Automatisierte Assessments und Traceability für den Modelltest

Mehr Effizienz, Traceability und Validität für den frühen funktionalen Test

Dr. Hartmut Pohlheim, Model Engineering Solutions

Es wird eine Methode vorgestellt, die es ermöglicht, Anforderungen auf allen Testsequenzen zu testen und automatisiert auszuwerten. Somit werden der Testaufwand und die Fehleranfälligkeit reduziert. Weiterhin wird eine Methode vorgestellt, die das geforderte Requirement-Tracing in der Praxis unterstützt. Die Kombination beider Methoden in einem effizienten Test-Tool führt zu einer signifikanten Steigerung der Testabdeckung und der Verlässlichkeit der Testergebnisse.

Immer komplexer werdende Software in Verbindung mit hohen Qualitätsansprüchen macht effizientes Testen unverzichtbar. Speziell durch die ISO 26262 ist es in der modellbasierten Entwicklung unerlässlich, effiziente Techniken zum Testen zu entwickeln. Die durch die Norm geforderte Testabdeckung in Verbindung mit Requirement-Tracing und der Empfehlung zu anforderungsbasiertem Testen führen dazu, dass bisherige Testmethoden überdacht werden müssen [1].

In diesem Zusammenhang stellen wir eine Methode zur automatischen Prüfung anforderungsbasierter Modelltests vor, mit der nicht nur eine höhere Testabdeckung gewährleistet werden kann, sondern darüber hinaus ein verlässliches Tracing durch ein speziell zu diesem Zweck entwickeltes Framework möglich ist.

Die Kombination der automatisierten Assessments und des automatisierten Requirement-Tracings sind in einem Test-Tool für Simulink®-, Embedded Coder®- und TargetLink®-Modelle integriert, mit dem die Forderungen der ISO 26262 mit erheblich geringerem Aufwand erfüllt werden können [2].

Integrierter Ansatz für den anforderungsbasierten Test

Wie der Name verdeutlicht, bilden die Anforderungen die Grundlage des anforderungsbasierten Testens. Ausgehend vom Lastenheft der Software beginnt der Tester damit, alle Stimulationen der Anforderungen abzuleiten, die zum Test der Software notwendig sind. Diese dienen als Input des zu testenden Modells. Nach einer Anzahl erfolgter Simulationen, auch Testsequenzen genannt, stehen die aufgezeichneten Output-Signale zur Verfügung. Mit diesen Signalen kann nun die Testbewertung bezüglich der Einhaltung geltender Anforderungen erfolgen. Mit der Erhebung der Execution-Coverage kann zusätzlich geprüft werden, ob die angewandte Menge an Stimulationen für eine Bewertung ausreichend ist (Abb. 1).

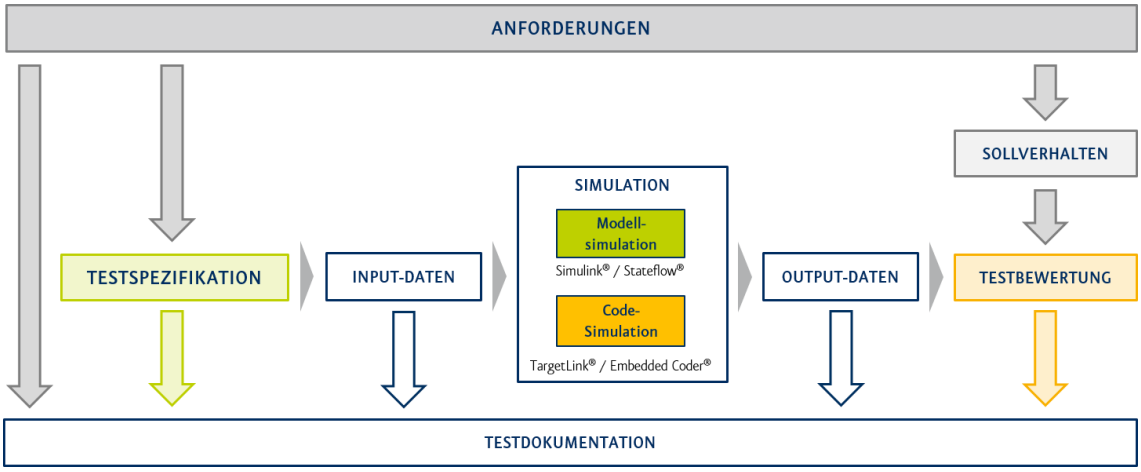

Abb. 1: Allgemeiner Ablauf des anforderungsbasierten Tests.

Klassischerweise erfolgte eine Software-Bewertung durch visuelle Begutachtung der Ausgangssignale. Dadurch müssen die Stimulationen oft sehr einfach gehalten werden, um die aufwendige manuelle Bewertung zu vereinfachen oder gar erst zu ermöglichen.

Dieser Beitrag präsentiert einen weiterentwickelten Ansatz zur verbesserten und vor allem automatisierten Bewertung anforderungsbasierter Testfälle. Insbesondere bricht er mit der strengen Verbindung von Testfällen und Anforderungen. Die automatische Bewertung erfolgt durch ein entsprechend entwickeltes Framework zur formellen Prüfung von Anforderungen. Dieser Ansatz zeichnet sich dadurch aus, dass eine einmalig in einem Assessment formulierte Prüfung wiederverwendbar ist und auf allen erdenklichen Systemantworten angewendet werden kann. Dies unterstreicht zum einen den allgemeingültigen Charakter von Anforderungen und hilft zum anderen Fehler aufzudecken, die zuvor nicht antizipiert worden sind.

Diese Methode beruht generell auf drei charakteristischen Elementen: der Anforderung, der Testspezifikation und den Assessments. Um deren starke Abhängigkeiten untereinander stets zu überblicken, ist es im Sinne des in der ISO 26262 geforderten Tracings unablässig, diese jederzeit nachzuverfolgen. Hierfür wurde das Framework *„Requirements, Coverage, and Traceability“* (RCT) entwickelt, welches auf komfortable Weise die Anforderungsverfolgung unterstützt.

Die Kombination beider Ansätze in einem Test-Tool ermöglicht somit einen integrierten Ansatz für einen modern gestalteten und effektiven Testprozess (Abb. 2).

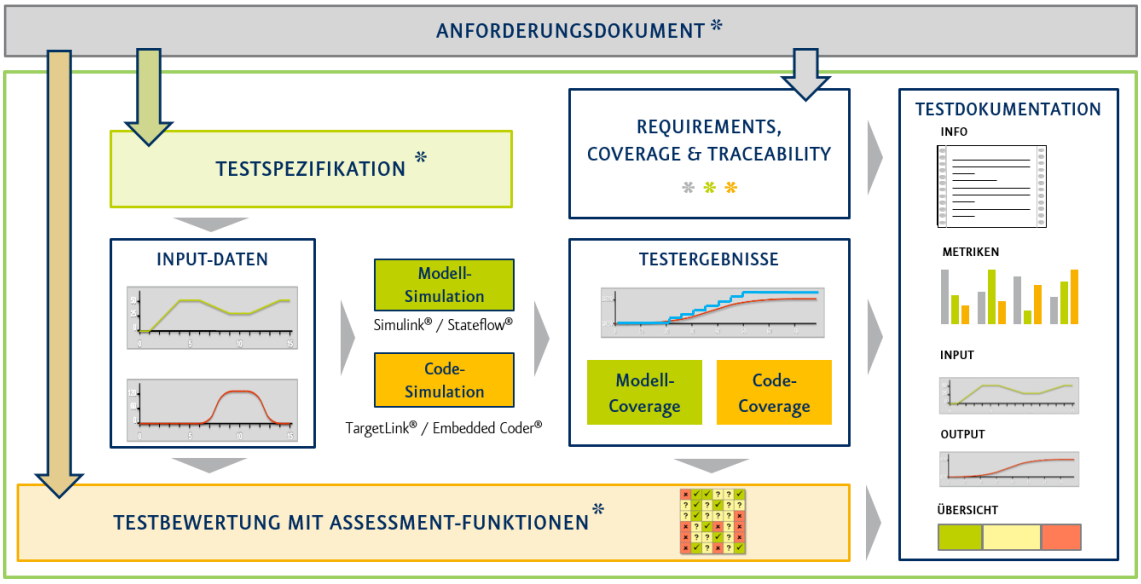

Abb. 2: Anforderungsbasierter Test mit Assessments und Anforderungsverfolgung.

Automatische Testevaluierung mit Assessments

Ein Grundprinzip der Assessments ist deren Wiederverwendbarkeit. Ein Assessment kann als eine Funktion verstanden werden, welches die aufgezeichneten In- und Output-Signale sowie die lokalen Signale und Parameter untersucht (Abb. 3).

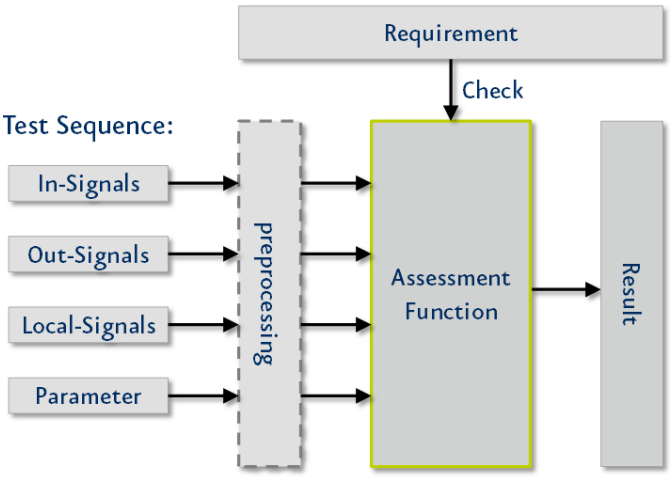

Abb. 3: Ein- und Ausgänge eines Assessments.

Hiermit prüft das Assessment die Invarianz des Systemverhaltens und stellt das Ergebnis für jede Sequenz einzeln bereit. Dabei sind drei Ergebniswerte möglich: *Passed* (das Verhalten entspricht der geltenden Anforderung), *Failed* (im Verlauf der Testsequenz

wurde eine Verletzung der Anforderung detektiert) und *NotTriggered* (die Testsequenz hat nicht zu einer Stimulation dieser Anforderung geführt).

Im Gegensatz zu einer direkten Prüfung von *n* Testsequenzen mit genau *n* erwarteten Ergebnissen (Abb. 4, links), liefert die Matrixevaluierung mit Hilfe von Assessments eine deutlich höhere Testabdeckung durch *m***n* Evaluierungen, da hier ein Satz von *m* Anforderungen, durch *m* Assessments geprüft wird und diese Prüfung auf allen *n* Testsequenzen erfolgt (Abb. 4, rechts). Die Matrixevaluierung ermöglicht zudem die Aggregation der Ergebnisse auf Ebene der Testsequenzen und Assessments. Dies führt unmittelbar zu einer höheren Verlässlichkeit der Prüfergebnisse.

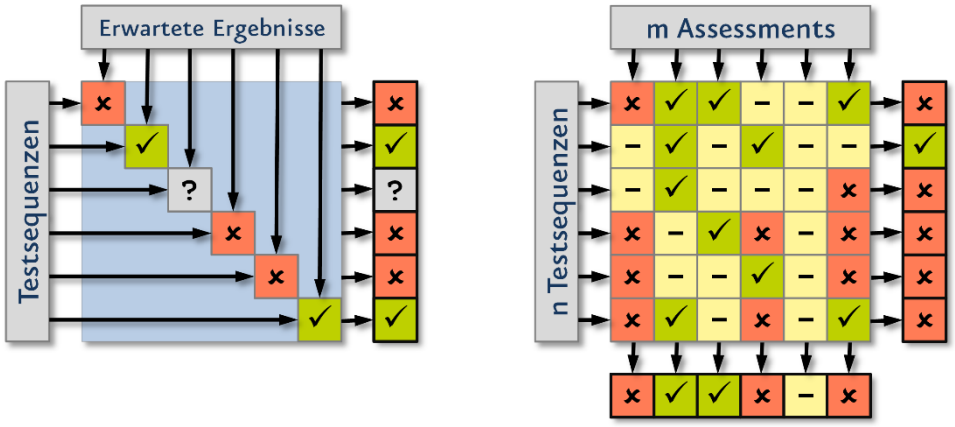

Abb. 4: Links: Geringe Testabdeckung durch Vergleich mit erwarteten Ergebnissen. Rechts: Hohe Testabdeckung durch Matrixevaluierung mit Assessments.

Um diese Reihe von Aufgaben erfüllen zu können, besteht ein Assessment aus einem beschreibenden Anteil und einem evaluierenden Ausführungsteil. Der erste Teil umfasst u.a. die Traceability-Informationen und einem Beschreibungstext. Im zweiten Teil finden als Post-Processing der Testsequenz der Signalzugriff und deren Bewertung statt (Abb. 5).

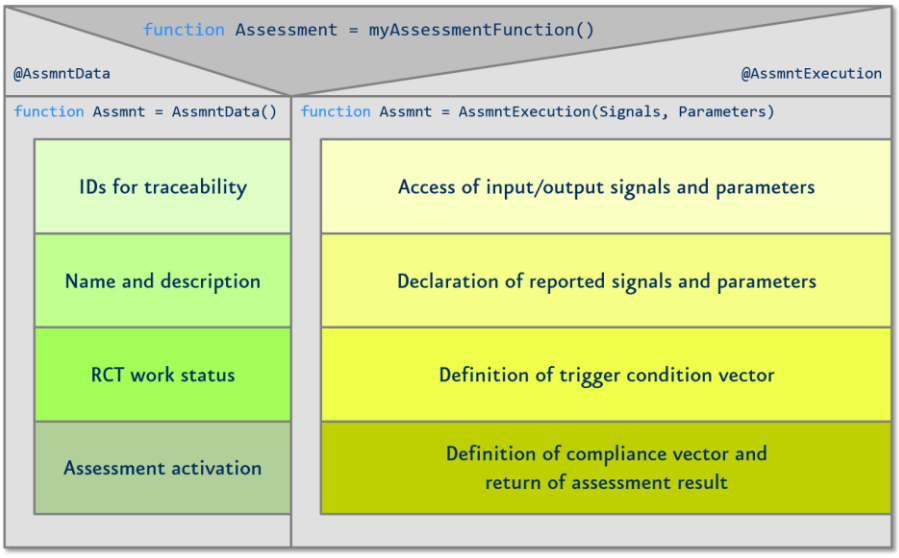

Abb. 5: Aufbau einer Assessment-Funktion.

Um einen schnellen Einstieg zu ermöglichen steht ein Template-Generator zur Verfügung. Anforderungen und benötigte Signale können ausgewählt werden und mit wenigen Eingaben wird automatisch ein passendes Template für die Assessments bereitgestellt.

Selbst zu befüllen sind im Beschreibungsteil der Name und die Description des Assessments (Abb. 6). Für die Evaluierung muss unter *Triggered* die Bedingung formuliert werden, unter der die Anforderung gültig ist. Mit den bereitgestellten Hilfsfunktionen kann daraus die Prüfung implementiert werden, wie im gezeigten Beispiel eine klassische „Wenn-Dann“-Beziehung. Zulässige Reaktionszeiten des Systems können ebenfalls berücksichtigt werden (Abb. 7).

Nachdem alle Testsequenzen zu der Evaluierung einer Anforderung beigetragen haben, können deren Ergebnisse aggregiert werden. Die Darstellung des Gesamtergebnisses mit zugehörigen Metriken unseres Beispiels ist in Abb. 8 gegeben. Hierbei genügt eine einzige Verletzung der Anforderung um als Endresultat ein *Failed* zu erhalten.

```
%% Requirement Text:
%       AP_ML_req_01:
%         Glide-slope control is armed when its Glide Slope Enable 'gs_cmd' and
%         Glide Slope Signal 'gs_signal' are both true.
```

```
%% Set here the descriptive (static) attributes of the assessment!
function Assmnt = AssmntData()

   % ID of assessment (string)
   Assmnt.ID = 'AP_ML_req01';

   % Name of assessment (string)
   Assmnt.Name        = 'Check Signal Armed';

   % Description of assessment behavior (cell array of strings)
   Assmnt.Description = {'Trigger: gs_cmd and gs_signal are both TRUE.' ''...
                         'Check: When both signals true then signal Armed is true.'};

   % IDs of covered requirements (cell array of strings)
   Assmnt.ReqIDs = {'AP_ML_req_01'};

   % Assessment work status (string)
   % Valid entries are e.g.:
   %  'created'            -  Item created.
   %  'reqchanged'         -  Item has to be inspected due to changed requirement.
   %  'inwork'             -  Item still in work.
   %  'reviewed'           -  Item passed the review process.
   Assmnt.Status = 'reviewed';
end
```

Abb. 6: Deskriptiver Teil einer Assessment-Funktion zur Dokumentation wichtiger Informationen.

```
%% Implement here the evaluation algorithm!
function Assmnt = AssmntExecution(Signals, Parameters)

   % Import of signals
   gs_cmd    = MTa_getSignal(Signals, 'gs_cmd');
   gs_signal = MTa_getSignal(Signals, 'gs_signal');

   Armed     = MTa_getSignal(Signals, 'Armed');

   % List of signals that should be shown in assessment analyzer (cell array of strings)
   Assmnt.ReportSignals = { 'gs_cmd', 'gs_signal', 'Armed'};

   % Import of parameters
   % Example = Parameters.Example;

   % Time intervals or events where the requirement should be triggered
   % (logical or numeric vector)
   Assmnt.Triggered = gs_cmd & gs_signal;

   % Test requirement
   if any(Assmnt.Triggered),

      % Vector of length of sequence containing non-zero at passed positions and
      % zero at failed positions (logical or numeric vector)
      delayTol = [0 0]; % Allowed delay for reaction

      % When 'Triggered' is TRUE then 'Armed' is TRUE
      Assmnt.Complied = MTa_checkReaction(Armed, Assmnt.Triggered, 'alltrue', delayTol);

   end
end
```

Abb. 7: Ausführbarer Teil einer Assessment-Funktion mit hinterlegtem Evaluierungsalgorithmus.

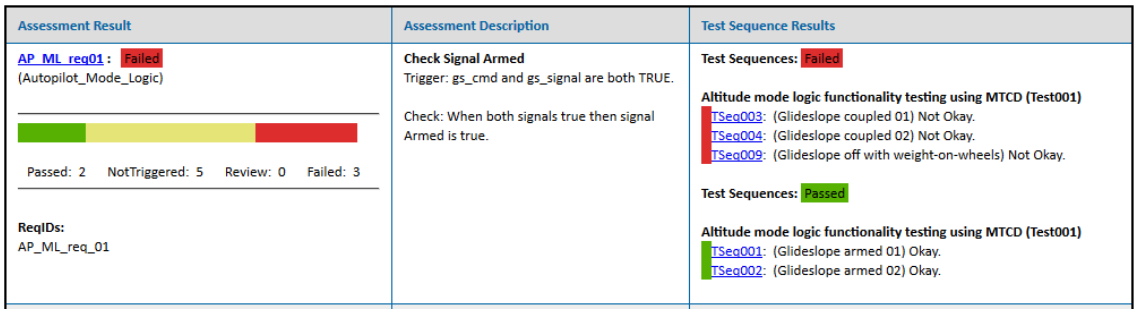

Abb. 8: Dokumentation der Testbewertung mehrerer Testsequenzen durch ein Assessment im Assessmentkatalog.

Anforderungsverfolgung und Darstellung der Ergebnisdaten

Charakteristisch für den vorgestellten Modelltest ist, dass Testsequenzen wie Assessments stets Bezug auf Anforderungen nehmen. Hieraus entsteht eine Vielzahl an Verknüpfungen (Abb. 9), deren Nachverfolgung in der Praxis oft eine große Herausforderung für den Tester darstellt.

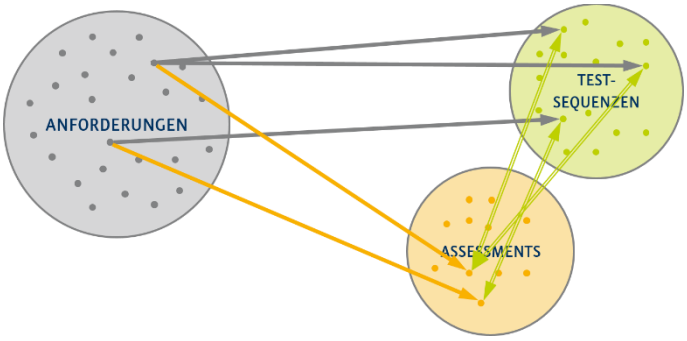

Abb. 9: Schematische Darstellung der Zusammenhänge zwischen Anforderungen, Testsequenzen und Anforderungen.

Um diesen Prozess zu unterstützen, haben wir das RCT-Framework entwickelt. Es folgt automatisiert den Verknüpfungen und ermöglicht somit die klare Nachverfolgung von Anforderungen. Als zentrale Elemente dienen hier die IDs der Anforderungen. Der Tester muss lediglich auf diese Bezug nehmen – sowohl innerhalb der Testspezifikationen, als auch der Assessments. Dieses Framework stellt hierfür die entsprechenden Möglichkeiten bereit (Abb. 10). Eine Methode zur Testspezifikation (MTCD), die den hier vorgestellten Prozess unterstützt ist in [3] vorgestellt.

Sind alle Bezüge hinterlegt, wird eine Berichtsübersicht bereitgestellt, die nicht nur alle geltenden Anforderungen zusammenfasst, sondern auch detaillierte Informationen über die jeweiligen Testsequenzen und Assessments liefert – sowie Angaben zu deren Bearbeitungsfortschritt macht (Abb. 11).

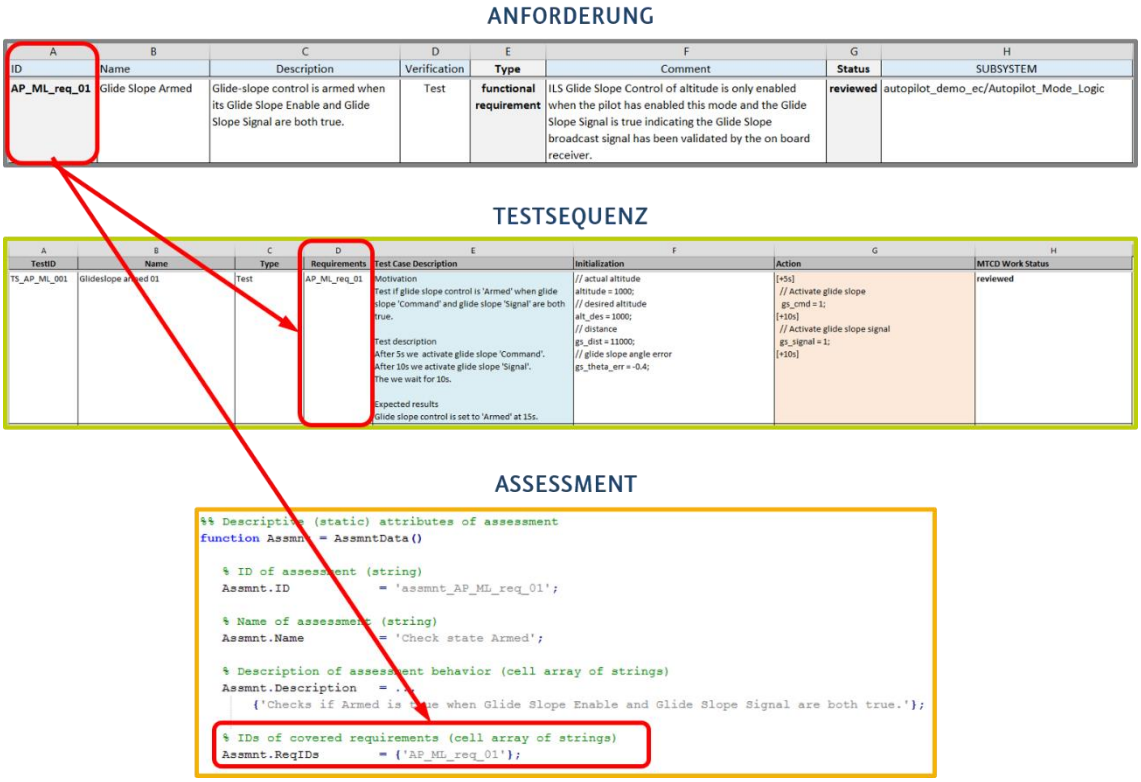

Abb. 10: Zum Prinzip der Anforderungsverfolgung mit Hilfe der ID.

ID	Description	Testability	Testability Review Status	Related Test Sequences	Number Of Test Sequences	Test Sequence State Monitor	Related Assessments	Number Of Assessments	Assessment State Monitor
AP_ML_in_01	This document describes a set of requirements that apply to the altitude mode logic portion of an autopilot controller. The autopilot controller can be activated by the pilot and has two mutually exclusive tasks: - In mode 'altitude control' the controller maintains and adjusts the altitude of the plane at the desired value. - In mode 'glide slope' it supports the approach for a landing when the plane is close to the airport.	No	reviewed		0			0	
AP_ML_hd_02		No	reviewed		0			0	
AP_ML_req_01	Glide-slope control is armed when its Glide Slope Enable and Glide Slope Signal are both true.	Yes	reviewed	TS_AP_ML_001 TS_AP_ML_002	2	100%	assmnt_AP_ML_req_01	1	100%
AP_ML_req_02	Glide Slope control becomes coupled when the control is armed and (Glide Slope Angle Error > 0) and Distance < GSMINDIST.	Yes	reviewed	TS_AP_ML_003 TS_AP_ML_004	2	45%	assmnt_AP_ML_req_02	1	100%
AP_ML_req_03	Altitude climb control is entered whenever: |Actual Altitude- Desired Altitude | > ALTMAXDIST. Units: Actual Altitude – feet Desired Altitude – feet	Yes	reviewed	TS_AP_ML_005 TS_AP_ML_007	2	75%	assmnt_AP_ML_req_03	1	10%
AP_ML_req_04	Altitude hold mode is entered whenever: |Actual Altitude- Desired Altitude | < 30*Sample*Period*(Pilot Climb Rate / 60)	Yes	reviewed	TS_AP_ML_006 TS_AP_ML_007	2	45%	assmnt_AP_ML_req_04	1	100%
AP_ML_req_05	Altitude hold control and altitude climb control are disabled when autopilot enable is false.	Yes	reviewed	TS_AP_ML_008	1	100%	assmnt_AP_ML_req_05	1	50%
AP_ML_req_06	Glide-slope control is disabled when the Weight on Wheels flag is true.	Yes	reviewed	TS_AP_ML_009	1	50%	assmnt_AP_ML_req_06	1	80%
AP_ML_req_07	Altitude control is disabled when the Weight on Wheels flag is true.	Yes	notreviewed	TS_AP_ML_010	1	80%	assmnt_AP_ML_req_07	1	70%

Abb. 11: Übersicht des RCT-Frameworks über die Anforderungen und aller mit ihnen verknüpften Testsequenzen und Test Assessments.

Das RCT-Dashboard enthält außerdem wichtige Angaben über den Erfüllungsgrad der Anforderungen und deren Abdeckung mit Testsequenzen und Assessment, was eine Aussage zur funktionalen Güte des getesteten Software-Moduls ermöglicht. Metriken zur Bewertung der Testqualität werden ebenfalls bereitgestellt. Darunter zählen u.a. der Fortschrittsgrad des Reviews von Testsequenzen und Assessments sowie die Werte für Modell- bzw. Code-Coverage (Abb. 12).

Results	Percentages	Target Value	Absolute
Requirements Compliance	43%	100%	3
Test Sequences Compliance	60%	100%	6
Assessments Compliance	43%	100%	3

Requirements	Percentages	Target Value	Absolute
Total Requirements			9
Requirements with Reviewed Testability	89%	100%	8
Testable Requirements	78%		7
Testable Requirements with Test Sequences	100%	100%	7
Testable Requirements with Assessments	100%	100%	7

Test Sequences	Percentages	Target Value	Absolute
Total Test Sequences			10
Reviewed Test Sequences	30%	100%	3
Stimuli Trigger Performance	86%		6
Test Sequence State Monitor	69%		6,9

Assessments	Percentages	Target Value	Absolute
Total Assessments			7
Reviewed Assessments	43%	100%	3
Assessment Trigger Performance	86%		6
Assessment State Monitor	73%		5,1

Execution Coverage	Percentages	Target Value	Absolute
Model Condition Coverage	83%	80%	20
Model Decision Coverage	69%	80%	22

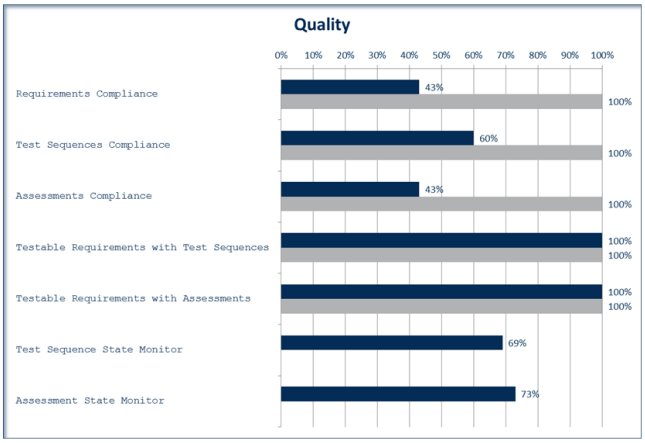

Abb. 12: Tabellarische und grafische Darstellung der Anforderungserfüllung und Testabdeckung eines Software-Moduls, wie es mit dem RCT-Framework dargestellt wird.

Umgang mit Anforderungsänderungen

Der Report des RCT-Frameworks dient zum einen als wichtiges Dokumentationselement. Zum anderen ist es ein effektives Arbeitsinstrument, in dem der Tester seinen Arbeitsaufwand ablesen kann, insbesondere bei Änderung des Anforderungsdokuments.

Durch das RCT-Framework kann auf einfache Weise das neue Anforderungsdokument in das bestehende Projekt geladen werden. Die Differenzanalyse von RCT identifiziert die geänderten Anforderungen und setzt den Bearbeitungstand der mit ihnen verknüpften Elemente zurück. Hierdurch sinken in der Praxis zunächst die Kennzahlen des Testprojekts. Des Weiteren wird die Teilmenge an betroffenen Testsequenzen und Assessments angezeigt, sodass eine gezielte und somit zeitsparende Revision dieser Elemente möglich ist – stets mit dem Ziel, die ursprünglichen Kennzahlen wieder zu erreichen bzw. zu übertreffen (Abb. 13).

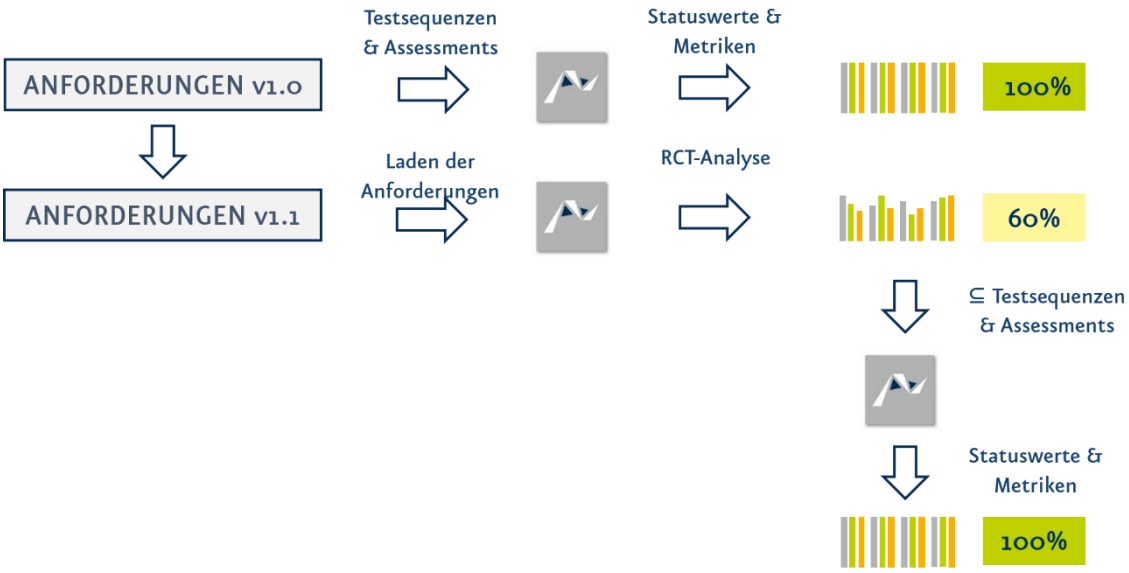

Abb. 13: Zum Prinzip des Umgangs mit Anforderungsänderungen mit dem RCT-Framework.

Zusammenfassung

Der vorliegende Betrag zeigt, wie es methodisch möglich ist, zu einer höheren Testabdeckung zu gelangen. Die Anwendung des hierfür entwickelten Frameworks zur automatischen Anforderungsprüfung mittels Assessments unterstützt den Anwender dadurch in der Praxis im hohen Maße. In Zukunft werden Testsequenzen mehr und mehr die Aufgabe übernehmen, dass zu testende System ausreichend zu stimulieren, um die hohen geforderten Coverage-Werte zu erzielen. Die Prüfung und Sicherstellung der Anforderungserfüllung erfolgt effizient durch die Assessments.

Zudem wurde gezeigt, dass auch das in der Norm geforderte Tracing tool-basiert und effizient unterstützt werden kann. Die Anwendung des entsprechenden Frameworks ermöglicht bereits im laufenden Testprozess nicht nur eine Bewertung der funktionalen Güte, sondern Aussagen über die Qualität des Tests als solches.

Die Kombination beider Ansätze zeigt, wie schon heute ein moderner anwendungsbasierter Modelltest gestaltet sein kann.

Literatur

[1] ISO, *ISO 26262: Road vehicles -- Functional safety*, International Standard ISO/FDIS 26262, 2011.

[2] Model Engineering Solutions GmbH, 2015. [Online]. Available: http://www.model-engineers.com/de/mtest.html.

[3] Schmidt, Tobias, et al. *Efficient Testing Framework for Simulink Models with MTCD and Automated Test Assessments in the Context of ISO 26262*. SAE International Journal of Passenger Cars-Electronic and Electrical Systems 7.2014-01-0306 (2014): 166-177.

Autor

Hartmut Pohlheim ist das stabile Rückgrat der Technologieentwicklung der Model Engineering Solutions GmbH. Er ist ein CTO aus Leidenschaft und stemmt Kundenprojekte sowie die interne Softwareentwicklung selbst unter hohem Zeit- und Qualitätsdruck. Besonders geschätzt wird sein unerschöpfliches Fachwissen und seine pragmatische Herangehensweise an herausfordernde Technologiethemen.

Kontakt

Internet: www.model-engineers.com
Email: pohlheim@model-engineers.com