

Testen im Automotive- & Embedded-Umfeld

Nur ein anderes Spiel?

<p>Automopoly, 2007</p>					Bahnhof Ingolst. Nord
					Technikträger 300.000 €
					SIL-System 50.000 €
					Ereignis Feld
					HIL-System 100.000 €
Bahnhof Golsburg 	Treiberdiagnose 17 BT	Meilenstein Gemeinschafts Feld	Impuls 4b 4 BT	Vernetzungstest 10 BT	Aktion Rückruf

Einleitung

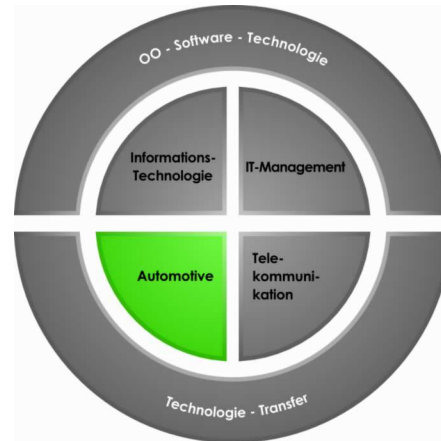
Kölsch und Altmann organisiert sich neben seinen Kernbereichen

- Informations-Technologie,
- OO-Software-Technologie und
- Telekommunikation

und den Querschnittsbereichen

- IT-Management und
- Technologie-Transfer

in einen weiteren Kernbereich „**Automotive**“.



Hierbei sind die Grenzen zwischen diesen Bereichen durchaus fließend. So wird IT-Management in allen SW- wie auch HW-Projekten betrieben, genauso wie z.B. technische Systeme wiederum auch als Komponenten in einem übergeordneten Informationssystem mitspielen können.

Nicht zuletzt halten Methoden des SW-Engineering und der strukturierten Entwicklung im Augenblick auch Einzug ins Automotive-Umfeld, das bisher eher von Methoden des System-Engineering geprägt war.

Dies dokumentiert sich insbesondere auch beim Testen: Bisher zeitaufwändige „händische Tests“ werden mehr und mehr von automatisierten Tests abgelöst bzw. zumindest in weiten Teilen ergänzt.

Mit den Anforderungen, die an einen Test-Ingenieur gestellt werden, wächst auch sein Stellenwert im Projekt. Seine Qualifikation ist der des SW- bzw. HW-Entwicklers keinesfalls untergeordnet. Die Anforderungen können sogar höher sein, schließlich muss er zwei Systeme verstehen bzw. beherrschen, das zu testende System *und* das Testsystem.

Software-Management wird betrieben, seit Software erstellt wird, die zugehörigen Methoden sind zwischenzeitlich etabliert und allgemein akzeptiert. Software-Management ist jedoch nur ein, wenn auch wesentlicher Teil des Test-Management (siehe Kap. [Test-Management](#)).

Durch die zunehmende Testautomatisierung liegt im Automotive-Umfeld bzgl. der einzelnen Testaufgaben besonderes Gewicht auf der Vorbereitung und Wartung der Testumgebung. (siehe Kap. [Testumgebung](#) bzw. [Testumfeld Automotive](#)). Sowohl bei der Spezifikation als auch bei der Durchführung der Tests (siehe Kap. [Testspezifikation & -durchführung](#)) muss insbesondere die heterogene Welt der eingesetzten Werkzeuge berücksichtigt und – im Rahmen von Outsourcing und der teilweise dislozierten Testlabors – den Schwierigkeiten bei der Verteilung und des Austauschs bzw. Upgrades insbesondere der HW-Anteile sowohl des Prüflings als auch der Testumgebungs-HW begegnet werden. Nicht zuletzt müssen die

Testen im Automotive- & Embedded-Umfeld

Testergebnisse (siehe Kap. [Testergebnisse](#)) zusammengetragen und dokumentiert werden.

Vor allem im Anwendungsumfeld Automotive, aber auch in der Wehrtechnik wird überwiegend mit dem V-Modell (siehe Kap. [V-Modell](#)) gearbeitet, an Hand dessen vor allem auch die Testaktivitäten auf den verschiedenen Teststufen (siehe Kap. [Teststufen und nicht funktionale Tests](#)), abgegrenzt (soweit möglich) in Verifikation und Validierung (siehe Kap. [Verifikation & Validierung](#)) erläutert werden.

Mehr und mehr werden Tests auch aus den erstellten Modellen generiert, so dass – neben dem Testen an sich – zusätzlich auch noch der Generator beherrscht werden muss (siehe Kap. [Modellbildung & Simulation](#)).

Test-Management

Neben dem Wissen um effektive Testtechniken und langjähriger Test-Erfahrung ist der Einfluss des Test-Managements nicht zu unterschätzen. Die ausgefeiltesten Techniken versagen, wenn die erforderlichen Ressourcen nicht in benötigter Qualität zu einem bestimmten Zeitpunkt zur Verfügung stehen.

Erfahrungen & Anforderungen

Speziell beim Testen besteht das Dilemma, dass hier gerne mal zuerst Sparmaßnahmen angesetzt werden, meist allerdings recht kurzsichtig. Verschiedenste Rückrufaktionen sind ein Indiz dafür, dass zu wenig getestet wurde und billigend in Kauf genommen wird, dass die Tests zum Kunden verlagert werden.

Andererseits sollte Testen auch nicht zum Selbstzweck werden. Die Erfahrung zeigt, dass die meisten Fehler im Testsystem selbst gefunden werden. Gründe hierfür sind meist komplexe Testumgebungen mit nicht genügend qualifiziertem bzw. geschultem Testpersonal, verbunden mit der Vorgabe von Testzielen, die sich nicht orientieren am obersten Testziel, nämlich vor allem „Fehler zu finden“. Meist werden die Teams der verschiedenen Teststufen auch viel zu spät zusammengestellt, so dass sie gerne mal eher zusammengewürfelt sind und nicht genügend Zeit bleibt, die Mitarbeiter auf ihre Aufgaben und den Test im Detail entsprechend zu planen und vorzubereiten.

Schulung der Mitarbeiter heißt hier nicht nur Werkzeug- und theoretische Prozessschulung, sondern bleibt eine Herausforderung über den ganzen Testzeitraum hinweg. Hier gilt es, die Mitarbeiter laufend weiter zu bilden, Konzepte und Prozesse so zu präsentieren (und immer wieder zu wiederholen), dass sie zum einen verstanden, zum anderen auch angenommen und dauerhaft gelebt werden. Voraussetzung hierfür ist vor allem, das Warum verstanden zu haben und an den Erfolgen teilhaben zu dürfen.

Ein sehr wesentlicher Skill für einen Test-Mitarbeiter ist Kommunikationsfähigkeit und –willigkeit, da der Tester sehr viele Schnittstellen hat, angefangen vom Kunden über die Entwicklungsabteilung bis hin zur Qualitätssicherung, um nur ein paar wesentliche zu nennen. Es ist auch nicht unüblich, dass die Test-Teams örtlich verteilt sind oder gar ganze Testpakete ausgelagert werden. Nicht zuletzt braucht Test auch Architektur, insofern ist Teamfähigkeit ein genauso tragendes Element wie das Hinausschauen über den eigenen Tellerrand.

Qualitätssicherung beschränkt sich nicht nur auf den Prüfling, genauso wichtig sind geeignete Maßnahmen auch beim Test. Angefangen mit einer tragenden Architektur sollten die implementierten Testfälle Programmierrichtlinien einhalten und gut strukturiert sein. Ein Testfall bzw. seine Teile sind immer auch Kandidaten, in anderen Testfällen bzw. übergreifenden Szenarien wiederverwendet zu werden.

Zu einem effektiven Test-Management gehört die Testplanung mit dem daraus resultierenden – und laufend fortgeschriebenen – Testplan. Hierin werden vor allem die Testumgebung, der Test selbst und die Dokumentation der Testergebnisse spezifiziert.

Testprozess: CMMI / SPICE

In den bezeichneten Anwendungsfeldern spielen immer mehr – und mehr als anderswo – Prozessmodelle und Assessment-Methoden eine Rolle. Es geht nicht nur um die Qualität der auszuliefernden Artefakte, sondern im Rahmen des Zusammenspiels vieler Stakeholder (angefangen mit Auftraggebern und Zulieferern) auch um die Qualität, Dokumentation und Nachvollziehbarkeit und nicht zuletzt ständige Verbesserung der eingesetzten Prozesse und um Methoden, deren Einhaltung (und ständige Verbesserung!) zu validieren sind. Hier haben sich, beide basierend auf der ISO15504, die Standards CMMI und speziell für die Automobilindustrie SPICE bzw. genauer AutomotiveSPICE etabliert.

CMMI lehnt sich wesentlich an die ISO12207 an, einem Prozessmodell für Software-Engineering und an die ISO15288, einem Prozessmodell für das System-Engineering und definiert Reifegrade für eine Organisation (z.B. eines ganzes Unternehmens bzw. eines Bereichs davon) bzw. auch nur für einzelne Prozessgebiete.

SPICE definiert kein eigenes Prozessmodell, sondern konzentriert sich auf das Assessment. Zugrunde gelegt werden hier in der Regel aber die Prozessmodelle aus der ISO12207, SPICE ist also – vereinfacht ausgedrückt – quasi das Bewertungsverfahren dieser Norm.

Beide (CMMI und SPICE) bewerten die eingesetzten, gelebten und verbesserten Prozesse. Um die Qualität der Produkte zu verbessern, ist es in der Regel dienlich, auch die Prozesse der Qualitätssicherung zu validieren. CMMI und SPICE prüfen allerdings nur auf Vorhandensein der Prozesse, nicht auf inhaltliche Qualität der dabei bearbeiteten Artefakte.

Des Weiteren ist der Reifegrad einer Organisation zum Einen nur definiert für einen bestimmten Zeitpunkt und kann sich ein halbes Jahr später schon wieder ganz neu darstellen, zum Anderen sind die Prozesse auch stark vom Auftraggeber bestimmt oder zumindest beeinflusst.

Test

Testumgebung

Bei der Beschreibung der Testumgebung wird definiert, welche HW und SW für welche Testebene benötigt werden. Hierzu gehören Simulationsumgebungen bzw. Test-Treiber, einzusetzende Werkzeuge, die Rechner- und Vernetzungsumgebung. Dies kann schon recht komplex sein, muss aber noch ergänzt werden um die Installation, Konfiguration und Wartung all der eingesetzten Werkzeuge und Umgebungen.

Hinzu kommt, dass speziell Werkzeuge, aber auch HW-Anteile eingekauft werden, und von externen Dienstleistern gewartet werden. Angefangen von der Auswahl des Equipments über den Einkauf, die Installation und Einrichtung und das laufende Customizing stellt sich die Frage, ob die Planung, Überwachung und Wartung dieser Bereitstellungen vom Projekt selbst geleistet werden sollen oder – vor allem wenn es mehrere Testprojekte mit ähnlicher Umgebung gibt – in ein explizites Testquerschnitts-Team ausgelagert wird.

Die Testlandschaft ist hier aktuell noch durchaus heterogen. Nur langsam entwickeln sich hier Standards, die in Gremien wie Autosar oder der ASAM vorangetrieben werden.

Im Rahmen der Testumgebung nicht unerwähnt bleiben soll hier, dass sowohl die Testdokumentation, die entwickelten Testfälle, die Prüfmuster, als auch die Testergebnisse geeignet abgelegt und archiviert werden müssen. Dies muss konzipiert und mit geeigneten Werkzeugen (z.B. Documentum und Clearcase) geplant und umgesetzt werden.

Testspezifikation & - durchführung

Neben der Abgrenzung, was der Test nicht leistet, wird hier definiert, welche Methoden (z.B. „Blackbox“) mit welchen Techniken in welchen Testebenen Schritt für Schritt eingesetzt werden. Hierzu gehört auch eine realistische Abschätzung des Aufwands für die einzelnen Aktivitäten.

Eine besondere Herausforderung bei technischen Systemen stellt das Timing dar. Angefangen damit, dass die Prozessoren für die Embedded-Systeme durchaus eingeschränkt sind, dies sowohl in ihrer Mächtigkeit als vor allem auch in Ihrer Performance, beeinflusst auch das Übertragungsprotokoll zwischen den einzelnen Subsystemen die Zeit, die vergeht, bis eine Stimulation in einer entsprechenden Reaktion des Systems mündet. So werden manche Botschaften zyklisch und regelmäßig übertragen, während andere nur bei Bedarf sofort gesendet werden. Eine entsprechende Verzögerung kann sich hierbei sowohl hinsichtlich der Stimulation als auch bzgl. der Verifikation ergeben. Eine Simulationsumgebung bzw. diese repräsentierende Werkzeuge qualifizieren sich vor allem dadurch, inwieweit und wie gut sie in Realzeit messen können. Auch bzgl. der angebundenen Hardware gilt es, beim Test die vorgesehenen Entstörzeiten zu berücksichtigen.

Neben diesem eher technischen Anteil (auch an Spezifikation) ist ein weiterer wesentlicher Teil die Planung der menschlichen Ressourcen und der zeitlichen Dimension unter Beachtung der vorgesehenen Meilensteine. Man bedenke hierbei auch, dass ein Plan schon veraltet ist, kurz nachdem er erstellt wurde. Planung ist keine einmalige Aktion, sondern ein laufender Prozess. Nicht zuletzt bleibt zu überwachen, dass das Testziel stückweise zumindest näher rückt. In einer Branche mit engen Zeitplänen sind die Testziele des Auftraggeber (ob intern oder extern) meist höher gesteckt als die selbst gesteckten, bisweilen gar illusionär (100% Testabdeckung?!). Diesen Spagat, zum Einen den Kunden zufrieden zu stellen, gleichzeitig aber nicht an unerreichbaren Testzielen bzw. engen Kostenplänen zu verzweifeln, gilt es zu beherrschen.

Testergebnisse

Bzgl. der Testergebnisse muss spezifiziert werden, wie die aus den Tests resultierende Testdokumentation aussehen soll. Auch Fragen der Archivierung der Testergebnisse – zusammen mit der Umgebung und dem Prüfling (Sample / SW-Version, Variante), in der bzw. mit dem sie produziert wurden – müssen geklärt werden.

Nicht zuletzt muss die Testdokumentation auch handhabbar sein, insofern,

- dass sie nicht so umfangreich ist, dass z.B. eine Browser sie gar nicht mehr laden kann (evtl. müssen die Tests dementsprechend aufgeteilt werden) oder
- dass sie auch eingängig lesbar und vollständig ist, damit sie auch von den Entwicklern und sonstigen Interessenten angenommen werden. Hierzu gehören z.B. auch präzise Timestamps für jede Stimulation und für jede Verifikation.

Im Sinne der **Traceability** sollte es auch möglich sein, dass für jeden Testfall die von ihm umgesetzten Requirements referenziert werden, genauso im Übrigen wie für jedes Requirement die Testfälle referenziert werden, die sie mehr oder weniger vollständig abdecken.

Anwendungskontext & Projektumfeld

Technische Systeme werden überwiegend (z.B. in der Automobilindustrie, der Wehrtechnik, der Luft- & Raumfahrttechnik) auf Grundlage des V-Modells entwickelt, zum Teil nicht zuletzt deshalb, weil der Auftraggeber letztlich oftmals die „Öffentliche Hand“ ist.

V-Modell

Das V-Modell sieht hier hinsichtlich der verschiedenen Integrationsstufen grundsätzlich folgende Testebenen vor, wobei das V-Modell keinen Prozess darstellt im Sinne von zeitlicher Abhängigkeit, sondern nur die einzelnen Artefakte zueinander in Beziehung setzt:

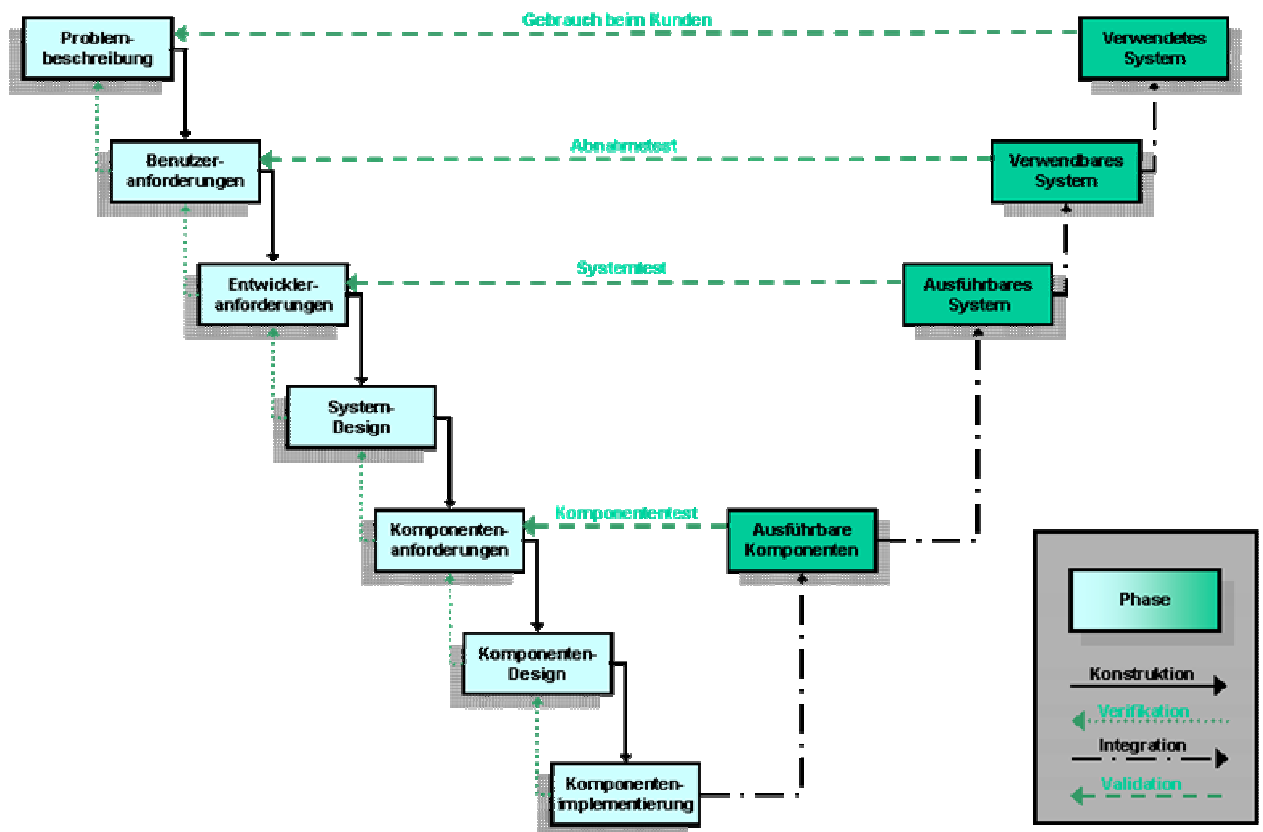


Abbildung 1: V-Modell und Testen

Automotive und Embedded-Systeme zeichnen sich vor allem aus durch

- Verteile und komplexe Systementwicklung (Nationen-übergreifend, OEM/Zulieferer)

Testen im Automotive- & Embedded-Umfeld

- Extrem hohe Anforderungen hinsichtlich Echtzeitverhalten, Zuverlässigkeit und Sicherheit
- z. Teil extreme Umweltbedingungen (Druck, Temperatur, mechanische Beanspruchung, ...)
- hohen Zeit- und Kostendruck
- extrem hohe Änderungs- und Konfigurationsanforderungen

Basierend auf diesen zusätzlichen Rahmenbedingungen wird das Testen im Automotive-Umfeld im allgemeinen Sprachgebrauch mit folgendem ∇ kategorisiert:

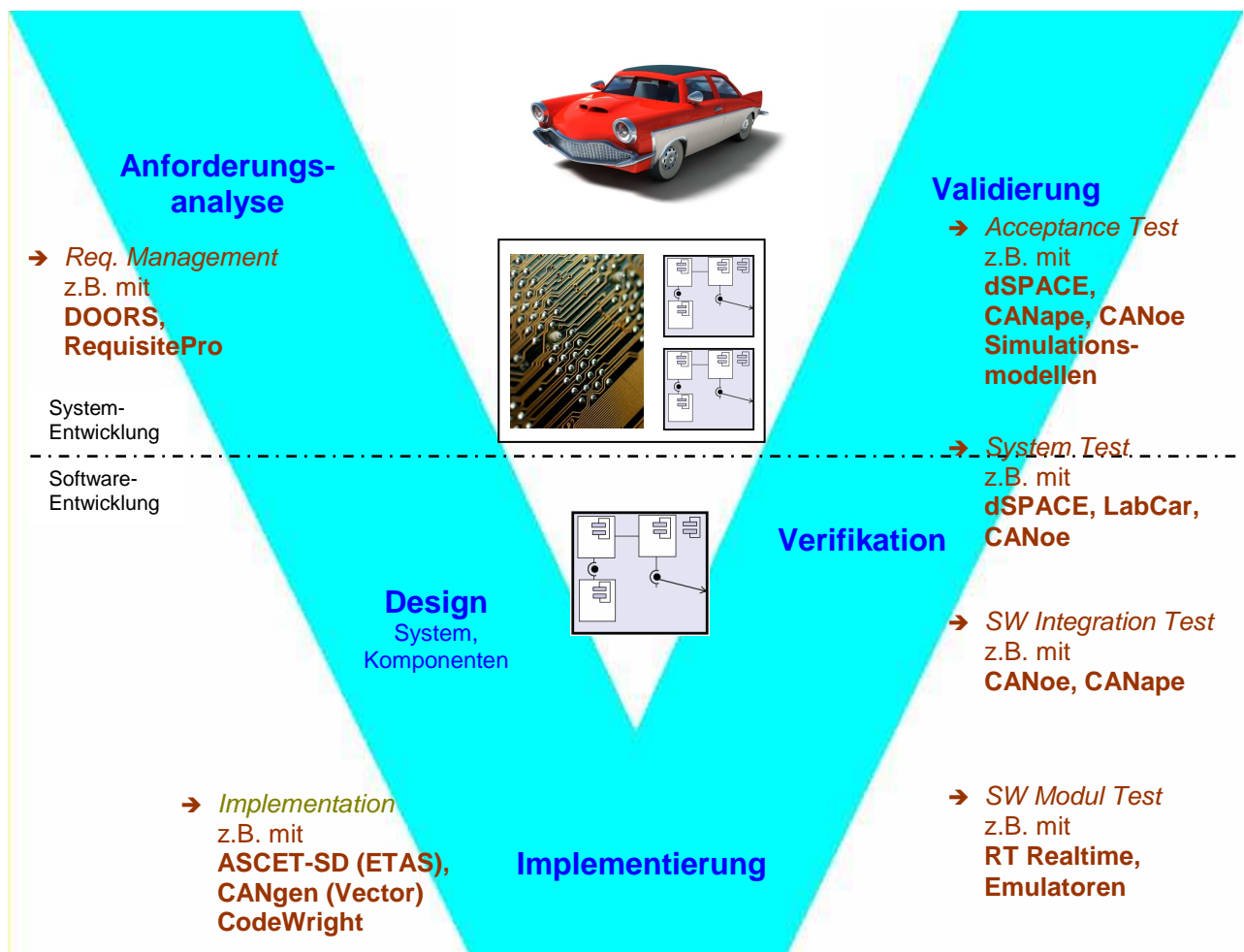


Abbildung 2: Tool-Einsatz

Naiv könnte man das ∇ in Schreibrichtung interpretieren, dass also von der Anforderungsanalyse bis zur Validierung die Teststufen genau in der Reihenfolge ablaufen, was grundsätzlich erst einmal nicht falsch ist. Nicht erst mit dem V-

Modell-XT, das das bisherige Modell auch um Ansätze agiler Software-Entwicklung erweitert, ist es heute aber State-of-the-Art, dass das ∇ oben eigentlich zu weit offen ist. Das ∇ oder Teile davon werden dabei nicht nur in mehreren Iterationen durchlaufen, sondern die auf gleicher Höhe liegenden Stufen laufen evtl. mit etwas Versatz auch parallel. Der Systemtest z.B. beginnt also nicht erst mit Ende des SW-Integrationstest, sondern muss auch vorbereitet werden. Die Testergebnisse können dazu führen, dass nochmals über das Design nachgedacht werden muss. Nicht zuletzt ist die linke Seite des ∇ auch gehalten, Ihre Artefakte so zu gestalten, dass sie auch verifizierbar und validierbar sind. Schon hierbei können die Test-Ingenieure mit im Boot sitzen.

Verifikation & Validierung

Im Hinblick auf die Teststufen wird explizit zwischen Verifikation und Validierung unterschieden, wobei die Grenzen durchaus fließend sind:

Verification: Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled [ISO/IEC 12207, Software life cycle processes]. In other words, verification ensures that

“you built it right”.

Validation: Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled [ISO/IEC 12207, Software life cycle processes.] In other words, validation ensures that

“you built the right thing”.

Diese Unterscheidung ist vor allem bei der Entwicklung von technischen Systemen essentiell, da hier nicht nur – wie bei anderen, größeren Systemen auch – die Anforderungen einem zeitlichen Wandel unterworfen sind, sondern des Weiteren auch bewusst hingenommen wird, dass die Anforderungen mit Abschluss der Anforderungsanalyse nicht komplett sind, sondern in weiteren Iterationen vervollständigt und – z.B. falls technisch nicht oder nur unter grober Verletzung der finanziellen oder zeitlichen Projektrahmenbedingungen umsetzbar – korrigiert werden.

Für die Verifikation können verschiedene Techniken angewendet werden, so z.B.

- Dynamische Tests (Testling ist mit integriert und wird ausgeführt)
 - Funktionale Tests: Tests, mit denen die Requirements verifiziert werden, die die Funktion des Systems beschreiben. Solche Tests werden in der Regel als Black-Box-Tests ausgeführt, da hierzu keine Kenntnis über die innere Struktur des zu testenden Systems notwendig ist.
 - Tests basierend auf der Struktur: Diese Tests setzen Kenntnisse über die Implementierung des Systems voraus, sind also White-Box-Tests. Hierbei werden einzelne Komponenten des Systems getestet.

- Zufallstests: Hierbei werden aus der Menge aller möglichen Testfälle (das sind mehr als die implementierten!) eine zufällige Auswahl getroffen bzw. die Testfälle in verschiedener Reihenfolge (Szenarien!) durchgeführt.

Hierbei kann der Testling zum Mitprotokollieren angehalten, wie auch aktives Debugging betrieben werden, um bei der Fehlersuche zu unterstützen. Zu diesem Zweck kann z.B. auch das Werkzeug CANape eingesetzt werden, um bei Bedarf entsprechende, interne Variablen auszulesen, die einen zu untersuchenden Ausschnitt des Systemzustands repräsentieren.

- Statische Analysen (Testling muss hierzu nicht ausgeführt werden)
 - Wesentliche Teile dieser Tests werden von Compiler, Linker und Lint automatisch ausgeführt. Ebenfalls automatisierbar sind Tests, die z.B. die Programmierrichtlinien überwachen. Des Weiteren kann z.B. in Reviews die korrekte Umsetzung der Requirements und Spezifikationen überprüft werden.

Für die Validation können verschiedene Techniken angewendet werden, so z.B.

- Formale Methoden: Hinausgehend über die statischen Tests bzgl. der Verifikation können hier auch diverse Methoden zum Einsatz kommen, die die Umsetzung der Requirements (Traceability), die Vollständigkeit und Konsistenz der geforderten Requirements und Systemmodelle und das Verhalten des System (HW & SW) untersuchen helfen.
- Fehlerinjektion: Hierbei werden bewusst Fehler in die HW bzw. SW des Systems eingespeist und das daraus resultierende Verhalten des Systems untersucht. Hierzu zählt z.B. bei einem elektrischen bzw. elektronischen System die Einspeisung eines Kurzschlusses bzgl. eines Ausgangs. Bei Zugriff auf internen Variablen des Systems können aber auch diese von außen geändert werden, um z.B. HW-Fehlerinjektionen zu simulieren.
- Abhängigkeits-, Fehleranalyse & Risikoanalyse: Mit diesen Analysemethoden kann die Auswirkung von Fehlern auf die verschiedenen Komponenten des Systems untersucht werden und darauf basierend eine Risikobewertung durchgeführt werden. Ein Methode hierfür ist z.B. die FMEA (Failure Modes And Effects Analysis). Des Weiteren können bei der Fehleranalyse Gegenmaßnahmen definiert werden.

Teststufen und nicht funktionale Tests

Modul- und Integrationstest sind Teststufen, die Ihre Wurzeln in der SW-Entwicklung haben. Hier gilt es, geeignete Simulationsmodelle zu erstellen, damit die Umgebung des Prüflings geeignet zu simulieren und diesen mit passenden Testdaten zu „füttern“. Nicht zuletzt sind die Ausgaben des Prüflings zu evaluieren. Neben der Auswertung von analogen Ausgängen (Oszilloskop), sind hier insbesondere MCD-Systeme wie CANape und CANoe im Einsatz, um die interne Zustandsübergänge in den Moduln zu validieren, aber auch die Informationen auf

Testen im Automotive- & Embedded-Umfeld

den verschiedenen Bus-Systemen (z.B. CAN, LIN) zu stimulieren bzw. zu untersuchen.

Während beim Integrationstest vor allem die Schnittstellen der und zwischen den Komponenten getestet werden, und hierfür auch Kenntnis über die Interna des Systems vonnöten sind, sollte der **Systemtest** als Black-Box-Test bzgl. eines bereits integrierten Systems ausgeführt werden. Getestet wird hier insbesondere nicht vorrangig gegen die Spezifikation, sondern – zumindest letztlich – gegen die Anforderungen an das System. Getestet wird hierbei HW und SW im Verbund (z.B. an einer Lastbox oder in einer HiL-Umgebung).

Lasttests werden bei Embedded-Systemen vor allem bzgl. der verschiedenen Busse durchgeführt. So gibt es eine ganze Menge Nachrichten, die zwischen den einzelnen Busteilnehmern ausgetauscht werden, vor allem auch Nachrichten, die nur bei Bedarf geschickt werden (im Gegensatz zu anderen, die laufend auf dem Bus sind). Des Weiteren wird unterschieden zwischen Nachrichten, die nur Informationen transportieren und solchen, die eine Handlungsanweisung an den Adressaten darstellen. Letztere sind z.B. bei der Diagnose zu finden. Die Evolution geht hier in Richtung von Bussen mit mehr Durchsatz (z.B. Flexray). Beachtet werden müssen im Rahmen der Massenproduktion aber immer auch die Kosten des einzelnen Bussystems. Im Zusammenhang mit überlasteten Bussen ist insbesondere die Sicherstellung der Priorisierung wichtiger Nachrichten ein besonderes Testziel.

In **Spannungstests** wird ein System dahingehend untersucht, ob es (insbesondere bei Über- und Unterspannung) das spezifizierte Verhalten zeigt, ob es einzelne Funktionen nur in bestimmten Spannungsbereichen aktiviert und sie bei Verlassen derselben deaktiviert. Nicht zuletzt müssen hier auch Konzepte der Entstörung (Hysterese) beachtet werden.

Diagnose (Monitoring, Fehlerspeicher, Anpasskanäle, Messwerte, Stellgliedtest, ...) besitzt im Automotive-Umfeld einen großen Stellenwert.

Hierbei haben sich die Standards

- ISO-14230-3 (KWP2000) und
- ISO-15765 (Diagnosis on CAN)

am Markt etabliert und sind in der ISO-14229-1 (UDS) aufgegangen. Auch die ASAM hat eine eigene Arbeitsgruppe MCD eingerichtet, die sich um die Standardisierung in diesem Bereich bemüht. Dass hier mehr oder weniger alle Automobilhersteller mitarbeiten, gibt zur Hoffnung Anlass, dass die Standardisierung auch in diesem Bereich weiter voranschreitet.

Entsprechend nehmen **Diagnosetests** einen nicht unerheblichen Anteil der Testkapazität in Anspruch und gehören sicherlich mit zu den anspruchsvollsten und umfangreichsten Testaktivitäten im Bereich Automotive.

Nicht zuletzt sind auch Vernetzungstests ein wichtiger Teil im Testprozess. Im Rahmen der stufenweisen Integration der einzelnen Teilsysteme bleibt auch nach dem erfolgreichen Test der einzelnen Teilsysteme durchaus noch Platz für Über-raschungen, vor allem in Bezug auf Übertragungskapazität des Netzwerks und das u. a. auch dadurch beeinflusste Timing.

Modellbildung & Simulation

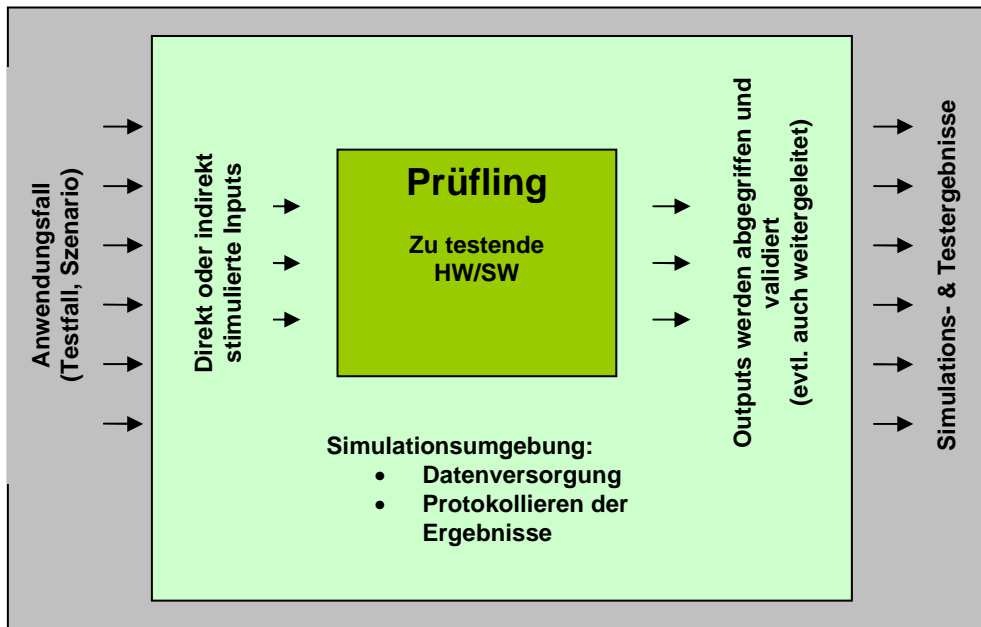


Abbildung 3: Simulation

Sowohl Informationssysteme als auch technische Systeme durchlaufen die verschiedenen Entwicklungsstadien Anforderungsanalyse, Analyse & Design, Implementierung, Test und Systemeinführung.

Während bei reiner Softwareentwicklung der Nutzen von Prototypen unbestritten ist:

- Geringer Herstellungsaufwand (gemessen am Gesamtprodukt)
- Analyse (zu einem frühen Zeitpunkt!), ob die Anforderungen überhaupt und ggfs. wie umsetzbar sind
- Risikobewertung und Kostenschätzungen

kommt bei der HW- bzw. kombinierter HW- und SW-Entwicklung noch der Aspekt hinzu, dass die Entwicklung und vor allem der Test von HW unter realen Bedingungen (z.B. Elektr. Stabilitätsprogramm, ESP) vergleichsweise teurer ist. Hierbei handelt es sich z. T. um hochkomplexe Sensorik- und Regelungssysteme, deren Test das Vorhandensein des kompletten Gesamtsystems voraussetzen würde. Simulationsmodelle bringen hier eine enorme Kostenersparnis, auch wenn hierdurch der Test unter realen Bedingungen nicht komplett ersetzt werden kann.

Teile von SW-Prototypen können durchaus auch im weiteren Projekt verwendbar sein, während HW – zumindest für den produktiven Einsatz – meist neu erstellt

Testen im Automotive- & Embedded-Umfeld

werden muss. Des Weiteren kann SW – bei geeigneter Infrastruktur – sogar noch nach der System Einführung ohne weiteres und kostengünstig ausgetauscht werden (Flash).

In diesem Sinne kommt auch der Simulation bei der Entwicklung von technischen Systemen eine in der Regel höhere Bedeutung zu als z.B. bei der Entwicklung von Informationssystemen. Beginnend vom Requirements-Management weiter über die Analyse der Systemanforderungen, der Erstellung der Systemspezifikation bis hin zu HW- bzw. SW-Design & Implementierung können so zum einen ein gemeinsames Verständnis erarbeitet und zum anderen Fehler bereits im Vorfeld gefunden, vom Zulieferer bzw. OEM analysiert und korrigiert werden.

Ergebnis der ganzen Anstrengungen ist ein validiertes Modell eines Systemkontextes, das mit den verschiedenen Verantwortlichen der Systementwicklung (nicht zuletzt mit dem Kunden) abgesprochen ist.

Des Weiteren werden große komplexe Systeme stufenweise integriert. Konsequenz für das Testen ist die Notwendigkeit, die nicht vorhandenen Systemkomponenten so zu simulieren, dass sie sich bzgl. des durchzuführenden Tests so verhalten wie später auch im Komplettsystem. So müssen beim Test von Fahrzeugkomponenten die nicht vorhandenen Komponenten simuliert werden, hängen diese z.B. alle an einem Bus, so ist also eine Restbussimulation erforderlich.

Testumfeld Automotive

Simulatorboxen, Lastboxen / Brettaufbau

Lastboxen fassen ausgewählte Aktuatoren, Sensoren, Bedienfелеlementen, Leuchten, Motoren, Relais, ... auf engstem Raum zusammen und sind so insbesondere für Handtests geeignet.

Lastboxen können allerdings auch an HIL-Systemen eingesetzt werden, hier dann allerdings ohne die – in der HIL-Umgebung *automatisch* stimulierten – Aktuatoren.

Ein Brettaufbau beschreibt eine andere – vor allem besser einsichtige – Anordnung der Elemente.

HIL-System

Hardware-in-the-Loop-Systeme zeichnen sich dadurch aus, dass elektronische oder mechanische Teilkomponenten eines Systems so an ein Computersystem angeschlossen werden, dass sie im Rahmen einer Simulation getestet werden können.

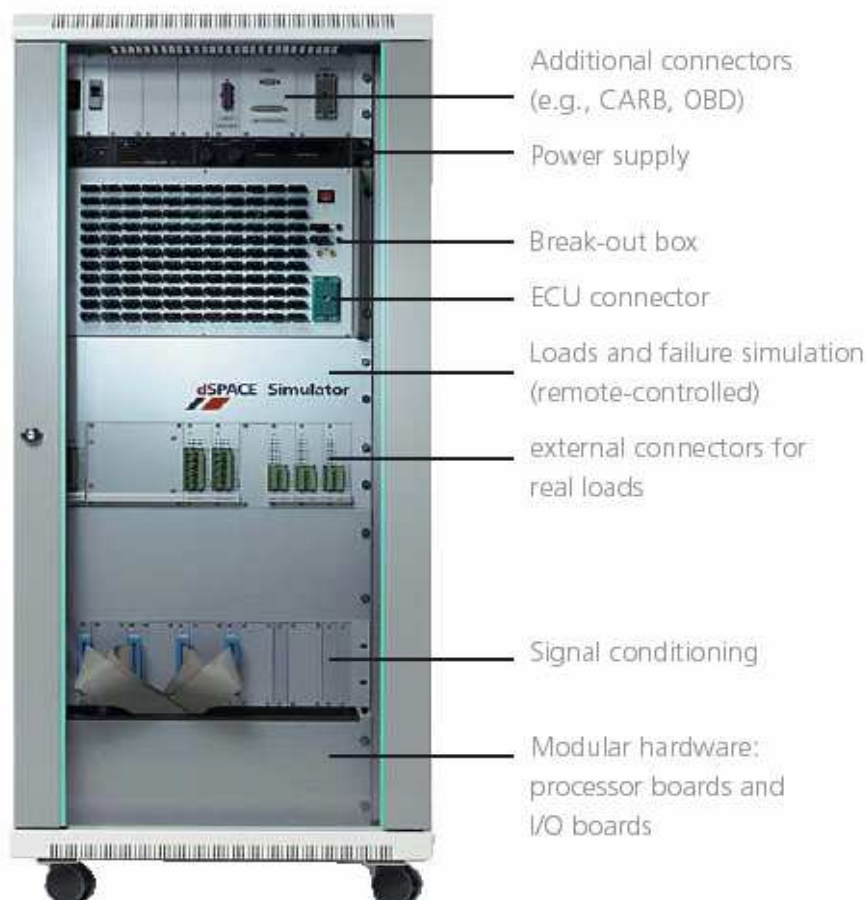


Abbildung 4: HIL-System

Dies hat im Rahmen der stufenweisen Integration von komplexen Systemen den Vorteil, dass Teile der Simulation durch konkrete HW und SW ersetzt werden. Dem/den Prüfling(en) wird eine aus seiner/ihrer Sicht hinreichende Umgebung simuliert, auf deren Einflüsse er entsprechend reagiert.

Die stufenweise Integration von komplexen Systemen ist in der SW-Entwicklung schon seit langem State-Of-the-Art und kann so auf Systeme mit Embedded-Anteilen übertragen werden. Insbesondere werden hierdurch automatisierte und damit auch regressive Tests unterstützt.

Zwecks realer Simulation bzgl. der an den Prüfling angeschlossener Lasten können diese ebenfalls konkret angeschlossen werden, Gemeinhin werden diese in einer so genannten Lastbox zusammengefasst. Nicht zuletzt bieten solche Systeme auch eine programmierbare Spannungsversorgung und in der Regel eine Fehlersimulationseinheit, mit deren Hilfe – über eine Programmschnittstelle steuerbar – elektrische Fehler, wie z.B. Kurzschlüsse oder Unterbrechungen, simuliert werden können.

Resümee

Die Welt ist komplex

Die Umgebung für den Test von Embedded-Systemen ist komplex,

- da hierbei nicht nur SW, sondern auch HW getestet wird,
- da die Systeme in der Regel auch vernetzt sind,
- da diverse Protokolle umgesetzt und damit zu simulieren bzw. zu testen sind,
- da viele und heterogene Werkzeuge im Einsatz sind.

Kölsch & Altmann hat in diesem Testumfeld mehrere Jahre Erfahrungen im Aufbau und Betrieb solcher Umgebungen, weniger was die Konfiguration der HW betrifft, als vor allem im Zusammenhang mit der Konfiguration und Anbindung an die automatisierte Test- und Simulationsumgebung. Genannt seien hier die Werkzeuge

- Matlab/Simulink
- dSPACE ControlDesk & AutomationDesk
- DiagRA (MCD-Werkzeug)
- CANape
- CANoe (u.a. CAPL-Skripts, DiVa)
- CANdela

Dabei wurden im Speziellen auch Teststellungen bearbeitet

- im Bereich Diagnose (z.B. auch Monitoring auf Basis von Fehlersimulation),
- im Zusammenhang mit Netzwerkmanagement und der Restbussimulation
- im Zusammenhang mit dem Aufweck- und Einschlafverhalten von Steuergeräten
- im Zusammenhang mit der Integration von DiagRA in die Testumgebung
- im Zusammenhang mit der Konfiguration einer ECU über CANape und des Auslesens von Zustandsinformation zum Zwecke der Verifikation von Testerwartungen

Insbesondere konnte Kölsch & Altmann auf Basis dieser Testumgebung die bislang hauptsächlich manuell an Lastboxen durchgeführten Tests überwiegend automatisieren und damit auch regressiv testbar gestalten, was angesichts dicht aufeinander folgender Releases und verschiedener Varianten im Hinblick auf die jeweilige erforderliche Freigabe unabdingbar ist.

Technik ist nicht alles

Das Anforderungsprofil eines Testingenieurs geht über das Testen hinaus:

- Das beginnt vor allem einmal mit guten Branchen-Know-how, welches man sich zwar erarbeiten kann, nicht jedoch ohne jegliches Basiswissen und selten während des Projekts zwischen eng gesetzten Meilensteinen.
- Der Tester ist Ingenieur, d.h. kann selbständig und strukturiert arbeiten, ist kommunikativ, teamfähig, kreativ und weder weniger wichtig noch weniger qualifiziert als der Software- bzw. Hardware-Ingenieur.
- Testfälle (Design & Implementierung) und Testergebnisse müssen dokumentiert und nachgewiesen werden, wegen der vielen Releases und Varianten besteht die dringende Anforderung, dass diese regressiv und automatisiert testbar sind.

Neben technischer Unterstützung ist es ständiges Bestreben von K&A, der Forderung seiner Kunden nachzukommen, nicht nur bei der technischen Umsetzung mitzuwirken, sondern Dienstleistungen auch im Rahmen der folgenden Aufgabstellungen anzubieten:

- Test-Management
(Testplanung, Ressourcenplanung, Werkzeugeinsatzplanung, Testkonzeption, ...)
- Schulung & Weiterbildung
(insbesondere in strukturierten Methoden der Software-Entwicklung und Testkonzepten)
- Qualitätssicherung
(vor allem der Testumgebung, der eingesetzten Werkzeuge und der entwickelten Testfälle)

Verzeichnisse / Links

Abkürzungsverzeichnis

<abbr>	<extension and/or explanation>
ASAM	Association for S tandardisation of A utomation- and M easuring Systems
BCM	B ody C omputer M odule, Bodycomputer (Audi)
CMMI	C apability M aturity M odel Integration
FMEA	F ailure M ode and E ffects A nalysis
HIL	H ardware I n the L oop
KWP	K ey W ord P rotocol
MCD	M easurement C alibration D iagnostics
RAM	R andom A ccess M emory, temporary memory
(P)ROM	(P rogrammable) R ead O nly M emory, permanent memory
(E)EPROM	(E lectrically) E rasable PROM
Duty Cycle	Prozentuales Verhältnis der An- zur Auszeit
PWM	P ulse W idth M odulation
SPICE	S oftware P rocess I mprovement and C apability D etermination
UDS	U niversal D iagnostic S ervices

Abbildungsverzeichnis

Abbildung 1: V-Modell und Testen

Abbildung 2: Tool-Einsatz

Abbildung 3: Simulation

Abbildung 4: HIL-System

Literatur

[<Kürzel Autorenname><2-stelliges Erscheinungsjahr>] <Name 1. Autor>, <Name 2. Autor>, ..., <Titel>, <Erscheinungsjahr (4-stellig)>

[ScZu06] Jörg Schäuffele, Thomas Zurawka,
Automotive Software Engineering, 2006

[ZiSc06] Werner Zimmermann, Ralf Schmidgall, Bussysteme in der
Fahrzeugtechnik, 2006