NeGSt

Neue Generation Signaltechnik

Sektorweite Initiative zur Sicherung der Zukunftsfähigkeit der Leit- und Sicherungstechnik

Teilbericht

AP 2300

Ansätze zur Optimierung toolgestützter Teststrategien

09.08.2013

Gefördert durch:

Bundesministerium für Wirtschaft und Technologie

aufgrund eines Beschlusses des Deutschen Bundestages

Laufzeit: 01.09.2011 – 31.08.2013

Projektträger: TÜV Rheinland Consulting GmbH

Änderungsverfolgung

Datum	Bearbeiter	Version	Inhalt
12.04.2013	Suter	0.1	Erstellung Grobkonzept
28.05.2013	Suter	0.2	Erstellung Feinkonzept
24.06.2013	Suter	0.3	Erste Ausgabe
02.07.2013	Suter	0.4	Abschluss
04.07.2013	Möller-Neustock	0.5	Review
05.07.2013	Suter	0.6	Abschluss
17.07.2013	Neustock	0.7	Review
09.08.2013	Neustock Winkler	1.0	Finalisierung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Innerhalb des Ergebnisberichts "NeGSt_Ergebnisbericht_2300AG5_Prozess der RAMS-Nachweisführung und unterstützende Werkzeuge" [NeGSt_Teilbericht] wurde dargelegt, dass die Beachtung der Sicherheitsanforderungen im gesamten RAMS-Prozess durchgängig, mögöochst toolgestützt, nachgewiesen werden muss. Es wurde festgestellt, dass die globalen Sicherheitsanforderungen des RAMS-Prozess ab der Phase des Designs einer Untersuchung, sowie einer Detaillierung unterliegen. Ab dieser Phase muss somit sichergestellt sein, dass die durchgängige Nachweiskette bis zur Erstellung des Sicherheitsnachweises eingehalten wird.

In diesem Dokument wird die Fehlerverfolgung parallel zum Entwicklungsprozess betrachtet, also bereits ab der Design- und Entwurfsphase, über das eng damit verbundene Thema „Erstellen, Durchführen und Verwalten von Tests“ bis zur Nachweisdarlegung im Sicherheitsnachweis (TSB). Im Folgenden wird nicht die besondere Klassifizierung der Sicherheitsanforderungen gemäß [NeGSt_Teilbericht] thematisiert, sondern ein generelles Konzept der Fehlerverfolgung, unabhängig von der Klassifizierung, erstellt. Weitere Analysen, inwieweit die Fehlerverfolgung gemäß RAMS-Anforderungen erfolgen kann (z.B. über eine spezielle Attribuierung), sollten in einer weitergehenden Evaluierung erfolgen.

Eine optimale Abstimmung der angewandten Prozesse und der verwendeten Tools wird im Sinne der sicherheitsorientierten Entwicklung angestrebt und spielt auch aus wirtschaftlicher Sicht, bezogen auf Aufwand, Kosten und nachhaltige Effizienz, eine wichtige Rolle.

Nachfolgend soll beispielhaft das Testkonzept der Firma Funkwerk für das Projekt „Lindaunis“ (ein ESTW-R vom Typ Alister 2.0) vorgestellt werden.

Es sollen Möglichkeiten für Verbesserungen im Testkonzept sowie in der Toolchain herausgearbeitet werden. Zusätzlich erfolgt eine Evaluierung des modernen Application Lifecycle Management (ALM) Tools „Polarion“, um die Aspekte des Managements von Fehlern zu betrachten.

2 Grundlagen und Definitionen zur Fehlererkennung, -verfolgung und zum Fehlermanagement

2.1 Fehlerdefinition

Ein Fehler (Ausfall) wird laut dem Deutschen Institut für Normung als ein „Merkmalswert, der die vorgegebenen Forderungen nicht erfüllt“ und als „Nichterfüllung einer Anforderung“ definiert. Die Anforderung wird hierbei als „Erfordernis oder Erwartung, welche festgelegt, üblicherweise vorausgesetzt oder verpflichtend ist“ definiert. Unterschieden werden erwartete Fehler und unerwartete Fehler.

Ein Fehler kann systematisch sein, d.h. er hängt von Fehlervoraussetzungen ab. Beim Fehlerauftreten unter bekannten Fehlervoraussetzungen, kann der Fehler reproduziert werden.

Ein Fehler kann nur vermieden werden, wenn die Fehlerursache bekannt ist. Da die Folgen eines Fehlers generell unerwünscht sind, wird er häufig als Schaden betrachtet, denn er tritt oft abrupt oder von Beginn an auf. Zur besseren Beurteilung eines Fehlers, wird dieser Gefährdungsklassen zugeordnet.

In der Technik sind Fehler überwiegend auf menschliche Fehler in der Konstruktionsphase oder im Produktionsprozess zurückzuführen. Die FMEA (Failure Mode and Effects Analysis oder Auswirkungsanalyse) versucht alle möglichen Fehler, die Fehlerfolgen und die schlimmsten möglichen Fehlerverkettungen systematisch zu erkennen und zu bewerten, um entsprechende Maßnahmen rechtzeitig vorzunehmen.

2.2 Gefährdungsklassen

Gefährdungsklassen dienen der Klassifizierung von relativen Auswirkungen eines technischen Fehlers. Durch eine genaue Definition der Gefährdungsklasse und der Umsetzungen aus der definierten Fehlerstrategie ist eine rechtzeitige Reaktion auf den Fehler sichergestellt und es ist möglich, die Art des Fehlers zu beurteilen. Es gibt folgende hier in Ansatz gebrachte Klassifizierungsmöglichkeiten:

- Klasse 1 (nicht gefährliche Ausfälle, Werte müssen in den Sicherheitsberechnungen nicht mit betrachtet werden; sie sind nicht relevant)
- Klasse 2 (gefährliche Ausfälle, sind relevante Werte für die Sicherheitsberechnung)

2.3 FMEA

Die FMEA dient dazu, Schwächen frühzeitig zu erkennen und liefert eine „Rangliste“ von Ausfallarten, welche verschiedene Auswirkungen auf das System haben. Diese Betrachtungen fließen dann in die RAMS-Berechnungen eines Systems mit ein.

2.4 Grundlagen

2.4.1 Sicherheitsintegritätslevel (SIL)

Das Sicherheits-Integritätslevel SIL (Safety Integrity Level) gibt die Ausfallgrenzwerte für eine Sicherheitsfunktion, die in der Betriebsart mit hoher Anforderungsrate oder in der Betriebsart mit kontinuierlicher Anforderung betrieben wird an.

Sicherheits-Integritätslevel	Rate gefahrbringender Ausfälle der Sicherheitsfunktion
4	$\geq 10^{-9} bis < 10^{-8}$
3	$\geq 10^{-8} bis < 10^{-7}$
2	$\geq 10^{-7} bis < 10^{-6}$
1	$\geq 10^{-6} bis < 10^{-5}$

Tabelle 1: Einteilung der Sicherheitsintegritätslevel

2.4.2 Fehlerbaumanalyse (FTA)

Die Fehlerbaumanalyse (FTA) ist ein „ableitendes“ bzw. „fortführendes“ Verfahren, welches ein System, bezogen auf seinen Komponenten, in einem booleschen Modell darstellt. Diese Darstellung dient dazu, die Wahrscheinlichkeit eines System-Ausfalls zu bestimmen.

2.5 Fehlererkennung

In der heutigen Zeit wird eine hohe Zuverlässigkeit von einem Gerät dadurch erzielt, dass es durch ein gutes Konzept, durch hohe Qualität der verwendeten Komponenten und durch einen gut überlegten Systemaufbau entwickelt worden ist. Dazu ist es besonders hilfreich, wenn Fehler schon in der Entwicklungsphase erkannt werden, um Schwachstellen rechtzeitig beheben zu können.

2.5.1 Ausfallbetrachtungen

Durch RAMS-Nachweisdokumentationen wird dargelegt, wie Fehler rechtzeitig erkannt und behoben werden. Die RAMS-Prozesse (Reliability-Zuverlässigkeit, Availability-Verfügbarkeit, Maintainability-Instandhaltbarkeit und Safety-Sicherheit) beschreiben den Systemlebenszyklus und beinhalten zum einen die Ergebnisse aus den Berechnungen der Zuverlässigkeit und der Sicherheit und zum anderen die Ergebnisse aus den Verfügbarkeits- und Instandhaltbarkeitsbetrachtungen.

Die Untersuchung der Ausfallraten (Anzahl der (gefährlichen) Ausfälle je Zeiteinheit) von Komponenten eines Systems hat hierbei eine große Bedeutung. Sie hilft dabei, rechnerisch eine Zuverlässigkeit vorauszusagen. Es ist somit möglich, einen Fehler durch vorsorgliche Reparatur oder Austausch zu vermeiden. Die Ausfallrate einer Baugruppe (System) wird mit Hilfe der FMEA oder der Siemens-Norm SN 29500 ermittelt. Die Ausfallraten für die verwendeten Bauteile werden vom Hersteller angegeben oder mit Hilfe der Daten aus der Siemens-Norm bestimmt.

Die Fehlerbaumanalyse (FTA) dient zur Bestimmung der Wahrscheinlichkeit eines Systems-Ausfalls. Wichtig ist auch die Betrachtung der Ursache für den Fehler und die Häufigkeit seines Auftretens. Daraus sind präventive Maßnahmen, wie z.B. einen neuen Aufbau des Systems, eine rechtzeitig durchgeführte Reparatur oder ein Geräteaustausch, abzuleiten.

2.5.2 Bauteilbetrachtung

Bauteile eines Systems werden gemäß ihrer Funktion zu Funktionsblöcken zusammengesetzt. Diese werden dann nach Beteiligung einer bestimmten Funktion in Serie oder parallel redundant zusammengeschaltet.

2.5.3 Maßnahmen zu einer höheren Wahrscheinlichkeit zur Aufdeckung von Fehlern

Eine Maßnahme zur Vereinfachung von Fehlerermittlungen und –behebung ist ein Diagnosesystem, welches technische Betriebsdaten systematisch und vollständig protokolliert.

Eine entsprechende Diagnoseschnittstelle dient dazu, Störungsdaten zu erfassen, aufzubereiten und statisch auszuwerten.

Die dazu notwendigen Informationen vom Stellwerk und der Bedieneroberfläche sollen sowohl aktuell als auch aus archiviertem Datenbestand dem Servicepersonal der LST zur Verfügung stehen. Darüber hinaus soll eine gut strukturierte Übersicht und Klassifizierung der Störungsarten geschaffen werden. Eine wichtige Hilfestellung dafür ist, dass die möglichen betroffenen Bauteile zu der aufgetretenen Störung angezeigt werden und somit eine schnelle Reparatur oder ein schneller Austausch des Bauteils bzw. der Baugruppe vorgenommen werden kann.

3 Ansätze zur Optimierung toolgestützter Umsetzung von Teststrategien

In diesem Abschnitt soll ein Überblick über die im Projekt ESTW-R Lindaunis angewandte Teststrategie zur Fehlererkennung und –management gegeben werden.

Die Ideen für verbesserte Konzepte, Prozesse und Tools sollen aus den Erfahrungen, die im Projekt ESTW-R Lindaunis gesammelt wurden, zusammengetragen werden. Um die Anforderungen an die Testumgebung besser nachvollziehen zu können, soll das Projekt, dessen Architektur und Toolchain kurz beschrieben werden.

3.1 Überblick über die Zielplattform

Beim Projekt handelt es sich um ein ESTW-R vom Typ Alister 2.0, ein auf speicherprogrammierbaren Steuerrungen (SPS) basierendes Stellwerk aus Standardindustriekomponenten (Commercial off-the-Shelf - COTS). Konkret besteht die Stellwerksplattform aus F30-HiMatrixsystemen der Firma Hima.

Man unterscheidet hier zunächst zwischen zwei Teilsystemen, dem Subsystem eines technisch verfahrensgesicherten Bedienplatzes und dem Subsystem Stellwerkskern.

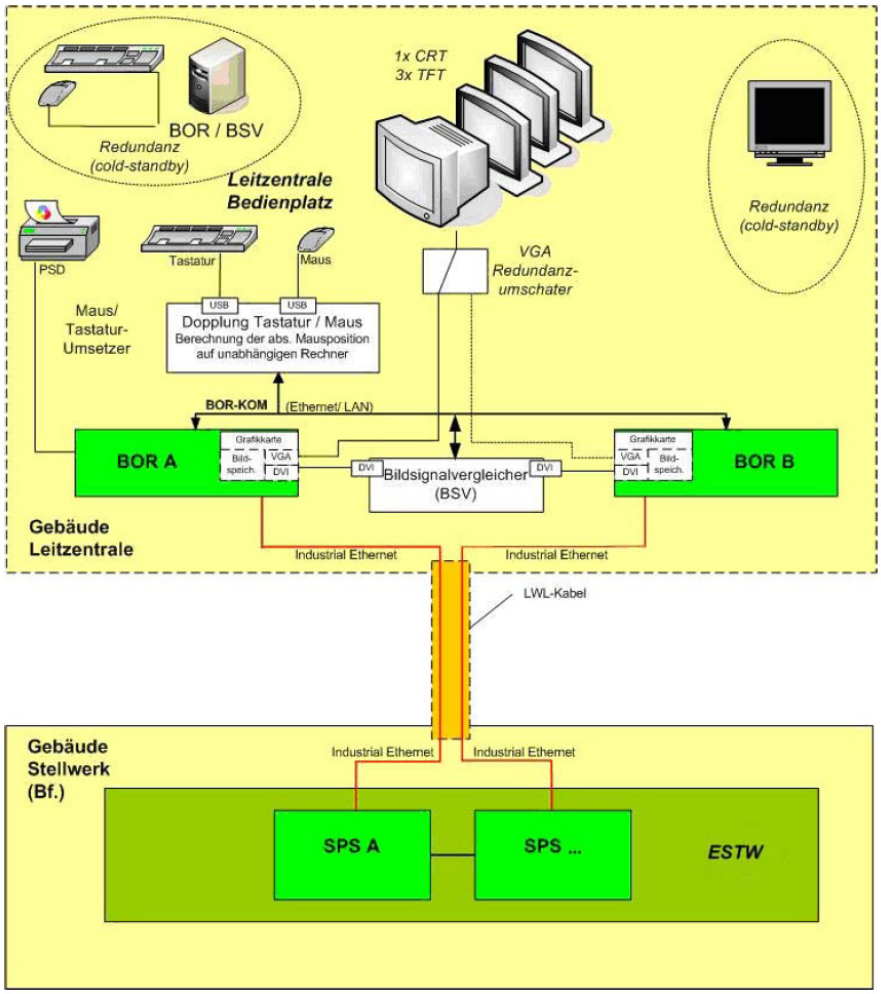

Abbildung 1: Überblick Systemkonzept

Der Stellwerkskern ist wiederum in die Teilsysteme „Generisches Produkt“ (GP) und „Generische Applikation“ (GA) unterteilt.

Das Generische Produkt besteht im Wesentlichen aus dem Stellwerkskern (SPS) und den Anschaltbaugruppen (xCM) zu Ansteuerung der Außenelemente. Das GP umfasst auch die Treiberebene für

die Außenelemente und soll die Stellwerkslogik der GA von den jeweils eingesetzten Hardwarekomponenten entkoppeln.

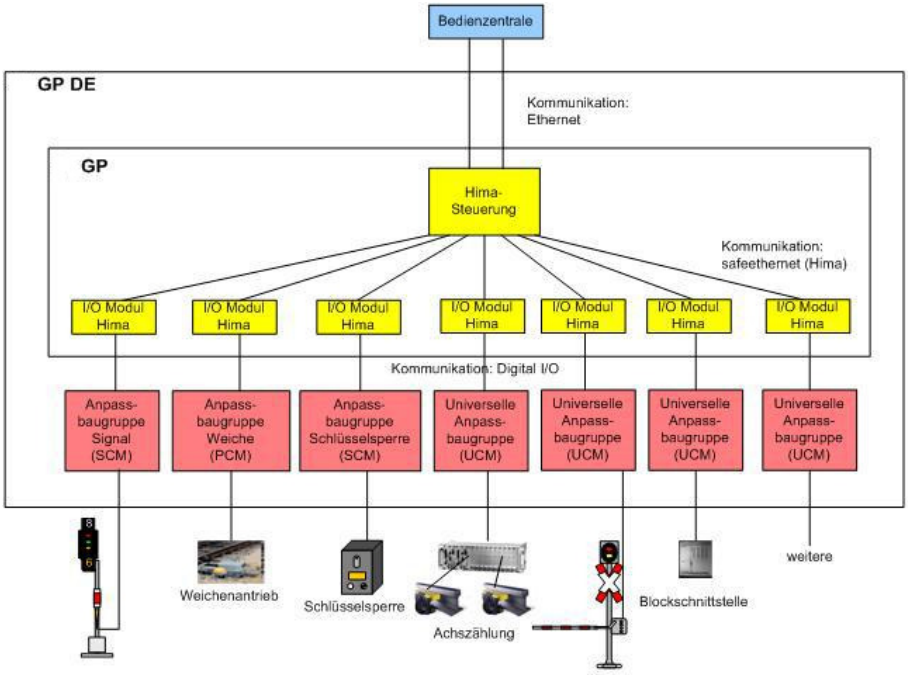

Abbildung 2: Generisches Produkt

Der GA-Teil beinhaltet hauptsächlich die Stellwerkslogik, wie z.B. das Einstellen und Auflösen von Fahrstrassen und ist für jedes Stellwerk gleich.

Die Anpassung auf die örtlichen Gegebenheiten erfolgt, gemäß dem generischen Ansatz durch Parametrisierung über einen SA-Anteil. SA definiert eine Spezifische Applikation und beinhaltet zum Beispiel die örtlichen Stellwerkselemente sowie die einzelnen Fahrstraßen.

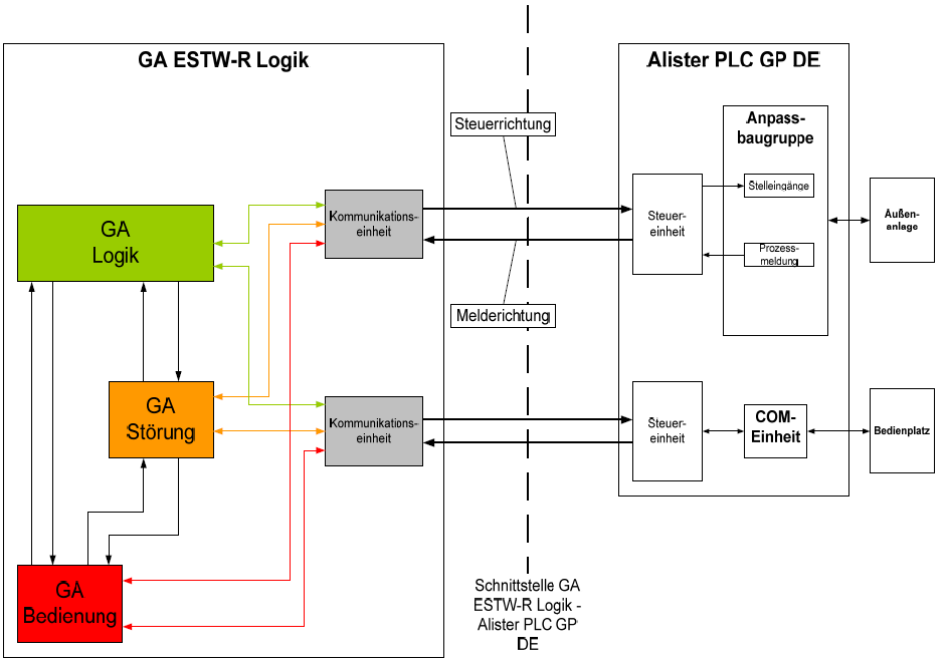

Abbildung 3: Schnittstellen der Teilsysteme GA und GP

3.2 Toolchain

Die Softwarekomponenten des Stellwerkskerns (GA und GP) werden funktional zum größten Teil in der Unified Modeling Language (UML) spezifiziert. Die Zustandsdiagramme werden toolunterstützt entwickelt und sind eingebettet in Worddokumente, welche die Systemanforderungen über ein Anforderungsmanagementtool einbetten und die Traceability zu Anforderungen, Versionen, etc. herstellt. Die Programmierung erfolgt grafisch mit ELOP II Factory und die Versionskontrolle erfolgt ebenfalls toolgestützt.

Für das Management von Fehlern, Defects und Testcases wird ein Testmanagementtool verwendet.

3.3 Testvorgaben

Um Aussagen zum Testkonzept treffen zu können, muss zwischen den einzelnen Teilsystemen des Stellwerkes unterschieden werden.

Begriff	Beschreibung
Bedienplatz	- Squish Tests
GA	- automatisierte Modultests
GP	- atomare Modultests (Überprüfung der korrekten Funktion einzelner Module bzw. Zustandsmaschinen) - Integrationsprüffälle Überprüfung der Funktionalität jeder Steuereinheit einzeln und im Zusammenwirken. Eine Teilmenge sind Prüffälle, die aus Betrachtungen der FMEA entstammen oder bestimmte Hardwareausfallszenarien testen.
Schnittstellentests	- GA/GP Prüffälle, die das korrekte Zusammenspiel der beiden Komponenten des Stellwerkkerns überprüfen - Kommunikationstestbett eine Java basierte Testumgebung, die die Schnittstelle zwischen den Rechnern des Bedienplatzes und dem Stellwerkskern überprüft.
Systemintegrationstests	- Überprüfen die korrekte Funktion aller Teilsysteme im Zusammenspiel

Tabelle 2: Testkategorien

Wie in Tabelle 2 dargestellt, sind die Testkategorien genauso vielfältig wie ihre Durchführung und Verwaltung.

Die bisher eingesetzten IBM-Tools boten intrinsisch jedoch noch keine einheitliche, durchgängige Handhabung der Testcases und ihres Managements. Darüber hinaus fehlt häufig die Möglichkeit einer Einbindung von proprietären 3rd Party Tools. Daher war es notwendig, die Toolchain durch eigene Entwicklungen bzw. Anpassungen zu ergänzen.

3.4 Testkonzept für das Generische Produkt

Am Beispiel der GP Prüffälle soll ein konkretes Testkonzept für ein Teilsystem des Stellwerks dargestellt werden.

Es werden Ansatzpunkte für eine Verbesserung bzgl. Verwaltung, Effizienz und Transparenz dargestellt, mit der Intension, die unterschiedlichen Rollen in der Nachweisführung gemäß CENELEC signifikant zu unterstützen.

3.4.1 Schritt 1: Erstellen der Prüffallspezifikation für eine Steuereinheit

GP Prüffälle definieren ein bestimmtes Prüfszenario für eine Steuereinheit, z.B. eine Weiche. Ein solches Prüfszenario wäre beispielsweise das Umstellen der Weiche von links nach rechts. Hierbei han-

delt es sich um einen Regelprüffall, also einen Prüffall, der eine Standardfunktion der Weiche, mit fehlerfreiem Verlauf, beschreibt.

Zum gesamten Umfang der Prüffälle gehört natürlich auch die Definition möglicher Fehlerfälle. Die Tabellenform wurde gewählt, weil sich so ein eindeutiger Ablauf des Prüffalls inklusive aller Teilschritte abbilden lässt. Hierbei wird ein eindeutiger Bezug zwischen

- korrekter Funktion der Steuereinheit,
- dem Verhalten der Software inkl. der Dokumentation auf Zustandsebene,
- der physikalischen Schnittstellen,
- der Schnittstellen zum Teilsystem GA und

Die Nachweisführung fordert, dass alle bekannten Regel- und Fehlerszenarien sowie jeder mögliche Zustandsübergang abzudecken sind.

3.4.2 Schritt 2: Prüffalldurchführung am Realsystem

Wenn eine Prüffallspezifikation fertig gestellt ist, d.h. die gesamte zu testende Funktionalität einer Steuereinheit spezifiziert so implementiert ist, werden die Prüffälle mit der originalen Hardware im Testlabor durchgeführt und dokumentiert. Voraussetzung ist, dass eine verifizierte und freigegebene Softwareversion der Steuereinheit verwendet wird, die wiederum eine verifizierte Version der Softwareanforderungsspezifikation voraussetzt. Für die Dokumentation wird ein Datenlogger verwendet, der an den physikalischen Ein- und Ausgängen angeschlossen wird.

3.4.3 Schritt 3: Prüffalldurchführung am Simulationssystem

Die Messdaten unter allen bekannten Funktionsszenarien dienen gleichzeitig als Grundlage für die Beschreibung eines simulierten Außensystems. Unter Anderem aus Kosten- und Platzgründen ist es schwer möglich, ein komplettes Testsystem unter Verwendung aller tatsächlichen Hardwarekomponenten aufzubauen. Dennoch bestehen der Wunsch und die Notwendigkeit nach einer solchen Testanlage.

Die Umsetzung ist möglich, indem man z.B. die Vielzahl aller Steuereinheiten wie Bahnübergänge, Weichen, Signale, etc. simuliert. Dieses Simulationssystem basiert ebenfalls auf einer SPS mit digitalen Ein- und Ausgängen. Ein simuliertes Außenelement besitzt ein Regel- sowie Fehlerverhalten, dass dem originalen Außenelement entspricht. Das Verhalten wird über Skripte beschrieben und kann leicht angepasst werden. Dies bietet auch den Vorteil, dass sich somit bestimmte komplexe Testszenarien und Zeitverhalten einstellen und testen lassen, die am Realsystem nur schwer einzustellen sind.

Die Simulation bietet nicht nur die Möglichkeit, ein ganzes Element zu simulieren, sondern auch Schnittstellen, um einzelne SW-Module bzw. Zustandsmaschinen zu testen. Diese atomaren Tests ermöglichen also auch, innere Zustände des Stellwerkskerns zu überprüfen.

Ein weiterer Vorteil des Simulationssystems ist die Speicherung von detaillierten Log-Dateien, um eine korrekte Prüffalldurchführung und deren Verlauf zu dokumentieren bzw. belegbar nachzuweisen.

3.4.4 Schritt 4: Vergleich von Original- und Simulationssystem

Um das Simulationssystem verwenden zu können, muss zuvor der Nachweis erbracht werden, dass sich die Simulation im Regelverhalten genauso verhält wie die originale Außenanlage. Dafür werden alle Prüffälle, die bereits mit den originalen Außenelementen durchgeführt wurden am Simulationssystem wiederholt. Dies erfolgt skriptgesteuert. Anschließend werden die Logdateien toolgestützt mit den Messungen der Außenanlage verglichen.

Den Vergleich übernimmt das eigens dafür entwickelte und auf Java/Eclipse basierende Tool AutoPruV. AutoPruV bietet ebenfalls die Möglichkeit, die Logdateien der Prüffälle mit den Vorgaben der Prüffallspezifikation und sogar der SW-Anforderungsspezifikation, d.h. den Zustandsdiagrammen, zu vergleichen.

Damit wird erreicht, dass Abweichungen von der Anforderungsspezifikation und Fehler offenbart werden.

Generell wird für jeden der zuvor angegebenen Schritte ein Bericht über die Durchführung angefertigt, welcher Randbedingungen und Ergebnisse dokumentiert. Gleichzeitig fließen diese Berichte in das Konfigurationsmanagement (KM) ein, und bilden zusammen mit den entsprechenden Dateien (Skripte, Logs, etc.) einen Versionsstand, der gegen eine weitere Bearbeitung geschützt wird.

Festzustellen ist, dass die eingesetzte Tool Suite zusammen mit den selbstentwickelten Ergänzungen und Schnittstellen ein erster Schritt in Richtung ALM sind.

3.5 Ansatzpunkte für Prozess- und Tooloptimierungen

Die eingesetzte IBM Rational Suite, also die gesamte Anzahl der einzelnen Applikationen dieser Suite, bietet bereits viele gemeinsame Schnittstellen und Abhängigkeiten, die für eine CENELEC-konforme Entwicklung notwendig oder mindestens hilfreich sind, wie z.B. Versionierung und Traceability. Gerade im Bereich der Testverwaltung offenbaren sich für den Bereich der Stellwerksentwicklung jedoch Grenzen.

Es ist grundsätzlich möglich, Testsuiten und Testcases anzulegen und/oder mit Anforderung oder Defekten zu verknüpfen. Eclipse-basierenden Tools sind jedoch stark auf das Testen von objektorientierten Anwendungen, wie z.B. in Java erstellte Projekte, ausgelegt. Somit wird z.B. ReqPro zwar formal für die Erstellung von Testcases und für die Verlinkung zu Anforderungen eingesetzt, eine direkte Verknüpfung zu den Testergebnissen und den dazugehörigen Daten ist aber nicht intrinsisch umsetzbar. Zusätzlich fehlt die Möglichkeit, die proprietären Testumgebungen direkt zu triggern. So ist es z.B. nicht möglich, automatisierte Regressionstests zu starten, auszuwerten und im definierten Fehlerfall Folgereaktionen auszulösen. Dies könnte z.B. die Erstellung eines Defekts beinhalten. Ein Ansatzpunkt wäre also alternative Verwaltungstools einzusetzen.

Die Möglichkeit eines modernen Application Lifecycle Management Systems (ALM) soll die Evaluierung eines Tools im Kapitel 3.6aufzeigen.

Grundsätzlich hat sich das oben vorgestellte Testkonzept als praktisch und zielgerichtet erwiesen. Die Bearbeitung der Prüffälle, die Durchführung und Dokumentation hat sich jedoch als sehr zeitaufwändig herausgestellt. Wiederholte Durchläufe, bedingt durch Änderungen an Spezifikation, Software und Prüffallspezifikationen wirken sich somit besonders negativ auf Entwicklungszeit und –kosten aus. Besonders die Fragmentierung der vielen einzelnen eingesetzten Tools und die Menge an händisch durchzuführenden Arbeitsschritten sollen einen weiteren Ansatzpunkt bieten: ein höherer Automatisierungsgrad und die Erarbeitung eines zentralen Testmanagement Systems.

Wie bereits zuvor erwähnt, wird bei der Auswertung der Prüffälle ebenfalls die Spezifikation mit einbezogen. Dies soll auch gleichzeitig den letzten zu nennenden Ansatzpunkt darstellen. Da auch die Modellierungstools eher für objektorientierte Anwendungen ausgelegt sind, sind auch bei der Modellierung im Bereich der Stellwerkstechnik auf SPS-Basis Einschränkungen hinzunehmen. Da objektorientierte Hochsprachen wie C++ oder Java in diesem Umfeld nicht zulässig sind und kein bislang evaluiertes Modellierungstool die volle angestrebte Funktionalität bietet, hat sich Funkwerk dafür entschieden, eine eigene Modellierungssprache, eine Domain Specific Language, zu erarbeiten. Diese wird in dem Forschungsprojekt MENGES, in Zusammenarbeit mit der Universität Kiel und einem weiteren Industriepartner entwickelt und sei an dieser Stelle nur erwähnt.

Denkbar wäre, direkt aus dem Modell den Code zu generieren („Model-Driven Development, ebenfalls evaluiert im MENGES-Förderprojekt). Dies setzt jedoch eine Akzeptanz in der Sicherheitsnachweisführung gemäß CENELEC voraus. Für SIL 4 Stellwerke ist dies nicht zeitnah geplant, es gibt aber durchaus erste beachtliche Erfolge im SIL0 / SIL2 Bereich.

3.5.1 Modellierungstool

Eine vollständige und eindeutige Modellierung bzw. UML-Spezifikation hätte den Vorteil, dass daraus automatisiert gültige Templates für die Prüffallerstellung generiert werden könnten. Schnittstellensignale, Initialisierungswerte und die Zustandsmaschinen sind immerhin bekannt. Dies bedeutet zum Einen, dass keine Fehler bzw. Inkonsistenzen, wie bei der händischen Erstellung entstehen können und

zum Anderen die Effizienz steigt. Bestimmte Fehler in der Spezifikation würden außerdem frühzeitig, d.h. während der Prüffallerstellung aufgedeckt werden.

Aus einer absolut eindeutigen UML-Spezifikation oder einer formalen Spezifikation, die programmtechnisch weiterverarbeitet werden kann, lassen sich auch automatisiert atomare Tests, also Schnittstellentests auf Modulebene, generieren und zwar für alle relevanten Permutationen der Zustandsübergänge unter Berücksichtigung, dass aus einer formalen Spezifikation automatisch Äquivalenzklassen für Prüffälle gebildet werden. Diese Anzahl von Prüffällen wäre von Hand nicht wirtschaftlich zu erstellen und auch nicht in der geforderten Qualität. Diese Maßnahme könnte zusätzlich die Softwareverifikation unterstützen bzw. diese auch ersetzten. Die 1:1 Umsetzung der UML-Spezifikation könnte somit nachgewiesen werden.

3.5.2 Testerstellung und -verwaltung

Zentraler Angelpunkt für ein gut integriertes Testmanagementsystem sollte eine zentral angelegte Testdatenbank sein. Sie sollte alle Prüffälle, sowie alle weiteren relevanten Daten, wie Ergebnisse, Logdateien, Messungen, Informationen zur Durchführung, Versionierung und Verifikation enthalten. Um auf der Datenbank arbeiten zu können, muss eine Client-Anwendung, eine Art Editor, entwickelt werden. Wie bereits zuvor erwähnt, sollten automatisch aus den Spezifikationen der Steuereinheiten Template-Informationen extrahiert werden, welche die Grundlage für die Maske des Editors bilden könnten. Abbildung 4 und Abbildung 5 zeigen beispielhaft Auszüge einer Prüffallspezifikation der Steuereinheit Weiche. Die Informationen werden bislang von Hand eingetragen. Der Inhalt einer solchen Prüffallspezifikation umfasst teilweise, je nach Komplexität der Steuereinheit, mehr als einhundert Prüffälle.

Nicht nur die Prüffallerstellung sollte automatisiert erfolgen, sondern auch Prüffalldurchführungen und Simulationsnachweise sollten direkt mit den Prüffällen verknüpft werden. Umwelttests, wie Salznebel-, Rüttel- und ähnliche Tests, sollten hier nicht ausgegrenzt sondern ebenfalls auf diese Weise erfasst werden.

Externe Prüftools wie AutoPruV sollten angepasst und integriert werden. Teilweise könnten Sie als Services permanent im Hintergrund laufen oder bei Bedarf manuell getriggert werden.

Prüfall ID	Prüfall Kurzbeschreibung	Identische Testszenarien	Prüfall durch zu führen an: 1 - Simulation 2 - originaler Hardwareaufbau GP DE Weiche mit Datenlogger zur Aufzeichnung der Schaltspiele für HaSi 3 - Testverzichtsfall (keine konkrete Testspezifikation, Ergebnis wird nicht benötigt)	Prüfall Typ: 1 - Normalfall 2 - betrieblicher Sonderfall 3 - Fehlerfälle der Außenanlage 4 - FMEA-Prüffälle (FMEA_XCM_XX) 5 - Prüffälle aus den Fehlerausfallanalysen der Baugruppen/Anpassbaugruppen (Hardware-XCM_XX)	Prüfallablauf mit Beschreibung	Eingangssignale vom PCM																Ausgangs-signale an das PCM				Signale von der Schnittstelle zu GA ESTW-R Logik			Signale an die Schnittstelle zu GA ESTW-R Logik									
						vUEL_a	vUEL_b	vUER_a	vUER_b	vA_a	vA_b	ASI	vSLer_b	vSRer_b	vSS_a	FsiLU	FsiLV	ErrSL_a	ErrSL_I	ErrSL_2	ErrSL_3	vSL_a	vSR_a	vSS_a	vUE_a	W.wLauf	W.a	W.Start	W.li	W.re	W.vUE	W.auf	W.AZD	W.PCM_FsiLX	W.Umlauf_li	W.Umlauf_re	W.W_abgeschalet
										x-1	x-1					x-1	x-1																				
200	Weiche Endlage links, Auffahrung	-	2	1	*BASIS_PF _ID: 116*	1	1	0	0	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	0	1
					Auffahrung, WA tritt ein, WUEL_a und WUEL_b erlöschen, Auffahrung	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	0	1
						1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	19	0	0	1
						1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	19	0	0	1
					Quittierung W.auf durch GA;	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	19	0	0	1
					Ende	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	19	0	0	1

Abbildung 4: Beispielprüffall inklusive aller Schnittstellen und Werte

Nach heutigem Kenntnisstand gibt es kein marktgängiges Werkzeug, welches die hier aufgeführten Anforderungen vollständig umsetzt.

Position der Zustandsmaschinen Alister PLC GP DE Weiche

Weichenumstellung					Weiche_Position_physikalisch		Aktuelle_Zustandsdaten_Weiche		Weiche_WA_Test		Weiche_WUE_Test		Weiche_Masch_Verriegel		Weiche_Ueberwachung_Betriebsmodus	
Weiche_umstellen_rechts	Weiche_umstellen_rechts_aus_WA	Weiche_umstellen_links	Weiche_umstellen_links_aus_WA	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code	Name des Zustands	Nummer des Zustands im Code
Weiche_links_neutral				1	Endlage_links	2	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1
Weiche_links_neutral				1	Stabilitaet_links	3	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1
Weiche_links_neutral				1	Weiche_aufgefahren	200	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1
Weiche_links_neutral				1	keine_physische_Endlage_aufgefahren	1	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1
Weiche_links_neutral				1	keine_physische_Endlage_aufgefahren	1	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1
Weiche_links_neutral				1	keine_physische_Endlage_aufgefahren	1	AZD	2	WA_Test_ok	7	WUE_Test_Ok	10	neutral	1	ErsSt_normal	1

Abbildung 5: Beschreibung des Verhaltens der Zustandsmaschinen während des Prüffalls

Ziel sollte es sein, die erworbenen Erkenntnisse aufzusetzen. Vielmehr sollen Prozesse und Tools im evolutionär entwickelt werden, die Aufgaben automatisieren, erleichtern, beschleunigen und weniger fehlerträchtig machen. Auch die Verfahren zur Erstellung der Prüfberichte einschließlich ihrer Formate können so in Abstimmung zwischen allen Rollen gemäß CENELEC abgestimmt werden. Es wäre denkbar, dass Verifizierer ebenfalls auf dem Datenbestand arbeiten und bestimmte Inhalte direkt und eindeutig beurteilen. Dafür müssten natürlich verschiedene Rollen im Testmanagement-Tool geschaffen werden. Eine weitere Funktionalität, die in diesem Zusammenhang angestrebt werden sollte, ist die Generierung von Dokumenten. Also die automatische Erzeugung von Dokumenten anhand von Information aus der Datenbank. Formale Vorgaben könnten durch Templates realisiert werden. Die automatisierte Dokumentengenerierung wäre auf viele Arten von Dokumenten des ganzen Entwicklungsprozesses anwendbar, wie unter anderem für Prüffallspezifikationen, Berichte zur Durchführung von Prüfungen, Checklisten und Verifikationen.

3.6 Evaluierung eines ALM-Tools

Ziel der Evaluierung ist die Erfassung der Möglichkeiten, die bestehenden Testtools besser in den Entwicklungs- und Nachweisprozess einzubinden und Abhängigkeiten zu schaffen. Hierbei handelt es sich aber nicht um eine allgemeine Beurteilung zur grundsätzlichen Eignung eines ALM-Tools für ein IT-Unternehmen, sondern es wird eine klare Abgrenzung zur Evaluation für den Einsatz in der Entwicklung von Stellwerksprojekten definiert.

In den nachfolgenden Unterkapiteln erfolgt eine gezielte Zusammenstellung von Informationen und ersten Erfahrungen im Umgang mit Polarion als möglicherweise geeignetes ALM-Tool.

Für die Evaluierung wurde die Polarion Version Stand April 2013 verwendet.

3.6.1 Grundkonzept

Grundsätzlich geht es um die Anbindung von externen, bereits bestehenden Testtools, die im Weiteren zusammengefasst bezeichnet werden als Test Automation Server (TAS).

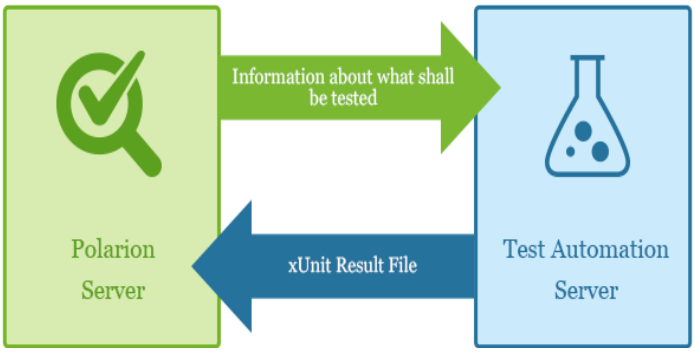

Abbildung 6: Grundkonzept der Testanbindung an Polarion

Abbildung 6 zeigt das Grundkonzept des Datenaustausches zwischen einem Polarion Server und einem Test Automation Server (TAS). Für die meisten Endanwender sind beide Server getrennte Systeme.

In unserem Fall besteht der TAS aus einer geeigneten Testanlage, gekoppelt mit einem Simulationssystem. Eine wichtige Voraussetzung ist, dass der TAS eine Art „Test Agent“ enthält, der eingehende Anfragen von einem externen Tool (hier vom Polarion-Server) entgegen nimmt. Dies ist wichtig, weil in der Testdurchführung oft komplexe 3rd Party Tools aufgerufen werden, welche in der Durchführungsumgebung verfügbar sein müssen.

3.6.1.1 Aufrufen eines Builds

Um den Prozess der gemeinsamen Nutzung von Informationen zwischen Polarion QA (dem Collaborative Test Management) und einem Test Automation Server zu starten, besteht die Möglichkeit den Build-Prozess mit einem der folgenden Tools aufrufen:

- Jenkins Integration Server (wird bei Funkwerk u.a. bereits eingesetzt)
- Hudson
- ANT (wird bei Funkwerk u.a. bereits eingesetzt)
- MAVEN (wird bei Funkwerk u.a. bereits eingesetzt)
- Polarion Build

Hinweis: Derzeit sind diese Anwendungen noch nicht für die SIL4-Stellwerksentwicklung akzeptiert.

3.6.1.2 Austausch von Testdaten

Sobald das entsprechende Build-Tool aufgerufen wurde, müssen entsprechende Skripte definiert und ausgeführt werden:

1. Testfälle von Polarion QA, für das was getestet werden soll, holen
2. zugehörige Test-Skripte für die Durchführung holen
3. Tests zum Test Automation Server senden

3.6.1.3 Testfalldurchführung

Für die Durchführung ist der TAS verantwortlich. Nachdem Polarion die Testdaten geliefert und den Test gestartet hat, wartet es nur auf die Ergebnisse. Die Ergebnisse werden vom TAS generiert und auf lokale bzw. Laufwerke im Netzwerk kopiert, auf welche Polarion nach jeder Testfalldurchführung zugreifen kann.

Angestoßen wird der Prozess durch das Senden von Test-Skripten; Abbildung 7 soll dies verdeutlichen.

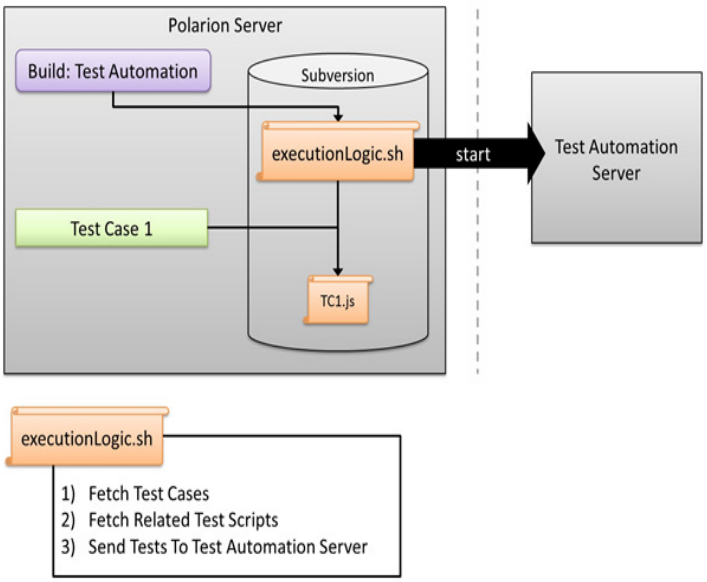

Abbildung 7: Anstoßen von Testfällen

3.6.1.4 Empfang von Ergebnissen

Mit dem Polarion Server ist es wie oben gezeigt möglich, externe Testtools einzubinden. Polarion ist darüber hinaus geeignet, die Traceability über alle Testartefakte nachzuweisen.

Zum Beispiel kann Eclipse-TPTP für automatisierte Last- und Performance-Tests und Selenium für automatisierte Tests der Benutzeroberfläche verwendet werden, um Ergebnisse jeder Testdurchführung zu Polarion zu importieren und dort für alle Rollen geeignet darzustellen.

Polarion kann auf diese Weise mit einem externen Tool, das die Testergebnisse in xUnit XML-Formate exportieren kann, zusammenarbeiten.

Um Testergebnisse von einem externen Test-Tool mit Polarion zu integrieren, muss jedes externe Tool so konfiguriert werden, dass eine xUnit Datei entsteht und in einen für Polarion zugänglichen Ordner geschrieben wird.

Polarion wiederum wird so konfiguriert, dass jeder Ordner auf das Vorhandensein einer Testergebnis-Datei regelmäßig überprüft wird. Ein geplanter Job namens xUnitFileImport ist in Polarion vorgesehen, der diese Aufgaben übernimmt.

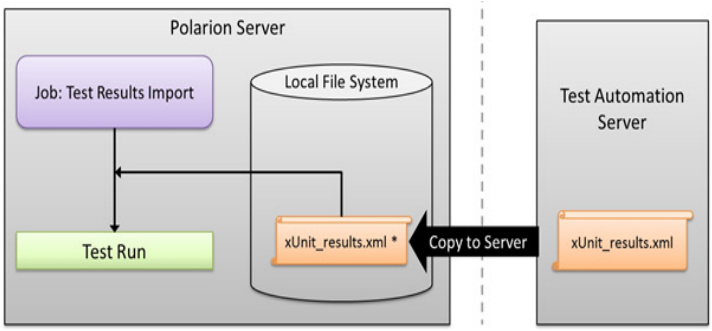

Abbildung 8: Rückgabe von Testergebnissen

Abbildung 8 zeigt die erforderliche Architektur für den Import der Testergebnisse. Man kann sehen, dass die Datei „xUnit_results.xml“ ursprünglich auf dem TAS in einem gültigen xUnit-Format erstellt ist. Hat der TAS die Lieferung der Daten abgeschlossen, dann kann auf die Ergebnisse zugegriffen werden.
Der Abgleich der Daten erfolgt automatisiert und wird konfigurierbar in festen Intervallen durchgeführt. Dabei wird geprüft, ob neue xml Ergebnisse auf der angegebenen Datenablage vorhanden sind. Der Import der Ergebnisse kann ebenfalls manuell angestoßen werden.
Es wird für jede gültige xml-Datei im angegebenen Verzeichnis erfolgreich ein neuer Testdurchlauf erzeugt. „Defect type Work Items“ werden für die Entwickler automatisch erzeugt, wenn Tests fehlschlagen. Am Ende wird die xml Datei (-en) automatisch gelöscht.

3.6.2 Positive Feststellungen aus der Evaluation

- Die getestete Performance der lokal installierten Anwendung war durchgängig akzeptabel. Zur Performance bei der Verwendung eines Servers im Netzwerk können noch keine Aussagen getroffen werden.
- Die lokale Installation war vollkommen problemlos.
- Testcases/Testruns sind importierbar und exportierbar im Excelformat. Damit besteht eine einfache Möglichkeit von Offlinetests. Dies könnten zum Beispiel Tests an den Installationsorten des Stellwerkes sein oder Tests, die von externen Stellen durchgeführt werden.
- Riskmanagement (bzw. –dokumentation) und FMEA sind in das Tool integriert und mit Tests verlinkbar.
- Bestimmte Testcases können direkt mit dem Tool erstellt und editiert werden (z.B. Systemintegrationstests und Checklisten).
- Schnittstellen zu externen Testservern sind implementiert.
- Neueste Erweiterung: Integration mit Matlab/Simulink

3.6.3 Negative Feststellungen aus der Evaluation

- Es traten vereinzelt „Hänger“ im IE-Browser aus; Speichern war dabei nicht möglich; nicht nachvollziehbare Fehlermeldungen traten auf
- Eine fehlende Verzeichnisstruktur erscheint behindernd für Attachments.
- Spezielle Tests wie ITG Prüffälle oder atomare Prüffälle können nicht direkt abgebildet werden sondern müssen über externe Tools verwaltet werden. Die externen Tools könnten aber mit Polarion verknüpft werden (siehe xUnit framework + xml format).

4 Fazit und Ausblick

Im Laufe des Entwicklungsprozesses elektronischer Stellwerke wurden viele Erfahrungen in den Bereichen Fehlererkennung und Teststrategie im Rahmen des Sicherheitsnachweises gesammelt. Aktuell werden verschiedene Tools oder Toolsuiten verwendet und auf die CENELEC-Prozesse angepasst und optimiert.

Die interne Toollandschaft hat Konzepte der heutigen ALM-Werkzeuge bereits vorgegriffen. Festzustellen bleibt jedoch auch, dass bei der Integration dieser Tools noch bestimmte Automatismen fehlen.

Nach einer Evaluierungsphase lässt sich feststellen, dass sich einige dieser Lücken durch die Verwendung des ALM-Tools Polarion schließen lassen. Eine vereinfachte Nachweisführung der Einhaltung aller RAMS-Anforderungen ist generell machbar und wirkt sich effizienzsteigernd auf die Gestaltung des Sicherheitsnachweises aus.

Ein besonderer Schwerpunkt der Betrachtung waren die Schnittstellen zu externen, proprietären Testsystemen. Diese ermöglichen die direkte Verknüpfung von Tests, Testsuiten und Fehlermanagement zu bestehenden RAMS-Anforderungen.

Diese Möglichkeiten, insbesondere die Handhabung von verteilten Anforderungen, erscheinen besonders wichtig im Hinblick auf den Einsatz in einem geplanten DB AG Testlabor (siehe „AP 1300 - Entwicklung eines Testkonzeptes“), welches eine zentrale Rolle in der Optimierung des Zulassungsprozesses für die DB AG einnehmen soll.

Ein weiteres Beispiel für einen praktischen Einsatz wäre die bereits im Kapitel 2.5.3 skizzierte Anbindung an eine Diagnosedatenbank (siehe Teilergebnisbericht "NeGSt_Ergebnisbericht_2300AG5_Feinkonzept und Prototyp für eine RAMS-Diagnoseplattform").

5 Anhang

5.1 Anhang 1: Referenzen

[1] Polarion 2013 User Guide

[2] Polarion QA for Test Managers

[3] User-Guide-Automated-Test-Execution-with-Polarion

5.2 Anhang 2: Abkürzungen

Abk.	Langform / Erläuterung
ALM	Application Lifecycle Management
CENELEC	European Committee for Electrotechnical Standardization
COTS	Commercial off-the-Shelf
ESTW-R	ESTW-Regional
FMEA	Fehler-Möglichkeits- und Einfluss-Analyse
FTA	Failure Tree Analysis (Fehlerbaumanalyse)
GA	Generische Applikation
GP	Generisches Produkt
ITG	Integrationsprüffälle
KM	Konfigurationsmanagement
LST	Leit- und Sicherungstechnik
RAMS	Reliability, Availability, Maintainability, Safety
ReqPro	(IBM-Tool) Rational Requisite Pro
SA	Spezifische Applikation
SIL	Safety Integration Level
SPS	Speicher Programmierbare Steuerungstechnik
SRS	Safety Requirements Specification
SW	Software
TAS	Test Automation Server
TSB	Technischer Sicherheitsbericht
UML	Unified Modelierung Language
XML	Extensible Markup Language