, die Kommunikation erheblich, da eine systematisch erstellte bildliche Darstellung die Zusammenhänge von Verhalten oder Struktur des Systems gegenüber von textuellen Anforderungen signifikant besser aufzeigen kann [9]. Dies steigert auch die Effizienz der Kommunikation erheblich, da nicht jeder Ingenieur sich selbst sein eigenes Bild von Struktur und Verhalten des Systems im Kopf „modellieren“ muss, um ganzheitlich das Verhalten eines Systemelements zu erfassen [10].

2.2 Redundante Anstrengungen und daraus resultierende Ineffizienzen

Isoliert arbeitende Ingenieurteams führen häufig zu doppelten Anstrengungen und redundanter Arbeit, da sie oft unabhängig in verschiedenen Projekten das gleiche Problem bearbeiten. Diese Redundanz verschwendet Ressourcen, erhöht die Entwicklungskosten und verzögert somit den Projektzeitplan, besonders bei großen Entwicklungsprojekten.

Die Lösung ist auch hier ein architekturzentrierter Modellierungsansatz (ACMBSE) [3], bei dem Systemteile Modellzerlegungsmodellen folgend dekomponiert und strukturiert abgelegt werden. Mit Verfahren zur Erstellung von standardisierten Schnittstellen und Services in einer Interfacedatenbank können zudem wiederverwendbare Modellbausteine geschaffen werden, die die Effizienz von Plattformprojekten signifikant erhöhen [11].

2.3 Inkonsistente Daten und Modelle

Das wichtigste Prinzip, das in der Projekthektik schnell vergessen wird, ist die einheitliche Informationsquelle (Single Source of Truth SSOT) [12]. Ohne eine einheitliche Arbeitsumgebung und Methodik zur Erstellung der notwendigen Engineeringartefakte entwickeln verschiedene Teams inkompatible Modelle, was die Integrations- und Validierungsprozesse erschwert. Diese Inkonsistenz führt dann oft zu Fehlverhalten aufgrund fehlerhaften Designs [6,7].

Eine modellgestützte und systematisch gepflegte Informationsquelle steigert die Effizienz erheblich und verringert den Aufwand für die Datensammlung so lange sichergestellt ist, dass alle Teammitglieder immer auf aktuelle Daten zugreifen können, was letztlich auch das Risiko von Projektfehlern minimiert. Wichtig ist ebenfalls, dass alle Disziplinen mit einer zwischen den Disziplinen abgestimmten Methodik und einheitlichen Prozessen arbeiten [6]. Ein Engineering-Framework, welches diese Arbeitsweise vorgibt, wird deshalb immer wichtiger, um das strukturierte Arbeiten mit SSOT-Daten umzusetzen.

2.4 Verzögerte Problemidentifikation

In isolierten Entwicklungsteams und Ingenieursumgebungen bleiben bereichsübergreifende Probleme oft bis spät im Entwicklungszyklus unentdeckt. Hardwareeinschränkungen, die die Softwareleistung beeinträchtigen, werden möglicherweise erst in der Integrationsphase erkannt, was die Behebung kostspieliger macht. Engineeringsilos behindern dabei zusätzlich die nötige Sichtbarkeit und Kommunikation [13].

Abhilfe schafft auch hier eine einheitliche disziplinübergreifende Methode in der Modellerstellung. Das erfordert, dass die Software- und Hardwarearchitekturmodelle von einem gemeinsamen Systemmodell abgeleitet werden.

Beste Ergebnisse erzielt man dabei, wenn alle Disziplinen im selben Modell mit demselben Regelwerk arbeiten. Besonders hervorzuheben ist, dass das Hardware-Software-Interface (HSI) [5] präzise formuliert werden muss. Zusätzlich ermöglichen ausführbare Funktionsmodelle auf System- und Disziplinebene eine frühzeitige Evaluierung des Systemverhaltens. Dies erfordert allerdings ein diszipliniertes Projektmanagement, um sich die Zeit zu nehmen, zuerst die Architektur zu verifizieren, bevor mit der eigentlichen Entwicklung begonnen wird.

2.5 Begrenzte Wiederverwendbarkeit

Eine nicht vorausschauende und nur im eigenen Projektkontext denkende Entwicklungsmannschaft erschwert die Wiederverwendung von erstellten Komponenten und Designs. Oft entwickeln Teams in Nachfolgeprojekten wiederholend maßgeschneiderte Kundenlösungen, anstatt existierende Komponenten zu nutzen, zu verbessern und diese wieder dem Entwicklungspool zur Verfügung zu stellen. Dies verringert die Gesamteffizienz, da Teams von bereits gelösten Problemen und getesteten Lösungen nicht profitieren.

Eine Lösung ist die Etablierung einer Organisationsstruktur, die den Fokus auf die Funktionsentwicklung legt und diese Lösungen den Plattformteams zur Verfügung stellt. Diese entwickeln daraus technische Lösungen, die dann in verschiedenen zukünftigen Kundenprojekten ebenfalls wiederverwendet werden können. Dies erfordert aber Weitsicht des Managements. Entsprechende Anfangsinvestitionen sind zu tätigen, die sich jedoch durch die Nutzung validierter und schnell integrierbarer Artefakte auszahlen. Voraussetzungen hierfür sind geeignete Integrationsreferenzmodelle und Integrationsspezifikationen. Das iSEF unterstützt dabei diese Integration mit vorgefertigten Modellbibliotheken.

2.6 Hemmende Innovation durch fehlende Kompetenz

Innovation gedeiht in Umgebungen, in denen sich verschiedene Perspektiven und Ideen gegenseitig befruchten. Aus der Erfahrung des Autors haben jedoch viele Systemingenieure kein ausreichendes Verständnis für Problemstellung der Hardware- und Softwaredisziplinen und erstellen stark abstrahierte Systemmodelle, die den Diszipliningenieuren wenig nützen. Kundenanforderungen werden zudem oft nur unzureichend analysiert, was die eigentliche Anforderung unklar lässt und Probleme in die Disziplinebene verschiebt.

Ein gutes Systemdesign erfordert ein tiefes technisches Verständnis für ein zu entwickelndes System und die daraus resultierende Systemarchitektur mit den im Anschluss folgenden Arbeiten der Hardware- und Softwareingenieure. Dieses Wissen sollte durch eine disziplinübergreifende sowie technisch fundierte Ausbildung mit wesentlichen Grundlagen der SW- und HW-Disziplinen vermittelt werden. Bestehende Programme wie CSE [14], INCOSE Zertifizierungen [15] oder die Ausbildung zum „Embedded System Architect“ [16] bieten richtige Ansätze, aber es fehlt der praktische Teil zur Erstellung einer durch die Disziplinen weiterverwendbaren Systemarchitektur.

Wir bei der invenio Systems Engineering GmbH entwickeln daher gerade ein Ausbildungsprogramm auf Basis des ACMBSE Ansatzes. Das iSEF bildet dazu das technische Fundament für die Modellierung. Diese Ausbildung wird 4 Stufen enthalten und ähnlich dem OCSMP Programm der OMG [17].

- Angewandte Systemmodellierung mit iSefML – Grundlagen und ACMBSE Methodik
- Angewandte Systemmodellierung mit iSEF – Formale Systemanalyse & Systemdesign
- Angewandte Systemmodellierung mit iSEF – Verhaltensmodellierung komplexer Systeme
- Angewandte Systemmodellierung mit iSEF – Aufbau domänenspezifischer Modellbibliotheken

3 INVENIO SYSTEMS ENGINEERING FRAMEWORK

Das invenio Systems Engineering Framework (iSEF) bietet wesentliche Lösungen zur Überwindung der Engineeringsilos durch die Einführung einer einheitlichen technischen Sprache und Methodik basierend auf UML/SysML [8]. Gegenwärtig implementiert in IBM Rhapsody, ermöglicht es eine benutzergeführte, strukturierte Ablage von Modellbausteinen und resultierend daraus ein kohärentes Datenmodell. Unterstützt durch definierbare Modellprüfungen stellt das iSEF sicher, dass alle Disziplinen einheitlich arbeiten. Ein gemeinsames Systemmodell fördert zudem die frühzeitige Erkennung und Behebung von potenziellen Problemen, was Systemintegration und Validierung erleichtert. Das iSEF unterstützt ebenfalls eine effiziente Entwicklung wiederverwendbarer Komponenten durch Referenzmodelle und Spezifikationen.

3.1 Die 3 Dimensionen der iSEF-Architekturbeschreibung

Mit zunehmender Komplexität von Systemen [1] müssen Systeme in kleinere Einheiten oder Subsysteme unterteilt und organisiert werden. Jedes Subsystem hat dann eine spezifische Aufgabe, die es Architekten und Designern ermöglicht, sich auf die Details dieses spezifischen Teils zu konzentrieren. Mit dem iSEF werden dazu basierend auf den Arbeiten in [4,5] drei Dimensionen der Systemarchitektur eingeführt: die Modellsichten auf das System (*Viewpoints*), die Abstraktionsebenen (*System Abstraction Levels*) und die Hierarchietiefe (*Hierarchy Depth*). Diese drei orthogonalen Dimensionen ermöglichen einen strukturierten Ansatz, wobei alle Architekturelemente in einem virtuellen 3D-Raum formal zusammengehörig abgelegt werden, wie in Abbildung 1 dargestellt. Um ein leichtes Verständnis zu entwickeln, werden durch den Modellierer immer aber nur Ausschnitte (Modellsichten) gezeigt, sodass alle Elemente gemäß UML/SysML [8] auf einem 2D-Raum transponiert sind (siehe Abbildung 1).

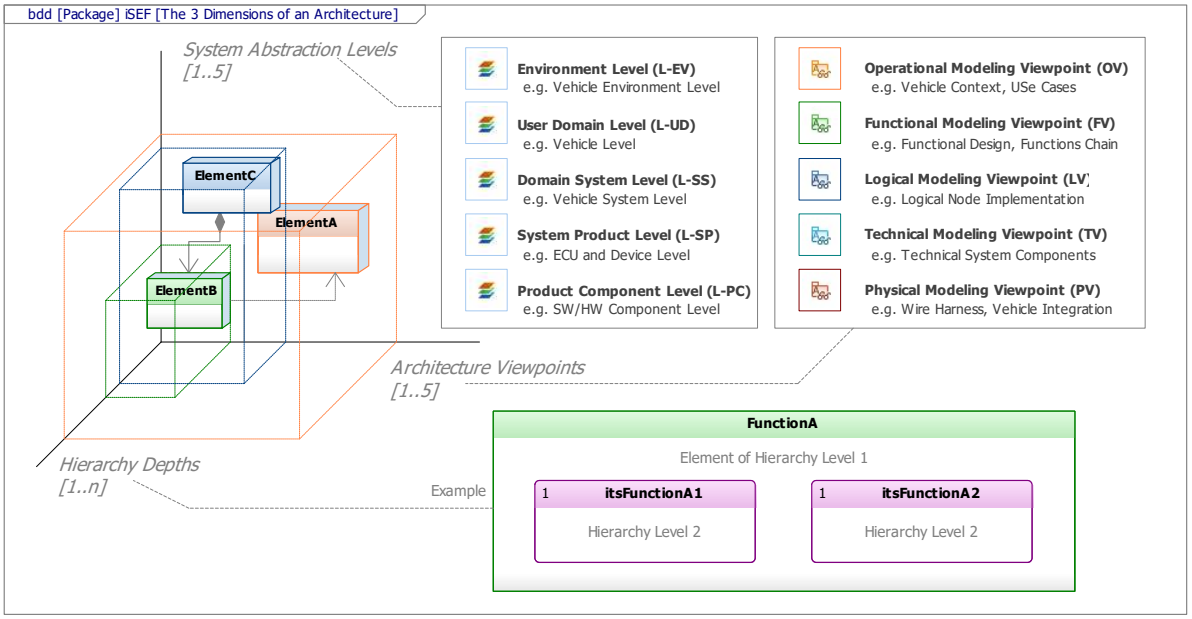


Abbildung 1: Die drei Dimensionen der Systemarchitekturbeschreibung

3.1.1 Die Modellsichten des iSEF

Im Rahmen der Entwicklung eines Frameworks [2] wurden entsprechend Abbildung 1 fünf architekturrelevante Modellsichten (*Viewpoints*) identifiziert, um ein System mit starkem Fokus auf Wiederverwendbarkeit zu analysieren, zu definieren und zu erstellen: Die operative Sicht (*Operational Viewpoint*), die funktionale Sicht (*Functional Viewpoint*), die logische Sicht (*Logical Viewpoint*), die technische Sicht (*Technical Viewpoint*) und die physische Sicht (*Physical Viewpoint*) [5]. Das iSEF-Regelwerk definiert dabei, welche Art von Element auf welchem Viewpoint und dem entsprechenden Abstraktionslevel erstellt werden darf. Das Tool Rhapsody unterstützt mit dem iSEF-Profil das korrekte Modellieren durch eine stringente Benutzerführung.

3.1.2 Verknüpfen der Elemente zwischen den Modellsichten

Um eine kohärente Architekturbeschreibung über alle Modellsichten sicherzustellen, führt das iSEF erweiterte SysML-Notationen und -Semantiken ein. Dabei werden Elemente aus verschiedenen Viewpoints miteinander verknüpft [5]: funktionale Elemente werden in logische, diese in technische und schließlich diese wieder in physische Strukturelemente durch Komposition integriert. Diese Verknüpfung wird durch Schnittstellen (Ports), wie in Abbildung 2 dargestellt, über die die Modellsichten hinweg verlinkt werden [5]. Dieses formale Zusammenspiel von Elementkomposition und Schnittstellenverlinkungen erlaubt letztendlich eine formale Modellierung mit automatisierter Konsistenzprüfung durch einen interaktiv arbeitenden Modelchecker bereits während der Modellierung.

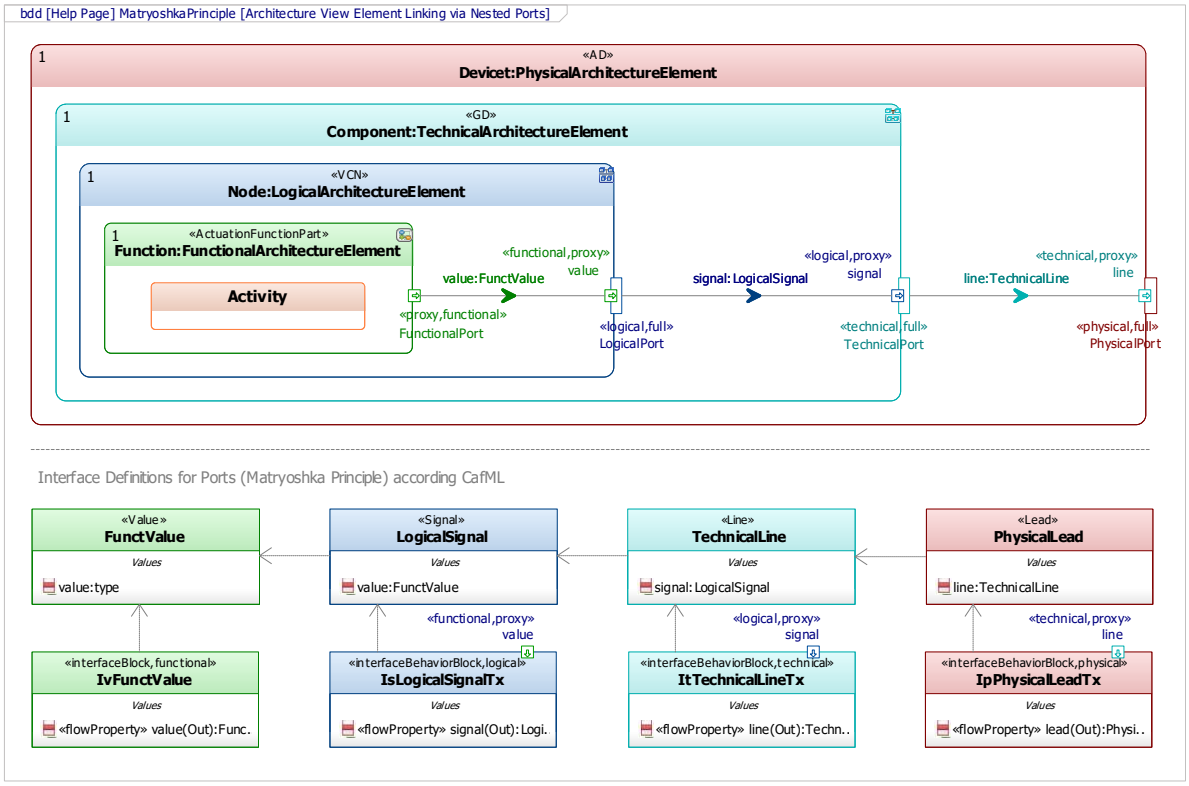


Abbildung 2: Linken von Elementen und Interfaces aufeinanderfolgender Viewpoints

Dank dieser Methode, ist eine formale Konsistenz der Schnittstellen durch verschachtelte Portmodellierung ersichtlich, bei der Proxy-Ports über die Interfacedefinition eines Interface BehaviorBlocks in Full-Ports eingebettet sind [5]. Funktionale Werte werden durch logische Signale gekapselt und über technische Schnittstellen übertragen, die an physische Schnittstellen gebunden sind. Dabei reflektieren die funktionalen, logischen, technischen und physikalischen Ports die Schnittstellen zur jeweiligen vorherigen Modellsicht. Dieses Prinzip wird auch das „Matroschka“ Prinzip genannt.

Die Methodik des Einbettens von Elementen eines Viewpoints in den darauffolgenden stellt die stärkste formale Lösung zur Verknüpfung aller Modellsichten dar und ist der zentrale Schlüssel des iSEF in der Architekturmodellierung. Besonders für Systeme, die eine hohe funktionale Sicherheit bis ASIL C/D erfordern, bietet diese Methode Konformität zu den Anforderungen der ISO 26262.

Ein weiterer aus der ACMBSE Methode resultierender Vorteil ist auch die flexible Zusammensetzung von wiederverwendbaren Strukturen über die Viewpoints hinweg, da die gezeigten Schnittstellen eines *InterfaceBehaviorBlocks* [5] auch eine Transformationsaufgabe übernehmen. So kann die logische Schnittstelle einer Komponente nach außen hin konstant erscheinen, obwohl im Inneren der Schnittstelle eine Werttransformation stattfindet, weil etwa ein Funktionsblock mit ähnlichem, aber nicht gleichem Interface eingebettet wurde. Modellierungswerkzeuge mit der implementierten Methode können diese Prinzipien nun applizieren, um Notationen und semantische Verifizierungen zu automatisieren und eine maschinelle Modellverifizierbarkeit zu erreichen.

3.1.3 Die Abstraktionsebenen des iSEF

Der funktionale und technische Umfang eines Systems und wie tief in das System hineingeschaut werden soll, wird durch *System Abstraction Levels* definiert [4], welche eine künstliche Abgrenzung, aber keine inhärente Eigenschaft des Systems sind. Sie dienen jedoch als Klassifizierung jedes möglichen Systems. Andererseits kann sich eine Systemabstraktionsebene auf den Prozess der Verallgemeinerung physischer, räumlicher oder zeitlicher Details eines Systems beziehen, oder die Abstraktion konzentriert sich stattdessen auf ein benutzerwahrgenommenes Modell oder eine Darstellung von physischen Informationen.

Bei dem iSEF wird die künstliche Abgrenzung in fünf *System Abstraction Level* unterteilt: der *Environment Abstraction Level* (L-EV), der *User Domain Abstraction Level* (L-UD), der *Domain System Abstraction Level* (L-DS), der *System Product Abstraction Level* (L-SP) und der *Product Component Abstraction Level* (L-PC). Eine detaillierte Beschreibung ist dazu in [4] zu finden und eine beispielhafte Darstellung des Umfangs der Abstraktionslevel ist in Abbildung 1 gegeben. Jedes System und seine Subsysteme können in eine der Systemabstraktionsebenen eingeordnet werden [2]. Bei der Zerlegung eines Systems kann ein Teilsystem sofort in eine niedrigere Abstraktionsebene eingeordnet werden, wenn keine weitere Hierarchie angewendet wird. Die Entwicklung beginnt dabei immer auf der Ebene des Systemkontexts, des System of Interests.

Systemabstraktionsebenen bieten mehrere Vorteile: Sie ermöglichen die Ableitung neuer Systemkontexte von Subsystemen mit klaren Interaktionen zwischen Systemelementen wie Funktionen, Knoten und Komponenten, und verhindern ebenfalls ein "Level-Hopping" zwischen Elementen verschiedener Abstraktionsebenen. Zudem erlauben sie die Implementierung maschinell überprüfbarer Dekompositionsregeln für Architekturelemente auf den jeweiligen Abstraktionsebenen aufgrund verpflichtender Dekompositionsmodelle. Dies stellt sicher, dass Teams, die den Regeln folgen, immer auch gleiche, zueinander kompatible Systemstrukturen erstellen.

Gestützt auf standardisierte Interfaces aus den Bibliotheken des iSEF werden somit Architekturen geschaffen, die die Kommunikation zwischen den Entwicklungsteams verbessern und wiederverwendbare Systembausteine bereitstellen.

3.1.4 Hierarchy Depth (Hierarchische Modelltiefe)

Wird ein System auf einer Abstraktionsebene analysiert und über alle Viewpoints hinweg modelliert, spricht man von einem vollen Architekturschritt. Dabei können System- und Funktionsbausteine auf verschiedenen Viewpoints unterschiedliche Granularität aufweisen. Die hierarchische Modelltiefe [4] definiert deshalb die Granularität der Zerlegung eines Systems innerhalb einer Systemabstraktionsebene eines bestimmten Viewpoints. Um die Zerlegung einfach zu halten, ist die bevorzugte Hierarchietiefe eins. Jedoch muss die Hierarchietiefe zwischen den Viewpoints innerhalb einer Systemabstraktionsebene nicht konstant sein, da die Granularität je nach Zerlegungsschema variieren kann. Damit eine standardisierte Systemarchitektur sichergestellt wird, bietet das iSEF eine Reihe von Referenzmodellen an, mit denen die Architekten angeleitet werden, die Systembausteine über die Modellsichten hinweg so zu gestalten, dass ein leicht wiederverwendbarer Architekturschritt modelliert werden kann. Eine beispielhafte Darstellung der hierarchischen Modelltiefe ist in Abbildung 1 gegeben.

3.2 Methodische Modellierungsschritte zum Systemdesign

Aufbauend auf den drei Dimensionen der Architekturbeschreibung, wie in [18] aufgezeigt, wird die System-modellierung mit ACMBSE und iSEF anhand von acht grundlegenden Schritten möglich. Diese umfassen die Feature-Definition, Anforderungsableitung, Systemkontext-Definition, Verhaltensanalyse, funktionales Design, logische Komponentenentwicklung, technische Systementwicklung und die Realisierung der physischen Architektur.

Wird die in [18] gezeigte Methode richtig angewendet, ergibt sich eine Architekturbeschreibung, die über alle Abstraktionsebenen und Modellsichten bis zur Disziplin-spezifischen Architektur von Hardware und Software kohärent abgebildet werden kann. Natürlich ist es nicht immer sinnvoll, den gesamten 3D-Raum an Elementen zu modellieren. Hier ist die richtige Balance gefragt [2], sich auf wesentliche Pfade innerhalb des Modellierungsraums zu fokussieren. Die im iSEF enthaltenen Tailor-Modelle unterstützen den Modellierer dabei, den erforderlichen Pfad zu finden und alle wesentlichen Elemente darzustellen.

4 EXZELLENZ DURCH ANGEWANDTES SYSTEMS-ENGINEERING

Ein wesentlicher Aspekt für die korrekte Erstellung der Systemarchitektur ist neben dem bereitgestellten Framework und den Modellbibliotheken auch die entsprechende Ausbildung der Ingenieure, diese Werkzeuge richtig zu nutzen und damit die Engineeringsilos aufzubrechen und eine kohärente Arbeitsweise zu fördern. Ein Ausbildungsprogramm, wie im Folgenden vorgeschlagen, kann dabei die entsprechende Methodik lehren. Es sollte dann fundiert der ACMBSE-Ansatz vermittelt und in praktischer Anwendung mit den Modellierwerkzeugen in umfassenden Praxisübungen trainiert werden. Die wesentlichen Elemente dieser Ausbildung sollten den folgenden Inhalt umfassen.

- Grundlagen des architekturzentrierten Systems Engineering mit Fokus der Vermittlung der Prinzipien des ACMBSE wie in diesem Paper erläutert
- Spezialisierungen der erweiterten iSEF Modelliersprache basierend auf SysML/SysML V2 und UML mit Fokus auf alle Ingenieursdisziplinen
- Lehre und Anwendung aller Dekompositionsmodelle der entsprechenden Modellsichten und Integration zu einem kohärenten Modell
- Anwendung der ACMBSE geschulten Methodik und Notation bei der Umsetzung konkret vorgegebener Architekturen speziell in der EE-Entwicklung

- Detailmodellierung simulationsfähiger funktionaler Modellstrukturen von Grund auf mit Anwendung der vorhandenen iSEF-Funktionsbausteine
- Modellieren von technischen und physischen Systemarchitekturen, die nahtlos in Disziplinarchitekturen überführt werden können
- Modellieren von Eco-System Architekturen und Cloud-Systemen zur Realisierung aktuell existierender Systeme
- Schulen und trainieren der interdisziplinären Zusammenarbeit zwischen Software- und Hardwaredisziplin und der darüberliegenden Systems Engineering-Disziplin
- Erstellen eines durchgängigen Modells vom Systemkonzept bis zur Realisierung in den Disziplinen während des Kurses in Gruppenarbeit und Selbststudium

Eine zentrale Rolle spielt hierbei, die Fähigkeit aller System- und Diszipliningenieure zur Zusammenarbeit zu etablieren, den modellierten Inhalt der jeweils anderen Disziplin zu verstehen und zu lesen und eben auch eigene Modelle so zu erstellen, dass sie der jeweiligen anderen Disziplin in gleicher Sprache vermittelbar und verwendbar sind.

5 FORSCHUNGSBEDARF UND AUSBLICK

Das invenio Systems Engineering Framework (iSEF) adressiert die Herausforderungen, die Engineeringsilos zu überwinden, indem es einen integrierten Ansatz fördert, der verschiedene Sichtweisen verknüpft und die Zusammenarbeit zwischen den Disziplinen Systems Engineering, Software-Engineering und Hardware-Engineering stärkt. Durch die Verwendung von Abstraktionsebenen und Modellsichten ist die Sicherstellung kohärenter Datenmodelle gegeben und es werden die Silos im Engineering abgebaut. Dies verbessert nicht nur die Kommunikation und Koordination, zwischen den Engineering Teams, sondern erhöht auch die Fähigkeit, bereichsübergreifende Probleme frühzeitig zu erkennen und zu lösen, gewährleistet Datenkonsistenz und fördert die Wiederverwendung von Komponenten und Artefakten, was letztlich Innovation und Effizienz in Projekten vorantreibt.

Eine effiziente Anwendung des iSEF erfordert neben dem eigentlichen Regelwerk auch einen gewissen Anfangsinvest in die Erstellung von Out-Of-The-Box anwendbaren Modellbibliotheken. In den nächsten Monaten ist hier noch Entwicklungsarbeit notwendig, insbesondere bei domänenspezifischen Signalbibliotheken und grundlegenden Funktionsbausteinen. Sind die Grundbibliotheken einmal vorhanden, kann dann über einen angestrebten „Model Store“ von und mit der SE-Community ein umfassendes ECO-System an standardisierten und kompatiblen Modellbibliotheken dem Nutzer zur Verfügung gestellt werden.

Ein weiterer wichtiger Aspekt ist der zukünftige Einsatz generativer Künstlicher Intelligenz (KI) in der Analyse und Modellierung von Systemen. Aktuell finden erste Untersuchungen statt, die den Einsatz von KI-Agenten vorsehen. Diese Agenten können sowohl die Evaluierung von Modellen auf ihre Plausibilität unterstützen als auch wiederkehrende Aufgaben in der Erstellung von Schnittstellen und standardisierten Systembausteinen übernehmen. Damit ermöglicht der Einsatz von KI eine effiziente Erstellung und Anpassung dieser Modelle, was einen erheblichen Qualitäts- und Zeitgewinn für die Systementwicklung bietet.

Zur breiteren Anwendung dieses Frameworks sollten zusätzliche Ausbildungsprogramme in Kooperation mit Hochschulen weiter ausgebaut und die Methodik didaktisch systematisiert werden. Dabei spielt die praktische Ausbildung eine entscheidende Rolle, um sicherzustellen, dass die Ingenieure die notwendigen Fähigkeiten und Kenntnisse zur effizienten Nutzung des iSEF erwerben.

6 REFERENZEN

[1] T. Weilkiens, Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur. Mit einem Geleitwort von Richard Mark Soley. Heidelberg, Germany: dpunkt.verlag, 2014.

[2] T. Weilkiens, SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML. Morrisville, NC: Lulu.com, 2016.

[3] M. D. Brooks and T. M. Wheeler, "Experiences in Applying Architecture-Centric Model-Based System Engineering to Large-Scale, Distributed, Real-Time Systems," in Proceedings of the 2007 IEEE/ACM International Conference on Software Engineering (ICSE 2007), 2007.

[4] E. Seidel and M. Forlingieri, "Moving towards Server-Zone Architecture with MBSE at Continental," in INCOSE International Symposium, Hawaii, USA, Jul. 2023. [Online]. Available: https://www.incose.org/symposium2023.

[5] E. Seidel and M. Forlingieri, "Architecture-Centric Model-Based Development At Continental," in Proceedings of the AOSEC International Symposium, Bangalore, India, Oct. 2023. [Online]. Available: https://www.aosec2023.org.

[6] "Systems and software engineering — Architecture description," ISO/IEC/IEEE 42010:2011, 2011.

[8] "OMG Systems Modeling Language Version 1.6," OMG SysML, Dec. 2019. [Online]. Available: https://www.omg.org/spec/SysML/1.6.

[9] "Study of Modern Modeling Techniques for Model-Based Systems Engineering Methodologies," IJERT, Feb. 2023. [Online]. Available: https://www.ijert.org/study-of-modern-modeling-techniques-for-model-based-systems-engineering-methodologies.

[10] "Mastering State Diagrams in UML: A Comprehensive Guide," Visual Paradigm Guides, 2023. [Online]. Available: https://guides.visual-paradigm.com/mastering-state-diagrams-in-uml-a-comprehensive-guide/.

[11] "Engineering Silos," Heykona, 2023. [Online]. Available: https://www.heykona.com/blog/engineering-silos.

[12] "Reimagining Engineering to Deliver More Projects More Efficiently," McKinsey & Company, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/reimagining-engineering-to-deliver-more-projects-more-efficiently.

[13] A. de Waal, M. Weaver, T. Day, and B. van der Heijden, "Silo-Busting: Overcoming the Greatest Threat to Organizational Performance," Sustainability, vol. 11, no. 23, Article 6860, Dec. 2019. [Online]. Available: https://doi.org/10.3390/su11236860.

[14] "Systems Engineering," OOSE. [Online]. Available: https://www.oose.de/unsere-themen/systems-engineering.

[15] "Certification," INCOSE. [Online]. Available: https://www.incose.org/certification. [Accessed: June 7, 2024].

[16] G. Muller, L. van Veen, and J. van den Aker, "Systems Engineering Education: From Learning Program to Business Value," MDPI Systems, 2023.

[17] "OCSMP Certification," OMG. [Online]. Available: https://www.omg.org/certification. [Accessed: June 7, 2024].

[18] E. Seidel "Architecture-Centric-Model-Based-Systems-Engineering-ACMBSE" invenio. [Online]. Available:https://www.invenio.net/systems-engineering/wp-content/uploads/sites/7/Architecture-Centric-Model-Based-Systems-Engineering-ACMBSE_DE.pdf. [Accessed: Oktober 30, 2024].