Whitepaper

Praxisnahe Einführung von Model-based Systems Engineering

Vorgehen und Lessons Learnt

fortiss

Praxisnahe Einführung von MBSE –
Vorgehen und Lessons Learnt

Autoren

Wolfgang Böhm

Technische Universität München,
Fakultät für Informatik,
80290 München
boehmw@in.tum.de

Manfred Broy

Technische Universität München,
Fakultät für Informatik,
80290 München
broy@in.tum.de

Maximilian Junker

Qualicen GmbH,
Lichtenbergerstr. 8,
85748 Garching b. München
maximilian.junker@qualicen.de

Andreas Vogelsang

Technische Universität Berlin,
Fakultät Elektrotechnik und Informatik,
Daimler Center for Automotive IT Innovations,
10587 Berlin
andreas.vogelsang@tu-berlin.de

Sebastian Voss

fortiss GmbH,
Guerickestr. 25,
80805 München
voss@fortiss.org

Inhalt

Kurzfassung

In der Entwicklung leistungsstarker, komplexer cyber-physischer Systeme hat die Anwendung eines modellbasierten Ansatzes eine Fülle von Vorteilen. Dadurch lassen sich Themen aus dem Anwendungsgebiet, aus der Architektur und aus der Implementierungsebene entkoppeln. Modelle erlauben es, sich auf unterschiedliche Aspekte zu konzentrieren und dadurch die Komplexität deutlich zu reduzieren. Arbeitet man mit wohlgewählten Modellierungskonzepten, so lässt sich das Verhalten von cyber-physischen Systemen und ihren Teilsystemen hinreichend genau durch geeignet gewählte Modelle beschreiben. Die Modelle dienen gleichermaßen zur Dokumentation der Entwicklung wie auch zur Unterstützung der weiteren Schritte. Im Extremfall versteht man die gesamte Entwicklung eines cyber-physischen Systems als Prozess, Schritt für Schritt für die anstehenden Entwicklungsaufgaben geeignete Modelle auszuarbeiten. Aus den Modellen heraus kann man die unterschiedlichsten Aufgaben im Entwicklungsprozess wie die Spezifikation der Anforderungen und der Architektur, die Simulation, die Generierung von Code und von Testfällen sowie auch die Dokumentation abdecken. Es wird dargestellt, wie die aus einer Reihe von Verbundprojekten des Bundesforschungsministeriums (BMBF) entstandene Methodik die modellbasierte Entwicklung für cyber-physischer Systeme vollständig abdeckt. Behandelt werden auch Ansätze für eine kontrollierte Einführung und für die Unterstützung der Migration in die neuen Entwicklungsparadigmen.

Einleitung

Technische Systeme mit hohem Softwareanteil, sogenannte Cyber-Physical Systems, wie etwa Fahrzeuge, Flugzeuge oder Produktionssysteme, haben immer stärkeren Einfluss auf unser tägliches Leben. Dabei nehmen deren Komplexität sowie die Erwartungen an deren Verfügbarkeit, Sicherheit, aber auch Benutzbarkeit und Funktionsumfang ständig zu. Cyber-Physical Systems werden heute stark verteilt über Organisationen, Disziplinen und Anwendungsdomänen hinweg entwickelt. Über die letzten zehn Jahre haben wir die wissenschaftlichen Grundlagen für die Entwicklung dieser Systeme geschaffen und in einer Reihe von großen Forschungsprojekten eine entsprechende Methodik aktiv mitgestaltet. In diesen sogenannten SPES-Projekten (SPES2020, SPES_XT, SPEDiT, CrESt), die durch das Bundesministerium für Bildung und Forschung gefördert wurden, sind wir mit insgesamt über 30 Partnern aus Wissenschaft und Wirtschaft zu dem Ergebnis gekommen, dass es in Zukunft nur mit einem modellbasierten Ansatz möglich sein wird, Cyber-Physical Systems zielgerecht zu entwickeln. In diesem Artikel fassen wir unsere wesentlichen Erkenntnisse in Bezug auf Modellbasiertes Systems Engineering (MBSE) der letzten zehn Jahre zusammen und beschreiben konkrete Hinweise für eine praxisnahe und effektive Einführung.

Was ist MBSE und was bringt es?

Dokumentenzentrierte vs. Modellbasierte Entwicklung

Dokumentenzentrierte Entwicklung

Heute werden Systeme in der Entwicklung weitestgehend über Dokumente beschrieben. Ziel der Dokumentation ist es zunächst, ein gemeinsames Verständnis für das System herzustellen. Dazu dienen die Dokumente als Kommunikationsbasis zwischen verschiedenen an der Entwicklung beteiligten Rollen und Organisationseinheiten. Die Dokumente sind außerdem Grundlage vertraglicher Beziehungen, beispielsweise zu einem Lieferanten. Schließlich bilden die Dokumente die Basis für Entwicklungsaktivitäten wie die Spezifikation des Systems oder dessen Test.

Um ein komplexes System wie ein Automobil zu beschreiben, werden somit in der Regel Hunderte Dokumente auf unterschiedlichen Abstraktionsstufen benötigt: Ausgehend von Dokumenten, die die übergeordneten Ziele des Unternehmens und der Stakeholder für das System beschreiben, werden Dokumente zur detaillierten Darlegung von Funktionen und Funktionsgruppen erstellt und darauf aufbauend wiederum Dokumente für eine detaillierte Komponentenspezifikation abgeleitet. Jedem Dokument sind üblicherweise Personen zugeordnet, die für dieses Dokument verantwortlich sind.

Die meisten dieser Dokumente sind in natürlicher Sprache (etwa Deutsch, Englisch oder andere Sprachen) verfasst. Zusätzlich können die Spezifikationen Diagramme oder Abbildungen, vielleicht sogar

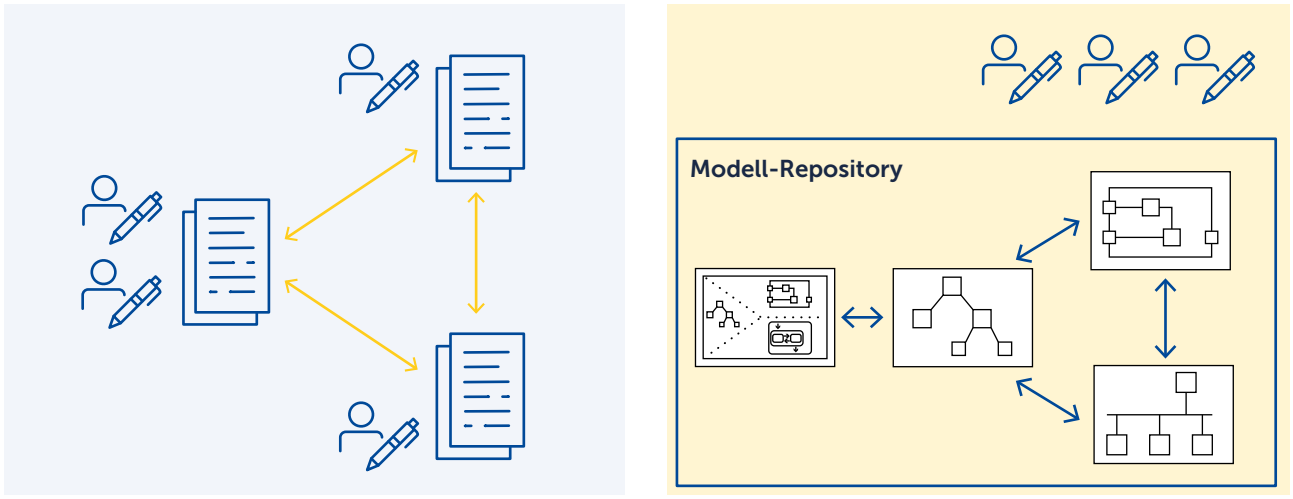


Abbildung 1: Dokumentenzentrierte und Modellbasierte Entwicklung

einige Modelle zur Beschreibung der Architektur oder des Verhaltens des Systems oder seiner Komponenten beinhalten. Um die Zusammenhänge zwischen den Teilen der Dokumente zu verstehen, verwenden die Autoren Textverweise zwischen einzelnen Elementen eines Dokuments oder zwischen Dokumenten. Das Resultat ist ein komplexes Gebilde von miteinander verknüpften Dokumenten, das über den Lebenszyklus des Produkts kaum in sich konsistent zu halten ist.

Die Formulierung in natürlicher Sprache birgt immer das Risiko falscher Interpretationen. Darüber hinaus gibt es Aktivitäten, die aufgrund einer solchen Beschreibung nicht einfach durchgeführt werden können, etwa Fragen wie „Passen die Schnittstellen zwischen Komponente A und B zusammen?" oder „Welche Funktionen beeinflussen die Funktion X?". Dazu sind in der Regel aufwendige Reviewprozesse notwendig. Außerdem kann auf Basis einer natürlichsprachlichen Beschreibung typischerweise keine Simulation des Systems durchgeführt werden.

Modellbasierte Entwicklung

Der Ansatz, der durch MBSE verfolgt wird, grenzt sich hiervon deutlich ab. Die Kernidee der Systemmodellierung ist es, nicht nur textuell zu beschreiben, welche Ziele, Funktionen, Komponenten, Schnittstellen oder Qualitätseigenschaften ein System erfüllt oder bereitstellt, sondern explizite Modellelemente dafür zu erstellen. Die Stärke der Modelle liegt dabei darin, dass sich diese auf wohldefinierte und gut verstandene Konzepte der Domäne stützen. Die Nutzung von vorgefertigten Modelltypen hat den Vorteil, dass diese bereits eine Reihe von Eigenschaften wie zum Beispiel Kompositionalität mitbringen. Das bedeutet, dass die Integration von zwei Teilen, die als Modell beschrieben sind, klar definiert und vorhersagbar ist. Wir haben in der Projektarbeit sehr viel Mühe in die wissenschaftliche Fundierung dieses Ansatzes investiert. Anstatt zu beschreiben, welche Elemente zusammengehören, oder textuelle Links zu setzen, erstellen wir Modellelemente für die Beziehungen (z.B. die Kommunikation über eine Schnittstelle). Eine Systembeschreibung ist dann nicht mehr dieses komplexe Gebilde von Dokumenten, sondern ist ein verflochtenes Netz von normierten Modellelementen, die ein gemeinsames Ganzes bilden – eine Instanz des Systemmodells. Das Systemmodell ist nicht mehr in einzelnen Dokumenten verteilt, sondern befindet sich in einem zentralen Modell-Repository. Während im dokumentbasierten Modus die verschiedenen Autoren unabhängig auf den verteilten Dokumenten arbeiten, wird in der modellbasierten Entwicklung stets auf dem einen zentralen Repository gearbeitet. Auf dieses zentrale Modell haben Stakeholder dann unterschiedliche Sichten, die exakt auf ihre jeweilige Rolle im Produktentstehungsprozess (z.B. Funktionsentwickler, Architekt etc.) zugeschnitten sind.

Der Übergang von Text zu Modellen bietet darüber hinaus den Vorteil, Mehrdeutigkeiten zu reduzieren. Je formaler das Modell, das verwendet wird, desto weniger Mehrdeutigkeit gibt es in der Beschreibung. Weiterhin bieten Modelle die Möglichkeit, automatische Analysen darauf anzuwenden, etwa um das Zusammenspiel der einzelnen Komponenten zu prüfen.

Abbildung 2: Ziele von MBSE

Ziele von MBSE

MBSE verspricht eine Reihe von Vorteilen gegenüber dokumentenzentrierter Entwicklung. Abbildung 2 zeigt die wichtigsten Ziele von MBSE in der Übersicht.

Beherrschung von Komplexität

Mithilfe von MBSE sind wir in der Lage, komplexe Abhängigkeiten in einem System besser zu verstehen und die Auswirkungen von Änderungen abschätzen zu können. Beispiele von solchen Abhängigkeiten sind Abhängigkeiten zwischen den Funktionen eines Systems (Beispiel: die Crash-Detection-Funktion eines Autos beeinflusst die Fensterheber-Funktion). Diese Abhängigkeiten wachsen überproportional mit der Größe des Systems und sind den meisten Beteiligten häufig nicht klar. Mithilfe der Verknüpfung zwischen Modellelementen (z.B. Eingabe-Ausgabe-Beziehung zwischen Funktionen) werden diese Abhängigkeiten jedoch explizit gemacht und sind dadurch einer Analyse zugänglich.

Unterstützung modularer Vorgehensweisen

Durch die geeignete Wahl der Modellierungselemente, allem voran ein leistungsfähiges Schnittstellenkonzept und ein modularer Kompositionsansatz, wird sichergestellt, dass auf Basis bewährter Entwicklungsprinzipien wie Kapselung, Schnittstellenabstraktion und Information Hiding eine modulare, arbeitsteilige Vorgehensweise gesichert ist.

Konfektionierte Systembeschreibung

Der modellbasierte Ansatz stellt geeignete Modellelemente für die unterschiedlichen Entwicklungsaufgaben zur Verfügung. Damit ist ein Satz von standardisierten Beschreibungsmitteln vorgegeben, die ein einheitliches Verständnis und geeignete Abstraktionsebenen unterstützen.

Konsistente Systembeschreibung

Durch die Nutzung einer (formalen) Modellierungssprache werden bestimmte Inkonsistenzen, beispielsweise die Verknüpfung von Eingaben und Ausgaben mit unterschiedlichen Typen (Datentypen oder unterschiedliche physikalische Einheiten), von vornherein vermieden. Außerdem können durch die oben erwähnten Analysen weitere Inkonsistenzen (z.B. im Verhalten von interagierenden Komponenten) bereits in einem frühen Entwicklungsstadium entdeckt und behoben werden. Dadurch wird es einfacher, eine konsistente Systembeschreibung zu erstellen.

Verbesserte Wiederverwendung

Ein zentrales Konzept in dem hier vertretenen MBSE-Ansatz sind Schnittstellen. Alle Elemente (Funktionen, Subsysteme, Komponenten etc.) sind durch eine Schnittstelle definiert, über die das Element mit seiner Umgebung interagiert und die die interne Implementierung des Elements kapselt. Diese Schnittstelle macht es einfacher, Modellelemente wiederzuverwenden, da die Schnittstelle

einen Vertrag darstellt zwischen demjenigen, der das Element beschreibt oder entwickelt, und demjenigen, der es nutzt. Außerdem können dadurch, dass alle Modellelemente in dem zentralen Modell-Repository liegen, Modellelemente auch leichter aufgefunden und in andere Modelle eingebunden werden.

Beherrschung von Varianten

Um eine größere Palette an Produkten anbieten zu können und gleichzeitig möglichst große Teile zwischen ähnlichen Produkten wiederzuverwenden, müssen Varianten von Produkten oder Produktteilen explizit beherrscht werden. Die Komplexität und die Anzahl der Varianten steigen mit der Produktgröße. Mit dokumentenzentrierten Ansätzen ist diese Vielfalt an Varianten mit verschiedensten Variationspunkten, die wieder untereinander in Abhängigkeiten stehen, nicht mehr beherrschbar. MBSE bietet hierzu aufgrund der feingranularen und präzisen Erfassung von Systemteilen und ihren Abhängigkeiten jedoch die Möglichkeit.

Automatisierung und Werkzeugunterstützung

Die semantisch fundierten Modellelemente erlauben eine Werkzeugunterstützung mit einem hohen Automatisierungsgrad, sowohl bei der Generierung von Modellen oder Code als auch durch Verifikation.

Verbesserte Kommunikation

Durch die Nutzung einer Modellierungssprache erhöht sich die Präzision der Beschreibung des Systems. Schon bei geringem Formalisierungsgrad der Modelle ist die Gefahr von Missverständnissen geringer. Dazu kommt, dass viele Modellierungssprachen Möglichkeiten der grafischen Beschreibung von Modellen bieten. Beide Aspekte tragen dazu bei, dass die Kommunikation der Stakeholder über das System vereinfacht wird.

Front Loading und verbesserter Prozess

Als Front Loading bezeichnet man das Verschieben von Aufwänden in frühere Phasen der Entwicklung, mit dem Ziel, durch Steigerung der Qualität in diesen Phasen die Gesamtaufwände zu verringern (siehe *Kosten und Qualität*). Durch das Front Loading werden, wie oben beschrieben, Fehler und dadurch notwendige Nacharbeiten vermieden. Dadurch wird der gesamte Prozess berechenbarer. Die Aufwände in den frühen Phasen sind besser planbar als die durch Fehler verursachten ungeplanten Änderungen. Zudem steigen die Kosten für die Beseitigung von Fehlern umso mehr, je später sie in der Produktentwicklung gefunden werden.

Kosten und Qualität

MBSE trägt dazu bei, die Lebenszyklus-Kosten der Systembeschreibung zu senken. Zwar ist der Aufwand zur Einführung von MBSE nicht zu vernachlässigen und insbesondere zu Beginn ist die Erstellung der Modelle mit einem höheren Zeitaufwand verbunden als die Beschreibung wie gewohnt mithilfe von Dokumenten, dennoch werden diese Kosten durch die Vermeidung von Fehlern (und damit Kosten durch Nacharbeiten) und die besseren Wiederverwendungsmöglichkeiten kompensiert. Durch Vorverlagerung verringert sich das Entwicklungsrisiko, da die Wahrscheinlichkeit, grundlegende Probleme erst spät im Entwicklungsprozess zu erkennen, gesenkt wird. Zudem steigt der Grad der Wiederverwendung von Artefakten (siehe *Verbesserte Wiederverwendung*) bei einem modellbasierten Entwicklungsprozess und hat somit positive Kostenauswirkungen gerade auch für zukünftige Produktentwicklungen. Die modellbasierte Entwicklung zeichnet sich durch einen höheren Automatisierungsgrad und Front-Loading-Aktivitäten aus. Studien zeigen beispielsweise, dass durch die Vorverlagerung von Testaktivitäten und die Möglichkeit der automatisierten Testfallerstellung aus Modellen Kosteinsparungen von zehn bis zwölf Prozent möglich sind [8]. Insgesamt führt die Nutzung von MBSE damit zu einer Verringerung der Kosten über den gesamten Lebenszyklus des Systems bei gleichzeitig höherer Qualität.

Herausforderungen bei der Einführung von MBSE

Bei der Einführung von modellbasiertem Systems-Engineering ist eine Reihe von Herausforderungen zu meistern. Für Unternehmen gibt es, wie zuvor in Kapitel *Ziele von MBSE* beschrieben, gute Gründe, von einer dokumentenzentrierten Entwicklung zu MBSE zu wechseln. Dabei ist es wichtig, auf diese besonderen Herausforderungen bei der Einführung von MBSE Rücksicht zu nehmen.

Um die Einführung möglichst zielführend und erfolgreich zu gestalten, ist es von großer Bedeutung, die individuellen Gegebenheiten jedes einzelnen Unternehmens herauszuarbeiten. Eine aus den Zielen abgeleitete systematische Einführung ermöglicht es, den individuellen Herausforderungen adäquat zu begegnen. Entscheidend ist es zu verstehen, dass viele der Herausforderungen nicht losgelöst voneinander betrachtet werden können, sondern sich gegenseitig bedingen.

Grundsätzlich unterscheiden wir bei der Einführung von MBSE zwei Gruppen von Herausforderungen: technisch-methodische sowie organisatorische Herausforderungen:

Technisch-methodische Herausforderungen

Technisch-methodische Herausforderungen beziehen sich auf die geeignete Auswahl einer Methodik, einer Modellierungssprache und eines Modellierungswerkzeugs. Diese drei Aspekte müssen getrennt voneinander betrachtet, aber gut aufeinander abgestimmt werden.

Methodik

Ein Systemmodell ist selbst ein komplexes Artefakt, das ohne eine dahinterliegende fundierte Methodik nicht effektiv und effizient erstellt werden kann. Zu einer MBSE-Methodik gehört beispielsweise die Definition der relevanten Modelltypen und ihrer Beziehungen. Ferner definiert eine Methodik Sichten (Views) auf das System. Eine Sicht beschreibt dabei einen speziellen Ausschnitt des Systemmodells, der etwa für einen Stakeholder oder eine Entwicklungsaktivität relevant ist. Durch die Nutzung von Sichten kann ein komplexes Gesamtmodell in mehrere weniger komplexe und der gegebenen Entwicklungssituation angepasste Modelle zerlegt werden. Beispiele für Sichten sind Funktionssicht und logische bzw. technische Architektursicht. Eine MBSE-Methodik beschreibt weiterhin Möglichkeiten zur Analyse und Generierung für die spezifischen Modelle. Ein Beispiel für eine MBSE-Methodik ist das SPES Modeling Framework, auf das wir in diesem Beitrag noch etwas ausführlicher eingehen werden.

Modellierungssprache

Eine Modellierungssprache definiert die Syntax und Semantik, mit der Modelle der Methodik beschrieben werden. Anders als eine Methodik legt die Modellierungssprache genau fest, welche textuellen oder grafischen Notationen erlaubt sind (Syntax). Die Semantik einer Modellierungssprache legt fest, welche Bedeutung die Notationen haben. Problematisch hierbei ist, dass viele der gängigen Modellierungssprachen (etwa SysML) bestenfalls eine lose definierte Semantik haben. In SysML ist beispielsweise nur für wenige Diagrammtypen eine präzise Semantik festgelegt (z.B. für State Charts), für andere ist nur eine lose, informelle Semantik beschrieben. Dadurch können Probleme ähnlich zu denen von textuellen Beschreibungen auftreten.

Werkzeuge

Um effektiv und effizient modellieren zu können, müssen die Methodik und die Sprache von einem passenden Werkzeug unterstützt werden. Nur so kann von den Möglichkeiten, die Modelle bieten, effizient Gebrauch gemacht werden. Beispielsweise können mithilfe von Werkzeugen automatische Analysen auf den Modellen durchgeführt oder Simulationen gefahren werden. Wichtig ist, dass das genutzte Werkzeug die gewählte Methodik und die Sprache sowohl syntaktisch als auch semantisch unterstützt.

Organisatorische Herausforderungen

Neben diesen technisch-methodischen Herausforderungen gibt es auch organisatorische Herausforderungen, die adäquat adressiert werden müssen.

Management-Unterstützung

Die Einführung von MBSE ist mit einem nicht zu unterschätzenden Aufwand (Upfront Invest) verbun-

den. Um die Vorteile von MBSE langfristig nutzen zu können, ist gerade in frühen Entwicklungsphasen Mehraufwand zu leisten. Die Erstellung von Modellen auf Basis einer durchgängigen Entwicklungsmethodik ist für viele Mitarbeiter neu und mit einem Zusatzaufwand verbunden. Neuartige Systementwicklungen durchlaufen zwangsläufig eine Lernkurve, die Ressourcen und Zeit kosten. Wenn dieser Mehraufwand nicht durch das Management und eine entsprechende Projektplanung unterstützt wird respektive das Management kein Verständnis aufbringt, dass zunächst mit längeren Entwicklungszyklen zu rechnen ist, ist die Einführung von MBSE zum Scheitern verurteilt: Erfolgreiche MBSE-Einführung ist nur durch konsequente Management-Unterstützung möglich. „Wann und mit wie viel Gewinn kann das Management rechnen?" ist eine häufig gestellte Frage. Da die Anwendungsszenarien und besonderen Gegebenheiten in Unternehmen zu unterschiedlich sind, gibt es unseres Erachtens keine pauschale und nur wenige quantifizierbare Aussagen zum Return on Invest (ROI) bei der Einführung von MBSE (siehe S.7, *Kosten und Qualität*).

Wissensvermittlung

Um eine durchgängige Entwicklung (und so die zahlreichen Vorteile von MBSE) zu erreichen, müssen unterschiedliche Abteilungen eng miteinander zusammenarbeiten, was einen schnellen Aufbau von Inhouse-Kompetenzen erfordert. Die Art und Weise der Zusammenarbeit hängt dabei eng mit der Methodik zusammen und muss an die Mitarbeiter der einzelnen Abteilungen herangetragen werden, die in die Entwicklung involviert sind. Insbesondere in Bereichen der Entwicklungsmethodik ist entscheidend, dass im Unternehmen schrittweise Wissen aufgebaut und an die einzelnen Mitarbeiter weitergegeben wird. Als besonders effektiv hat sich das weiter hinten im Kapitel *Strukturiere Einführung mit dem MBSE-Reifegradmodell* beschriebene Coaching-Modell erwiesen.

Akzeptanz und Change-Management

Der Wechsel von einer dokumentenzentrierten zu einer modellbasierten Entwicklung ist mit tiegreifenden Änderungen in der Systementwicklung verbunden. Gewohnte Arbeitsabläufe und Entwicklungsschritte ändern sich. Die Einführung von MBSE hat somit einen Einfluss auf jeden einzelnen Mitarbeiter. Um die nötige Akzeptanz der Mitarbeiter zu erlangen, sind mehrere Faktoren von großer Bedeutung: Neben der Auswahl eines „guten" Startprojekts, das in frühen Phasen erste Vorteile von MBSE verdeutlicht und neben dem initialen Mehraufwand auch spätere Arbeitserleichterungen aufzeigt, sollte unbedingt die notwendige Management-Unterstützung vorhanden sein, um die Mitarbeiter bei der Umstellung zu motivieren. Dabei ist ein Change-Management-Prozess unumgänglich, der sich um tagtägliche Bedarfe der Mitarbeiter kümmert und vor allem die Möglichkeit zur Wissensvermittlung durch Coachings und Schulungen bietet, um frühestmöglich auch die Vorteile von MBSE zu vermitteln.

Schrittweiser Roll-out

Die Einführung von modellbasiertem Systems-Engineering kann im Regelfall nicht von heute auf morgen und für die gesamte Organisation gleichzeitig erfolgen. Eine schrittweise und systematische Einführung ist hier die zielführende Variante. Ein überschaubares Pilotprojekt ermöglicht es etwa dem Unternehmen, MBSE im Unternehmenskontext anzuwenden und erste Erfahrungen mit der Methodik und den eingesetzten Werkzeugen zu sammeln. Gleichzeitig kann man auch erste Schwerpunkte setzen, indem man sich besonders auf methodisch ausgereifte Artefakte eines Entwicklungsschritts (wie ein funktionaler Prototyp) konzentriert. Immer ist dabei auf die speziellen Gegebenheiten und Ziele eines einzelnen Unternehmens einzugehen, heißt: Eine speziell auf das Unternehmen zugeschnittene Einführung von MBSE, die die besonderen Gegebenheiten berücksichtigt, ist von entscheidender Bedeutung. Eine strukturierte und in der Praxis erprobte Einführung von MBSE stellen wir im nächsten Kapitel vor.

Methodisches Vorgehen für MBSE

Wie zuvor ausgeführt, ist die Einführung von MBSE keine triviale Aufgabe, denn MBSE bricht mit vielen traditionellen Systems-Engineering-Ansätzen. Dabei erfordert es auch ein grundsätzliches Umdenken, wie ein System entwickelt wird. Das Umdenken betrifft insbesondere die Entwicklungsmethodik und die damit erstellten Artefakte. Mit dem SPES Modeling Framework wurde ein Rahmenwerk für eine umfassende, werkzeug- und modellierungssprachenunabhängige Methode für MBSE entwickelt, das einen umfangreichen Satz von konkreten Modellierungstechniken und -aktivitäten bietet, die gleichzeitig auf einem formalen mathematischen Fundament aufbauen. Entscheidend für ein erfolgreiches Anwenden von MBSE sind die vier Grundprinzipien der SPES-Methodik:

- Schnittstellen überall
- Modulare Komposition
- Vier Sichten auf das System
- Teilsysteme und Komponenten auf verschiedenen Granularitätsebenen

MBSE – Begriffe und Konzepte

Wir stellen zunächst einige Begriffe und Konzepte aus dem Rahmenwerk vor, die für den modellbasierten Entwicklungsansatz von entscheidender Bedeutung sind. Dabei verzichten wir bewusst auf eine formale Definition der Begriffe und wählen stattdessen einen pragmatischen Ansatz.

Was ist ein System?
Im Fokus modellbasierter Entwicklung steht stets ein klar umrissenes System. Ein System besteht aus einer Menge von Elementen, die derart organisiert sind, dass sie gemeinsam eine bestimmte Funktion (oder eine Menge von Funktionen) erfüllen [6], und zeichnet sich dadurch aus, dass es durch eine Schnittstelle eindeutig von seiner Umgebung abgegrenzt ist. Durch die Schnittstelle ist festgelegt, was Teil des Systems und was Teil seiner Umgebung ist. Grundsätzlich bildet das gesamte Universum außerhalb eines Systems die Umgebung des Systems. Wir betrachten jedoch nur den Teil, der für das System relevant ist, d.h. unmittelbare Auswirkungen auf das System hat oder durch das System beeinflusst wird (Kontext). Elemente des Kontexts können weitere Systeme, Dienste und Daten oder physikalische und technische Prozesse sein, die auf das System einwirken. Auch Menschen, die mit dem System interagieren, befinden sich in dessen Kontext.

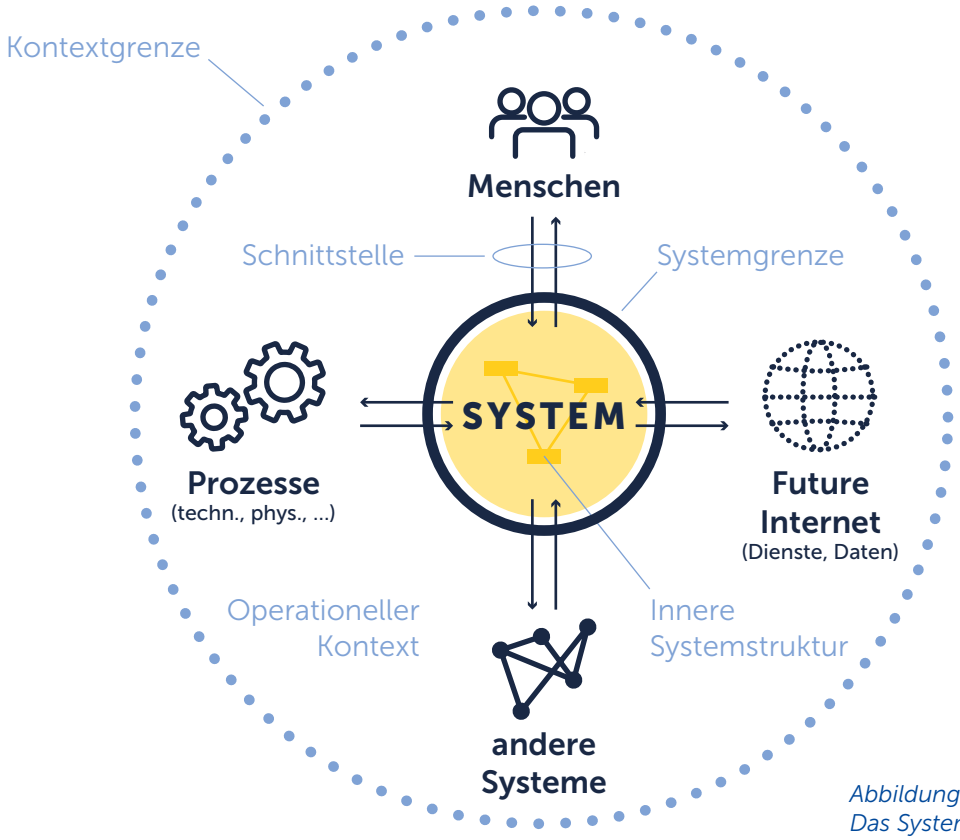


Abbildung 3:
Das System in seinem Kontext

Die Schnittstelle spezifiziert die Struktur (Syntax), die das System seiner Umgebung über die Systemgrenze bereitstellt, sowie das Verhalten (Semantik) des Systems, das an seiner Grenze beobachtet werden kann. Die Schnittstelle realisiert zwei wesentliche Prinzipien des Systems-Engineering: Kapselung und Information Hiding (siehe S. 6, *Unterstützung modularer Vorgehensweisen*). Zudem weist ein System eine innere Struktur auf, bestehend aus zueinander in Beziehung stehenden Elementen. Der Systembegriff und das damit eng verknüpfte Schnittstellenkonzept sind die wesentlichen Bausteine der SPES-Methodik, auf die wir im Folgenden näher eingehen.

Da das System ausschließlich über seine Schnittstelle mit seiner Umgebung kommuniziert, spielt die Systemumgebung, die über ein Kontextmodell erfasst wird, eine wichtige Rolle. Kontextmodelle erlauben beispielsweise auch, das Verhalten des Systems bereits in frühen Phasen der Entwicklung zu simulieren.

Abbildung 3 verdeutlicht den in der Methodik zugrunde gelegten Systembegriff grafisch.

Systemmodell und Formalisierung
Im MBSE-Ansatz werden Systeme durch Modelle repräsentiert. Ein Systemmodell ist ein konzeptuelles („generisches") Modell zur Beschreibung von Systemen und deren Eigenschaften. Es beschreibt, was ein System als Ergebnis einer Konzeptualisierung ausmacht. Systemmodelle definieren die Bestandteile des Systems und dessen Struktur, die wesentlichen Eigenschaften und sonstige Gesichtspunkte, die bei der Entwicklung zu berücksichtigen sind. Systemmodelle legen fest, worüber Anforderungen reden (Subjekt des Diskurses).

Eine entscheidende Frage beim modellbasierten Ansatz ist, in welcher Sprache und mit welcher Präzision die einzelnen Modelle beschrieben werden. Grundsätzlich sind auch textuelle Systembeschreibungen denkbar. Jedoch schaffen erst formalisierte Systemmodelle die Grundlage für eine präzise Beschreibung von Systemeigenschaften (Formalisierung). Sollen Eigenschaften durch logische Prädikate dargestellt („formalisiert") werden, so ist ein mathematisches Modell für Systeme erforderlich. Der hier vorgestellte methodische Ansatz basiert in weiten Teilen auf einem solchen mathematischen Modell.

Sichten, Viewpoints und Views
Um die Entwicklung des Systems und dessen Systemmodelle möglichst systematisch zu strukturieren, definiert man Sichten (Views) auf das Systemmodell, die beispielsweise eine Repräsentation des Systems aus der Perspektive einer bestimmten, für die Systementwicklung wichtigen Rolle (Kunde, Projektmanager, Entwickler, Systemarchitekt, Qualitätsmanager etc.) darstellen. Diese Views definieren somit Abstraktionen des Systems, indem sie sich auf bestimmte Aspekte konzentrieren, die für die jeweilige Rolle wichtig sind. Die strukturierte Sammlung von Notationen sowie Konstruktions-, Interpretations- und Analysetechniken zur Erstellung der Sichten, die im Entwicklungsprozess eines Systems angewandt werden, werden Viewpoints genannt: Ein Viewpoint ist immer der Standpunkt, von dem aus man auf das System blickt, eine View ist das, was man von diesem Standpunkt aus sieht (siehe ISO-Definition 42010, [7]).

Analyse- und Synthesemechanismen
Der Formalisierungsgrad bestimmt die Analyse- und Synthesemöglichkeiten, die automatisiert auf Basis der erstellten Modelle durchgeführt werden können, d.h., dass bestimmte Prüfungen und Analysen oder auch das Generieren von Modellen automatisiert mithilfe von Werkzeugen vorgenommen werden können. Beispielsweise kann geprüft werden, ob ein funktionales Modell den modellierten Anforderungen gerecht wird, ob die technische Architektur alle Echtzeitanforderungen erfüllt und vieles andere mehr. Mithilfe von (formalen) Kontextmodellen kann zudem das Verhalten des Systems simuliert werden. Neben dem Formalisierungsgrad der Modelle spielen bei der Frage nach der Automatisierbarkeit gewisser Entwicklungsschritte natürlich auch die Möglichkeiten, die die verwendeten Werkzeuge bieten, eine entscheidende Rolle. Automatisierbarkeit ist ein entscheidender Vorteil modellbasierter Entwicklung.

Das SPES Modeling Framework

Wir geben im Folgenden einen kurzen Überblick über das SPES Modeling Framework. Die Modelle des Frameworks decken dabei weite Teile des Systementwicklungsprozesses ab.

Den Kern des Frameworks bildet das universelle Schnittstellenkonzept, das für alle Elemente Schnittstellen definiert, die jeweils aus der Schnittstellensyntax und einer Beschreibung des an der Grenze des Elements beobachtbaren Verhaltens bestehen. Seien es Anforderungen, Funktionen, logische oder technische Komponenten, jedes dieser Elemente wird über seine Schnittstelle beschrieben. Verschiedene Elemente stehen über ihre Schnittstellen miteinander in Verbindung.

Die Grundidee des SPES Modeling Frameworks ist es, ein System unter verschiedenen Views (Abstraktionen) zu betrachten und unterschiedliche, mög-

lichst formale und in sich konsistente Modelle ein und desselben Systems zu erarbeiten. Diese Views definieren Ausschnitte aus dem Systemmodell, die für die aktuelle Entwicklungsaufgabe alle relevanten Informationen zur Verfügung stellen. Ein Beispiel für eine solche View ist die Konzentration auf die Systemlogik im logischen Viewpoint.

Views im SPES Modeling Framework sind Anforderungen (Requirements View), funktionale Sicht (Functional View) sowie logische Architektur (Logical View) und technische Architektur (Technical View), die das System in die beteiligten logischen oder technischen Komponenten zerlegen.

Für übergreifende Themen (sog. Crosscutting Topics) werden die Modelle der Viewpoints entsprechend ergänzt. Das erlaubt beispielsweise, Aspekte der funktionalen Sicherheit von Systemen zu analysieren.

Granularitätsebenen

Um die Komplexität des Systems und des zugehörigen Entwicklungsprozesses beherrschbar zu machen, werden ausgewählte Architekturkomponenten nach dem Prinzip „Teile und herrsche" als eigenständige (Sub-)Systeme betrachtet, für die wiederum Modelle und Views nach dem SPES-Ansatz erstellt werden. Dadurch entsteht eine Reihe von Granularitätsebenen, auf denen ein System und dessen Subsysteme betrachtet werden. Die Modellierung auf verschiedenen Granularitätsebenen ändert jeweils das Subjekt des Diskurses (Scope) und ist ein wichtiges Mittel in der modellbasierten Entwicklung, um die Systemkomplexität zu reduzieren und den Entwicklungsprozess insgesamt handhabbar zu gestalten. Auf der obersten Granularitätsebene befinden sich immer die Modelle, die das betrachtete System als Ganzes abbilden. Es folgen (unterschiedlich viele) weitere Granularitätsebenen, die – sukzessive – immer feinere Teilsysteme betrachten und mehr Details modellieren. Das mathematische Modell FOCUS, auf dem das SPES Modeling Framework basiert, stellt die Konsistenz der Modelle von System und Subsystemen sicher. Granularitätsebenen helfen dabei, (1) die Komplexität des betrachteten Systems zu beherrschen, (2) Prüfungen am System auf verschiedenen Ebenen der Komplexität durchzuführen, (3) Entwicklungsaufgaben, beispielsweise an Zulieferer, zu verteilen und (4) einzelne Modelle mehrfach wiederzuverwenden.

Neben Abstraktion und Granularität ist Konsistenz ein wichtiges Merkmal der Modelle im SPES Modeling Framework. Wir unterscheiden zwischen horizontaler und vertikaler Konsistenz. Zwei Modelle sind horizontal konsistent, wenn sie zu

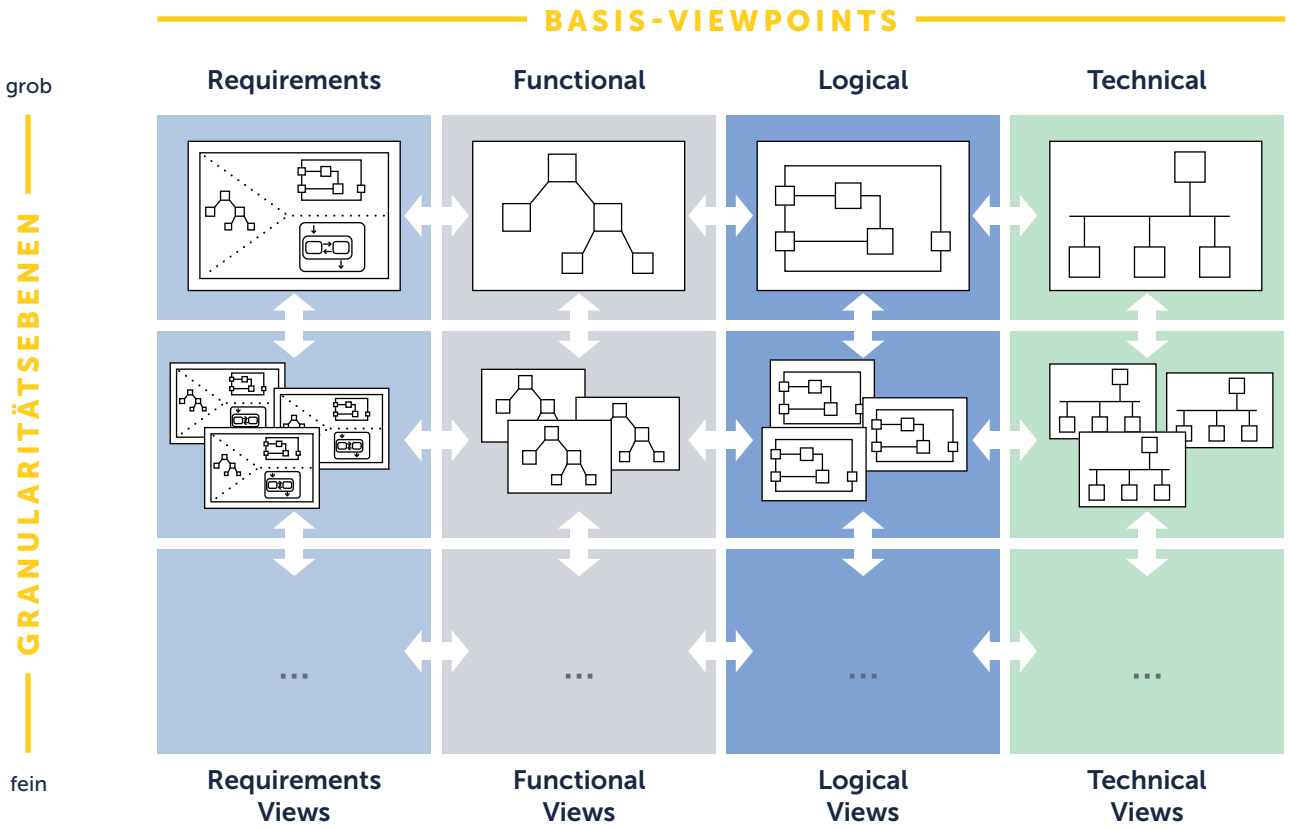


Abbildung 4: SPES-Matrix

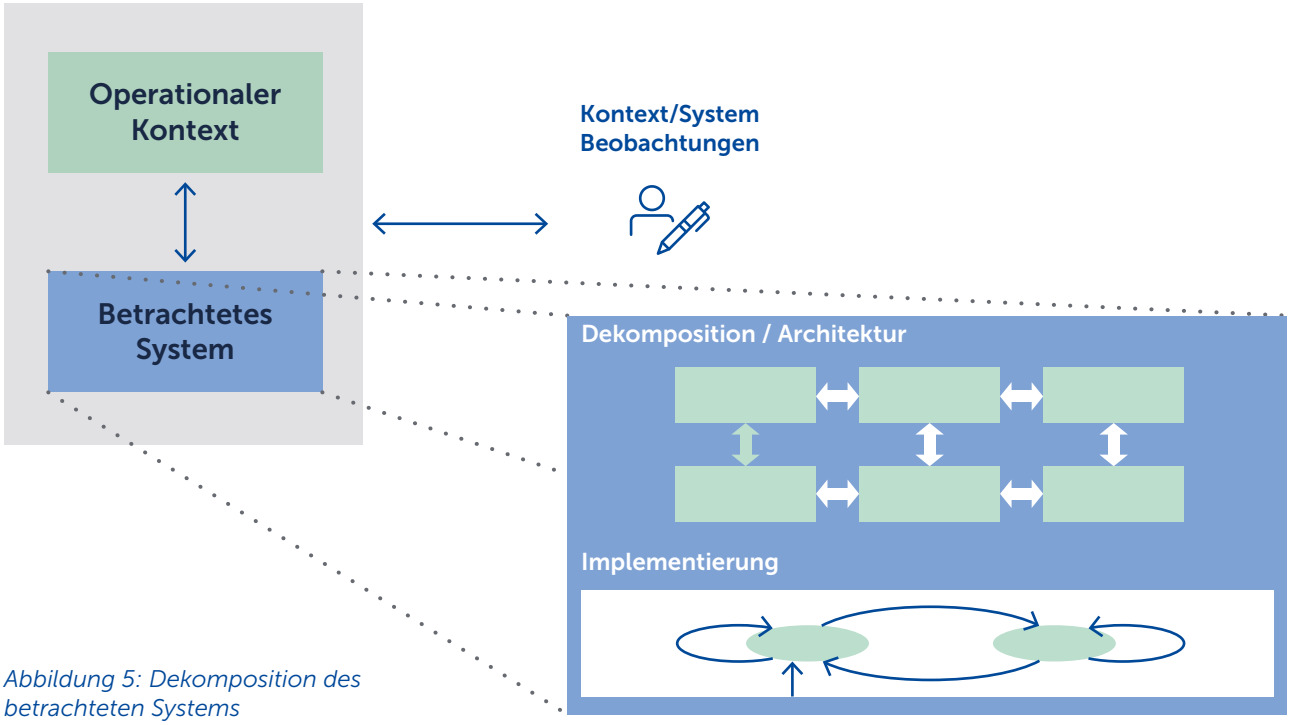


Abbildung 5: Dekomposition des betrachteten Systems

unterschiedlichen Views auf dasselbe System (d.h. innerhalb einer Granularitätsebene) gehören und keine widersprüchlichen Eigenschaften des betrachteten Systems repräsentieren. Zwei Modelle sind vertikal konsistent, wenn sie auf unterschiedlichen Granularitätsebenen zu einer View gehören und dabei keine widersprüchlichen Eigenschaften des betrachteten Systems in Bezug auf die spezifische View repräsentieren.

Views und Granularitätsebenen werden anschaulich in der sogenannten SPES-Matrix zusammengefasst (siehe Abbildung 4).

Requirements Viewpoint
Die Modelle des Requirements Viewpoints betrachten den sogenannten Problemraum und strukturieren das Entwicklungsproblem systematisch. Fragen der Architektur oder der Implementierung spielen hier noch keine bzw. eine stark untergeordnete Rolle. Zu den typischen Adressaten dieses Viewpoints gehören etwa Anwender, Kunden, Anforderungsmanager und Tester. Die Analyse- und Spezifikationstätigkeit im Requirements Viewpoint kann in vier Teilbereiche zerlegt werden: Kontext-, Ziel-, Szenario- und Anforderungsmodellierung im engeren Sinne. Besonders hervorzuheben ist die Modellierung der Grenzen des betrachteten Systems (Scope) und dessen relevanter Umgebung (Systemkontext), die in den Kontextmodellen zusammengefasst sind. Da wir das Verhalten des Systems immer an der Systemgrenze beobachten, spielen Kontextmodelle bei einer Vielzahl von Analysen eine wichtige Rolle, so z.B. bei der Simulation. Beachte: Da sich wie oben beschrieben der System Scope auf unterschiedlichen Granularitätsebenen ändert, ändert sich auch der zugehörige Systemkontext.

Die entstandenen Modelle werden vielfach Teil des zwischen Auftragnehmer und -geber geschlossenen Vertrags (Lasten- bzw. Pflichtenheft) und damit zum entscheidenden Kriterium dafür, ob der entsprechende Vertrag, auf dessen Grundlage das jeweilige System entwickelt wurde, als erfüllt gelten kann oder nicht. Formalisierte Modelle unterstützen wiederum bei der automatisierten Überprüfung.

Functional Viewpoint
Die Modelle des Functional Viewpoint betrachten das System aus einer reinen Funktionssicht heraus. Die in dieser Sicht zu entwickelnden Modelle bauen unmittelbar auf den Modellen des Requirements Viewpoint auf. Die Szenariomodelle sowie die Modelle der lösungsorientierten Anforderungen des Requirements Viewpoints werden im Functional Viewpoint in ein Blackbox-Modell (funktionale Architektur) überführt, welches die Anforderungen als Benutzerfunktionen formalisiert und sie in eine umfassende funktionale Systemspezifikation integriert. Dadurch wird eine Reihe von Analysen möglich, mit denen beispielsweise Inkonsistenzen in Anforderungen aufgedeckt werden können. Die Spezifikation von Systemfunktionen erlaubt es zudem, bereits früh im Entwicklungsprozess Schnittstellen und funktionales Verhalten eines Systems zu beschreiben und damit zusammen mit den Kontextmodellen einen funktionalen Prototyp zu entwickeln.

Ein weiterer zentraler Aspekt der Modellierung im Functional Viewpoint ist das Aufdecken und Modellieren von Abhängigkeiten zwischen einzelnen Systemfunktionen. Unerwünschte Abhängigkeiten können so möglichst früh vermieden, Widersprüche aufgedeckt und fehlende Anforderungen ergänzt werden.

Whitebox-Modelle dienen schließlich dazu festzulegen, aus welchen Teilfunktionen sich die Funktionen des Gesamtsystems zusammensetzen. Damit ist das Whitebox-Modell zugleich eine – sehr abstrakte – Beschreibung einer möglichen Realisierung der Benutzerfunktionen.

Logical Viewpoint
Während das funktionale Blackbox-Modell den Anknüpfungspunkt an den Requirements Viewpoint darstellt, knüpft das funktionale Whitebox-Modell an den Logical Viewpoint an, wo die Architektur des Systems und seiner Subsysteme, zunächst noch plattformunabhängig, spezifiziert wird. Die Funktionen des Whitebox-Modells werden auf Komponenten der logischen Architektur abgebildet. Da diese Abbildung im Allgemeinen n:m ist, ist eine geeignete Strukturierung der funktionalen Whitebox-Modelle mit Blick auf eine mögliche Architektur sinnvoll. Die logischen Komponenten des Systems können weiter in Teilkomponenten zerlegt werden, bis eine Implementierung durch eine Zustandsmaschine oder Code-Stücke möglich ist. Alternativ können Komponenten als eigenständige Subsysteme auf einer weiteren Granularitätsebene betrachtet werden. Jedes derart betrachtete Subsystem ist – auf der jeweiligen Granularitätsebene – im oben skizzierten Sinne ein definiertes System mit entsprechender Schnittstelle und eigenem Kontext sowie entsprechendem Verhalten. Über diesen Mechanismus kann beispielsweise auch eine Zuliefererschnittstelle realisiert werden.

Technical Viewpoint
Im technischen Viewpoint befinden sich die Modelle der technischen, plattformabhängigen Realisierung. Insbesondere wird dort das System in Komponenten zerlegt, die dann durch die verschiedenen Engineering-Disziplinen, also beispielsweise mechanische oder elektrische Bauteile, oder Softwaresubsysteme (d.h. Komponenten, die rein in Software implementiert werden) auf der nächsten Granularitätsebene weiter realisiert werden. Für Subsysteme, die rein in Software realisiert werden, stellt der technische Viewpoint weitere für die Softwareentwicklung wichtige Modelle und Methoden zur Verfügung. Beispiele hierfür sind Methoden zum automatischen Erzeugen von Taskschedules oder zum optimalen Deployment auf eine gegebene Hardware / Middleware-Architektur.

Integration in den Entwicklungsprozess

Die Reihenfolge der Viewpoints, wie sie in der SPES-Matrix gezeigt werden, ist bewusst so gewählt, dass sich die Modelle „früher" Phasen der Systementwicklung (z.B. Anforderungserhebung) links befinden, während rechts Modelle „späterer" Phasen (z.B. technische Umsetzung) zu finden sind. Damit spiegelt sich eine durchaus typische Reihenfolge bei der Erstellung der Modelle in der Anordnung der Viewpoints in der Matrix wider. Gleichwohl ist zu betonen, dass keinerlei Notwendigkeit besteht, hinsichtlich der Erstellung der einzelnen Modelle „von links nach rechts" und „von oben nach unten" vorzugehen. Welche Views – d.h. welche Modelle einzelner Viewpoints und Granularitätsebenen – überhaupt erstellt werden und in welcher Reihenfolge, hängt allein vom jeweiligen Projektkontext ab.

Da die formale Basis der SPES-Methodik auch Unterspezifikation unterstützt, ist es möglich, die Modelle iterativ Schritt für Schritt zu erweitern und sukzessive zu verfeinern. Das bedeutet insbesondere, dass der modellbasierte Ansatz den Grundprinzipien der agilen Entwicklung nicht widerspricht und auch nach dem Vorgehensmodell SCRUM gearbeitet werden kann. Modelle und modellbasierte Techniken unterstützen hierbei den SCRUM-Prozess. Besonders hervorzuheben ist noch der Aspekt der Kommunikation im Sprint-Team als ein zentrales Element des agilen Manifests (siehe [5]), der durch Modelle maßgeblich unterstützt wird.

Auch Techniken wie beispielsweise „Continuous Integration" können gezielt eingesetzt werden. Zu betonen ist dabei, dass diese Form des agilen Vorgehens dann eben nicht codezentriert, sondern modellzentriert ist. Es wird an Modellen gearbeitet, die fortlaufend in das Gesamtmodell integriert werden. Automatische Analysen auf Basis der Modelle verifizieren die Korrektheit des Systemmodells. Code wird dann schließlich aus den Modellen entwickelt oder kann gar automatisch generiert werden.

Grundsätzlich gilt jedoch: Das SPES Modeling Framework macht keinerlei Vorgaben über die Reihenfolge, in der die verschiedenen Modelle der SPES-Matrix erstellt werden sollten. Somit lässt sich mit der SPES Methode auch iteratives oder inkrementelles Entwicklungsvorgehen implementieren. Wie oben bereits erwähnt, erlaubt der Mechanismus der Granularitätsebenen die Integration verschiedenartigen Entwicklungsvorgehens und unterschiedlicher Entwicklungswerkzeuge, wie sie beispielsweise bei mechatronischen Systemen notwendig sind, und bildet die Schnittstelle für Zulieferer und für die Wiederverwendung von Komponenten.

Strukturierte Einführung mit dem MBSE-Reifegradmodell

Die im vorherigen Kapitel genannten Herausforderungen zeigen, dass MBSE systematisch und mit Bedacht eingeführt werden sollte. Zusammen mit Industriepartnern haben wir in den letzten Jahren einen Einführungsprozess entwickelt, der eine zielgerichtete und schrittweise Einführung von MBSE in einem Unternehmen oder einer Abteilung ermöglicht. Dieser Einführungsprozess erlaubt eine kontinuierliche Erfolgskontrolle und Steuerung der Einführung von einzelnen MBSE-Methoden, die das Unternehmen schrittweise näher an die gesteckten Ziele heranbringen.

Abbildung 6 zeigt eine Übersicht über den Einführungsprozess. Dieser ist ein iterativ inkrementeller Prozess, der MBSE in einem Unternehmen schrittweise einführt.

Planung: Der Einführungsprozess beginnt mit einer zwei- bis dreitägigen Planungsphase. In dieser Planungsphase werden zunächst die Ziele definiert, die das Unternehmen oder die Abteilung mithilfe der MBSE-Einführung erreichen will. Diese Definition ist besonders wichtig, um entscheiden zu können, für welche MBSE-Methoden sich die Einführung mit Blick auf die Ziele am meisten lohnt und auf welche Methoden zunächst verzichtet werden kann. Als Zweites wird der aktuelle Stand des Unternehmens oder der Abteilung in Bezug auf MBSE mithilfe eines Assessments ermittelt. Eine Gap-Analyse ermittelt die Differenz zwischen den Zielen und dem aktuellen Stand. Ein Teil des in der Gap-Analyse ermittelten Einführungsbedarfs wird dann als Einführungsmaßnahme für die kommende Einführungsiteration ausgewählt. Eine Einführungsmaßnahme kann zum Beispiel die Einführung einer MBSE-Technik für einen bestimmten Zweck sein.

Umsetzung: In der anschließenden drei- bis sechsmonatigen Umsetzungsphase werden die festgelegten Einführungsmaßnahmen umgesetzt. Dazu muss die konkrete MBSE-Methode definiert, Mitarbeiterinnen und Mitarbeiter geschult, Werkzeuge und auch Prozesse angepasst und die konkrete Technik natürlich erprobt werden. Es ist wichtig, vorab festzulegen, wann eine MBSE-Methode als „eingeführt" gilt (ähnlich einer „Definition of Done"). Es kann zum Beispiel festgelegt werden, dass eine MBSE-Methode erst dann als eingeführt gilt, wenn mindestens zehn Mitarbeiter in der Lage sind, die Methode anzuwenden, ein entsprechendes Methodenhandbuch existiert, die Werkzeuge die Methode hinreichend unterstützen und es ein Pilotprojekt gibt, in dem die Methode erfolgreich angewendet wurde. Unser Einführungsprozess sieht vor, dass die Umsetzungsphase durch regelmäßige Coaching Sessions (etwa alle zwei Wochen) von MBSE-Experten begleitet wird. In diesen Coaching Sessions präsentiert das Umsetzungsteam aktuelle Ergebnisse der Umsetzung. Die MBSE-Experten begutachten die erstellten Modelle, geben Hinweise zur Verbesserung und beantworten methodische Fragen. In den ersten Iterationen der Einführung ist es sinnvoll, externe MBSE-Experten für das Coaching einzusetzen. Wenn ausreichend MBSE-Know-how in der Organisation verfügbar ist, kann das Coaching auch von eigenen Mitarbeitern durchgeführt werden.

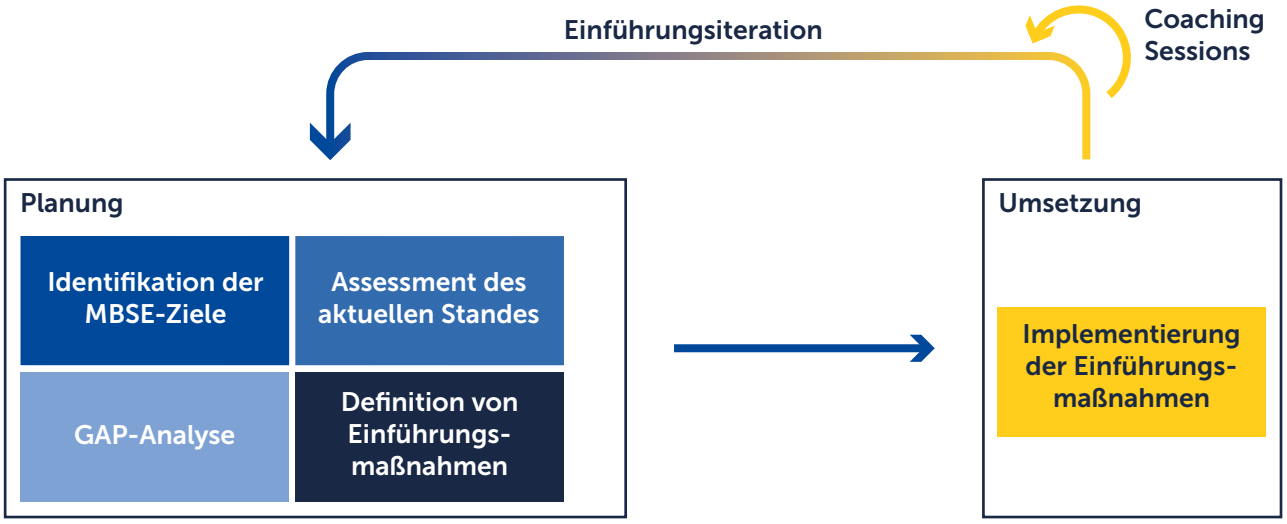


Abbildung 6: MBSE-Einführungsprozess

Engineering Functions	Focus Areas	Capabilities (A–G)										
Context Analysis	Operational Context			A	B	C	D	E		F	G	
	Knowledge Context		A	B	C		D					
Requirements	Scoping	A	B	C	D	E						
	Goal Modeling	A				B	C	D	E	F		
	Scenario Modeling		A	B	C	D	E	F		G		
	Requirements Modeling			A			B	C	D	E	F	G
System Functions	Sys. Function Modeling	A			B		C					D
	Sys. Function Specification			A	B		C		D	E	F	
	Event Chain Modeling				A		B	C			D	
	Mode Modeling				A		B		C		D	
System Architecture	Architecture Modeling		A	B	C	D		E	F	G		
	Component Modeling			A		B	C	D		E		
Integration & Testing	System Behavior Testing				A	B	C			D		
	System Quality Testing				A				B	C	D	
Technical Implementation	Technical Arch. Modeling		A	B	C		D	E	F			
	Technical Comp. Modeling			A			B		C		D	
SW Development	SW Execution Platform		A	B		C	D	E	F			
	SW Architecture				A		B	C	D	E		
	Deployment					A	B			C	D	
	SW Comp. Implementation								A		B	

Maturity

Abbildung 7: Das MBSE-Reifegradmodell

Einführungsiteration: Nach Abschluss der Umsetzungsphase findet erneut eine Planungsphase statt, in der die Ziele gegebenenfalls angepasst werden, wiederum der nun aktuelle Stand des Unternehmens in Bezug auf MBSE ermittelt wird und neue Einführungsmaßnahmen für die folgende Iteration definiert werden.

Das MBSE-Reifegradmodell

Um den oben skizzierten Einführungsprozess optimal zu unterstützen, haben wir ein MBSE-Reifegradmodell entwickelt, das genutzt werden kann, um insbesondere die Planungsphase zu unterstützen. Abbildung 7 zeigt eine Übersicht über das Reifegradmodell. Den Kern des Reifegradmodells bildet eine Menge von Capabilities (in der Abbildung durch die Buchstaben A–G bezeichnet), die durch den Einsatz von Modellen erlangt werden können. Die Capabilities sind Engineering Functions und Focus Areas zugeordnet, die in den folgenden Unterkapiteln detailliert vorgestellt werden.

Engineering Functions und Focus Areas

Das MBSE-Reifegradmodell strukturiert den Entwicklungsprozess in verschiedene *Engineering Functions*, die jeweils einen Aspekt des Engineerings betrachten. Angelehnt an die Viewpoints im SPES Modeling Framework, auf das wir im Folgenden noch näher eingehen werden, unterscheiden wir im MBSE-Reifegradmodell die folgenden Engineering Functions:

- Context Analysis: sämtliche Aktivitäten, die auf die Identifizierung, Spezifikation und Analyse des Kontexts abzielen, in den der Entwicklungsgegenstand eingebettet ist.
- Requirements: sämtliche Aktivitäten, die auf die Erhebung, Spezifikation und Analyse von Anforderungen abzielen.
- System Functions: sämtliche Aktivitäten, die auf die Identifikation, Spezifikation und Analyse von Funktionen abzielen, die an der Systemgrenze zur Verfügung gestellt werden.

- System Architecture: sämtliche Aktivitäten, die auf die Spezifikation und das Design einer logischen Lösung durch das Zusammenspiel von logischen Komponenten abzielen. Bei der logischen Lösung abstrahieren wir noch von technischen Implementierungsdetails.

- Integration & Testing: sämtliche Aktivitäten, die auf Integration und Testen des Systems in Hinblick auf die Übereinstimmung mit seinen Anforderungen abzielen.

- Technical Implementation: sämtliche Aktivitäten, die auf die Spezifikation und das Design einer technischen Umsetzung abzielen.

- Software Development: sämtliche Aktivitäten, die auf die Spezifikation und die Implementierung von reinen Software-Komponenten abzielen.

Jede Engineering Function ist noch mal unterteilt in verschiedene *Focus Areas*, die innerhalb der Engineering Functions den Fokus auf einen bestimmten Aspekt dieser Engineering Function legen.

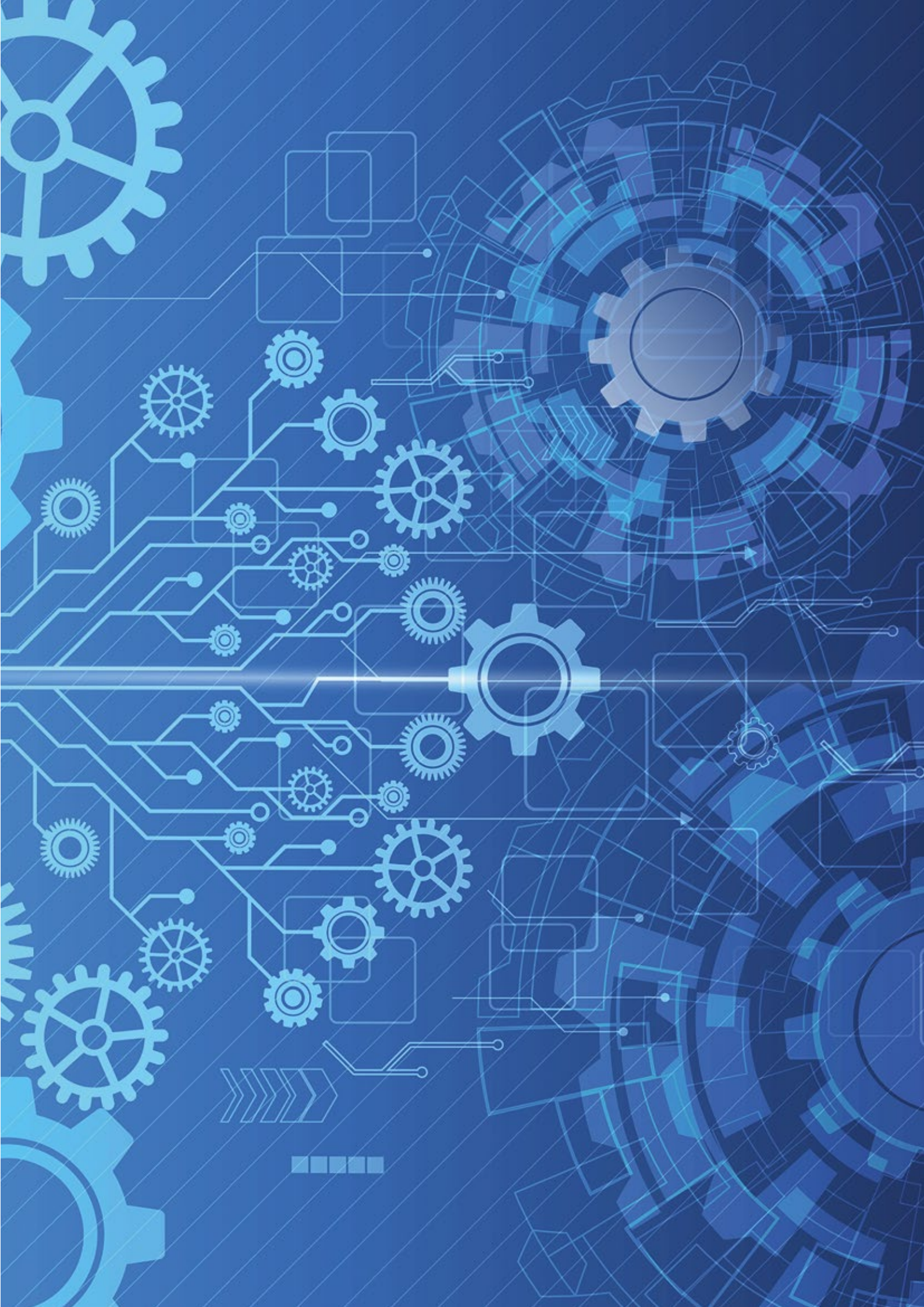
Capabilities

Capabilities sind das Kernstück des Reifegradmodells. Eine Capability beschreibt die Fähigkeit, die notwendig ist, um ein bestimmtes Ziel zu erreichen. Unser MBSE-Reifegradmodell beschreibt Fähigkeiten, die durch den Einsatz von Modellen erlangt werden können. Eine Capability wird über ein Template beschrieben, das alle relevanten Informationen zu der Fähigkeit enthält (siehe Tabelle 1).

Die Capabilities innerhalb einer Focus Area bauen in der Regel aufeinander auf, sodass die Capabilities innerhalb einer Focus Area mithilfe von Buchstabenkürzeln aufsteigend sortiert werden (zum Beispiel: Goal Modeling: A – Goal Modeling: D). Es ist wichtig zu erwähnen, dass die Anzahl der Capabilities von Focus Area zu Focus Area variieren kann. Eine besondere Bedeutung für das Reifegradmodell haben die Vorbedingungen einer Capability. Vorbedingungen referenzieren andere Capabilities, die als Voraussetzung für die aktuelle Capability bereits vorhanden sein müssen. Vorbedingungen können Capabilities innerhalb einer Focus Area oder auch Capabilities aus anderen Focus Areas referenzieren. Die Capability in Tabelle 1 hat als Vorbedingung, dass Ziele definiert wurden (Capability *Goal Modeling: A*) und dass Systemfunktionen definiert wurden, zu denen die Ziele in Beziehung gesetzt werden können (Capability *System Function Modeling: A*). In der Matrix des Reifegradmodells sind diese Abhängigkeiten dadurch berücksichtigt, dass eine Capability in der Matrix immer rechts von deren Vorbedingungen angeordnet ist. Dadurch entstehen die Lücken in der Matrix (siehe Abbildung 7).

Name	Die Beziehungen zwischen Zielen und Systemfunktionen sind identifiziert und dokumentiert.
Ziel	Systemfunktionen definieren Charakteristiken des zu entwickelnden Systems. Daher ist es besonders wichtig, dass die Umsetzung einer Systemfunktion auch wirklich einen Mehrwert bringt, sprich zur Erfüllung eines Ziels beiträgt. Systemfunktionen, die zu keinem Ziel beitragen, sollten gelöscht oder angepasst werden. Ziele, zu denen es keine Systemfunktionen gibt, stellen in der Abnahme des Systems große Risiken dar.
Aktion	Ziele, die das System erfüllen soll, werden systematisch festgehalten und mit Systemfunktionen in Beziehung gesetzt, wenn diese zur Erfüllung der Ziele beitragen. Diese Verlinkung kann entweder manuell durch einen Engineer erfolgen oder auch durch (teil-)automatisierte maschinelle und linguistische Analysen unterstützt werden.
Vorbedingungen	• Goal Modeling: A • System Function Modeling: A

Tabelle 1: Beispiel für die Beschreibung einer Capability



MBSE-Einführung mit dem MBSE-Reifegradmodell

Das MBSE-Reifegradmodell kann nun benutzt werden, um den aktuellen Stand der Fähigkeiten eines Unternehmens, einer Abteilung oder auch nur einzelner Projekte bezüglich MBSE zu ermitteln. Diese Ermittlung erfolgt im Rahmen eines leichtgewichtigen Assessments, das in der Regel nur zwei bis drei Stunden in Anspruch nimmt. Die Ergebnisse eines solchen Assessments sind:

- Eine systematische Beschreibung der Ziele einer Organisation in Bezug auf MBSE.
- Eine Abbildung der aktuellen Fähigkeiten der Organisation auf Capabilities des MBSE-Reifegradmodells.
- Die Identifikation von Lücken zwischen den Zielen der Organisation und den aktuellen Fähigkeiten.
- Die Identifikation von Lücken in der aktuellen Anwendung der MBSE-Methodik.
- Die Identifikation von konkreten nächsten Schritten zur Verbesserung der bestehenden MBSE-Methodik.

Abbildung 8 zeigt schematisch, wie sich Ziele und Fähigkeiten im Reifegradmodell abbilden. Blau markiert sind die vorhandenen Fähigkeiten, die weißen Felder zeigen die erhobenen Unternehmensziele an. Gelb markiert sind Fähigkeiten, die zur Erreichung der Ziele notwendig sind. Grün sind mögliche Vorschläge für erste Verbesserungsschritte.

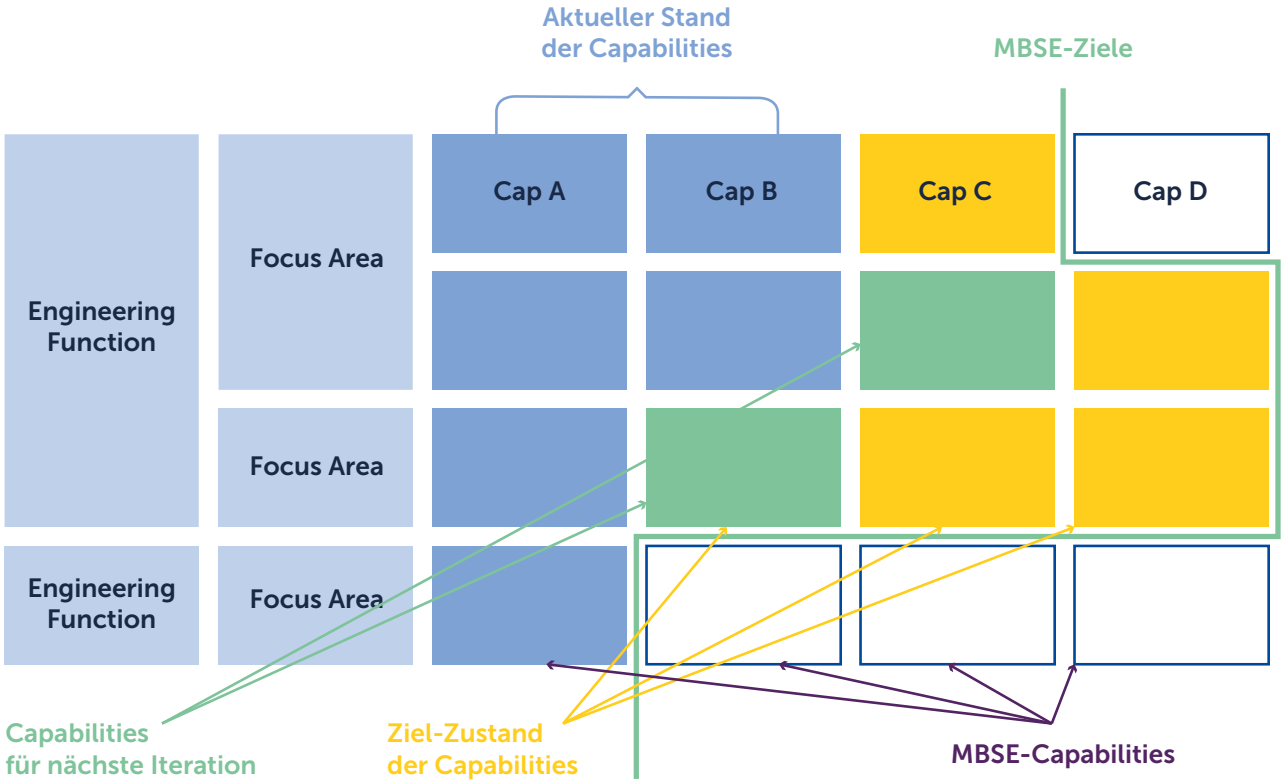


Abbildung 8: Ziele, aktuelle Fähigkeiten und zu entwickelnde Fähigkeiten im MBSE-Reifegradmodell

Zusammenfassung

Der dargestellte Ansatz wurde in einer Reihe von drei Verbundprojekten, gefördert durch das BMBF, entwickelt. An diesen Verbundprojekten war jeweils eine beträchtliche Anzahl von wissenschaftlichen Partnern und auch Industrieunternehmen beteiligt. Um die Zusammenarbeit intensiv zu gestalten, wurde in den Projekten deshalb weitgehend davon abgesehen, spezifische Werkzeuge zu benutzen oder spezifische Modellierungssprachen, sondern es wurde darauf Wert gelegt, den Ansatz und die verschiedenen Elemente des Ansatzes weiterzuentwickeln. Was entstanden ist, lässt sich dann auf unterschiedliche Werkzeuge oder Prozesse modellbasierter Entwicklung zuschneiden.

Weiterführende Arbeiten

[1] Broy, Stolen: „Specification and development of interactive systems: focus on streams, interfaces, and refinement", 2001

[2] Broy: „A logical basis for component-oriented software and systems engineering", 2010

[3] Pohl, Hönninger, Achatz, Broy: „Model-Based Engineering of Embedded Systems", 2012

[4] Pohl, Broy, Daembkes, Hönninger: „Advanced Model-Based Engineering of Embedded Systems Extensions of the SPES 2020 Methodology", 2016

[5] Schwaber: „Agiles Projektmanagement mit Scrum". Microsoft Press, 2007

[6] „*IEEE Std 610.12-1990,* Standard Glossary of Software Engineering Terminology", 1990, IEEE

[7] „ISO 42010, Systems and software engineering – Architecture description", 2011, ISO

[8] Sascha Kirstan: „Kosten und Nutzen modellbasierter Entwicklung eingebetteter Softwaresysteme im Automobil" Dissertation 2011, Technische Universität München

Impressum

Herausgeberin
fortiss gemeinnützige GmbH
Guerickestraße 25, 80805 München
E-Mail: info@fortiss.org
www.fortiss.org

Autoren
Wolfgang Böhm
Manfred Broy
Maximilian Junker
Andreas Vogelsang
Sebastian Voss

Lektorat
Lektorat Süd, München

Gestaltung
Sonja Taut

Druck
Cewe Print GmbH
Enderstraße 92c, 01277 Dresden

ISSN
2700-2977

Bildnachweise
Titel: shutterstock ©Nickolay Grigoriev
Seite 7: fortissGmbH
Seite 9: shutterstock ©metamorworks
Seite 10: shutterstock ©SFIO CRACHO
Seite 19: shutterstock ©chanut iamnoy
Seite 22: fortissGmbH ©Kathrin Kahle

1. Auflage, Juni 2020

fortiss ist das Forschungsinstitut des Freistaats Bayern für softwareintensive Systeme und Services mit Sitz in München. Die WissenschaftlerInnen am Institut arbeiten in Forschungs-, Entwicklungs- und Transferprojekten mit Universitäten und TechnologieFirmen in Bayern, Deutschland und Europa zusammen. Schwerpunkte sind die Erforschung modernster Methoden, Techniken und Werkzeuge der Softwareentwicklung, des Systems- & Service-Engineering und deren Anwendung auf kognitive cyberphysische Systeme wie das Internet of Things (IoT).

fortiss ist in der Rechtsform einer gemeinnützigen GmbH organisiert. Gesellschafter sind der Freistaat Bayern (Mehrheitsgesellschafter) und die Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

fortiss GmbH
Guerickestraße 25
80805 München
Deutschland
www.fortiss.org
Tel: +49 89 3603522 0
E-Mail: info@fortiss.org

fortiss