Die SOPHISTen

»Anforderungen und Architekturen für komplexe Systeme«

SOPHIST

Die SOPHISTen

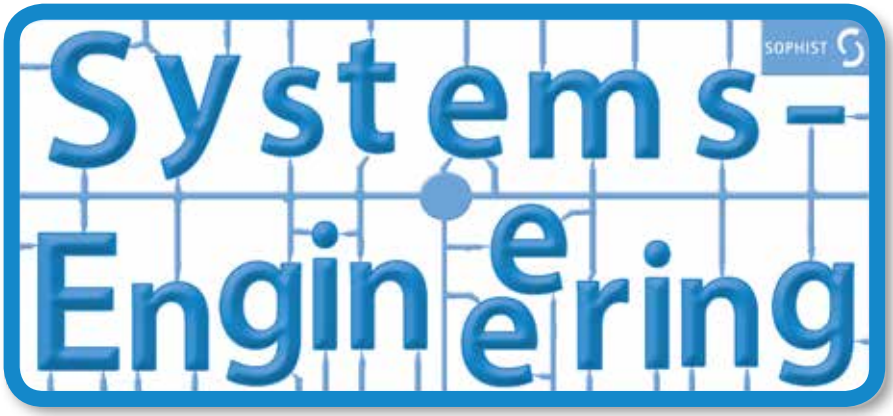
SOPHIST GmbH
Vordere Cramergasse 13
90478 Nürnberg
Deutschland
www.sophist.de

1. Auflage 2017

Copy Editing & Herstellung: Roland Kluge, SOPHIST GmbH
Druck: Flyeralarm

Copyright 2017 SOPHIST GmbH

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung der SOPHIST GmbH urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.
Es wird darauf hingewiesen, dass die in der Broschüre verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.
Alle Angaben und Programme in dieser Broschüre wurden mit größter Sorgfalt kontrolliert. Weder Autoren noch die Firma SOPHIST GmbH, etc. können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieser Broschüre stehen.



Die SOPHISTen

»Anforderungen und Architekturen für komplexe Systeme«

www.sophist.de

Inhaltsverzeichnis

1. Systems-Engineering - Einleitung ..... Seite 6
1.1 Motivation ..... Seite 7
1.2 Der Begriff System ..... Seite 9

2. Überblick über die Tätigkeiten ..... Seite 11

3. Systemanalyse ..... Seite 14
3.1 Überblick ..... Seite 14
3.2 Anforderungen ermitteln ..... Seite 16
3.3 Anforderungen dokumentieren ..... Seite 20
3.4 Vorgehen in der Analyse ..... Seite 30

4. Systemarchitektur ..... Seite 32
4.1 Black-Box-Sicht ..... Seite 34
4.2 Funktionale Sicht ..... Seite 36
4.3 Physikalische Sicht ..... Seite 38
4.4 Vorgehen in der Architektur ..... Seite 43

5. Weitere Aspekte des Systems-Engineerings ..... Seite 45
5.1 Weitere Tätigkeiten in der Entwicklung ..... Seite 45
5.2 Einordnung in den Lebenszyklus des Systems ..... Seite 47
5.3 Verwendung einer Produktbeschreibung ..... Seite 49
5.4 Die weichen Herausforderungen ..... Seite 52

6. Quellenverzeichnis ..... Seite 54

1. Einleitung

Seit über 20 Jahren beschäftigt sich SOPHIST nun mit Anforderungen. Wir haben in dieser Zeit zahlreiche Firmen bei der Einführung und Durchführung von gutem Requirements-Engineering unterstützt.

Dabei wurden wir mit den unterschiedlichsten Anwendungsgebieten konfrontiert. Diese reichten von gesamten Flughafen-Towern über reine Software-Anwendungen bis hin zu einzelnen ASICs (Application Specific Circuit).

Der Bedarf an der Unterstützung hat sich in dieser Zeit jedoch gewandelt. Neben der Ermittlung und Dokumentation von Anforderungen, die sich an den Betrieb des betrachteten Produkts richten, rücken die Anforderungen aus den anderen Lebenszyklen des Produkts immer mehr in den Fokus. Weiterhin wird die Notwendigkeit immer größer, die Analyse der Anforderungen als integralen Bestandteil der weiteren Entwicklung zu sehen.

Diese Integration wird im Bereich der reinen Softwareentwicklung mit den agilen Ansätzen gut unterstützt. Diese Ansätze funktionieren in einem komplexen System, das neben Software auch aus elektrischen, elektronischen oder mechanischen Anteilen besteht, allerdings nur bedingt.

An dieser Stelle setzt Systems-Engineering an, das in einem interdisziplinären Ansatz sowohl das gewünschte Produkt als Gesamtsystem als auch alle Bereiche in seinem Lebenszyklus betrachtet.

Seit einigen Jahren trainieren und vermitteln wir diese Ideen in unseren Projekten, woraus die Idee zu dieser Broschüre entstand. In ihr wollen wir Ihnen unsere Ansätze und Methoden darstellen, wobei uns natürlich bewusst ist, dass diese immer wieder an die Gegebenheiten in einzelnen Projekten und Organisationen angepasst werden müssen.

Die hier vorgestellten Ansätze und Methoden spiegeln das Wissen von vielen SOPHISTen wider. Somit muss an dieser Stelle allen SOPHISTen gedankt werden, die ihr Wissen aus zahlreichen Projekten in einen Topf geworfen haben und den Inhalt im Rahmen von heißen Diskussionen kräftig umgerührt haben. Nur dadurch konnten wir das Beste daraus abschöpfen und Ihnen hier vorstellen.

Besonderer Dank gilt jedoch:
- Dr. Stefan Queins für die Erstellung der Inhalte
- Chris Rupp für das inhaltliche Review
- Karin Schlör für das sprachliche Review
- Roland Kluge für das Design und Layout

1.1 Motivation

In den letzten Jahren gewinnt der Begriff des Systems-Engineerings immer mehr an Bedeutung. Begründet wird dies durch eine steigende Erwartungshaltung von Nutzern gegenüber technischen Systemen. Wurde z.B. eine Waschmaschine früher mit einem Drehknopf und mehreren Tasten bedient, so besitzen heute moderne Waschmaschinen einen Touchscreen und WiFi-Verbindung. Aber auch ein steigender Qualitätsanspruch führt zu immer komplexeren Systemen, da immer kompliziertere Algorithmen, ausgefeiltere Elektronik und immer höher anmutende Mechanik zum Einsatz kommen muss. Zusammenfassend führt diese steigende Erwartungshaltung zu den Smart-Eco-Systems, die uns Menschen in der bestmöglichen Art, die zurzeit vorstellbar ist, unterstützen sollen.

Die steigende Komplexität der Systeme spiegelt sich natürlich in der wachsenden Komplexität einzelner Komponenten wider, aus denen sich ein System zusammensetzt. Damit eng verknüpft ist der Komplexitätszuwachs bei den Abhängigkeiten und Schnittstellen zwischen diesen Komponenten oder sogar zwischen ganzen Systemen (Stichwort Digitalisierte Fabrik). Beides führte in den letzten Jahren zu der Erkenntnis, dass die Entwicklung solch komplexer Systeme mehr als die Entwicklung der einzelnen Komponenten umfassen muss.

Heutzutage wird unter einem komplexen technischen System ein Zusammenbau aus mehreren Komponenten verstanden, die aus verschiedenen Gewerken stammen. So kann sich ein solches System aus elektrischer und elektronischer Hardware, aus mechanischen Komponenten, aus Software und, je nach Anwendungsgebiet, weiteren Gewerken (Optik, Hydraulik, etc.) zusammensetzen. Eine Herausforderung des Systems-Engineerings ist somit, die frühere multidisziplinäre Entwicklung hin zu einer interdisziplinären Entwicklung zu vereinen.

Diese Erkenntnis führt zu einer weiteren Herausforderung des Systems-Engineerings: Der gesamte Lebenszyklus eines Systems (und nicht nur der operative Einsatz) muss bei der Entwicklung eines Systems mit betrachtet werden. Bewegen wir uns z.B. auf der Seite eines Zulieferers, so sollte die Entwicklungsabteilung schon in einer Angebotsphase koordiniert integriert werden. Des Weiteren müssen Anforderungen an das System aus Sicht der Fertigung, Verpackung, Transport, Pflege etc. mit berücksichtigt werden.

Diese neuen (und zum Teil auch alten) Herausforderungen führen zwangsläufig dazu, über eine Entwicklungsdisziplin nachzudenken, deren Tätigkeiten oberhalb der Prozesse der einzelnen Gewerke liegt, diese miteinander koordiniert und so die korrekte Entwicklung der einzelnen Bestandteile sicherstellt, damit deren Zusammenbau als gesamtes System die Wünsche der Anwender erfüllt. Zudem besteht die Hoffnung, die Effizienz der Entwicklung durch das Systems-Engineering zu steigern. Hier soll gerade die Steigerung des Aufwands in den frühen Phasen der Entwicklung helfen. Damit sollen die folgenden Punkte für die

späteren Phasen der Entwicklung ausgeschlossen bzw. abgeschwächt werden:

- *Überraschungen durch falsche Risikoeinschätzung .*
- *Änderungen aufgrund von Fehlern bzw. falschen Entscheidungen in den vorderen Phasen.*

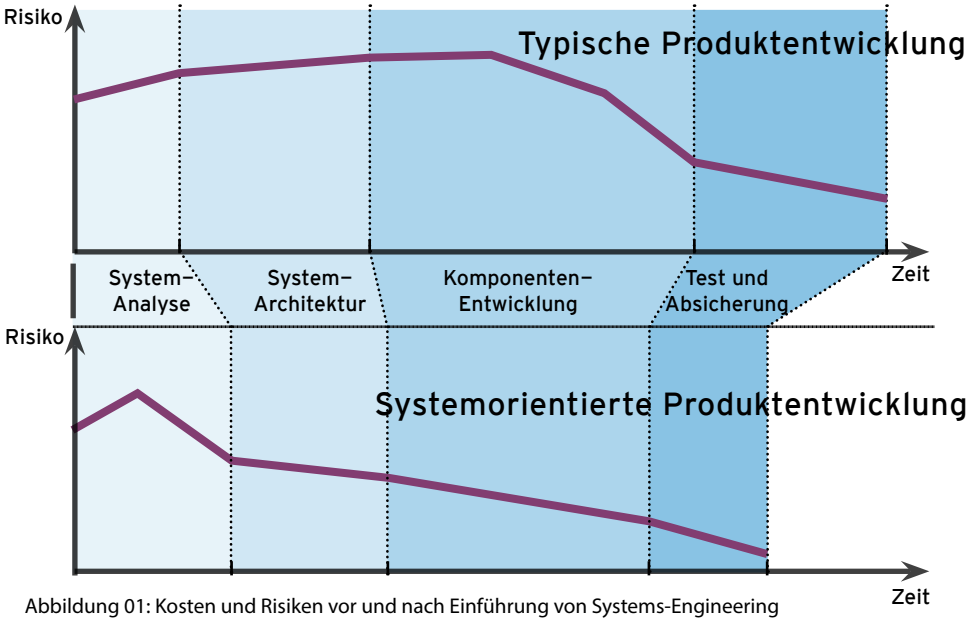


Abbildung 01: Kosten und Risiken vor und nach Einführung von Systems-Engineering

Abbildung 1 zeigt, angelehnt an [Incose15], die benötigte Entwicklungszeit und den qualitativen Verlauf der Risikoentwicklung ohne und mit Verwendung von Systems-Engineering. Man sieht, dass bei Verwendung von Systems-Engineering das Risiko früh abnimmt und dass sich die Gesamtentwicklungszeit verkürzt. Der Grund hierfür ist die Verlängerung der frühen Phasen. In wie weit sich diese Angaben in der Realität bestätigen, lässt sich besonders im Vorfeld einer Entwicklung nur sehr schwer abschätzen. Mehr zu möglichen Metriken zur Messung des Effizienzgewinns findet sich in Kapitel 5.

1.2 Der Begriff System

Der Begriff System stammt aus dem Griechischen und kann mit „aus mehreren Einzelteilen zusammengesetztes Ganzes“ übersetzt werden.
Im allgemeinen Sprachgebrauch wird der Begriff „System“ sehr breit verwendet.
Beispiele hierfür sind:

- *das Zusammenspiel mehrerer Menschen*
- *eine Firma mit ihren Mitarbeitern und Gütern*
- *das Zusammenwirken von Hochs und Tiefs (ein Wettersystem)*
- *eine Sonne mit ihren Planeten*
- *ein Dolby Surround System*
- *ein Auto*

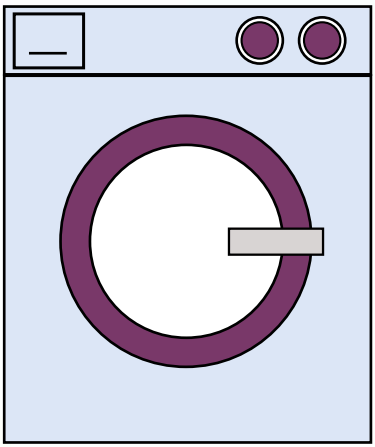
Bei allen Verwendungen des Begriffs System sind zwei Charakteristika zu finden: Zum einen besteht ein System immer aus mehreren kleineren Teilen und zum anderen würde das System ohne das Zusammenspiel dieser Teile nicht funktionieren.
Dies sind auch die beiden entscheidenden Eigenschaften von technischen Systemen (im Folgenden kurz „System“).

Systeme können in einzelne Komponenten zerlegt werden, die das Systemverhalten durch ihre eigenen Eigenschaften und durch ihr Zusammenspiel erreichen.

Auch bei technischen Systemen ist die Bandbreite beliebig. So können sehr komplexe als auch relativ einfach aufgebaute Anwendungen als System bezeichnet werden. Auffällig ist jedoch, dass den Systemen eine beliebig tiefe Hierarchie zu Grunde liegt. Das heißt, Systeme können in Bestandteile zerlegt werden, die wiederum in kleinere Teile zerlegt werden können. Diese Hierarchisierung macht auch sehr komplexe Systeme beherrschbar.

Als Beispiel sei hier das System „Waschmaschine“ genannt (siehe Abbildung 2).

Wie man in dem Beispiel sehen kann, taucht der Begriff „System“ im Namen der Strukturelemente an mehreren Stellen auf unterschiedlichen Hierarchieebenen auf. Ein Grund hierfür ist die Fokussierung auf unterschiedliche Ebenen. Der Vertrieb des Herstellers sieht als „System“ die gesamte Waschmaschine, da dies für ihn sein Betrachtungsgegenstand, sein Produkt, ist. Für ihn ist das Trommelsystem nur ein Teil des Produkts. Für die Organisationseinheit, die sich mit der Entwicklung des Trommelsystems beschäftigt, ist dieses Teilsystem jedoch der Betrachtungsgegenstand auf ihrer obersten Ebene, und wird demnach in dieser Organisationseinheit als „System“ betrachtet. Die Waschmaschine wäre dann für das Trommelsystem das sogenannte „Supersystem“.



Waschmaschine

- Trommelsystem
  - Tür
  - Türkontakt
  - Türschloss
  - Trommel
  - Trommelmotor
- Bedieneinheit
- Wasserkreislauf
  - Heizsystem
  - Einlassventil
  - Auslassventil
  - Pumpe
- Gehäuse
  - Rahmen
  - Gewichte
- Waschmittelzufuhr

Abbildung 02: Waschmaschine als System

Der Betrachtungsgegenstand, der dem „System" entspricht, muss demnach immer relativ zu der Organisationseinheit festgelegt werden, die den Betrachtungsgegenstand verantwortet. Das Sprichwort „One man's ceiling is another man's floor" sollte, gemäß einer Top-Down-Zerlegung im Systems-Engineering, also besser umformuliert werden zu „One man's floor is another man's ceiling".

Ein weiterer Standardbegriff aus dem Systems-Engineering ist aus dem zuvor geschilderten Umstand abgeleitet. Als „Subsystem" wird ein Teil eines Systems bezeichnet, das wiederum als System angesehen werden kann, und demnach aus weiteren Subsystemen oder gewerksspezifischen Komponenten (z.B. reinen Hardware- oder Software-Komponenten) aufgebaut ist.

2. Überblick über die Tätigkeiten

Die Haupttätigkeiten im Bereich des Systems-Engineerings spielen sich in der Entwicklung ab, wobei der Systems-Engineer die Einflüsse der weiteren, am Produktlebenszyklus beteiligten Disziplinen mit berücksichtigen muss. In diesen Disziplinen wird er in seiner Rolle jedoch keine Tätigkeiten verantworten, wobei er mit seinem übergreifenden Wissen über das Produkt in diesen Disziplinen unterstützend tätig werden kann. In dieser Broschüre werden wir die Tätigkeiten einer Systementwicklung anhand eines vereinfachten V-Modells vorstellen. Dabei werden wir uns auf den linken, den konstruktiven Ast des V-Modells konzentrieren.

Die Entwicklungstätigkeiten werden im Weiteren aus didaktischen Gründen in eine logische Reihenfolge gebracht. Dies soll aber nicht implizieren, dass ein solches wasserfallartiges Vorgehen durchgeführt werden muss. Ab einer gewissen Ebene wird es möglich sein, mehr iterativ-inkrementell oder sogar agil (besonders in der Softwareentwicklung) vorzugehen.
Die folgende Abbildung gibt nur einen Überblick über die wichtigsten Tätigkeiten im Systems-Engineering. In Kapitel 5.1 werden weitere Tätigkeiten eingeführt, die die folgende Abbildung vervollständigen.

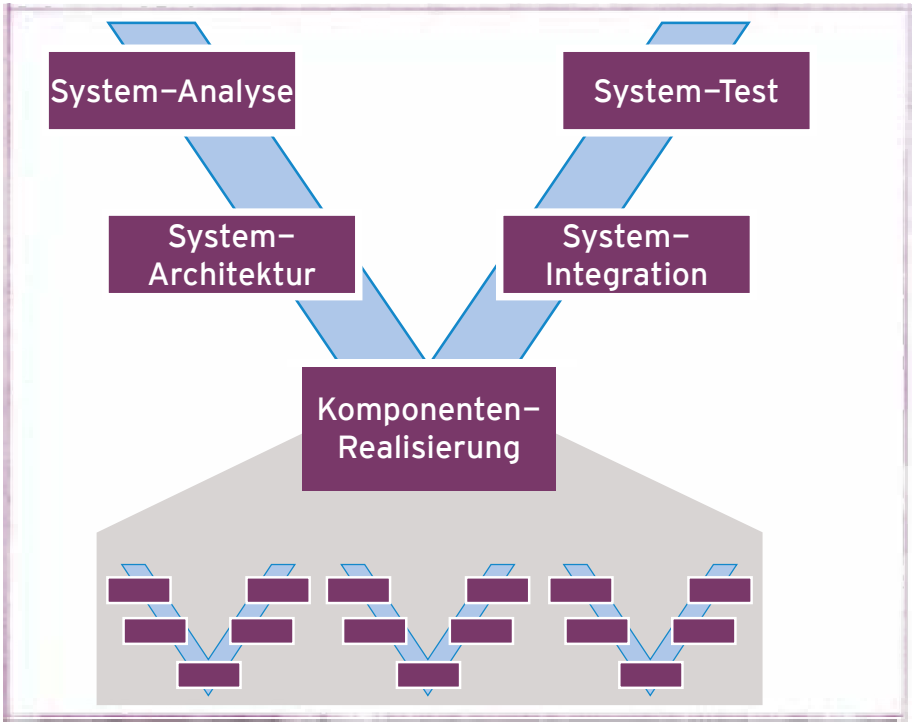


Abbildung 03: Grobe Einordnung der SE-Tätigkeiten

Laut Abbildung 3 wird auf dem linken Ast des V's zunächst eine Analyse und dann eine Architektur für das betrachtete System durchgeführt. Daraus folgen dann Komponenten, für die dann wiederum jeweils ein komplettes V durchlaufen wird. Im besten Fall erhält der Systems-Engineer für die oberste Ebene die getesteten Komponenten zurück, die er dann zu dem gewünschten System integrieren kann. Den Abschluss der Systementwicklung bildet dann der Systemtest.

Im Allgemeinen muss das Systems-Engineering mehr als zwei Ebenen betrachten, da nicht direkt aus der ersten Zerlegung die gewerksspezifischen Komponenten entstehen. Dabei würde dann zur Realisierung der Komponenten aus Abbildung 3 deren Entwicklung mit Systems-Engineering-Methoden bis auf eine Ebene durchgeführt werden, die dann nur aus gewerksspezifischen Komponenten besteht.

Aus dieser Betrachtung ergibt sich für den linken Ast des V-Modells eine Abfolge von Analyse- und Architekturschritten, bis eine Ebene von Komponenten erreicht wird, die

- entweder gewerksspezifisch ist und der entsprechenden Entwicklungsabteilung übergeben wird
- oder als komplexere Komponente einer anderen Organisationseinheit bzw. einem Zulieferer übergeben wird.

Abbildung 4 stellt diesen Zusammenhang für ein System über drei Ebenen graphisch dar.

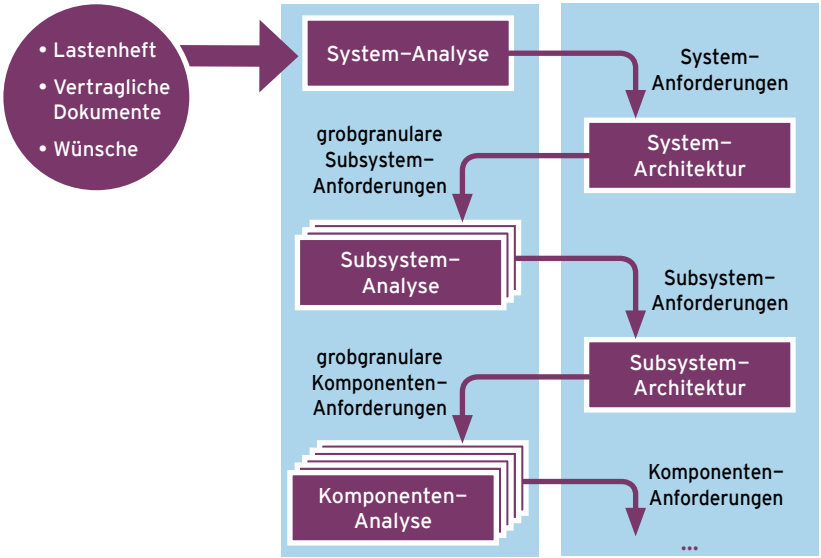


Abbildung 04: Abfolge von Analyse und Architektur auf mehreren Ebenen

Die Eingabe in den Prozess bilden Anforderungen, die von unterschiedlichen Anforderungsgebern geliefert werden können. Befinden wir uns auf Seiten eines Auftraggebers, z.B. eines **OEMs** (Original Equipment Manufacturer), so werden diese meist von einem Produktmanagement als Eingabe in den Entwicklungsprozess erstellt. Auf Seiten eines **Zulieferers** werden die Anforderungen von dem Auftraggeber vorgegeben. In beiden Fällen ist es üblich, diese Anforderungen der Entwicklung in Form eines Lastenhefts zur Verfügung zu stellen. Dabei kann nicht davon ausgegangen werden, dass diese Lastenhefte Anforderungen in der benötigten Qualität beinhalten. Dies ist ein Grund, eine Analyse der gegebenen Anforderungen vorzunehmen. Auf Seiten eines Zulieferers kann ein weiterer Grund sein, die Anforderungen an das zu erstellende System mit Systemen zu vergleichen, die in früheren Projekten entwickelt wurden und so als Grundlage für das neue System dienen können.

Die Ausgabe der Analysephase sind belastbare Anforderungen an das System aus Sicht der Entwicklung. Sie werden im Allgemeinen in Form eines Pflichtenhefts notiert.

Mit diesem Pflichtenheft kann nun der erste Architekturschritt durchgeführt werden. Hierbei werden die Subsysteme identifiziert und die Schnittstellen zwischen ihnen festgelegt. Diese Schnittstellen folgen aus der dritten Aufgabe in der Architektur, der Verteilung der Systemanforderungen auf die Subsysteme. Es entstehen neue Anforderungen an die Subsysteme, die wiederum als Lastenheftanforderungen für die Subsysteme angesehen werden können. Sie bilden nun die Eingabe in den Analyseschritt auf der Subsystemebene, wobei dieselben Ziele wie auf Systemebene verfolgt werden. Der nachfolgende Architekturschritt führt zu Anforderungen an die Komponenten, die zur Entwicklung an die entsprechenden Organisationseinheiten übergeben werden können.

Obwohl der Systems-Engineer die Komponentenentwicklung nicht mehr verantwortet, so ist er daran zumindest begleitend beteiligt. Er beobachtet und sammelt Informationen, um zu entscheiden, ob sich eine der folgenden Änderungen ergeben:

- An den Anforderungen auf Systemebene: Änderungen in einer Komponente führen dazu, dass die anfangs gestellten Systemanforderungen nicht realisiert werden können. Die geänderten Anforderungen müssen abgeklärt, neu auf die Komponenten verteilt und die Schnittstellen angepasst werden.

- An den Anforderungen an einzelne Komponenten: Zwar meldet eine Komponente den Bedarf an Änderungen an ihren Anforderungen, diese können jedoch dadurch aufgefangen werden, dass die betroffene Systemanforderung anders auf die Komponenten aufgeteilt wird und bei Bedarf die Schnittstellen angepasst werden.

3. Systemanalyse

3.1 Überblick

Die Systemanalyse hat zur Aufgabe, die von außen an das System gestellten Anforderungen in Anforderungen zu überführen, die den Bedürfnissen der weiteren Entwicklung genügen.

Die in der Systemanalyse entstehenden Anforderungen (im Folgenden auch „Systemanforderungen") sollten dabei zumindest den folgenden Qualitätskriterien genügen:

- **Vollständigkeit:** Auf einer hohen, abstrakten Ebene müssen alle Anforderungen erfasst sein. Dabei müssen diese dann soweit detailliert sein, dass sie die Herausforderungen, die auf die weitere Entwicklung zukommen, beschreiben. Wird ein agileres Vorgehen angestrebt, so müssen zumindest die architekturrelevanten Anforderungen detailliert genug beschrieben sein.

- **Eindeutig:** Da das Systems-Engineering im Allgemeinen eine verteilte Entwicklung bedeutet, sollten alle entstehenden Artefakte eindeutig dokumentiert werden. Wir schlagen die Verwendung der UML [RuQu12] oder der SysML [SysML] vor, da sie mit ihrer definierten Syntax und Semantik die notwendigen Voraussetzungen mitbringen.

- **Korrektheit:** Entgegen der klassischen Definition von Korrektheit ist damit an dieser Stelle gemeint, dass sich die Anforderungen auf den richtigen Betrachtungsgegenstand beziehen müssen. Auch wenn der Anforderungsgeber eine Anforderung an eine bestimmte Komponente gerichtet hat, so muss doch untersucht werden, ob nicht auch andere Komponenten von dieser Anforderung betroffen sind.

Anforderungen an ein System lassen sich in verschiedene Arten einteilen , die gerade in der Dokumentation unterschiedlich behandelt werden:

- Funktionale Anforderungen
- Qualitätsanforderungen
- Anforderungen an Daten
- Technische Randbedingungen
- Rechtlich-vertragliche Anforderungen

Für die Ermittlung und Dokumentation der funktionalen Anforderungen schlagen wir ein an die Use-Case-Analyse [Cockburn00] angelehntes Vorgehen vor. Zunächst werden die abstraktesten Funktionen des Systems gesucht und dann sukzessive genauer untersucht.

Dadurch entsteht eine Hierarchie von funktionalen Anforderungen, die um die Qualitätsanforderungen ergänzt werden, welche die jeweilige Funktion genauer beschreiben. Zum Einsatz kommen hier die **Use-Case- und Aktivitätsdiagramme** der SysML.

Eine andere, ergänzende Sicht erhält man, wenn man mittels eines **Zustandsautomaten** die nach außen hin sichtbaren Zustände des Systems inklusive der Übergänge zwischen ihnen und den in den Zuständen durchgeführten Funktionen beschreibt. Die Betrachtung aus unterschiedlichen Sichten erlaubt es dem Systems-Engineer, die Vollständigkeit der Anforderungen zu gewährleisten.

Die Anforderungen an die Daten des Systems werden in einem **Informationsmodell** beschrieben. Dieses dient gleichzeitig der Definition der Begriffe, die in den Anforderungen verwendet werden.

Die weiteren Anforderungstypen werden im Allgemeinen **natürlich-sprachlich** beschrieben. Die Vollständigkeit dieser Anforderungen wird durch die Vorgabe einer (produktspezifischen) Vorlage zur Gliederung der Anforderungen unterstützt [Rupp14, Kapitel 12]. Ihre Eindeutigkeit kann mit Hilfe der Master-Schablonen[Rupp14, Kapitel 10] unterstützt werden.

Die einzelnen Methoden und Techniken, die während einer Systemanalyse zum Einsatz kommen können, stellen wir anhand der beiden konstruktiven Tätigkeiten vor, die der Systems-Engineer in dieser Phase durchführen muss. Er muss Anforderungen ermitteln und sie dokumentieren.

3.2 Anforderungen ermitteln

Bei der Ermittlung der Anforderungen kommt auf den Systems-Engineer eine besondere Herausforderung zu, da er die Schnittstelle zu den Stakeholdern des Gesamtsystems darstellt (vgl. Abbildung 4). Die Stakeholder können aus anderen Organisationen oder einer Fachbereichsabteilung kommen, die nicht die Sicht der Entwicklung auf das System haben. Somit fällt dem Systems-Engineer die Aufgabe zu, die ihm gestellten Anforderungen in solche Anforderungen zu übersetzen, die von den weiteren Bearbeitern verstanden werden. Hier hilft die QFD-Methode, die unter anderem diesen Übersetzungsschritt formalisiert (siehe Abschnitt 3.2.1).

Als zweites Hilfsmittel werden hier Regeln zur Untersuchung von natürlich-sprachlichen Beschreibungen eingeführt (das *RE*gelwerk). Da die Anforderungen aus einem Produktmanagement oder von einem externen Kunden meist in natürlicher Sprache dokumentiert sind, bietet diese Art der Untersuchung ein großes Potenzial, um auf sogenannte Defekte in den gegebenen Anforderungen zu kommen (siehe Abschnitt 3.2.2).

Die dritte hier genannte Unterstützung wird durch eine geregelte Art der Dokumentation gebildet, durch die Defizite im Wissen des Systemanalytikers bewusst werden. Abschnitt 3.3 wird darauf eingehen.

Neben diesen drei Arten Anforderungen zu ermitteln, kommen in der Systemanalyse auch einige der aus dem klassischen Requirements-Engineering bekannten Techniken [Rupp14, Kapitel 6] zum Einsatz. Gerade die dokumentbasierten Techniken und die Beobachtungstechniken versprechen einen Wissensgewinn für den Systemanalytiker.

3.2.1 Quality Function Deployment

Die Methode des Quality Function Deployment [Akao92] geht auf den Japaner Yoji Akao zurück, und unterstützt in einem teamorientierten Ansatz die Entwicklung von Produkten, indem die Kundenwünsche sukzessive weiter in Richtung technische Anforderungen entwickelt werden. Dabei stehen sowohl die gewünschten Qualitätseigenschaften des Produkts als auch die Einbeziehung aller betroffenen Disziplinen (Entwicklung, Fertigung, Test, etc.) im Fokus.

In der ersten, für uns hier interessantesten Phase, werden die Kundenwünsche mit den mehr technischen und messbaren Systemeigenschaften (auch als Entwicklungsparameter bezeichnet) in Relation gesetzt. Dies führt zu einem guten Ausgangspunkt für die weitere Entwicklung, da hier Abhängigkeiten zwischen und Prioritäten für Anforderungen gefunden und entsprechend dokumentiert werden.

In der Praxis hat sich als Vorgehen das sogenannte House of Quality durchgesetzt, bei dem die Schritte der QFD in einer komplexen, einem Haus ähnelnden Matrix dokumentiert werden. Das Ausfüllen kann in bis zu 12 Schritten durchgeführt werden. Wir wollen hier nur die ersten vier dieser 12 Schritte darstellen, die die direkte Übersetzung der Kundenanforderungen in die Systemeigenschaften adressieren.

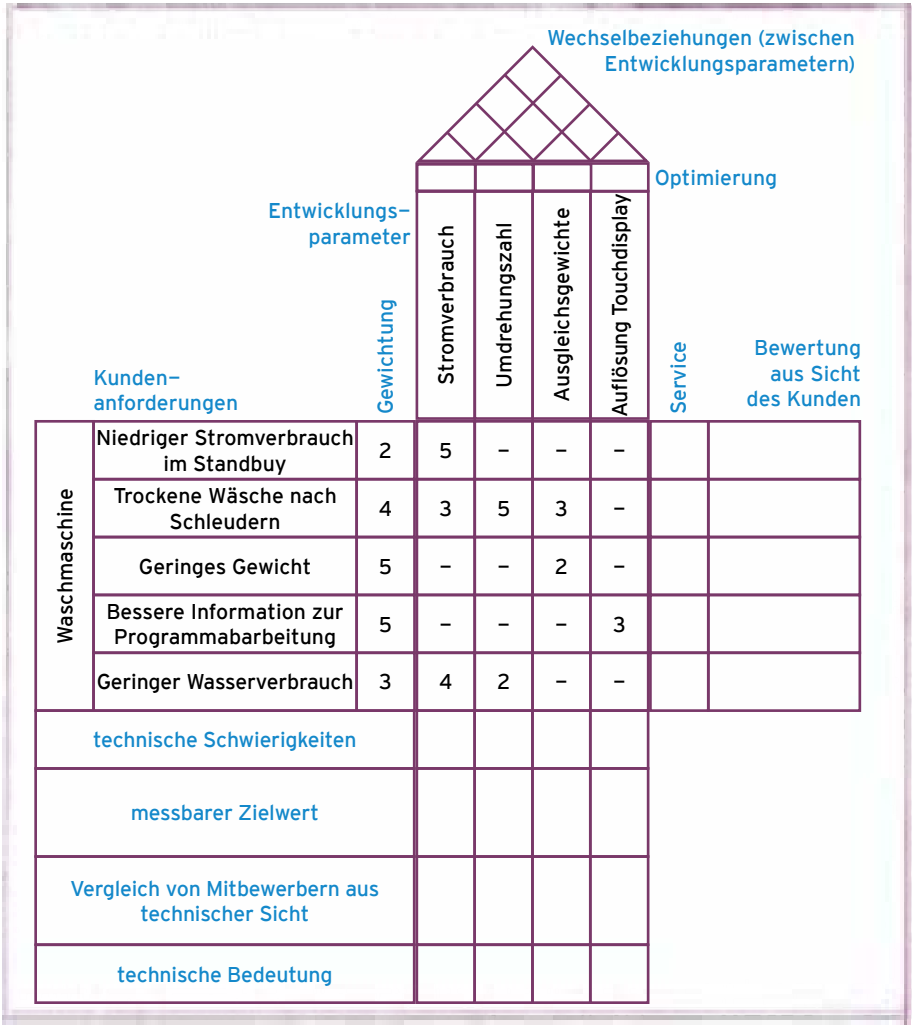


Abbildung 05: House of Quality

Im ersten Schritt werden die wichtigsten Kundenanforderungen identifiziert und eingetragen. Danach können sie, zusammen mit dem Produktmanagement, gewichtet werden (Schritt 2). Der kreative, im Team durchgeführte Schritt besteht nun darin, die Entwicklungsparameter zu finden, auf die die Kundenanforderungen einen Einfluss haben (Schritt 3). Abschließend werden die beiden Typen von Anforderungen in Relation gesetzt. Hiermit wird ausgesagt, wie stark die Auswirkung der Kundenanforderung auf eine, Entwicklungsparameter ist. Diese Gewichte und die Gewichtung der Kundenanforderungen bieten eine nachvollziehbare Grundlage zur Festlegung der Systemanforderungen.

Diese sehr abgekürzte Darstellung der QFD soll nur einen Eindruck über die Anwendbarkeit bei der Ermittlung der Systemanforderungen vermitteln. Die in der Literatur vorgestellte Methode bietet weitere Schritte, die sowohl mehr als auch verlässlichere Informationen zu den Systemeigenschaften erzeugt [Akao92].

Die maßgeblichen Einträge, Kundenwünsche und Systemeigenschaften, sind natürlich sehr stark von der jeweiligen Anwendung abhängig. Weiterhin ist die Anwendung dieser Methode sehr zeitaufwändig, da die Güte des Ergebnisses der QFD sehr stark von den richtig gewählten Systemeigenschaften abhängt. Hier kommt aber der eigentliche Zweck dieser Methode zum Tragen: Die stetige Weiterentwicklung eines Produkts, um auf neue Kundenwünsche eingehen zu können. Somit kann für ein Produkt (oder eine Reihe von ähnlichen Produkten) einmal Zeit in das Finden der Systemeigenschaften investiert werden, und diese dann bei einer erneuten Anwendung der QFD-Methode für ein ähnliches System wiederverwendet werden.

3.2.2 *RE*gelwerk

In den meisten Fällen sind die Anforderungen, die als Eingabe in die Systemanalyse vorgegeben werden, in natürlicher Sprache dokumentiert. Gerade ein Produktmanagement möchte sich nicht von einer formaleren Sprache in seiner Kreativität einengen lassen.

Deswegen besteht ein großer Bestandteil bei der Ermittlung von Aufgaben darin, Informationen aus textuellen Anforderungen abzuleiten. Dabei werden die textuellen Anforderungen mit natürlich-sprachlichen Techniken analysiert, um die folgenden Defizite in den Anforderungen zu erkennen

- Interpretationsspielräume
- Fehlende Anforderungen
- Inkonsistenzen zwischen Anforderungen

In der Literatur existieren zahlreiche Ansätze zur Untersuchung von natürlich-sprachlichen Anforderungen. Diese reichen von theoretischen, aber automatisierbaren Ansätzen [Krisch14] bis hin zu pragmatisch einsetzbaren Regelwerken.

Als Beispiel für die zweite Art von Unterstützung nennen wir hier zwei Regeln des SOPHIST-*RE*gelwerks [Rupp14, Kapitel 7]. In diesem Regelwerk werden 18 Regeln definiert, die es erlauben, verzerrte (falsche), getilgte (fehlende) oder generalisierte (zu abstrakte) Informationen zu erkennen.

Regel 1:
Falls ein Satz im Passiv definiert ist, so fehlt häufig der Handelnde in diesem Satz.

Vorgegebene Anforderung:
Die Trommel soll angetrieben werden.

Aktion für den Analytiker:
Nachfragen, wer die Trommel antreiben soll.

Regel 2:
Eine Nominalisierung in einem Satz deutet darauf hin, dass ein weiterer Prozess im System definiert werden muss.

Vorgegebene Anforderung:
Nach der Initialisierung muss die Waschmaschine „Bereit" auf dem Display anzeigen.

Aktion für den Analytiker:
Herausfinden, ob der Prozess der Initialisierung bereits beschrieben ist, oder ob es relevant ist, ihn zu beschreiben.

3.3 Anforderungen dokumentieren

Bei der Dokumentation der Systemanforderungen gehen wir in dieser Broschüre von einer Mischung zwischen natürlich-sprachlichen Anforderungen und einer modellbasierten Darstellung in SysML aus. Im Regelfall reicht für die modellbasierte Dokumentation eine solche semiformale Notation aus. In sehr sicherheitskritischen Anwendungsbereichen (Militärischer Bereich, Teile eines Fahrzeugs), in denen z.B. auch nur zertifizierter Programmcode eingesetzt werden darf, können auch formalere Dokumentationstechniken wie temporale Logik oder Prädikatenlogik [Barwise05] eingesetzt werden.

Klassischerweise wird die Analyse mit einer Systemabgrenzung begonnen. Das hier vorgestellte weitere Vorgehen in der Analyse richtet sich, wie in Abschnitt 3.1 eingeführt, nach der Use-Case-basierten Analyse. Dadurch können wir die in Use-Case- und Aktivitätsdiagrammen dokumentierten funktionalen Anforderungen als „führend" ansehen.

Die natürlich-sprachlichen Anforderungen zur Präzisierung der so notierten funktionalen Anforderungen werden in das entstehende SysML-Modell integriert. Ergänzt werden diese Anforderungen durch die Modellierung eines Informationsmodells, in dem die Anforderungen an die Daten beschrieben werden. Abgerundet wird das Bild durch die Einführung von Zustandsautomaten, um entweder die wichtigen Systemzustände oder Zustände wichtiger Entitäten aus unserem System darzustellen.

Dieses Vorgehen ist mittlerweile zum Standard bei der Dokumentation von Anforderungen geworden. In [ALM16] werden auch die Notationselemente vorgestellt, die man typischerweise bei der modellbasierten Dokumentation von Anforderungen einsetzt. Dabei geht man davon aus, dass alle in einem Modell notierten Informationen als Anforderungen gelten und nicht zusätzlich als natürlich-sprachliche Anforderungen dokumentiert werden müssen. Die Dokumentation der weiteren Anforderungstypen (siehe Abschnitt 3.1) wird im Regelfall natürlich-sprachlich und nicht im Modell durchgeführt. Dabei können Schablonen zur Formulierung guter natürlich-sprachlicher Anforderungen verwendet werden. Da diese jedoch nicht spezifisch für das Systems-Engineering sind, sondern für die Formulierung von Anforderungen im Allgemeinen gültig sind, werden wir diese hier nicht weiter vorstellen, sondern verweisen auf [Rupp14, Kapitel 10].

Abweichungen und Ergänzungen zu dem hier vorgestellten Vorgehen können je nach Anwendungsdomäne natürlich immer sinnvoll sein. Bezüglich der hier dargestellten Beispiele gilt, dass sie keinen Anspruch auf Vollständigkeit erheben.

3.3.1 Block-Definitions-Diagramm für den Systemkontext

Mit Hilfe eines Block-Definitions-Diagrams wird die Einordnung des Systems in seinen technischen Kontext gezeigt.

Der technische Kontext zeigt die mit dem betrachteten System in Beziehung stehenden Nachbarsysteme und Benutzer.

Diese Einordnung wird im Allgemeinen entweder aus einer darüber liegenden Architektur oder vom Produktmanagement vorgegeben. Es ist eine sehr einfache Sicht auf das System, in ihr können aber gut die Systeme und Anwender des betrachteten Systems dargestellt werden. Weiterhin können Sie zusammen mit den Assoziationen die Ein- bzw. Ausgaben des Systems darstellen.

Dieses Diagramm wird in der Architektur in eine formalere Darstellung (als Black-Box-Sicht auf den Betrachtungsgegenstand) überführt (siehe Abschnitt 4.1)

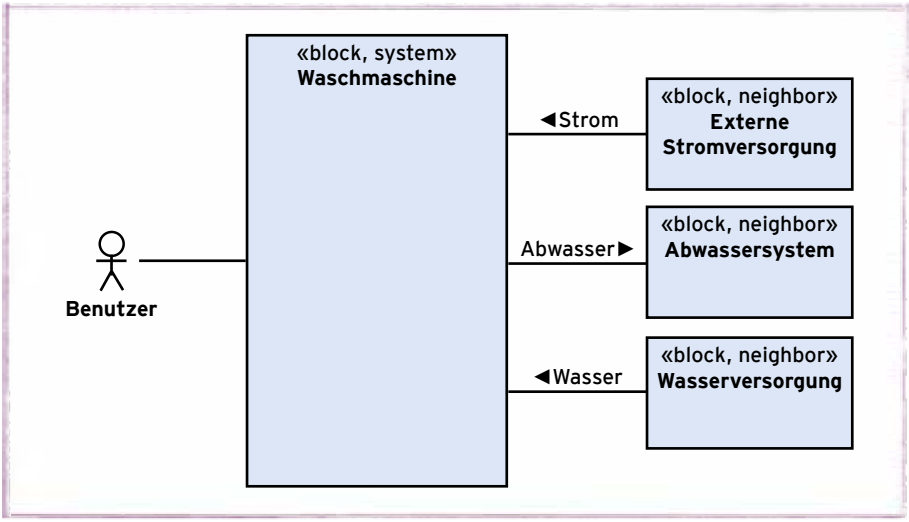


Abbildung 06: Kontextdiagramm der Waschmaschine

3.3.2 Use-Case-Diagramm für Systemprozesse

Use-Cases (dt.: Anwendungsfälle) beschreiben auf sehr abstrakter Ebene die Funktionen, die das System einem Akteur des Systems zur Verfügung stellt. Dabei ist mit Akteur entweder ein Benutzer des Systems oder ein Nachbarsystem gemeint, das mit dem betrachteten System in Verbindung steht. Aus der Literatur [Cockburn00] leitet sich die folgende Definition für einen Use-Case ab:

Ein Use-Case ist eine typische Interaktion des Benutzers mit dem System mit einem fachlichen Wert.

Aus dieser Definition folgen zwei wichtige Kriterien für Use-Cases, aus denen sich die weitere Vorgehensweise ableitet.

- Einem Use-Case liegt ein **Ablauf** zugrunde. Deswegen werden wir die Use-Cases mit Ablaufdiagrammen genauer beschreiben.

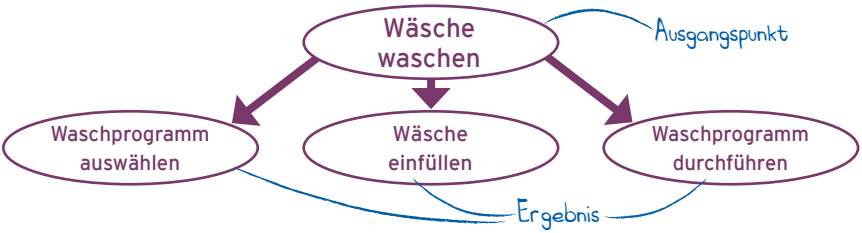
- Ein Use-Case dient einem **fachlichen Ziel**. Die geforderte Funktionalität folgt aus einem übergeordneten Prozess. Dies kann ein Geschäfts- oder Behandlungsprozess sein, den ein Anwender unter anderem mit dem betrachteten System durchführt. Oder es ist ein Prozess, der von dem Supersystem durchgeführt wird und zu dem das betrachtete System nur einen Teil zuliefert.

Da der Begriff „Use-Case" sehr häufig mit einem menschlichen Akteur in Verbindung gebracht wird, wir im Systems-Engineering aber auch die mehr technischen Prozesse betrachten müssen, werden wir im Folgenden den Begriff „Systemprozess" anstelle eines Use-Cases verwenden.

Für das Finden von Systemprozessen gibt es mehrere Möglichkeiten, wobei natürlich immer das Lastenheft als primäre Eingabe dient.

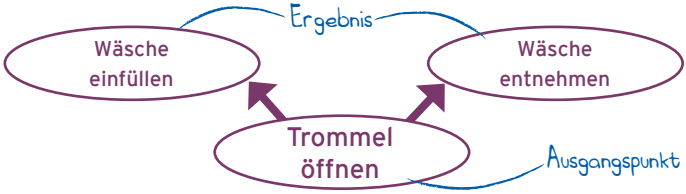
▶▶ 1. Das Top-Down-Vorgehen

Stellen Sie sich die Frage, weshalb ein Akteur das System in Anspruch nehmen möchte. Die Antwort auf diese Frage folgt im Allgemeinen aus den Geschäftsprozessen. Da diese aber sehr selten vollständig angegeben sind, lohnt es sich, diese Sicht von außen im Rahmen der Systemanalyse einzunehmen. Die Antworten auf die Frage werden im Allgemeinen zu Funktionen führen, die entsprechend der Kriterien für Use-Cases weiter zerlegt werden können [Rupp14, Kapitel 8].



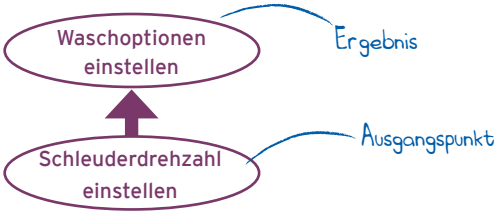
▶▶ 2. Das Bottom-Up-Vorgehen

Wenn Vorgängersysteme zur Verfügung stehen, können Sie Bottom-Up vorgehen, indem Sie dort solche Funktionen identifizieren, die nach außen hin sichtbar sind und somit eine Interaktion mit einem Akteur beinhalten. Für jede so gefundene Funktion können Sie sich nun die Frage stellen, welchem größeren Zweck diese Funktion dient. Damit finden Sie eine abstraktere Funktion des Systems, die eher einen Kandidaten für einen Systemprozess darstellt. Wenden Sie die gleiche Frage für die neu gefundene Funktionalität an bis Sie einen Prozess finden, der nicht mehr alleine durch das System unterstützt wird. Damit haben Sie einen Systemprozess identifiziert.



▶▶ 3. Schnittstellenanforderungen

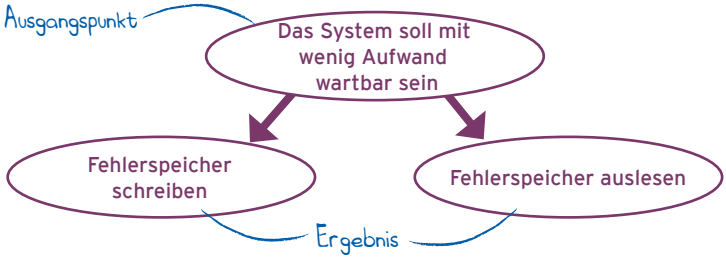
Falls bereits Schnittstellenanforderungen zu benachbarten Systemen oder zum Benutzer vorliegen, können Sie die Liste der bisher gefundenen Systemprozesse überprüfen, indem Sie die an den Schnittstellen geforderte Funktionalität den Systemprozessen zuordnen. Somit erhalten Sie unter Umständen Hinweise auf fehlende Systemprozesse.



▶▶ 4. Analyse der Qualitätsanforderungen

Die letzte hier genannte Möglichkeit besteht in der Analyse der im Lastenheft geforderten Qualitätsanforderungen. Manchmal werden Lösungen für die dort geforderten Anforderun-

gen aus Funktionen bestehen, die das System realisieren soll. Dieser Schritt kann für neue Systeme sehr kreativ sein, für bekannte Typen von Systemen werden Sie aber auf bewährte Lösungen aufbauen können. Ein großer Vorteil dabei besteht in der nicht benötigten Präzisierung der Lastenheftanforderung. Bevor Sie mit Ihrem Kunden diskutieren, wann etwas „schön", „schnell" oder „anmutig" ist, können Sie die Auswirkungen auf das System beschreiben und mit dem Kunden abklären, ob diese Lösung seine Ursprungsanforderung löst.



Als Ergebnis dieser Betrachtung entsteht ein Use-Case-Diagramm mit den für das System relevanten Systemprozessen.

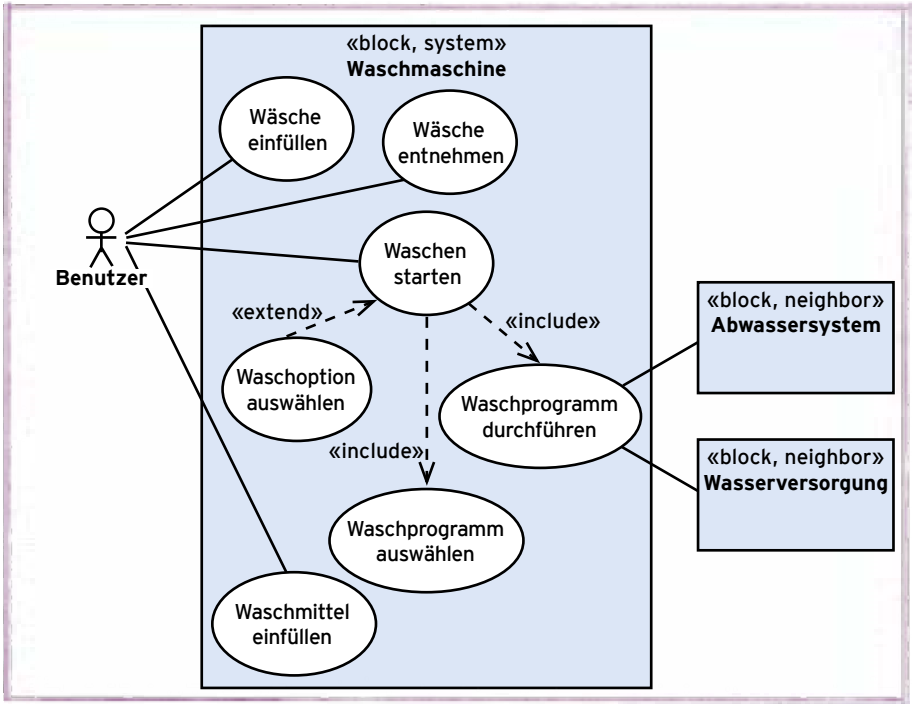


Abbildung 07: Use-Case-Diagramm mit den Systemprozessen

3.3.3 Aktivitätsdiagramm als Ablaufbeschreibung

Mit Hilfe von Aktivitätsdiagrammen können Sie die einem Systemprozess zugrunde liegenden Abläufe beschreiben. Gegenüber der klassischen Use-Case-Betrachtung werden Sie hier aber nicht nur auf die Aktionen eines Benutzers eingehen, sondern vielmehr auch die Prozessschritte betrachten, die das System durchführen soll. Aber beachten Sie dabei, dass Sie in der Systemanalyse von dem Betrachtungsgegenstand „System" ausgehen, und Sie keine Anforderungen an Teile des Systems (Subsysteme oder Komponenten) stellen.

Aktivitätsdiagramme bringen die Benutzereingaben und Systemaktionen für einen Systemprozess in einen Ablauf.

Bei dieser Tätigkeit können sehr komplexe Abläufe entstehen. Versuchen Sie die einzelnen Aktionen in Ihren Diagrammen hierarchisch zu gliedern, um auf einer Ebene in einem Diagramm die Übersichtlichkeit zu gewährleisten. Bezüglich der in der Systemanalyse angestrebten Detaillierung für die funktionalen Anforderungen gilt hier:

- Die Lastenheftanforderungen sollten sinnvoll den Schritten im Ablauf zugeordnet werden können.
- Alle Anforderungen, die Sie als Analytiker aus der fachlichen Domäne heraus der weiteren Entwicklung vorgeben möchten, sollten ebenfalls sinnvoll einem Schritt im Ablauf zugeordnet werden können.

Damit kann eine Vielzahl von Aktivitätsdiagrammen entstehen, die entweder direkt einem Systemprozess oder einer Aktion in einem solchen Diagramm zugeordnet sind.

Abbildung 8 zeigt den Ablauf des Systemprozesses „Waschen starten".

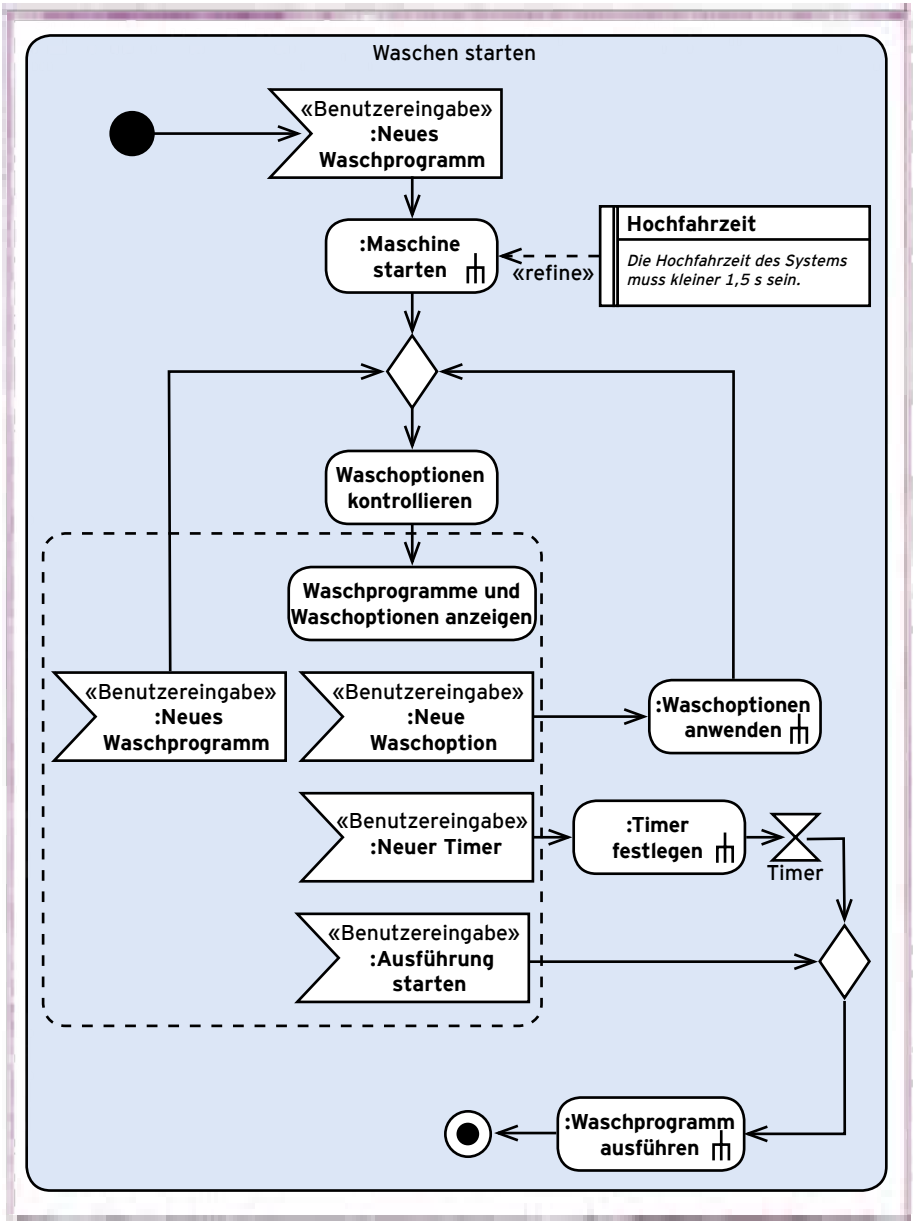


Abbildung 08: Aktivitätsdiagramm für einen Systemprozess

Die folgende Abbildung zeigt die Verfeinerung der Aktion „Waschoptionen anwenden“ aus Abbildung 8.

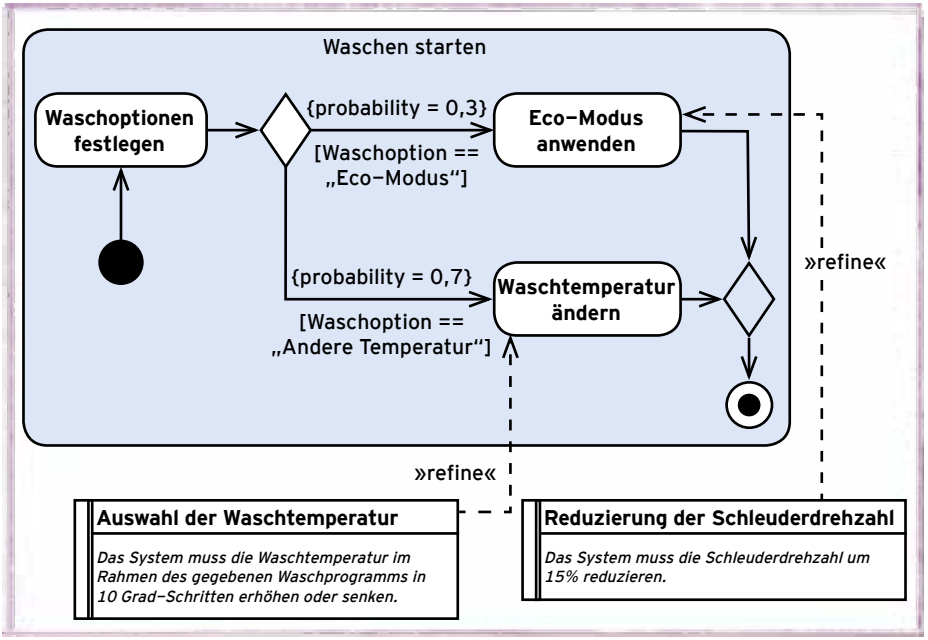


Abbildung 09: Beispiel einer Verfeinerung

Zusätzlich zu den Abläufen sind natürlich-sprachliche Anforderungen dargestellt, die Modellelemente in den Diagrammen z.B. durch Qualitätsanforderungen weiter beschreiben können. Sie werden dann verwendet, wenn die natürlich-sprachliche Darstellung geschickter als die Darstellung in SysML erscheint.

3.3.4 Klassendiagramm als Informationsmodell

Im Rahmen einer modellbasierten Dokumentation von Anforderungen enthält ein Informationsmodell zwei Arten von Informationen, die jedoch eng zusammen hängen.

In einem Informationsmodell werden die fachlich relevanten Begriffe zusammen mit ihren fachlich relevanten Daten definiert.

Beide Anteile des Informationsmodells stellen wiederum Anforderungen dar. Eine Aktion „Waschoptionen auswählen“ müsste dadurch präzisiert werden, dass die auswählbaren Optionen aufgezählt werden. Dies kann dadurch erspart werden, dass diese speziellen Optionen im Informationsmodell modelliert werden. Somit kann ein Informationsmodell Anforderungen enthalten und gleichzeitig als andere Darstellungsform eines Glossars angesehen werden.

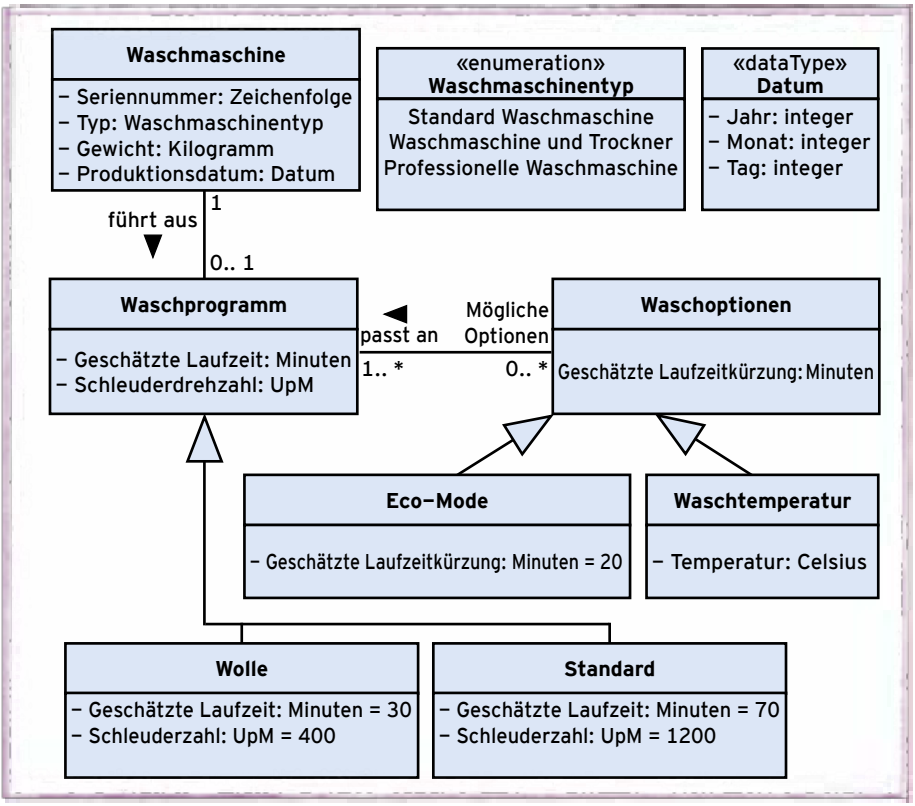


Abbildung 10: Informationsmodell der Waschmaschine

Abbildung 10 zeigt einen Ausschnitt aus dem Informationsmodell der Waschmaschine. Ergänzt werden kann es durch die Hinzunahme von Schnittstellen, um insbesondere die fachlichen Daten, die über die Schnittstellen ausgetauscht werden, zu dokumentieren.

3.3.5 Zustandsautomaten

Die Zustandsautomaten bilden die letzte Diagrammform, die üblicherweise zur Beschreibung von Anforderungen genutzt wird. Gerade bei technischen Systemen bietet diese andere Sichtweise auf das System einige Vorteile, da ein enger Zusammenhang zwischen dieser Sicht und der funktionsorientierten Sicht aus den Kapiteln 3.3.2 und 3.3.3. besteht.

Zustände des Systems geben die Bedingungen an, wann ein Systemprozess einem Akteur zur Verfügung steht oder sie definieren Bedingungen, die in den Abläufen zu Entscheidungen führen.

Damit folgt eine Festlegung für die Systemzustände, die in der Systemanalyse relevant sind. Das System weist nach außen hin in Abhängigkeit vom aktuellen Systemzustand ein unterschiedliches Systemverhalten auf. Entweder dadurch, dass dem Benutzer gewisse Systemprozesse nicht zur Verfügung stehen, oder dass sich die Abläufe der Systemprozesse unterscheiden.

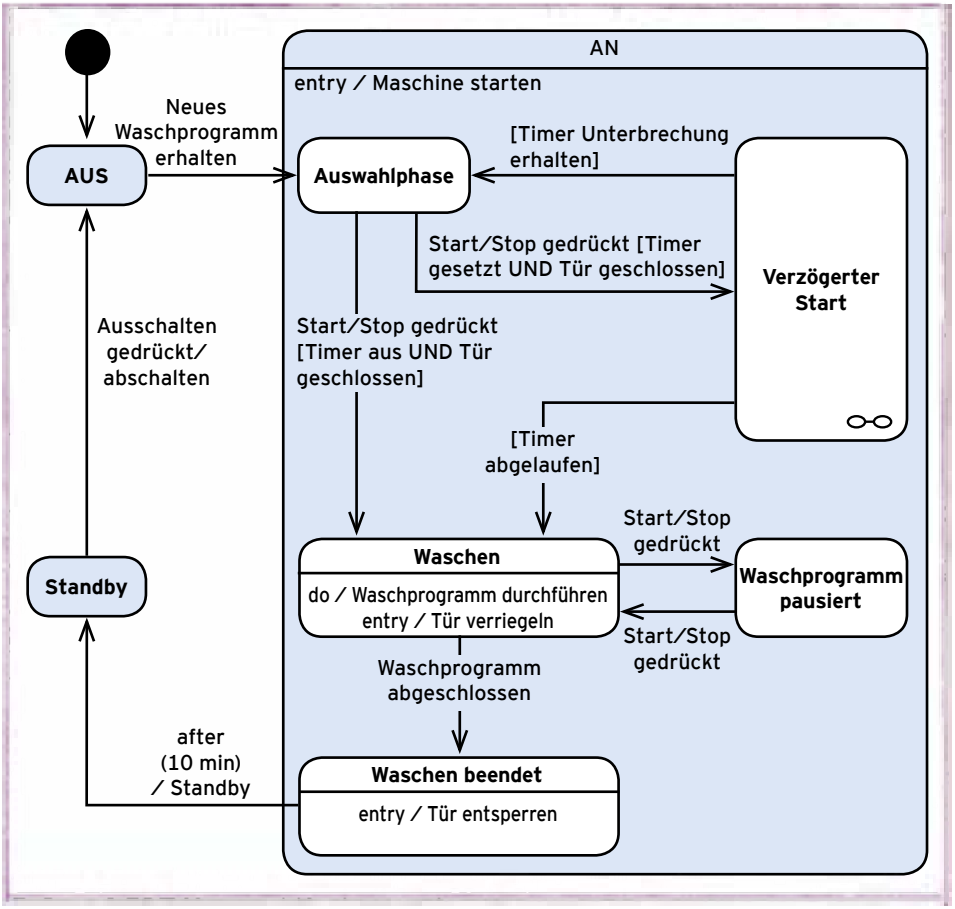


Abbildung 11: Zustände der Waschmaschine

Abbildung 11 zeigt eine typische Eigenschaft der Zustandsautomaten von technischen Systemen: Sie besitzen keinen Endzustand. Weiterhin kann hier auch von der Möglichkeit der Verfeinerung Gebrauch gemacht werden (Zustand „Verzögerter Start“ und Zustand „An“).

3.4 Vorgehen in der Analyse

Zum Abschluss der Vorstellung der Systemanalyse möchten wir nun die oben angesprochenen Tätigkeiten und entstehenden Diagrammtypen in einen Ablauf bringen, der als Anhaltspunkt für ein Vorgehen in einem Projekt dienen kann.

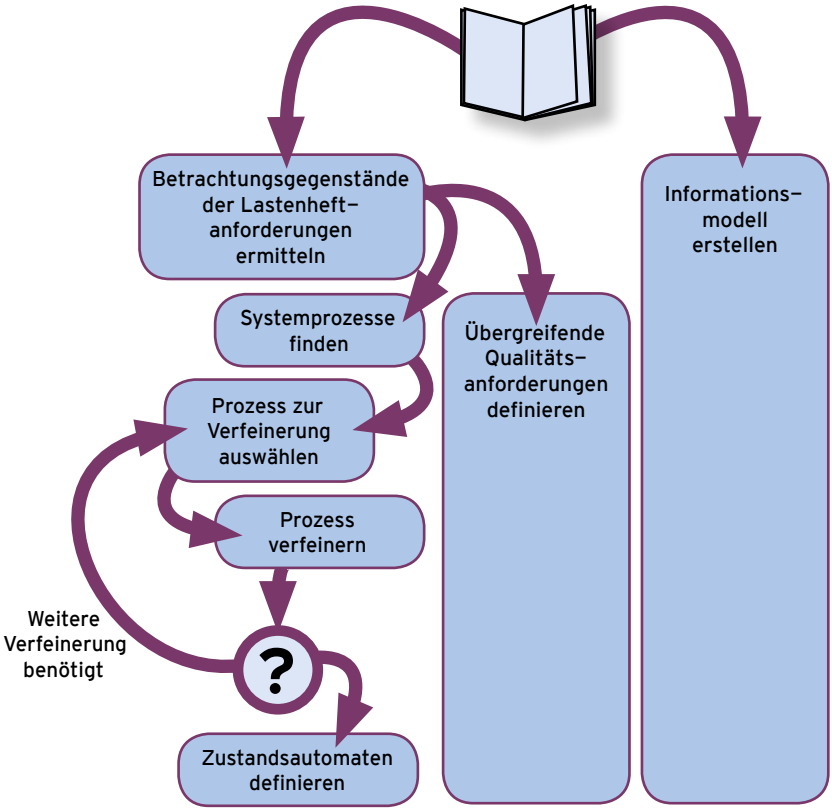


Abbildung 12: Vorgehen in der Systemanalyse

Dabei gehen wir von einem Lastenheft als Eingabe aus, das entweder von einem externen Kunden oder einem Produktmanagement erstellt wurde. In beiden Fällen kann nicht davon ausgegangen werden, dass die Inhalte des Lastenhefts der von der weiteren Entwicklung benötigten Qualität genügen. Deswegen ist der erste Schritt, den Betrachtungsgegenstand von jeder im Lastenheft gegebenen Anforderung zu prüfen. Falls eine solche Anforderung nicht das Gesamtsystem, sondern nur eine Komponente dessen adressiert, so muss sich der Analytiker die Frage stellen, aus welcher Anforderung auf Systemebene die betrachtete Anforderung resultiert. Existiert zum Beispiel ein Systemprozess, oder muss er noch definiert werden?

Gleichzeitig mit dieser Betrachtung der Lastenheftanforderungen bietet es sich an, die verwendeten Begriffe im Informationsmodell festzuhalten. Bei allen weiteren Arbeiten in der Systemanalyse sollten Sie bei Bedarf das Informationsmodell erweitern bzw. ändern.

Nach diesem Schritt können Sie die Ebene der Systemprozesse vervollständigen. Damit haben Sie einen guten Ausgangspunkt für die Entscheidung, welchen Systemprozess Sie als Erstes verfeinern wollen. Bei jedem Schritt in die Tiefe sollten Sie bemüht sein, diesen Schritt vollständig zu machen. Dies bedeutet zum Einen, den Ablauf auf dieser Ebene vollständig zu definieren, und zum Anderen, die jeweils benötigten Qualitätsanforderungen festzulegen.

Somit entsteht eine vollständige Beschreibung für einen Systemprozess und Sie haben die Wahlmöglichkeit, welche Schritte Sie als nächstes genauer betrachten möchten. Bei dieser Entscheidung sollten Sie die für die weitere Entwicklung kritischen Abläufe favorisieren.

Spätestens am Ende, gerne aber auch nach der Fertigstellung wichtiger Abläufe, sollten Sie die bisher gewonnenen Anforderungen durch die Modellierung eines Zustandsautomaten oder dessen Erweiterung validieren.

Die von den Abläufen losgelösten Qualitätsanforderungen können während der gesamten Analysephase definiert werden, auch wenn sie größtenteils direkt aus der Betrachtung des Lastenhefts folgen werden.

Nach diesen Schritten liegen Systemanforderungen in der benötigten Qualität (vollständig und widerspruchsfrei) und in einer bewusst gewählten Detaillierung vor.

4. Systemarchitektur

Die Systemarchitektur bildet den ersten Schritt der Realisierung.

Unter dem Begriff der Systemarchitektur werden die folgenden Kernaufgaben verstanden:

- ***Die Bestandteile des Systems identifizieren bis auf die kleinste Ebene, die für die Betrachtung im Rahmen der Systems-Engineering Relevanz hat***
- ***Für diese Bestandteile die Anforderungen aus den Systemanforderungen ableiten***
- ***Die Schnittstellen zwischen den Bestandteilen festlegen***

Ähnlich wie in der Systemanalyse werden auch in der Systemarchitektur drei Tätigkeiten durchgeführt: das Ermitteln, die Dokumentation und das Überprüfen der Systemarchitektur.

Das Finden einer geeigneten Architektur ist im Allgemeinen ein sehr kreativer Prozess, der sehr von der Erfahrung der beteiligten Entwickler abhängt und oftmals auf alte Produkte mit ihren Architekturen aufsetzt. Unterstützt wird diese Tätigkeit durch das Bilden verschiedener Sichten auf das System.

Zum Überprüfen der getroffenen Entscheidungen existieren einige Ansätze, die in die folgenden drei Gruppen eingeteilt werden können:

- Ein **Review** mit den Gesamtverantwortlichen der gefundenen Komponenten sollte auf jeden Fall durchgeführt werden. Es stellt sicher, dass die Kommunikation der Entscheidungen zu den Entwicklern, die auf diesen Entscheidungen ihre Entwicklungen aufbauen, stattfindet.

- Eine **Bewertung** der aktuellen Architektur kann z.B. mit Hilfe der QFD durchgeführt werden. Damit können Sie z.B. die Stabilität Ihrer Architektur überprüfen. Weiterhin können Sie spezielle Aspekte wie z.B. das Datenaufkommen an Schnittstellen untersuchen. Hier kommen Techniken (z.B. statische und dynamische Kopplung) zum Tragen, wie sie aus der Softwarearchitektur bekannt sind [Starke14]. Auch eine FMEA kann als überprüfende Technik angesehen werden, da sie eine bestehende Architektur auf Schwachstellen durch die Betrachtung der Fehlerauswirkungen überprüft [Pfeufer14].

- **Formale Ansätze** dienen zur Überprüfung, ob die gewählte Systemarchitektur spezielle Aspekte aus den Anforderungen erfüllt [Pflüger14]. Diese Aspekte beziehen sich hauptsächlich auf die Qualitätsanforderungen wie Bandbreite eines Busses, Stromaufnahme des Systems oder auch die benötigte Rechenleistungen der Prozessoren.

Im Folgenden werden wir die Dokumentation der benötigten Informationen mit Hilfe der SysML genauer vorstellen und dabei motivieren, wie diese Dokumentationstechniken die Ermittlung und Überprüfung der Architektur unterstützen kann.

Für die Dokumentation der Systemarchitektur wird zunächst das Kontextdiagramm aus Abschnitt 3.3.1 in eine formalere Darstellung, die **Black-Box-Sicht**, überführt (Abschnitt 4.1).

Für die Darstellung der Zerlegung des Systems stellen wir zunächst die **funktionale Sicht** vor (Abschnitt 4.2). Sie hilft Ihnen bei der Identifikation der Komponenten, die an einer Funktionalität des Systems beteiligt sind.

Als zweite Sicht auf die Struktur des Systems stellen wir die **physikalische Sicht** vor (Abschnitt 4.3), in der wir die Komponenten eindeutig zuteilen und die somit als Grundlage für die Strukturierung der gesamten Dokumentation des Systems dienen kann (siehe auch Abschnitt 5.3).

Darüber hinaus können Sie noch weitere Sichten auf die Struktur des Systems bilden, um weitere Aspekte, die bei der Systemarchitektur relevant sind, darzustellen. Dies kann z.B. eine Darstellung der Fertigung des Systems sein (eine Produktionssicht) oder eine Netzwerksicht, in der ausschließlich die Kommunikationswege zwischen PCBs dargestellt sind.

Prinzipiell richtet sich die Wahl der benötigten Sichten nach Ihrem System und dem von Ihnen gewählten Vorgehen. Deshalb können die hier angesprochenen Sichten nur als Vorschlag verstanden werden. Eine andere Art der Definition der Sichten findet sich in [Pohl et al.12].

4.1 Black-Box-Sicht

Die Black-Box-Sicht stellt das System in seinem technischen Kontext zusammen mit seinen Schnittstellen dar.

Die benötigten Informationen für dieses Diagramm sind im Wesentlichen die Schnittstellen des Systems. Sie können in zwei Gruppen unterteilt werden:

- Schnittstellen, die für die Entwicklung in dem Lastenheft oder aus einer darüber liegenden Systemarchitektur (siehe Abschnitt 4.3) vorgegeben sind.
- Schnittstellen, die während der Entwicklung festgelegt werden.

Die Schnittstellen der ersten Kategorie können direkt zu Beginn modelliert werden. Das so entstandene Diagramm sollte während der gesamten Entwicklung immer wieder angepasst werden, um Änderungen direkt zum Auftraggeber oder zu dem Systemarchitekten des Supersystems kommunizieren zu können.

Als Ergebnis liegt ein Internal-Block-Diagramm wie in Abbildung 13 vor.

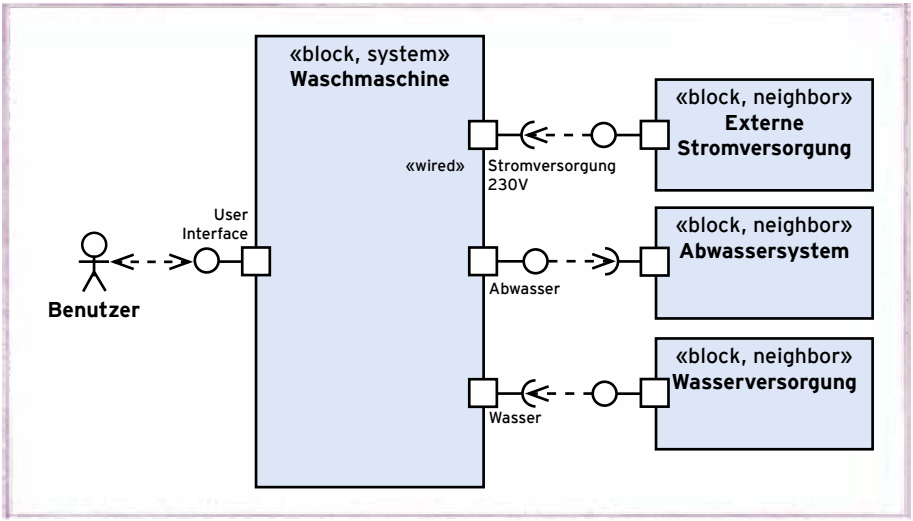


Abbildung 13: Black-Box-Sicht der Waschmaschine

In diesem Diagramm fällt die Verwendung der beiden Notationsmittel „Ports" und „Interfaces" der SysML auf. Es hat sich bewährt, die verschiedenen Informationen einer Schnittstelle in unterschiedlichen Modellelementen darzustellen.

Für die Zuordnung der Informationen zu den Notationsmitteln gilt dabei:

- Einem **Port** werden die mehr physikalischen Eigenschaften der Schnittstelle zugeordnet, also zum Beispiel der Typ der Verbindung des Wasseranschlusses oder der Typ des Steckers einer LAN-Verbindung.

- Dem **Interface** werden die fachlichen Daten mit Hilfe eines Verweises auf das Informationsmodell zugewiesen. Welche Daten werden über eine LAN-Verbindung ausgetauscht, oder welche Informationen werden dem Benutzer angezeigt?

Die Richtung der Schnittstellen (Bietet das System eine Schnittstelle an oder wird sie vom System benötigt?) hängt in der Systemarchitektur von der fachlichen Betrachtung und nicht von der technischen Umsetzung ab. So wird von der Waschmaschine frisches Wasser und Strom benötigt (benötigte Schnittstellen) und sie liefert das Abwasser (bereitgestellte Schnittstelle). Diese Zuordnung ist nicht immer eindeutig, jedoch sollten Sie eine prinzipielle Art der Festlegung finden, die Sie dann durchgängig einsetzen können.

4.2 Funktionale Sicht

Die funktionale Sicht stellt die Komponenten des Systems mit ihren Schnittstellen dar, die an der Realisierung einer Funktion beteiligt sind.

Dieser Inhalt der funktionalen Sicht ist dazu prädestiniert, eine zentrale Aufgabe der Architektur zu unterstützen: die Zerlegung der geforderten funktionalen Systemanforderungen in funktionale Anforderungen an die beteiligten Komponenten. Dabei werden ebenso die einer Systemfunktion zugeordneten qualitativen Anforderungen zerlegt und den Komponenten zugeordnet.

Bei der Erstellung dieser Sicht werden Sie die Komponenten und ihre Schnittstellen verwenden können, die bereits in der physikalischen Sicht definiert wurden. Sollten diese nicht ausreichend sein, so können Sie in diesem Schritt natürlich neue Komponenten und neue Schnittstellen definieren. Diese sollten Sie allerdings ebenfalls in der physikalischen Sicht hinzufügen, um so die dort geforderte Vollständigkeit zu erreichen (siehe Abschnitt 4.3).

Welche Systemfunktionen Sie in der funktionalen Sicht betrachten, hängt von der Komplexität der jeweiligen Systemfunktion und damit der Anzahl der beteiligten Komponenten ab. Sie sollte aber auf jeden Fall eine Funktion sein, die Sie in der Systemanalyse im Rahmen der Ablaufbeschreibung identifiziert und modelliert haben (Abschnitt 3.3.3). Sie wird in der Architektur in einen Block überführt und in einem Internal-Block-Diagramm dargestellt.

Für das in Abbildung 14 dargestellte Beispiel wurde bewusst eine einfache Funktion gewählt. Sie würde in einer Verfeinerung der Aktivität „Waschprogramm ausführen“ aus Abbildung 8 in der Systemanalyse definiert werden. Trotzdem lässt sich gut darstellen, dass bei vollständiger Betrachtung mehr Komponenten und Schnittstellen als direkt offensichtlich an der Funktion beteiligt sind. So muss z.B. die Fixierung der Trommel so ausgelegt sein, dass die geforderte Drehzahl der Trommel erreicht werden kann, ohne dass sich die Trommel vom Rahmen löst.

Aus der Betrachtung der Systemfunktion folgen neben den Schnittstellen auch funktionale und nichtfunktionale Anforderungen an die jeweiligen Komponenten. Diese könnten ebenfalls in der funktionalen Sicht definiert werden. Jedoch bietet es sich an, sie wie die Schnittstellen in der physikalischen Sicht zu definieren, da in dieser Sicht alle Informationen zu einer Komponente aus verschiedenen Sichten und verschiedenen Diagrammen zusammengefasst werden können. Damit stellen die funktionale Sicht und auch die anderen Sichten nur einen Teil aller Informationen aus der physikalischen Sicht dar. Diese zentrale Platzierung der Informationen und die Ausschnittsbildung in verschiedenen Sichten erscheinen zunächst

kompliziert, jedoch wird diese Art der Dokumentation von den einschlägigen Modellierungswerkzeugen sehr gut unterstützt.

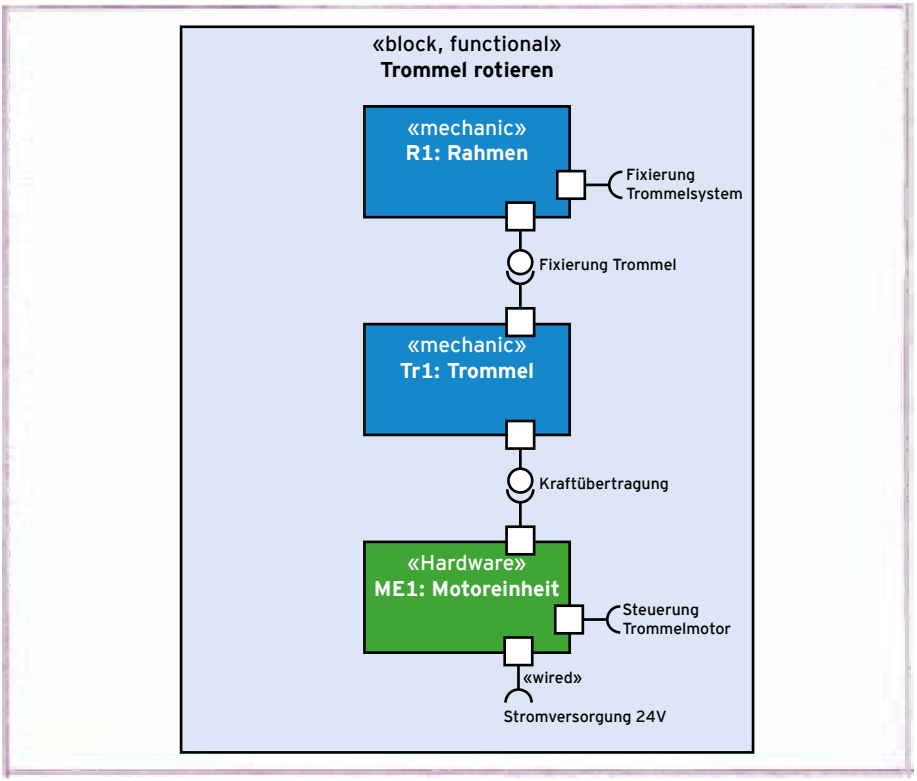


Abbildung 14: „Trommel rotieren“ in der funktionalen Sicht

4.3 Physikalische Sicht

In der physikalischen Sicht werden die Bestandteile eines Systems inklusive ihrer Schnittstellen und Anforderungen dargestellt.

Das Ziel dieser Sicht ist es, die Zerlegung des Systems bis hinunter zu den kleinsten, in der Systemarchitektur betrachteten Komponenten in einer Baumstruktur zu beschreiben. Damit ist die Einordnung einer Komponente eindeutig in eine, eventuell mehrstufige, Hierarchie gegeben. Dadurch bietet sich diese Sicht auch für die Strukturierung aller Dokumentationen von der Beschreibung der Systemanforderungen über die Systemarchitektur bis hin zu den Anforderungen an die Komponenten an (siehe Abschnitt 5.3).

In dem in dieser Broschüre gewählten Beispiel gehen wir von einer dreistufigen Hierarchie aus. Das System wird in Subsysteme zerlegt, die dann in gewerksspezifische Komponenten zerlegt werden. Für die Darstellung eines solchen Hierarchieübergangs werden in der SysML zwei Diagrammtypen benötigt: Ein Block-Definitions-Diagramm und ein Internal-Block-Diagramm.

In einem Block-Definitions-Diagramm werden die Bestandteile eines Betrachtungsgegenstandes definiert. In Abbildung 15 wird demnach beschrieben, aus welchen Typen von Subsystemen die Waschmaschine besteht.

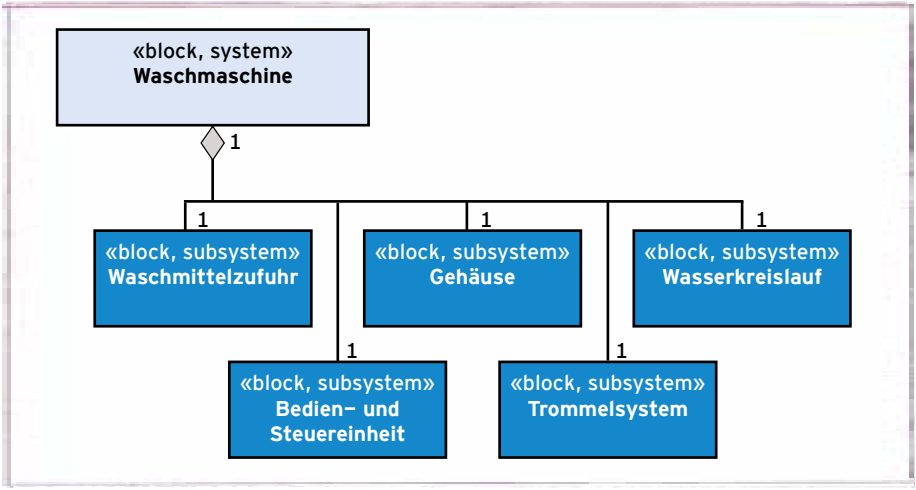


Abbildung 15: Block-Definitions-Diagramm der Waschmaschine

Dieses Diagramm bildet die Grundlage für das wesentlich interessantere Diagramm, das Internal-Block-Diagramm, das die folgenden, zusätzlichen Informationen beinhaltet:

- Interne Schnittstellen der Subsysteme untereinander
- Verantwortlichkeiten für die Realisierung der Systemschnittstellen

Abbildung 16 zeigt auszugsweise beide Informationen

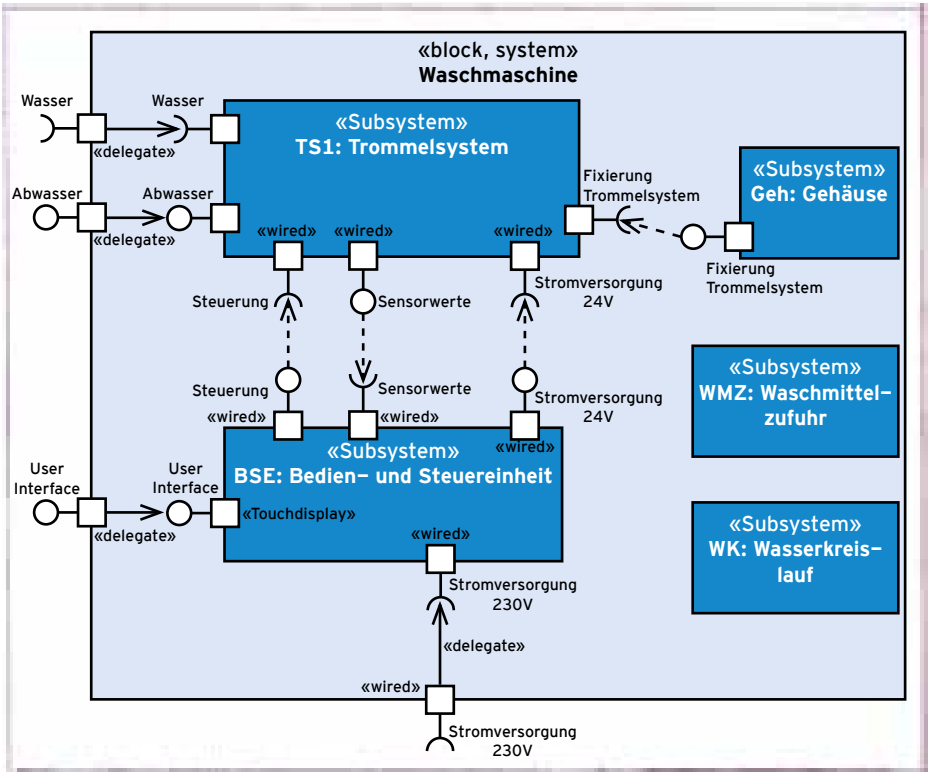


Abbildung 16: Internes Block-Diagramm der Waschmaschine

Die internen Schnittstellen resultieren aus der Verteilung der Systemanforderungen, die zum Teil in der funktionalen Sicht gefunden wurden (z.B. die Übergabe der Sensorwerte vom Trommelsystem zur Bedien- und Steuereinheit). Ihre Darstellung bzgl. der Richtung und der Aufteilung in Ports und Interfaces folgen den selben Prinzipien, die in Abschnitt 4.1 eingeführt wurden. Die <<delegate>>-Beziehung zeigt, welcher Teil des Systems für die Realisierung einer Systemschnittstelle zuständig ist. Diese Information ist

essenziell, da die Systemschnittstellen in letzter Instanz durch die gewerksspezifischen Komponenten realisiert werden müssen. Somit muss sich diese Delegation von Schnittstellen auch in der nächsten Verfeinerungsstufe fortsetzen.

Für die Darstellung des Aufbaus der Subsysteme folgen wir den gleichen Prinzipien wie bei dem System. Die Black-Box-Sicht der Subsysteme leitet sich dabei direkt aus dem Internal-Block-Diagramm der darüber liegenden Ebene (Abbildung 16) ab, weswegen sie im Allgemeinen nicht noch einmal dargestellt werden muss.

In Abbildung 17 sind die Bestandteile des Trommelsystems definiert, deren Zusammensetzung mit den Schnittstellen in Abbildung 18 gezeigt wird.

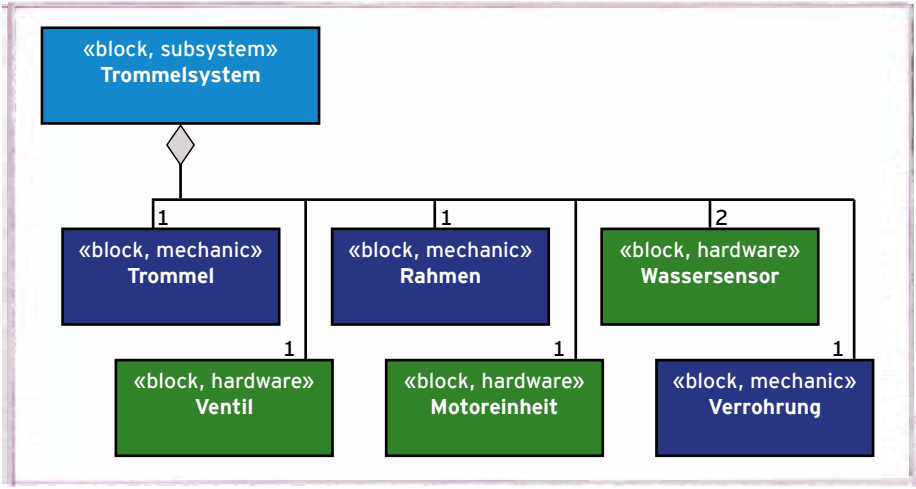


Abbildung 17: Block-Definitions-Diagramm für das Trommelsystem

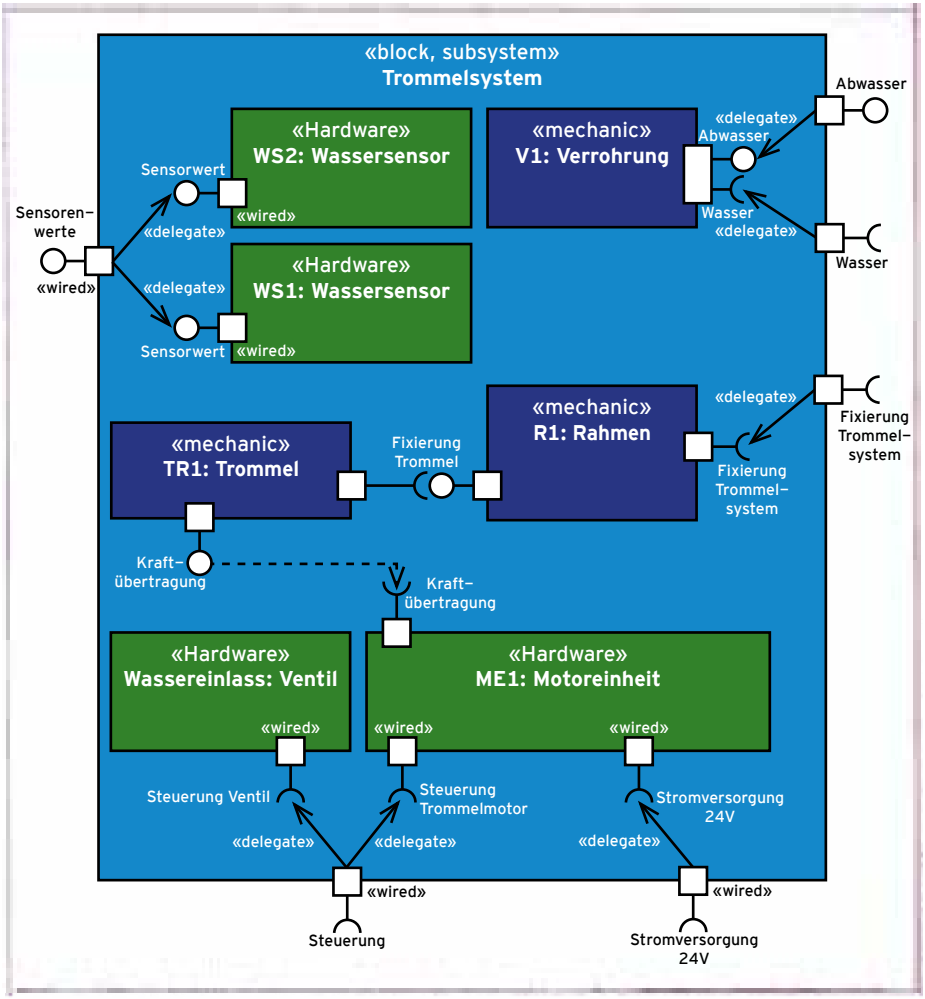


Abbildung 18: Internes Block-Diagramm für das Trommelsystem

In Abbildung 18 wird auch die Notwendigkeit der Unterscheidung in die beiden Diagrammtypen ersichtlich. In dem Block-Definitions-Diagramm wird definiert, dass das Trommelsystem aus zwei Wassersensoren besteht. Hingegen zeigt ein Internal-Block-Diagramm den Unterschied zwischen den beiden Verwendungen (In Abbildung 18 als „WS1“ und „WS2“).

Als letzte Information innerhalb dieser Sicht wird noch beschrieben, welche Anforderungen an die Komponenten bestehen, und aus welchen Systemanforderungen sie folgen.
In Abbildung 19 sind diese Informationen für eine Anforderung an die Motoreinheit dargestellt, die aus der in Abschnitt 4.2 betrachteten Funktionalität des Trommeldrehens folgt.

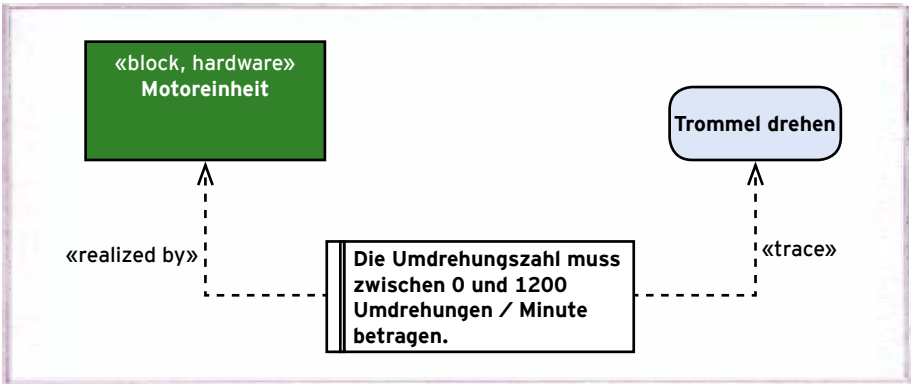


Abbildung 19: Darstellung der Komponentenanforderungen

Die in Abbildung 19 aufgezeigte Verfolgbarkeit zu den Systemanforderungen unterstützt zwei wichtige Tätigkeiten. Zum einen wird der dargestellte Architekturschritt an sich unterstützt, indem Sie überprüfen können, ob Sie alle (wichtigen oder kritischen) Funktionen des Systems auf die Komponenten verteilt haben. Zum Anderen werden Sie in der späteren Entwicklung diese Verfolgbarkeit bei einer Impact-Analyse ausnutzen können. Falls sich eine Systemanforderung ändert, können Sie die Auswirkungen auf die daraus abgeleiteten Komponentenanforderungen analysieren.

4.4 Vorgehen in der Architektur

Wie in den vorigen Abschnitten erwähnt, wird sich die Architektur nach der Realisierung der Systemanforderungen, speziell der funktionalen Anforderungen, richten. Damit ergibt sich insgesamt der folgende Ablauf für das Vorgehen in der Systemarchitektur.

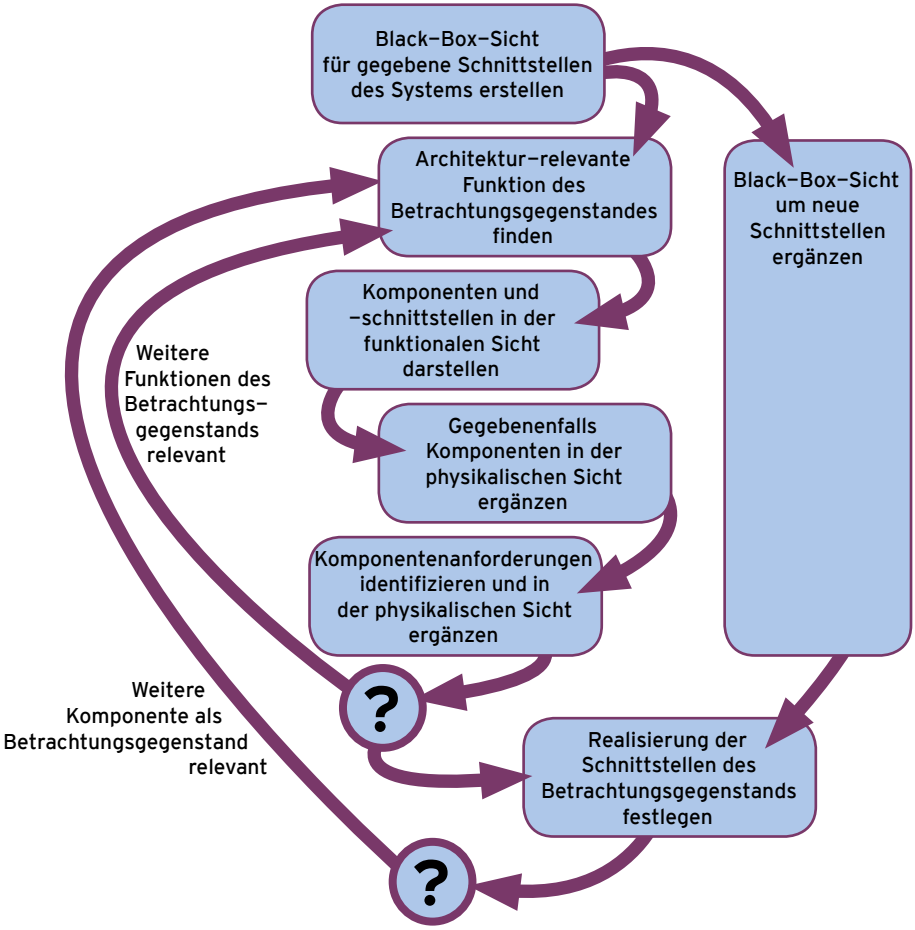


Abbildung 20: Vorgehen in der Systemarchitektur

In Abbildung 20 sind zwei Schleifen dargestellt, die die beiden Freiheitsgrade repräsentieren, die Sie als Systemarchitekt haben. Die innere Schleife stellt dar, ob Sie weitere Funktionen in der funktionalen Sicht betrachten wollen. Die äußere Schleife lässt Ihnen die Möglichkeit, eine zuvor gefundene Komponente weiter zu zerlegen.

Die vollständige Durchführung aller Tätigkeiten aus Abbildung 20 für alle Ebenen der Systemzerlegung wird im Allgemeinen nicht notwendig sein. Wenn Sie also z.B. sicher sind, ein Subsystem weiter in seine Bestandteile zu zerlegen, so müssen Sie nicht zusätzlich die Systemanforderungen auf dieses Subsystem verteilen. Überspringen Sie für diese Tätigkeit dieses Subsystem und verteilen Sie die Systemanforderungen direkt auf dessen Komponenten (siehe auch Abbildung 19). Sie sollten jedoch für jeden gefundenen Betrachtungsgegenstand die Schnittstellen modellieren, wobei diese von der Zerlegung auf der darüber liegenden Ebene schon zum Teil vorgegeben werden.

5. Weitere Aspekte des Systems-Engineerings

5.1 Weitere Tätigkeiten in der Entwicklung

Wir haben in dieser Broschüre mehr Wert auf den linken, den konstruktiven Ast des Vs gelegt, da hier die Weichen für eine erfolgreiche Systementwicklung gestellt werden. Nichtsdestotrotz ist der Systems-Engineer auch an weiteren Tätigkeiten im Systementwicklungsprozess beteiligt.

Die erste Aufgabe nach der Entwicklung der Komponenten besteht in der Integration dieser Komponenten zu den Subsystemen bzw. zu dem System. Hierbei muss sichergestellt werden, dass die entsprechenden Betrachtungsgegenstände rechtzeitig für die nachfolgenden Tests in der benötigten Qualität zur Verfügung gestellt werden. Die eigentliche Integration wird häufig von den einzelnen Gewerken durchgeführt. So werden zum Beispiel die Software-Entwickler für die Integration ihrer Software mit der entsprechenden Hardware sorgen, und die Mechanik-Abteilung wird diese Hardware in ein Gehäuse einfügen.

Auch bei den Tätigkeiten, die sich mit dem Testen der Entwicklungsgegenstände beschäftigen, ist der Systems-Engineer nur indirekt beteiligt. Er muss jedoch die Qualität der folgenden Ergebnisse aus diesen Tätigkeiten sicherstellen:

- **Testkonzept:** Wird der Entwicklungsgegenstand durch das im Testkonzept definierte Vorgehen ausreichend getestet?
- **Testfälle:** Erlauben die definierten Testfälle, die im Testkonzept definierten Ziele zu erreichen?
- **Testdurchführung:** Wurden alle Tests entsprechend erfolgreich durchgeführt?

Bei der Testdurchführung können Testfälle natürlich auch mit einem negativen Ergebnis beendet werden. In diesem Fall wird der Systems-Engineer im Allgemeinen die erste Anlaufstelle sein, um zu entscheiden, welche Komponenten für das negative Testergebnis verantwortlich waren und dementsprechend informiert werden müssen.

Neben diesen integrativen und testenden Tätigkeiten müssen in einem Systems-Engineering-Prozess noch Tätigkeiten definiert werden, die die Weitergabe der getroffenen Entscheidungen sicherstellen. Dazu können nach den Analyse- und Architekturschritten Review-Schritte eingeführt werden, die z.B. durch die Technik eines Walk-Throughs [Rupp14, Kapitel 14] alle Informationen an die betroffenen Entwickler weitergeben.

Diese Review-Schritte sollten jedoch nur eine Absicherung darstellen. Prinzipiell sollten an allen Entwicklungsschritten die Entwickler beteiligt sein, die in den nachfolgenden Entwicklungsschritten verantwortlich sind. Im Einzelnen bedeutet dies:

- **Analyse:** Integrieren Sie hier den Systemarchitekten, um in den Anforderungen direkt auf kritische Stellen für die Realisierung hinweisen zu können.
- **Architektur:** Hier sollten die Verantwortlichen an den Entscheidungen beteiligt sein, die die in der Architektur gefundenen Komponenten verantworten. Nur sie haben das Detailwissen, um die Architekturentscheidungen fundiert treffen zu können.

Die letzten hier dargestellten Tätigkeiten beschäftigen sich mit der Bewertung des Systems-Engineerings und dem Gewinn, den es für den gesamten Entwicklungsprozess bringt. Dazu erscheint eine Metrik über die Änderungsanträge (Change Requests) am vielversprechendsten. Verfolgen Sie, auch über vergleichbare Projekte hinweg, ob sich die Anzahl der Änderungen verringern. Hierbei sollten Sie zwischen zwei Arten von Verursachern für die Änderungen unterscheiden:

- Änderungen durch die Stakeholder
- Änderungen durch die Realisierung

Die erste Ursache kann nicht vollständig durch eine verbesserte Analysephase ausgeschlossen werden, da sich Änderungswünsche bei den Stakeholdern immer ergeben können. Demnach sollte durch das Systems-Engineering gerade die zweite Ursache von Änderungen minimiert werden. Damit haben Sie gezeigt, dass die frühe und komponentenübergreifende Sichtweise auf das System einen Vorteil für die Entwicklung bringt.

5.2 Einordnung von Systems-Engineering in den Lebenszyklus des Systems

Bisher haben wir über die Beteiligung des Systems-Engineerings an einem Entwicklungsprozess gesprochen, ohne diesen jedoch in einen übergeordneten Prozess einzuordnen, der alle Phasen im Lebenszyklus eines Systems beschreibt. Solche Prozesse sind im Allgemeinen sehr unternehmensspezifisch ausgestaltet, ähneln sich jedoch auf einer abstrakten Ebene. In den frühen Phasen ist aber immer wieder eine Unterscheidung zwischen den Prozessen bei einem OEM (Original Equipment Manufacturer) und einem Zulieferer zu erkennen.

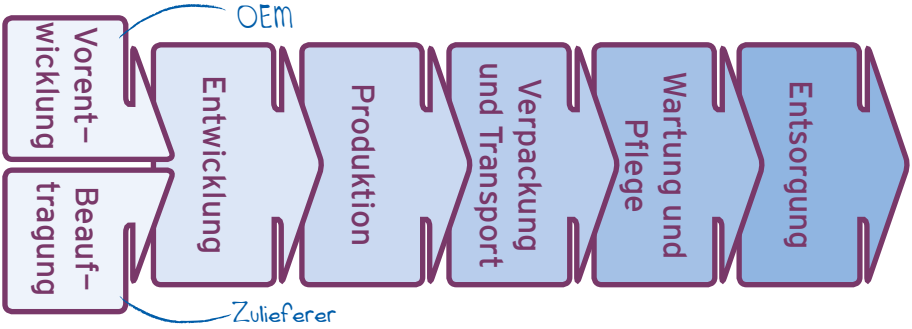


Abbildung 21: Abstrakter Lebenszyklus eines Systems

Bei vielen OEMs wird zunächst eine Vorentwicklung durchgeführt, während bei einem Zulieferer am Anfang eines Projekts die Beauftragung, also die Angebotsphase steht. Bezüglich der Beteiligung des Systems-Engineers an diesen beiden Phasen gilt das Folgende:

Vorentwicklung: Hier ist die Aufgabe des Systems-Engineers, aus dem Lastenheft die Anforderungen soweit zu analysieren, bis die kritischen Themen identifiziert werden können. Für diese Themen wird dann eine detailliertere Analyse durchgeführt. Sie dient als Startpunkt für eine vorläufige Systemarchitektur, um die an den kritischen Themen beteiligten Komponenten zu identifizieren und Anforderungen an diese zu stellen.

Bei diesem Schritt sollten auf jeden Fall die Komponentenentwickler beteiligt sein. Nach der Entwicklung der Komponenten wird der Systems-Engineer die Konformität der entstandenen Komponenten bzw. ihres Zusammenbaus zu den Anforderungen überprüfen. Falls in Vorentwicklung für unterschiedliche Themen verschiedene Konzepte getrennt voneinander entstanden sind, ist es zusätzlich notwendig, die Möglichkeit zur Integration dieser Konzepte zu bewerten.

Beauftragung: In dieser Phase wird im Allgemeinen der linke Ast des Vs durchlaufen, um ein abschätzbares Realisierungskonzept zu erhalten. Dabei müssen aber nicht alle im Prozess geforderten Artefakte erzeugt werden. Gleichzeitig wird das Lastenheft kommentiert, um zusammen mit dem Realisierungskonzept als Grundlage für ein Angebot zu dienen. Der Systems-Engineer sollte neben den Komponentenentwicklern an allen Schritten beteiligt sein, da er sich verantwortlich für alle Artefakte fühlen sollte, die aus Sicht der Entwicklung für ein Angebot geliefert werden müssen. Es können sein:

- Kommentiertes Lastenheft mit Anpassungen und Annahmen
- Realisierungskonzept und evtl. Alternativen
- Bill of Material
- Risikoliste

Die Beteiligung des Systems-Engineers an der Entwicklung haben wir in den vorigen Kapiteln beschrieben. Für die hinteren drei Phasen im Lebenszyklus gilt, dass der Systems-Engineer die aus diesen Phasen folgenden Anforderungen an das System mit berücksichtigen muss.

Zusätzlich wird er, gerade aus der Wartung und Pflege, mit Änderungswünschen konfrontiert werden. Seine Aufgabe ist es, diese Wünsche entgegenzunehmen, ihre Auswirkungen zu analysieren und die daraus folgenden Änderungen in den Anforderungen an die Komponenten zu definieren. Dabei hilft ihm die im folgenden Abschnitt eingeführte Produktbeschreibung.

5.3 Verwendung einer Produktbeschreibung

Für die Spezifikation eines Systems, also die Dokumentation der Systemanforderungen, existieren in der Literatur zahlreiche Vorlagen ([Robertson12], [Rupp14, Kapitel 19]). Wir wollen an dieser Stelle jedoch einen Schritt weitergehen.

Am Ende der Entwicklung haben Sie ein System entwickelt, das, hoffentlich, alle im Vorfeld gestellten Systemanforderungen erfüllt. Falls Sie wie vorgeschlagen vorgegangen sind, werden Sie auch Anforderungen für die Komponenten des Systems dokumentiert haben. Häufig tritt der Fall ein, dass das System und einzelne Komponenten zu mehr in der Lage sind, als die Anforderungen verlangen. Somit unterscheidet sich das, was das System oder eine Komponente kann, von dem, was das System oder die Komponente können sollte.

Es wäre also wünschenswert, am Ende der Entwicklung eine Beschreibung des Systems und seiner Komponenten zu besitzen, die die tatsächlichen Eigenschaften und nicht die Anforderungen beschreibt. Dabei sollte diese Beschreibung nicht nur die Eigenschaften des Systems, sondern auch die wichtigsten Aussagen über die Realisierung beinhalten.
Ein solcher Übergang von Anforderungen zu Eigenschaften kann zahlreiche Tätigkeiten nach der Entwicklung unterstützen:

- In der **Wartung** können Fehlerfälle leichter nachvollzogen werden, da sich die Beschreibung auf die tatsächliche Realisierung bezieht.
- Im Rahmen der **Pflege** des Systems werden Änderungen und Erweiterungen benötigt, deren Auswirkungen leichter analysiert werden können.
- Die Erstellung weiterer **Dokumentationen** (Benutzer-, Wartungshandbuch, etc.) setzt auf das tatsächliche und nicht auf das geforderte Verhalten auf.
- Die **Nachfolgeprojekte** mit einem vergleichbaren Inhalt können die Ergebnisse aus dem Projekt verwenden und Änderungen an dessen Ergebnissen definieren.

Gerade der letzte Punkt birgt auf Seite eines Zulieferers ein großes Potenzial. Dies veranschaulicht das folgende Beispiel.

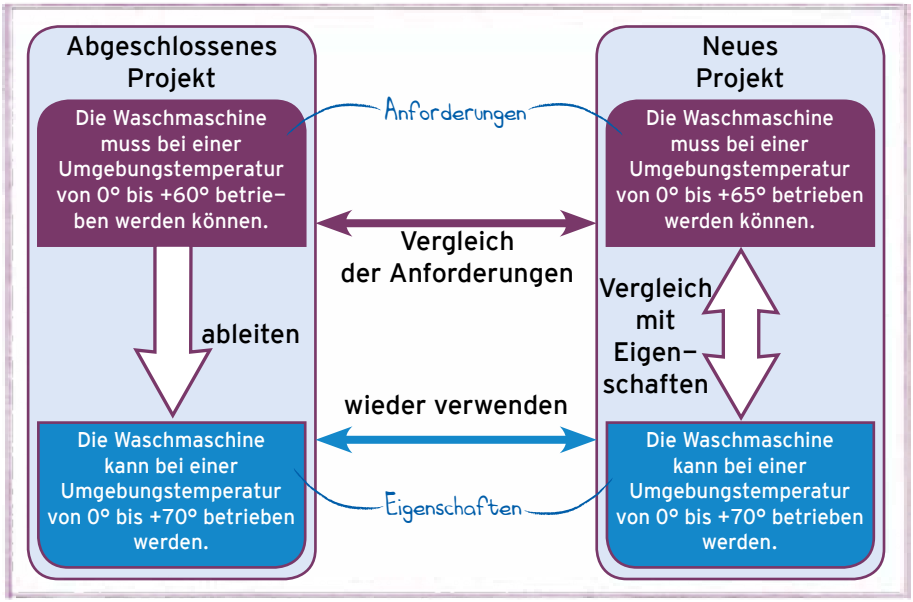


Abbildung 22: Wiederverwendung von Eigenschaften

Der Vergleich der „alten" Anforderung mit der neuen Anforderung würde die Notwendigkeit von Anpassungen liefern. Wird jedoch die tatsächliche Eigenschaft des Systems für den Vergleich herangezogen, so erfüllt das aktuelle Produkt schon die neu geforderten Anforderungen.

Die Struktur einer solchen Produktbeschreibung ähnelt der Struktur, die Sie bei der Dokumentation der Anforderungen an das System und seiner Komponenten verwenden würden. Gegenüber einer reinen Anforderungsbeschreibung müssen wir sie jedoch erweitern, um die Realisierung eines Betrachtungsgegenstandes auf beliebiger Ebene (und damit die Anforderungen / Eigenschaften an die Komponenten) beschreiben zu können.

Dabei können wir uns den Umstand zu Nutze machen, dass für alle Ebenen von Betrachtungsgegenständen immer wieder ähnliche Typen von Anforderungen / Eigenschaften beschrieben werden müssen. Es entsteht eine übergreifende Struktur, die sich auf allen Ebenen immer wieder aus zwei Teilen zusammensetzt: Ein Teil zur Beschreibung der Anforderungen / Eigenschaften des jeweiligen Betrachtungsgegenstands und ein Teil zur Beschreibung seiner Bestandteile. Diese hierarchische, sich immer wieder ähnelnde Struktur hilft dem Leser bei dem Verständnis des gesamten Systems.

1 Einleitung
2 Definitionen
3 Einsatzbedingungen
4 Schnittstellen
5 Betriebsfunktionen der Waschmaschine
6 Weitere Eigenschaften
7 Anforderungen an weitere Tätigkeiten
8 Zusätzliche Artifakte
9 Realisierung der Waschmaschine
  9.1 Systemarchitektur
  9.2 Subsystem Trommelsystem
  9.3 Subsystem Bedien– und Steuereinheit
  ...

Abbildung 23: Exemplarische Produktstruktur für die Waschmaschine

Abbildung 23 gibt ein Beispiel für eine umfassende Produktbeschreibung vor.
Kapitel 1 bis 8 beschreiben die Anforderungen / Eigenschaften des Systems, während ab Kapitel 9 dessen Realisierung beschrieben wird. Die Untergliederung in Kapitel 9 ergibt sich aus der Struktur des Systems (siehe Abbildung 15), wobei für jedes Subsystem (und jede Komponente auf tieferer Ebene) potenziell die Gliederung auf Systemebene (Kapitel 1 bis 9) eingesetzt werden kann.

Diese Struktur wird auf dem linken Ast des Vs top-down mit Anforderungen und Architekturergebnissen gefüllt. Nach der Realisierung und dem Test der Komponenten und der darüber liegenden Ebenen können die Anforderungen dann bottom-up durch die tatsächlichen Eigenschaften ersetzt werden.

5.4 Die weichen Herausforderungen

Wenn Sie diese Broschüre bis hierhin aufmerksam gelesen haben, kann Ihnen der Gedanke kommen, dass Systems-Engineering eine Disziplin ist, die nur von Personen durchgeführt werden kann, die Spezialisten in allen betroffenen Disziplinen und Gewerken sind. Das wäre natürlich wünschenswert, ist jedoch nicht realistisch.

Zum Einen haben wir in dieser Broschüre die Aufgaben der Rolle Systems-Engineer beschrieben. Diese Rolle kann in einem Projekt natürlich von mehreren Personen wahrgenommen werden. Wahrscheinlich ist dies auch schon in Ihren Projekten der Fall, ohne dass die Rolle explizit benannt ist, und die Tätigkeiten werden von bekannten Rollen (z.B. von einem Mechanik-Entwickler) durchgeführt. Ein definierter Systems-Engineering-Prozess mit der entsprechenden Rolle soll dafür sorgen, dass die benötigten Tätigkeiten aus System-Ebene bewusst und zum richtigen Zeitpunkt unter Einbeziehung der betroffenen Personen durchgeführt werden.

Zum Anderen benötigt ein Systems-Engineer nicht das Spezialwissen aus den verschiedenen Disziplinen im Detail. Er muss jedoch genügend Wissen haben, um Entscheidungen von den richtigen Personen einzufordern, sie in Zusammenhang zu anderen Entscheidungen zu setzen und sie bei Bedarf zu weiteren Beteiligten zu kommunizieren.

Damit können die Fähigkeiten eines guten Systems-Engineers in drei Ebenen eingeteilt werden.

- **Inhaltlich:** Ein Systems-Engineer sollte Wissen in der Anwendungsdomäne besitzen.
- **Methodisch:** Er sollte die Methoden und Techniken, die in den verschiedenen Tätigkeiten zum Einsatz kommen, kennen und können.
- **Persönlich:** Da Systems-Engineering viel mit Kommunikation zu tun hat, sollte er teamorientiert und konfliktbereit sein. Weiterhin benötigt er Abstraktionsvermögen und sollte Vertrauen zu Entscheidungen von Dritten haben.

Falls Sie solche Personen in Ihrem Unternehmen kennen, haben Sie gute Kandidaten für die Rolle des Systems-Engineers gefunden. Falls nicht, so können Sie diese Rolle auch mit mehreren Personen, die unterschiedliche Skills besitzen, besetzen. So kann zum Beispiel ein interner Mitarbeiter mit dem benötigten inhaltlichen Wissen durch eine Person von außen, die das methodische Wissen beisteuern kann, unterstützt werden.
Die Rolle des Systems-Engineer und den entsprechenden Prozess einzuführen, ist aber eine Aufgabe, die nicht unterschätzt werden darf. Ein Grund hierfür kann die Tatsache

sein, dass viel wertvolles Wissen zentralisiert und dokumentiert wird. Dies kann auf Widerstand stoßen, da die bisherigen Wissensträger das Gefühl bekommen, entbehrlich zu werden. Ein weiterer Grund kann sein, dass Aufwand in frühe Phasen eines Projekts investiert wird, der sich erst in späteren Phasen bezahlt macht.

Desweiteren gelten für die Einführung eines Systems-Engineering-Prozesses die gleichen Aussagen (und Herausforderungen) wie bei jedem Einführungsprozess. Ein Beispiel ist in [Rupp14, Kapitel 22] gegeben.

6. Quellenverzeichnis

[ALM16] www.ireb.org/de/downloads/, Handbuch zum CPRE Advanced Level Modeling, letzter Zugriff: September 2017

[Akao92] Yoji Akao: QFD – Quality Function Deployment, Verlag Moderne Industrie, Landsberg 1992

[Barwise05] Jon Barwise: Sprache, Beweis und Logik. Band I: Aussagen- und Prädikatenlogik, Mentis Verlag, 2005

[Cockburn00] Alistair Cockburn: Writing Effective Use Cases, 1. Auflage, Addison-Wesley Professional, 2000

[Krisch14] Krisch, Jennifer. „Kontextbasierte lexikalische Kontrolle von Anforderungsdokumenten." Proceedings of the XVI EURALEX International Congress, Bozen. 2014.

[Incose15] Incose: Systems Engineering Handbook, A Guide for System Life Cycle Processes and Activities 4. Auflage, Wiley John and Sons, 2015

[Pfeufer14] Hans-Joachim Pfeufer: FMEA – Fehler-Möglichkeits- und Einfluss-Analyse, Carl Hanser Verlag, 2014

[Pflüger14] Andre Pflüger: Modellgetriebene Validierung von System-Architekturen gegen architekturrelevante Anforderungen: Ein Ansatz zur Validierung mit Hilfe von Simulationen, University of Bamberg Press, 2014

[Pohl et al.12] Klaus Pohl, Harald Hönninger, Reinhold Achatz, Manfred Broy: Model-Based Engineering of Embedded Systems, Springer Verlag, 2012

[Robertson12] Robertson, S.; Robertson J.: Mastering the Requirements Process. Getting Requirements Right. 2nd Edition. Pearson, 2012.

[Rupp14] Chris Rupp: Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil, 6. Auflage, Hanser Verlag, 2014

[RuQu12] Chris Rupp, Stefan Queins: UML Glasklar – Praxiswissen für die UML-Modellierung, 4. Auflage, Hanser Verlag, 2012

[Starke14] Gernot Starke: Effektive Softwarearchitekturen – Ein praktischer Leitfaden, 6. Auflage, Hanser Verlag, 2014

[SysML] www.omgsysml.org, Offizielle Homepage der OMG zur SysML, letzter Zugriff: September 2017

WISSEN for free

(Englisch)

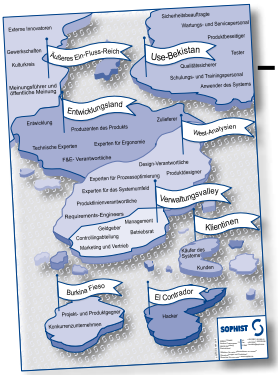
– [Broschüren]

Geballtes Wissen von den SOPHISTen

Kostenfrei zu bestellen unter:

www.sophist.de/wissen-for-free

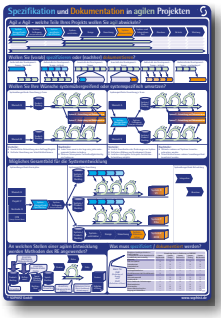
[Plakate]

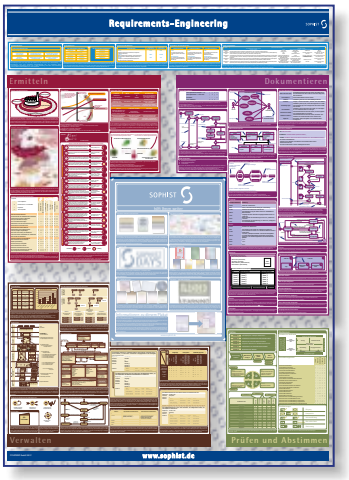


Stakeholderlandkarte
(Deutsch & Englisch)

Das legendäre SOPHIST UML2 Plakat

Das neue modulare SOPHIST Requirements-Engineering Plakat





Verschiedene „Kerne“ zum modifizieren!

» Grau is alle Theorie, maßgebend is auffen Platz. «

Adi Preißler

Genau. Deshalb müssen sich Manschaften auch auf den Platz optimal vorbereiten. Am Besten geht das mit Profitrnern wie den **SOPHIST**en.

Wir coachen Kunden wie Siemens, Daimler, die Bundesagentur für Arbeit, Roche, Volkswagen, Robert Bosch, Lufthansa Systems, EnBW, VHV Versicherung und viele andere

- Anforderungen an Systeme zu hinterfragen, zu verstehen und zu dokumentieren,
- Wiedersprüche und Redundanzen in Modellen zu erkennen,
- Notationen richtig anzuwenden und perfekt zu modellieren,
- eine Architektur zu wählen, die länger als bis zur Abnahme aktuell ist,
- agil und angepasst zu arbeiten.

Das Ganze bieten wir in Form von Beratung, Coaching, Trainings und Vorträgen.

Wir bewahren Sie vor der Abseitsfalle.
Sprechen Sie mit uns über Ihren Trainingsplan unter:
+49 (0) 911 40 900 - 0

Oder schreiben Sie uns eine E-Mail an heureka@sophist.de

Das Trainingsformat für schnelleren Lernerfolg!

Sie möchten die Prüfung zum CPRE-Foundation Level ablegen und dabei schnell und effektiv das benötigte Wissen erlernen?

Dann buchen Sie doch einfach unser BLENDED LEARNING-Training.

Certified Professional for Requirements Engineering - Foundation Level

- Lernen Sie die theoretischen Inhalte für die Zertifizierung und testen Sie Ihr Wissen ortsungebunden mit Hilfe unserer E-Learning Platform im Vorfeld.
- An den 2 Tagen der Präsenzveranstaltung steht die Vorbereitung auf die Prüfung im Mittelpunkt.
- Der Trainer geht gezielt auf Ihre Ihre Ergebnisse der E-Learning-Platform und Ihren Fragen mit viel Praxisbezug ein.
- Sie erhalten zudem ausführliche Seminarunterlagen, das Buch „Basiswissen Requirements Engineering“ und vieles mehr!

Hier finden Sie weitere Informationen zu diesem Training:
www.sophist.de/bl-cpre-fl/

Systems-Engineering

SOPHIST beschäftigt sich nunmehr seit über 20 Jahren mit Anforderungen, also der Ein- und Durchführung von gutem Requirements-Engineering. Über die Jahre hinweg haben wir darin viele Erfahrungen aus unterschiedlichsten Anwendungsgebieten gesammelt. Dabei haben wir festgestellt, dass neben der Ermittlung und Dokumentation von Anforderungen, die sich an den Betrieb des betrachteten Produkts richten, die Anforderungen aus den anderen Lebenszyklen des Produkts immer mehr in den Fokus rücken.

Eine adäquate Betrachtung der Anforderungen wird in der reinen Softwareentwicklung mit agilen Ansätzen gut unterstützt. Diese Ansätze funktionieren in einem komplexen System, das neben Software auch aus elektrischen, elektronischen oder mechanischen Anteilen besteht, allerdings nur bedingt. An dieser Stelle setzt **Systems-Engineering** an, das in einem interdisziplinären Ansatz sowohl das gewünschte Produkt als Gesamtsystem, als auch alle Bereiche in seinem Lebenszyklus betrachtet. Darüber hinaus ist die Architektur zur Aufteilung des Produkts in seine Komponenten ein integraler Bestandteil des Systems-Engineerings.

Die Investition von Aufwänden in die frühen Phasen der Systementwicklung hilft Ihnen, Risiken zu minimieren und späten Überraschungen vorzubeugen.