Critical software

Modellbasierte Fehler- und Safety-Analysen komplexer Systeme

Komplexität beherrschen durch modulare und digitalisierte Prozesse

criticalsoftware.com
info@criticalsoftware.com

Der Schauplatz für modellbasierte Fehler-Analysen

Mehr Komponenten, mehr Funktionen, mehr Interaktionen – wie lässt sich die zunehmende System-Komplexität unter Kontrolle halten?

Die Entwicklung vielschichtiger kritischer Systeme fordert heute mehr denn je einen guten Durchblick aller beteiligten Komponenten und ein gutes Verständnis der Zusammenhänge, auch über die Grenzen des eigentlichen Produkts hinaus. Seit Jahrzehnten helfen Modellbildungen und Simulationen, komplexe Wechselwirkungen besser zu beschreiben. Insbesondere in den letzten 10 Jahren hat der Begriff des „Modelbased System-Engineering" (MBSE) bei der System- und Funktionsentwicklung verstärkt Einzug gehalten, oft in Verbindung mit der Sprache SysML („Systems Modelling Language").

Dennoch bestimmen auch heute in vielen Entwicklungsabteilungen noch textbasierte Dokumente das Bild, angereichert mit flachen Grafiken und verwaltet in lokalen Dateien und Verzeichnissen. Oftmals erstellt auf der Basis klassischer Bürosoftware-Tools, kommen diese Lösungen bereits bei der „nominalen" Funktionsentwicklung, deren Verifikation und eindeutigen Nachvollziehbarkeit an ihre Grenzen.

Besonders kritisch wird es, wenn das nicht-nominale Verhalten zu betrachten ist - bei Analysen, welche die Untersuchung und Nachweise der funktionalen Sicherheit und Zuverlässigkeit eines Systems zum Ziel haben, speziell in Situationen bei denen letztlich Menschenleben auf dem Spiel stehen können.

"All models are wrong but some are useful."
George E.P. Box

Unterschiedlichste Bauteil-Arten mit individuellen Fehlermodi, jede x-fach verwendet, irreguläre Wechselwirkungen, sich umstrukturierende und rekonfigurierende Topologien, ein diffiziles Zusammenspiel von Software und Hardware, starke Vernetzung und Interaktion mit anderen Systemen, der Umwelt und menschlichen Akteuren, „hidden links" – im Ergebnis eine extreme Fülle von Betriebszuständen, die bei der Risiko-Analyse zu erfassen und zu bewerten sind.

Wie können Sie als System-Verantwortlicher sicherstellen, dass - bei solch einer Vielfalt von Text-Reports aus der Feder vieler verschiedener Verfasser - Ihre Safety-Bewertungen das reale Gefährdungspotential hinreichend abbilden? - lückenlos, widerspruchsfrei und vor allem: „im Falle des Falles" selbsterklärend und leicht nachvollziehbar, auch nach Jahren, wenn der damalige Autor bereits weitergezogen ist? – Wird zudem das einmal erarbeitete wertvolle Knowhow zu Defekten und deren Auswirkungen so eingesetzt, dass es auch für den Betreiber des Systems Vorteile bringt, z.B. im Sinne hoher Zuverlässigkeit, guter Diagnose und Fehlersuche oder zielführender integrierter Wartungskonzepte?

Interessanterweise gibt es auch für solche Analysen bezüglich Bauteil-Fehlern bereits seit vielen Jahren etablierte Ansätze und kommerzielle Lösungen, um insbesondere dieser nochmals gesteigerten Komplexität modellbasiert gerecht zu werden. Hier spricht man von „Modelbased Safety-Analysis" (MBSA).

In diesem Whitepaper werden wir die Rolle „klassischer" – oder analoger -Modellbildung bei Safety-Analysen erkunden und anschließend auf einige tool-gestützte, digitale MBSA-Ansätze eingehen.

© Critical Software. All rights reserved.

Einige Begriffe zu Safety-Analysen

Bevor wir uns der modellbasierten Herangehensweise an das Thema Fehleranalyse nähern, lassen Sie uns zunächst einige Begriffe klären::

Fehlermodell: Eine gedankliche, physikalische, mathematische, logische oder anderweitige Beschreibung, wie sich ein definiertes System-Element bzw. eine Komponente im Fehler-Fall vermutlich verhält. Pro Element sind auch mehrere Fehler-Fälle (Fehler-Modi) denkbar mit jeweils eigener Verhaltensbeschreibung. Es ist wichtig zu verstehen, dass die Beschreibung der Auswirkung im System-Verbund nicht Teil des Fehlermodells ist.

Fehler/Fault: Ein originärer Defekt an einem definierten Modul oder Bauteil innerhalb eines Systems (rot in Abbildung 1). Wird das Modul bzw. Bauteil repariert oder ausgetauscht, sollte dieser Fehler verschwinden.

Abhängiger Fehler/Error: Eine Diskrepanz zwischen dem beabsichtigten Verhalten eines System-Elements und seinem tatsächlichen Verhalten innerhalb der Systemgrenze (gelb in Abbildung 1). Die Ursache dafür liegt in anderen Elementen, nicht in der Komponente selbst.

Versagen/Failure: Eine System-Funktion zeigt ein Verhalten, das im Widerspruch zu ihrer Spezifikation steht (orange in Abbildung 1). Ein interner Fehler führt nicht zwangsläufig zu einem Funktionsversagen. Falls z.B. Fehlertoleranztechniken oder Redundanzen aktiv sind, kann die Gesamt-Funktionalität an der System-Grenze sehr wohl der Spezifikation entsprechen.

Symptom: Eine auf höherer System-Ebene bzw. an der System-Grenze wahrgenommene Verhaltensbeobachtung, die Anlass gibt, über bestimmte Zusammenhänge und Zustände im Inneren des Systems zu schlussfolgern (orange in Abbildung 1). Die Beobachtung kann sowohl der Spezifikation entsprechen als auch im Konflikt mit ihr stehen.

Kern-Ursache/Root-Cause: Eine mögliche system-interne Erklärung für ein Symptom, hier insbesondere ein möglicher interner Fehler-Zustand (aus einem oder einer Kombination mehrerer Faults), dessen Eintritt einen plausiblen Grund für das beobachtete Verhalten auf höherer Ebene bzw. das Symptom-Bild liefern kann.

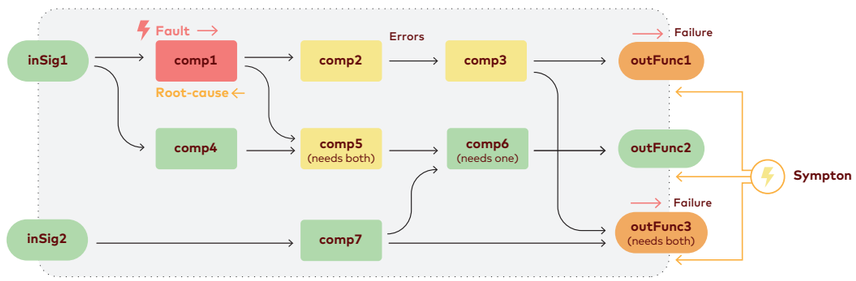


Abbildung 1 Exemplarisches vernetztes System mit Fehler und Auswirkungen auf Funktionen

© **Critical Software.** All rights reserved.

Der „klassische“ analoge Weg

Begriffe wie „Modellierung“ oder „modellbasierte Entwicklung“ veranlassen häufig Assoziationen an fortschrittliche, computergestützte Virtualisierungsmethoden mit zeitgemäßen Software-Tools, eingesetzt, um komplexe eingebettete System zu entwickeln.

Tatsächlich lässt sich der Begriff aber viel weiter verstehen und auch für sicherheits-kritische Analysen gibt es einige klassische analoge Methoden der Modellierung. Hier eine kleine Auswahl:

Abbildung 2 Britischer hölzerner Pferde-Simulator zum Test von Abläufen und Prozess-Fehlern im Kampf, vor 1915

https://commons.wikimedia.org/wiki/File:Horse_simulator_WWI.jpg

A) PHYSIKALISCHE MODELLIERUNG

Eine der vermutlich ältesten Modellierungs-Methoden überhaupt ist es, reale Zusammenhänge an einem physikalischen Nachbau zu reproduzieren. An diesem Abbild lassen sich – oft in angepasstem Maßstab oder sonstwie vereinfachter Form - in kontrollierter Umgebung Experimente durchführen, die im echten Leben so nicht machbar sind. Architektur-Modelle von Gebäuden oder Labor-Aufbauten technischer Anlagen sind typische Beispiele physikalischer Modellierung, um statische oder dynamische Verhaltensweisen zu untersuchen und aus den Erkenntnissen Rückschlüsse auf das evtl. erst noch zu bauende reale Gegenstück und mögliche Optimierungen zu ziehen.

Prinzipiell können auf diese Weise auch die Auswirkungen von nicht-nominalem Verhalten getestet werden, durch „Injizieren“ bestimmter Fehler in den Nachbau. Hierfür muss die Fehlerart bzgl. ihres grundsätzlichen Verhaltens bekannt sein und auch die möglichen Effekte bereits vorab bedacht werden. So dürften sich z.B. in einem Test-Aufbau der elektrischen Versorgung in einem Flugzeug Fehler wie Leitungsunterbrechungen und deren Auswirkungen relativ leicht testen lassen. Schwieriger würde es – in diesem Beispiel - bereits bei der absichtlichen Provokation von Kurzschlüssen zwischen verschiedenen Leitungen, die ggf. zur Zerstörung von Modellteilen (beispielsweise realen Subsystem-Modulen oder Boards) bis zum Verlust des kompletten Aufbaus führen könnten.

Es zeigt sich: Auch wenn die Verwendung eins physikalischen Modells bestimmte Experimente überhaupt erst möglich macht: die Erstellung des Modells ist zeit- und kostenaufwändig und die Tests wollen sehr gut geplant sein.

B) MENTALE MODELLIERUNG

Doch auch ohne ein konkretes materielles Pendant des zu realisierenden Produkts zu bauen, sind wir viel dichter an der Idee der Modellierung dran als uns oft bewusst ist. Denn auch die systematische gedankliche Erfassung von Zusammenhängen mit dem Ziel eines besseren Verständnisses des betreffenden Systems stellt durchaus eine wichtige Art der „Modellierung“ dar. Innerhalb einer angenommenen Grenze stellen wir uns anhand bekannter Informationen oder unseres Vorab-Wissens bestimmte Elemente, interne Verknüpfungen, Abläufe und Zusammenhänge vor, um daraus auf das Verhalten des Ganzen zu schließen: wir simulieren „mental“, um Schlüsse zu ziehen und von diesem Knowhow-Gewinn bei der System-Entwicklung zu profitieren.

In der Tat ist diese Variante der „modellbasierte Methodik“ in sehr vielen Projekten anzutreffen, oft sogar als verketteter Prozess: auf Basis vorgelagerter Dokumente bildet sich der Experte „im Kopf“ ein Modell des zu entwickelnden Systems im aktuellen Reifegrad, analysiert und entwickelt es ein Stück weiter anhand eigenen Wissens und manifestiert am Ende die Arbeitsergebnisse erneut in einem Dokument.

Eine im Rahmen von Safety-Analysen sehr häufige Analyseform auf Basis mentaler Modelle ist die „Failure Mode and Effect Analysis“, FMEA. Mit der Vorstellung einer gegebenen Funktionalität eines mehr oder weniger komplexen System-Verbunds werden hier in strukturierter Weise - für einen angenommenen internen Fehler nach dem anderen – auf mögliche Auswirkungen bzw. Versagensfälle auf oberster System-Ebene geschlussfolgert, multi-disziplinär im Team. Die Analyserichtung ist induktiv, d.h. gemäß der Kausalität von Ursache zur Auswirkung, und in der System-Hierarchie „bottom-up“, also von den Komponenten hoch zur System-Ebene.

Auch die umgekehrte Denkrichtung – von der Versagensbeobachtung bzw. dem irregulären Symptom an der Systemgrenze hinunter zu möglichen Kern-Ursachen (Root-Causes) - kommt in der Fehleranalyse-Praxis oft vor: bei der Diagnose technischer Systeme, z.B. durch erfahrene Mechaniker in der Werkstatt bei der Ermittlung des auszutauschenden Bauteils. Dabei greifen sie ggf. zurück auf vorab durchdachte und definierte Fehlersuch-Prozeduren in Form von Entscheidungsbäumen („Decision-Trees“, DT).

So unkompliziert und „tool-frei“ die mentale Modellierung auch eingesetzt werden kann, ist dieser Vorteil auch gleichzeitig ihr entscheidender Nachteil: das Knowhow in den Köpfen der Experten ist sehr individuell und kann nur durch mehr oder weniger gut geschriebene text-basierte Dokumente oder Schaubilder weitergegeben werden.

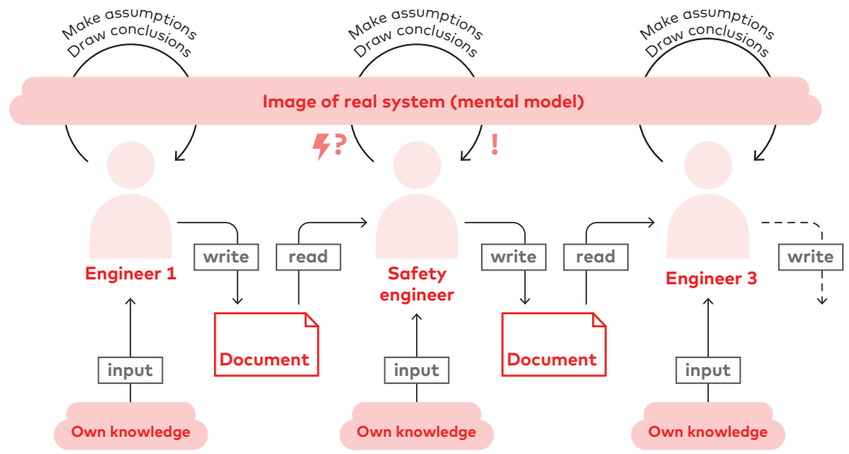


Abbildung 3 Kette mentaler Modellierungsschritte bei der Systementwicklung, z.B. zur Fehleranalyse

© **Critical Software.** All rights reserved.

C) GRAFISCHE MODELLIERUNG

Die Individualität mentaler Modellierung erfordert naturgemäß unterstützende Dokumentation zur hinreichenden Visualisierung des jeweiligen Modells. Hier kommen grafische Abbildungen, Schemata oder Notationen ins Spiel, die helfen sollen, den Gedankengang des Autors für Andere leichter nachvollziehbar zu machen. Aus Gründen der Übersichtlichkeit gilt dabei die Devise: „ein System, viele Sichten“: durch Fokussierung jeweils auf ausgewählte Aspekte des echten Systems, geeignete Vereinfachungen und oft auch die Verwendung standardisierter Gestaltungselemente und Symbole entstehen individuelle grafische „Modelle“ der Realität. SysML-Anwender kennen dies' in Form der verschiedenen vordefinierten Diagrammtypen.

Seien es SysML-Notationen, Zeichnungen, Blockschaltbilder, Prozessdiagramme, Elektrik- oder Hydraulikpläne einer Maschine: Illustrationen machen es möglich, die Abhängigkeiten zwischen Systemteilen und Hierarchien besser zu überblicken, zu diskutieren und auch gedanklich zu simulieren. Ein weiterer Aspekt ist die systematische Darstellung und Benennung der Bild-Elemente, ein wichtiger Beitrag zu einer klaren und unmissverständlichen Kommunikation mit allen Beteiligten.

Neben Reliability-Block-Diagrammen (RBD) sind Fehlerbäume bzw. Fault-Trees (FT) die seit Jahrzehnten im Safety- und Reliability-Bereich wohl bekannteste Form grafischer Modellbildung - die in einer Baumstruktur visualisierte angenommene Versagenslogik des Systems, vom Entwicklerteam deduktiv ermittelt vom Top-Event bis hinunter zu möglichen Root-Causes oder Kombinationen davon.

Obwohl im Wesentlichen nur grafisch aufgebaut aus Blöcken für Ereignisse und für Boolesche Verknüpfungen, erlauben FTs dennoch eine quantitative Berechnung der Wahrscheinlichkeit des Top-Events in Abhängigkeit der Ausfallwahrscheinlichkeiten der Elementar-Komponenten bzw. –Knoten.

So unterstützen grafische Modelle die System- und Safety-Entwicklung entlang des gesamten Prozesses, von der Diskussion verschiedener Architekturen in einer frühen Entwurfsphase (beispielsweise im Rahmen einer PSSA / Preliminary System Safety Analysis gemäß ARP4761) und der Ableitung von Safety-Anforderungen bis hin zur Validierung des finalen Systems „as built“ (wie dem numerischen Nachweis der Gesamt-Wahrscheinlichkeiten von Top-Hazards).

Jedoch, trotz ihrer Ausdruckstärke bleibt auch hier ein wesentlicher Schwachpunkt: Schlussfolgerungen vom skizzierten Modell und dem darin angenommenen Verhalten der Komponenten auf das Zusammenspiel und die Funktionalität des Gesamtsystems im Nominal- und Fehler-Fall geschieht im Kopf des Menschen, basierend auf individuellen und ggf. undokumentierten Annahmen. Damit sind die Qualität, Nachvollziehbarkeit und Persistenz der Ergebnisse letztlich abhängig von der individuellen Expertise der Fachleute und ihrer Verfügbarkeit im Unternehmen.

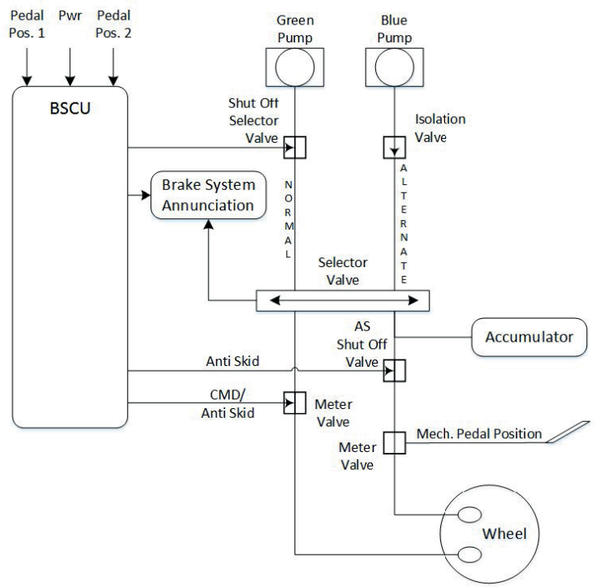


Abbildung 4 System-Diagramm, Beispiel aus ARP4761

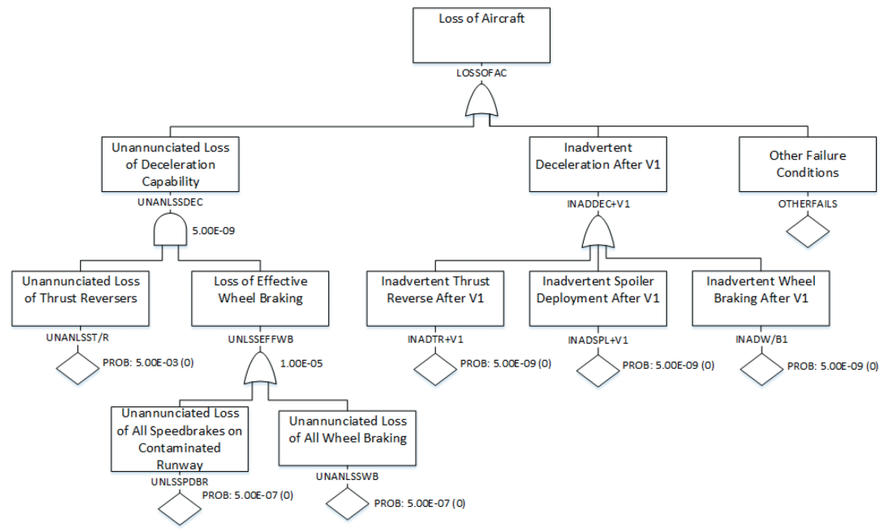
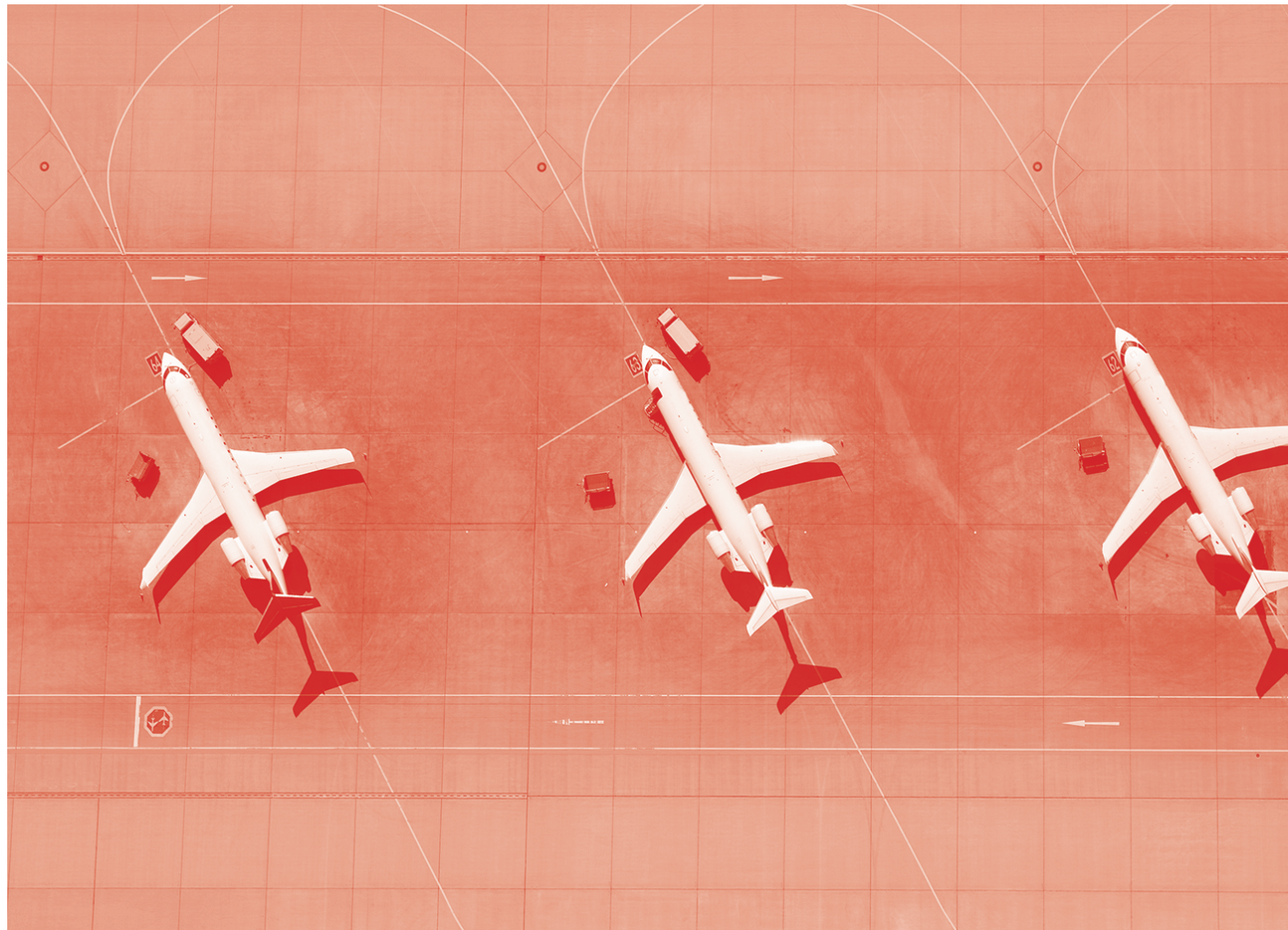


Abbildung 5 Fault-Tree, Beispiel aus ARP4761



© Critical Software. All rights reserved.

Der „ausführbare“ digitale Weg

Hier kommen nun die eigentlichen ausführbaren Modelle ins Spiel. Sie machen es möglich, durch rechnergestützte Simulation das jeweilige Verhalten des echten Systems nachzubilden. Auch diese Technik wird bereits seit vielen Jahrzehnten gelehrt und ist im Engineering etabliert, bisher aber überwiegend zur Abbildung und Analyse des nominalen Verhaltens.

Die relevanten Variablen, Parameter und Konstanten der zu beschreibenden Einheiten und Funktionalitäten werden – im Rahmen getroffener Annahmen - in einer Modellierungssprache erfasst und durch Gleichungen, Zustandsautomaten, x-y-z-Kennfelddaten oder andere analytische Formen in Beziehung zueinander gebracht.

In vielen Werkzeugen unterstützt durch grafische Benutzer-Schnittstellen, entsteht so durch die spezifizierten Eingangs-Ausgangs-Verbindungen zwischen den einzelnen Modellbausteinen implizit ein ausführbares mathematisches Gesamtmodell des stationären oder dynamischen Verhaltens.

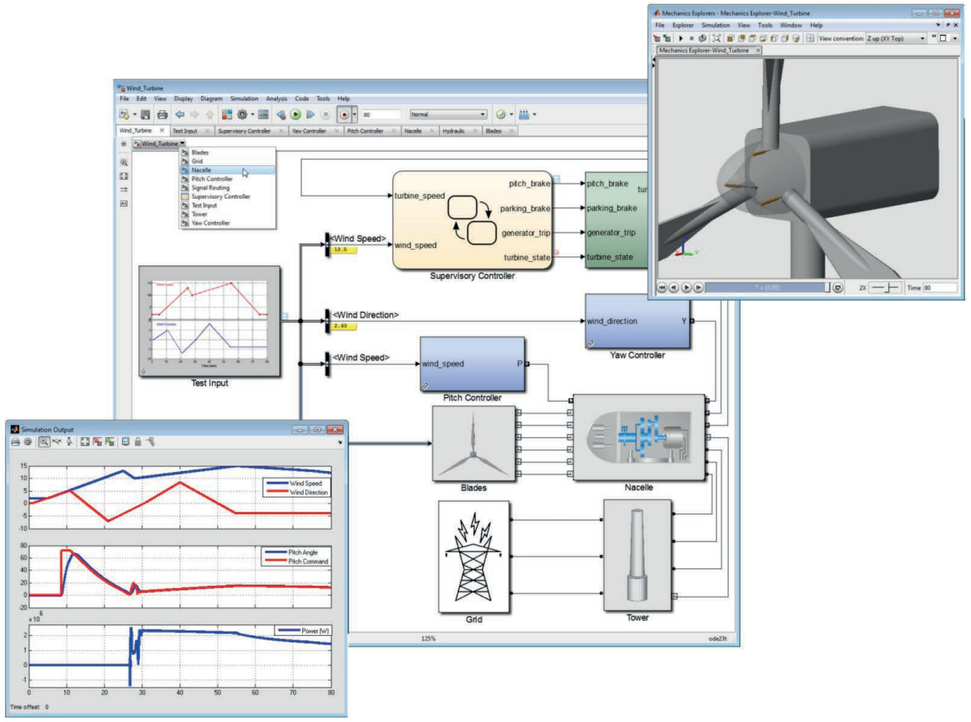


Abbildung 6 Beispiel-Bild Simulink-Modell (*https://commons.wikimedia.org/wiki/File:Simulink_model_of_a_wind_turbine.tif*)

© Critical Software. All rights reserved.

A) KAUSALE BLOCK-ORIENTIERTE MODELLIERUNG

Folgt das Modellierungsprinzip den angenommenen Wirkketten des Systems, spricht man von „kausaler Modellierung“, auch bezeichnet als „block-orientierte“ oder „signal-orientierte“ Modellierung. Hierbei drücken die Beschreibungen innerhalb der Modellblöcke eine klare Abhängigkeit der Ausgangsgrößen von den Eingangsgrößen einer jeden Einheit aus. Um diese Auswerterichtung besser zu veranschaulichen, befinden sich oft in den grafischen Abbildungen bzw. „Icons“ der Modellteile die Eingänge links und die Ausgänge rechts. Die grafische Anordnung der Blöcke ebenso wie die implizite Auswertung ist „gerichtet“.

Ein populärer Vertreter dieses Modellierungsansatzes ist Simulink: die zuvor durch System-Analyse ermittelten System-Variablen sowie Differential- und algebraischen Gleichungen (DAE) zur Beschreibung des Verhaltens werden auf unterster Ebene aus elementaren mathematischen Operator-Bausteinen zusammengesetzt. Gruppiert und mit definierten Schnittstellen versehen („Ports“) ergeben sich wiederverwendbare neue Bausteine, aus denen so das gesamte mathematische Beschreibungsmodell des realen Systems aufgebaut wird.

Nach Vorgabe geeigneter Initialisierungswerte und Eingangsprofile kann dieses Berechnungssystem entsprechend des analytischen Datenflusses anschließend vom Simulationsalgorithmus gelöst und die numerischen Werte oder Zeitverläufe der zuvor unbekannten Systemgrößen angezeigt werden. Für viele ingenieur-technische Analyse-Aufgaben ist diese Art der Modellierung seit Jahrzehnten üblich und viele wiederverwendbare projekt-übergreifende Modell-Bibliotheken sind entstanden.

Wie sieht es nun mit Fehler- und Safety-Analysen aus? Wie unterstützt uns das Modell, um die Zusammenhänge zwischen lokalen Defekten und deren systemweiten Auswirkung besser zu verstehen? – Nun, per Simulation analysieren lässt sich nur das, was zuvor auch modelliert worden ist. Die Beschreibung des Nenn-Verhaltens muss also erweitert werden um Modelle von Fehlern. Dabei stellen sich einige grundsätzliche Fragen:

1. Wie lassen sich Fehlermodi von Bauteilen im Modell organisieren?
Die im realen Leben auftretenden Fehler sind oft konkreten Bauteilen zugeordnet und haben eine spezifische Bezeichnung („unterbrochen“, „klemmt offen“, ...). Es gilt also, neben dem „intakt“-Verhalten eines System-Bausteins auch alternative Beschreibungen für einen oder sogar mehrere verschiedene Fehler-Modi im Simulationsmodell zu berücksichtigen, inklusive einer geeigneten Auswahl-Technik.

2. Was ist mit der ursprünglichen Struktur der Modell-Gleichungen?
Einige, rein parametrische Fehlermodi können sehr einfach im bestehenden kausalen Gesamtmodell berücksichtigt werden, beispielsweise eine numerisch fehlerhafte Sensor-Charakteristik oder ein „Bit-Flip“. Die Wirkkette bleibt grundsätzlich erhalten. Andere System-Fehler können jedoch die gesamte Struktur des zuvor für das Nominal-Verhalten aufgestellten DAE-Systems beeinflussen. Wenn beispielsweise im Falle irregulärer topologischer Veränderungen (z.B. „Kurzschluss im elektrischen Netzwerk“) sich unvorhergesehene Flüsse und damit neue Bilanz-Terme ergeben, wäre ein signifikanter Modell-Umbau notwendig [Ref01].

3. Gilt die angenommene kausale Berechnungskette noch?
Vielleicht ja – vielleicht nicht. Bei vielen realen Einzel- oder gar Mehrfach-Fehlern, die wir analysieren möchten, gilt das im Nenn-Modell angenommene globale Eingangs-Ausgangs-Verhalten nicht mehr. So kann ein Rohrleitungs-Leck lokal die Strömungsrichtung umkehren oder ein irrtümlich angetriebener bzw. nicht hinreichend abgesicherter E-Motor ungeplante Spannungen im Netz entgegen der angenommenen normalen Wirkkette induzieren – und so Änderungen im Modell erforderlich machen. Oder, denken Sie an ein geschlossen hängenden Fluid-Ventil: hier reicht es nicht aus, dem Ausgangsdruck einfach einen Wert "Null" zuzuweisen.

Im Safety-Bereich mit all den oben genannten Herausforderungen bringt das kausale Modellierungsprinzip bei der quantitativen Fehler-Analyse realer Bauteile somit einige Herausforderungen mit sich. Ein derartiger Ansatz zur numerischen

Simulation lässt sich hier am ehesten auf die Modellierung der traditionell prozeduralen Steuerungslogik und die Entwicklung SW-seitiger Diagnose-Funktionen oder Mitigations- und Backup-Strategien anwenden.

Jenseits der numerischen Simulation können kausale Input-Output-Modelle dennoch sehr hilfreich sein, um bei der Entwicklung sicherheitskritischer Systeme den Gesamt-Überblick zu behalten, inklusive der Hardware. Das Management der vielen internen Abhängigkeiten zwischen Anforderungen, Systemfunktionen, betrachteter Lösungsprinzipien und Allokation zu Architektur-Elementen auf verschiedensten Ebenen der Systemhierarchie bis hin zur Zuordnung passender Tests ist eine recht anspruchsvolle Aufgabe. Mit klassischem, dokumentenbasiertem Vorgehen sind hier schnell die Grenzen erreicht, Engineering-Fehler schleichen sich ein, Work-Products passen nicht mehr zueinander, die erforderliche Nachvollziehbarkeit (Traceability) geht verloren.

Solche Probleme lassen sich vermeiden durch über den gesamten V-Prozess integrierte modellbasierte Tools, mit zentraler Datenhaltung für Anforderungen, Funktionen, technischen Lösungen, Komponenten, Interfaces, SW-Code, Dokumente etc.

Und sehr hilfreich für das Safety-Bewusstsein: es lassen sich bereits während der Entwicklung die Abhängigkeiten im Fehlerfall visualisieren. So ist es z.B. bei der ESCAPE-Technologie möglich, an beliebiger Stelle im Netzwerk einen Defekt zu „injizieren" und auf Knopfdruck die davon möglicherweise betroffenen System-Umfänge zu ermitteln und anzuzeigen, gemäß dem FMEA-Gedanken, „downstream" (Abbildung 7). Umgekehrt kann das Versagen der Toplevel-Funktionalität vorgegeben und die möglichen „Root-Causes" dazu ermittelt werden wie bei einer Diagnose, „upstream" (Abbildung 8). Bereits in der frühen Entwurfsphase lassen sich so die Vor- und Nachteile bestimmter Architekturen evaluieren und gegeneinander abwägen sowie „vergessene" Beziehungen quer über alle System-Ebenen, Funktionalitäten und Hardware-Einheiten aufdecken, im gesamten System [Ref02].

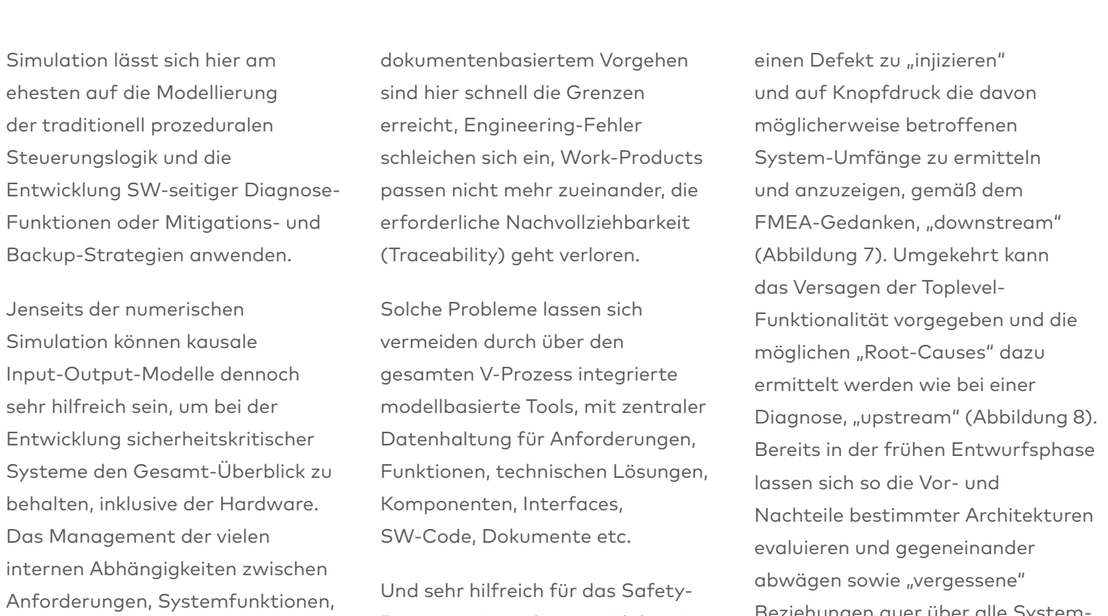


Abbildung 7 Modellbasierte Fehler-Analyse im Abhängigkeits-Netzwerk: a) Vorgabe („Injizierung") lokaler Fehler (rot) und Ermittlung des betroffenen System-Umfang und der Schnittstellen (orange)

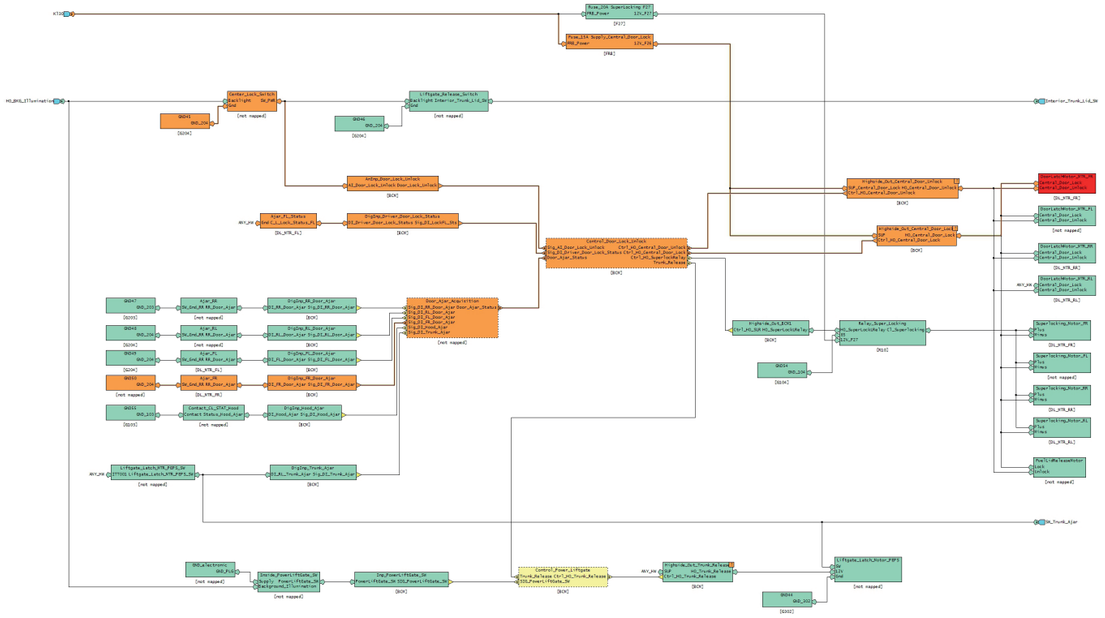


Abbildung 8 Modellbasierte Fehler-Analyse im Abhängigkeits-Netzwerk: b) Vorgabe Versagensfall an einer Schnittstelle (rot) und Ermittlung möglicher Verursacher im Systemverbund (orange)

Bei einer weiteren Kategorie von Modellierungswerkzeugen liegt der Fokus weniger auf der sicherheitstechnischen Analyse eines Systems, sondern auf einer formal korrekten und verifizierbaren Methodik zur Entwicklung sicherheitskritischer Software. Durch die funktionale Eigenart von SW-Systemen kommt auch hier das kausale Modellierungsparadigma zur Anwendung. So ermöglicht z.B. das Tool SCADE eine sehr konsistente Darstellung von Zustandsautomaten und des internen Datenflusses, aus dem am Ende SW-Code für Anwendungen höchster Sicherheits-Anforderungen generiert wird.

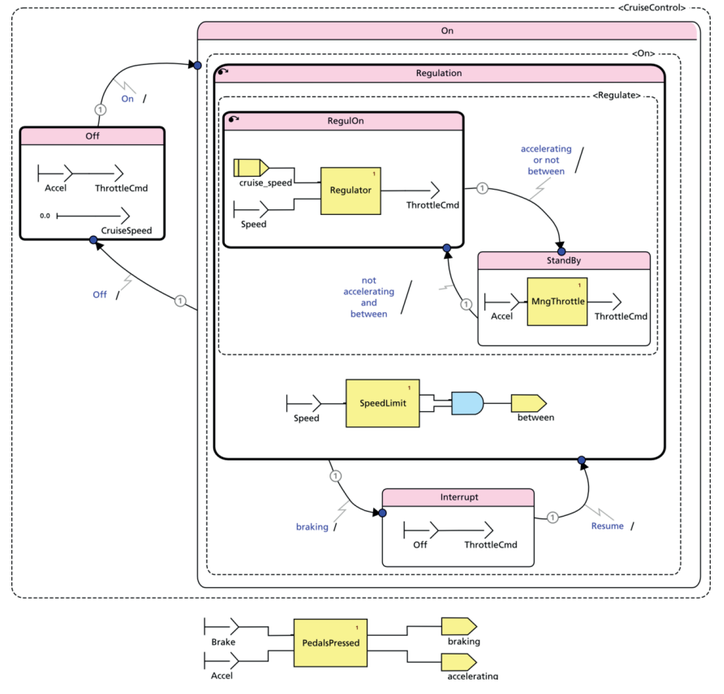


Abbildung 9 Modellbasierte Entwicklung sicherheitskritischer Software eines Tempomat-Systems mit SCADE (https://commons.wikimedia.org/wiki/File:SCADE-cruise-control-design.png)

© Critical Software. All rights reserved.

B) PHYSIKALISCHE SYSTEM-MODELLIERUNG, AKAUSAL

Die Notwendigkeit, eine bestimmte Wirkungskette anzunehmen und die Berechnungsrichtung bereits vorab zu durchdenken, ist ein oftmals ein Handicap der kausalen Modellierung. Wie wäre es, wenn das gesamte Modell sich allein nach der realen physikalischen System-Struktur ausrichtete und diese 1:1 beschriebe – mit Komponenten, Schnittstellen, Topologien und Hierarchien?

Für diese „deklarative" Darstellung eignet sich sehr gut das Paradigma der Objekt-Orientierung: jedes benötigte Modell-Element ist ein Abbild einer geeigneten Typ-Klasse. Gleichgültig, ob diese nur für einen schlichten Variablen-Datentyp steht wie *Current* mit einer zugeordneten Einheit *Ampere*, für eine Schnittstelle (Port) wie *Pin* mit mehreren internen Variablen, ein Bauteil wie *SolenoidValve* mit je zwei Schnittstellen für den elektrischen und zwei für den hydraulischen Teil oder gar ein komplexes Gesamtsystem wie *DialysisMachine* oder *JetEngine* - jedes Attribut im Modell entstammt einer definierten Klasse, sozusagen als Blaupause. Das ist das Prinzip der Sprache Modelica [Ref03], mit einem breiten Spektrum von Anwendungen in Automotive, Avionik, Raumfahrt, Robotik, Energietechnik und auch bei nichttechnischen Systemen aus Bio-Mechanik und Medizin. Etliche darauf aufbauende Tools haben sich etabliert.

Ein interessanter Aspekt der Sprache ist es, dass wir nicht mehr über die Datenfluss-Richtung nachzudenken brauchen. Der Auswerte-Algorithmus kümmert sich darum, ob bei einer bestimmten Analyse die Variable innerhalb einer Schnittstelle als Eingangs- oder als Ausgangsgröße dient. Das Modell ist daher „akausal".

So bleibt das Wissen über die interne Struktur und das Verhalten um ein reales physikalisches Element im Modell an genau einer Stelle beschrieben, gekapselt und nur über die Repräsentanten der echten (elektrischen, mechanischen, hydraulischen, Bus-, ...) Schnittstellen zugänglich. Wir erhalten einen sehr leicht verständlichen und navigierbaren Modell-Aufbau, der z.B. ebenso einem E-CAD-Tool entstammen könnte (Abbildung 10) oder sich unmittelbar aus der im Engineering bekannten „Stückliste" („Bill of Materials", BoM) herleitet.

Diese Klarheit in der Darstellung des Modells hilft sehr, während der Entwicklungsphase das Systemverständnis zu erhöhen, Fehler zu vermeiden oder auch die Modellierung an PDM- oder QM-Standardprozesse wie teile-orientierte Versionsverwaltung anzubinden. Jetzt ist es sehr intuitiv und einfach möglich, ein und dasselbe Modell für verschiedene Arten von Analysen zu erweitern und zu nutzen, ohne die interne Topologie ändern zu müssen. Das Knowhow-Investment ins Modell ist damit gesichert, über Mitarbeiterwechsel hinweg. Allein auf den Algorithmus kommt es an.

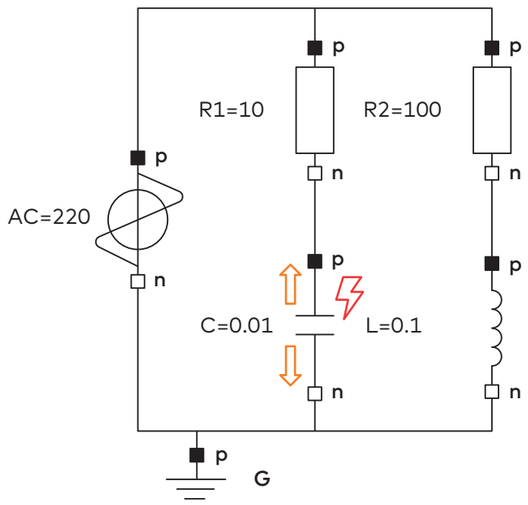


Abbildung 10 Physikalisches System-Modell eines elektrischen Schaltkreises (aus: [Ref03]); bi-direktionale Fehlerauswirkung

Hinsichtlich Fehler-Analysen oder gar RAMS-Analysen lassen sich die zuvor bei der kausalen Modellierung genannten Fragen hier sehr leicht beantworten: alles, was zu einem bestimmten Bauteil-Typus gehört, wird in der entsprechenden Modellklasse hinterlegt: Bezeichnungen der Fehlermodi, alternative lokale Verhaltensmodelle oder auch spezifische Parameter wie z.B. Ausfall-Wahrscheinlichkeiten. Das erlaubt, gezielt einzelne oder mehrere Fehler im Systemmodell zu aktivieren und durch Simulation deren Auswirkungen in alle Richtungen zu ermitteln - die FMEA lässt sich automatisieren.

Umgekehrt ermöglicht das akausale „Zwei-Richtung"-Paradigma, dasselbe Modell auch in Gegenrichtung auszuwerten. An beliebiger Stelle im Systemmodell lassen sich Werte an Modellvariablen zuweisen, selbst wenn sie dem Nominalverhalten widersprechen sollten. Dieses beobachtete Verhaltensmuster bildet ein „Symptom" (siehe Abbildung 1), um mit Hilfe eines Algorithmus zur sog. „modellbasierten Diagnose" automatisch auf mögliche Root-Causes zurück zu schließen [Ref04].

So hilft uns der „virtuelle Spielplatz" auch dabei, bereits früh im Prozess das Fehler-code-/BITE-Konzept bzw. die sogenannte Diagnostic Coverage (DC) zu optimieren.

Seien es derartige industrie-erprobte modellbasierte Anwendungen für Diagnose, Monitoring und RAMS-Analyse, formale Sprachen zur Safety-Analyse wie AltaRica [Ref05] oder ganz neue Ideen wie smartIflow auch für Rail-Anwendungen [Ref06]: zweifellos bilden modellbasierte Safety-Analysen selbst ein spannendes Gebiet innerhalb des Ingenieurwesens, nicht nur hinsichtlich der Modellierung, sondern auch in der Herangehensweise und im persönlichen Mindset in den Engineering-Projekten.

© **Critical Software.** All rights reserved.

Das Versprechen Modellbasierter Analysen

„Modellbasiert“ ist ein vielversprechender Weg, auch bei Fehler- und Versagensfällen in komplexen Systemen den Überblick zu behalten und dazu bei rechnergestützter Umsetzung wertvolles Engineering-Knowhow nachhaltig nutzbar zu machen.

Auch wenn die Modellierung zu Beginn mühsam und zeitaufwändig erscheint, so zahlt sich diese Investition mehrfach aus. Durch ein besseres Verständnis der System- und Ausfall-Abhängigkeiten, durch mehr Vertrauen in die Ana¬lyse-Ergebnisse, durch schnellere Design-Iterationen und Optimierung von Backup- und Mitigation-Strategien im späteren Betrieb der Anlage kann eine gute Modellbildung in der Tat die Art und Weise revolutionieren, wie Systeme betrieben und sicher gehalten werden.

Eingebettet in eine durchgängige Prozess-Kette ohne „Medien-Brüche“ wird das Modell zur zentralen Wissens-Referenz und Grundlage in einem soliden Prozess zur Entwicklung sicherheitskritischer Systeme, dient als „ausführbare Spezifikation“ und kann als virtuelles Test-Bett viel V&V-Aufwand einsparen.

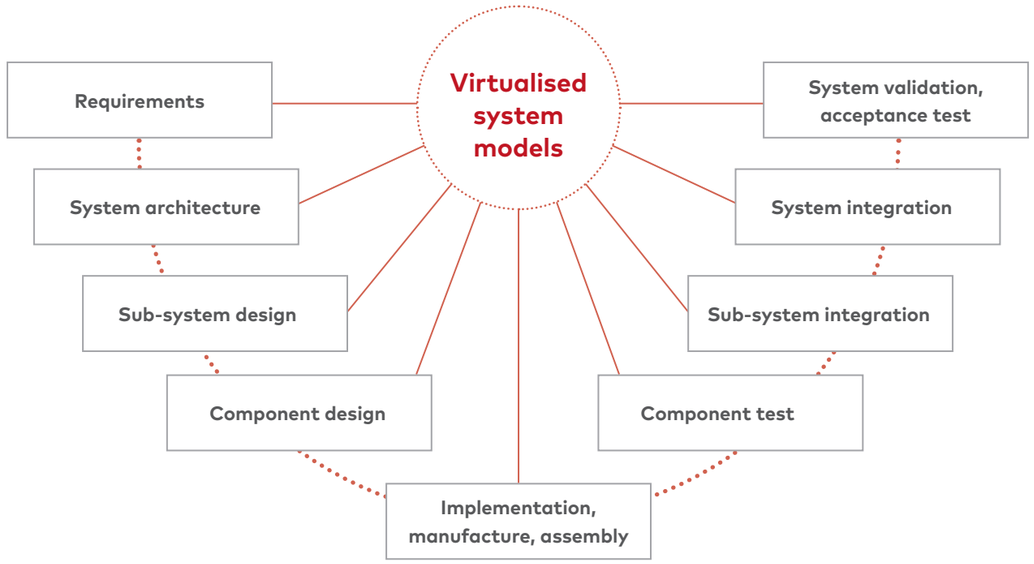


Abbildung 11 Das Virtuelle System als zentrale Referenz im Prozess der Systementwicklung

Das Critical-Modell: Wie wir helfen können

Wir bei Critical Software haben Erfahrung in der Anwendung von Modellierungsmethoden, um die Effektivität und Sicherheit von missionskritischen Systemen in einer Vielzahl von Branchen zu gewährleisten, darunter Automobil, Eisenbahn, Luft- und Raumfahrt und vielen anderen. In ähnlicher Weise können die Software-Validierungseinrichtungen, die wir für eine Reihe von Branchen bereitstellen, problemlos die umfassenden Tests, Verifikationen und Validierungen durchführen, die für sicherheitskritische Systeme in allen Bereichen erforderlich sind.

Unsere über 20-jährige Erfahrung nicht nur in der Entwicklung, sondern auch im Testen und Verifizieren komplexer und vielschichtiger Systeme hat uns die Möglichkeit gegeben, auf diesem Gebiet innovativ zu sein. Indem wir die Agile Methodik zur Entwicklung, Prüfung und Verifizierung hochintegrierter Systeme sowie traditionellere Methoden einschließlich Wasserfall anwenden, bieten wir einen dynamischen Ansatz zur Entwicklung und Validierung kritischer Software, der den Kunden in den Mittelpunkt stellt.

Referenzen:

[Ref01] Joshi A., Heimdahl, M.P.E., Miller S.P., Whalen M.W., Model-Based Safety Analysis, NASA/ CR-2006-213953, University of Minnesota, Minneapolis, Minnesota/ Rockwell Collins Inc., Cedar Rapids, Iowa; *https://shemesh.larc.nasa.gov/fm/papers/Joshi-CR-2006-213953-Model-Based-SA.pdf*

[Ref02] J. Kaiser, G.F. Soto, X.Tang/J.Li/ G. Guo/C. Wen, J. Brandscheid, EXCEED: Integrated, model based development of the E/E-System of CHERY"s new vehicle platform, Day of System Engineering, TdSE 2017, Paderborn, Germany; *http://www.3e-motion.com*

[Ref03] P. Fritzson, Introduction to Object-Oriented Modeling and Simulation with Modelica Using OpenModelica, Open Source Modelica Consortium v2012; *https://www.modelica.org/publications*

[Ref04] Fasol D., Münker B., Bunus P., A Model-Based Safety and Dependability Methodology for Missile Safety Engineering, International System Safety Society Congress ISSC2015, San Diego; *https://icomod.com/mbsa_ISSC2015*

[Ref05] Prosvirnova, T., Batteux, M., Brameret, P.A., Cherfi, A., Friedlhuber, T., Roussel, J.M., and Rauzy, A., The AltaRica 3.0 project for model-based safety assessment. IFAC Proceedings Volumes, 46(22), 127 – 132. doi: *http://dx.doi.org/10.3182/20130904-3-UK-4041.00028*

[Ref06] Lunde R., Hönig, P., Müller C., Reasoning about Different Orders of Magnitude of Time with smartIflow, University of applied Sciences, Ulm, Germany; *https://smartiflow.bitbucket.io/*

Um mehr über unsere Arbeit zu erfahren, wenden Sie sich bitte an: info@criticalsoftware.com

© Critical Software. All rights reserved.

We are CMMI Maturity Level 5 rated.

For a list of our certifications & standards
visit our website.

criticalsoftware.com
info@criticalsoftware.com