Eine vielseitige Testkomponente auf Grundlage des EULYNX-Teilsystems Weiche

A versatile test component based on the EULYNX Subsystem Point

Daniel Schwencke

Mit dem Aufkommen standardisierter modularer Architekturen in der Leit- und Sicherungstechnik (LST) gewinnt die externe Validierung und Verifikation entsprechender Teilsysteme der Signaltechnik-Hersteller zunehmend an Bedeutung, um z. B. die Konformität mit dem Standard und die Interoperabilität mit anderen Teilsystemen nachzuweisen. Dies kann durch das Testen einzelner Teilsysteme bis hin zu vollständigen Systemen erfolgen. Die Einsicht, dass dasselbe Teilsystem dabei in unterschiedlichen Tests unterschiedliche Rollen einnimmt, führte zu der Idee einer „vielseitigen Testkomponente". Der Beitrag berichtet über eine Umsetzung, die die Teilsysteme Weichencontroller und Weichenantrieb umfasst und auf der EULYNX-Spezifikation des Teilsystems Weiche basiert.

1 Einleitung

1.1 Testbedarf durch standardisierte modulare Architekturen

Seit mehreren Jahren entwickeln europäische Eisenbahninitiativen standardisierte modulare Architekturen: EULYNX [1] konzentriert sich auf die Schnittstellen (engl. interface – IF) des Stellwerks; die Reference CCS Architecture (RCA) [2] deckte den erweiterten Bereich der – vor allem streckenseitigen – LST-Systeme ab; und die Open CCS On-board Reference Architecture (OCORA) [3] kümmert sich um IF zugseitiger LST-Komponenten. Zusätzlich beschäftigt sich der Europe's Rail System Pillar mit der Integration dieser Architekturen in einer „System Pillar Architecture" [4].

Während in der Vergangenheit Modularisierung, IF-Design und das detaillierte Verhalten von LST-Teilsystemen größtenteils herstellerinterne Themen waren, sind sie durch die neuen Architekturen stärker vorgeschrieben. Dies führt zu austauschbaren Teilsystemen, die potenziell von unterschiedlichen Herstellern stammen. Dies wiederum induziert einen erhöhten Verifikations- und Validierungsbedarf, um – gegenüber dem externen Systemintegrator und den Behörden – nachzuweisen, dass die interagierenden Teilsysteme tatsächlich das beabsichtigte Verhalten des Gesamtsystems realisieren. An erster Stelle ist die Konformität der Teilsysteme zur standardisierten Spezifikation zu verifizieren, was idealerweise bereits alle möglichen wünschenswerten Eigenschaften, die in die Spezifikation eingearbeitet sind, implizieren würde. In der Praxis ist dies jedoch nicht ausreichend; Sicherheit, Robustheit, die erfolgreiche Integration von Teilsystemen usw. benötigen weitergehende Verifikation. Zunehmend beschäftigen sich die Standardisierungsinitiativen mit standardisierter Verifikation: Mit Baseline 4 Release 1 hat EULYNX begonnen, „Zertifizierungstestfälle" für seine „Teilsysteme" bereitzustellen, d. h. für diejenigen standardisierten Stellwerksschnittstellen, die auch standardisiertes Verhalten des verbundenen Feldelement-

With the advent of standardised modular signalling architectures, the external validation and verification of corresponding subsystems from signalling suppliers has grown increasingly in importance, e.g. to prove conformity with the standard and interoperability with other subsystems. This can be done by testing single subsystems through to complete systems. The insight that the same subsystem will play different roles in different tests has led to the idea of a "versatile test component". This article reports on a realisation involving the point controller and point machine subsystems, based on the EULYNX Subsystem Point specification.

1 Introduction

1.1 Demand for testing induced by standardised modular architectures

European railway initiatives have been developing standardised modular architectures for several years: EULYNX [1] focuses on interlocking interfaces (IF), the Reference CCS Architecture (RCA) [2] has covered the wider scope of the (mostly trackside) command control and signalling (CCS) system and the Open CCS On-board Reference Architecture (OCORA) [3] deals with the IF of onboard CCS components. Moreover, Europe's Rail System Pillar has concerned itself with integrating these architectures into a "System Pillar Architecture" [4].

Whereas modularisation, IF design and the detailed behaviour of CCS subsystems was previously largely an internal supplier matter, with the new architectures they are now prescribed to a greater extent. This leads to interchangeable subsystems, potentially provided by different suppliers. This in turn leads to an increased need for verification and validation in order to prove to both the external system integrator and the authorities that the interacting subsystems will actually realise the intended system behaviour. In the first place, conformity to the standardised specification has to be verified and, ideally, this would already imply various desirable properties incorporated into the specification. However, this is not sufficient in practice; safety, robustness, successful subsystem integration, etc. will require further verification. Increasingly, the standardisation initiatives care about standardised verification: Baseline 4 Release 1 of EULYNX started providing "certification test cases" for its "subsystems", i.e. for those standardised interlocking IF that also include standardised behaviour for the connected field element controllers. EULYNX is also looking into formal verification [5, 6], but for now testing seems to be the envisaged means of verification [7].

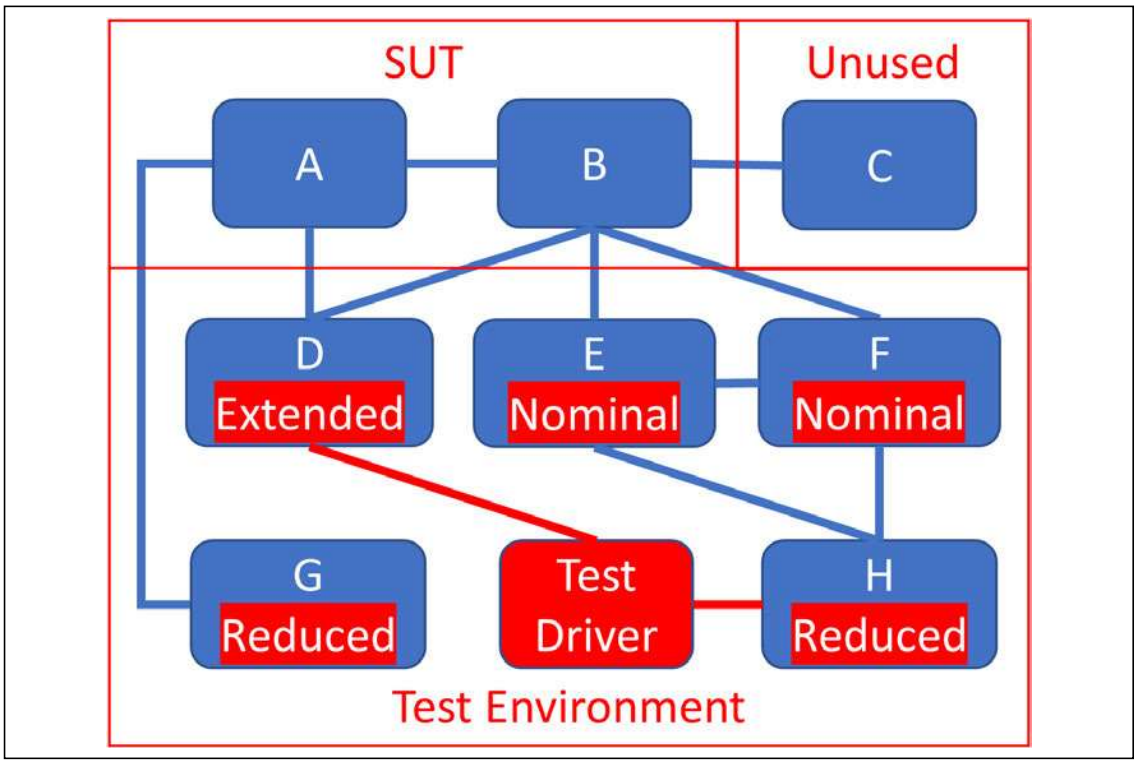

Bild 1: Beispielanordnung für Tests (in rot) in einem Systemkontext (in blau)
Fig. 1: An example set-up for testing (in red) within a system-of-systems context (in blue)

Controllers umfassen. EULYNX untersucht zwar auch die Nutzung formaler Verifikation [5, 6], bislang scheint jedoch Testen die anvisierte Verifikationstechnik zu sein [7].

1.2 Die Idee einer vielseitigen Testkomponente

Bild 1 zeigt ein System untereinander verbundener Teilsysteme A bis H (blaue Kästen und Verbindungen). Abhängig vom Ziel und Gegenstand des Tests, kann jedem Teilsystem beim Testen eine Rolle zugeordnet werden (s. rote Kästen als ein mögliches Beispiel):

- entweder Teil des getesteten Systems (engl. system under test – SUT)
- oder Teil der Testumgebung
- oder nicht für die Testanordnung benötigt.

Jedes Teilsystem, das Teil der Testumgebung ist, wird darüber hinaus Funktionalität in einem gewissen Ausmaß umsetzen (Text mit rotem Hintergrund als Beispiel):

- entweder reduzierte Funktionalität (z. B. als Adapter oder einfacher Simulator)
- oder nominelle Funktionalität (wie eine Teilsystemimplementierung)
- oder erweiterte Funktionalität (z. B. als fehlgeschlagenes, defektes oder manipuliertes Teilsystem).

Schließlich werden einige Teilsysteme der Testumgebung mit dem Testtreiber verbunden sein (rote Verbindungen), der die Testausführung gemäß einem gegebenen Testfall steuert; andere jedoch nicht. Weitere Unterschiede (nicht in Bild 1 dargestellt) können zwischen simuliertem Teilsystem und realem Produkt sowie in der Wahl der Konfigurationsparameter eines Teilsystems bestehen.

Entsprechend ist es nicht schwer sich vorzustellen, dass ein Teilsystem beim Testen auf vielfältige Weise genutzt werden kann; und die Notwendigkeit, alle Systemteile zu testen, sowie die Überlegungen zu verschiedenen Verifikationszielen in Abschnitt 1.1 zeigen, dass mehrere dieser Nutzungsarten tatsächlich in Testkampagnen benötigt werden. Für jede Nutzung desselben Teilsystems eine neue Version zu implementieren, wäre aufwendig und würde redundante Arbeit bedeuten,

1.2 The idea of a versatile test component

fig. 1 shows a system of interconnected subsystems A to H (the blue boxes and connections). For testing, each subsystem will be assigned a role (consider the red boxes as an example) depending on the purpose and scope of the test:

- either part of the system under test (SUT)
- or part of the test environment
- or not used in the test set-up.

Moreover, each subsystem that is part of the test environment will be required to realise functionality at a certain extent (the text with the red background as an example):

- either reduced functionality (e.g. acting as an adapter or a simplified simulator)
- or nominal functionality (acting as a subsystem implementation),
- or extended functionality (e.g. acting as a failed, defective or manipulated subsystem).

Finally, some test environment subsystems will be connected to the test driver (the red connections) that controls the test execution according to the given test case, but others may not be. Further differences (not illustrated in fig. 1) may be simulated vs. real product subsystems and the chosen subsystem configuration parameters.

Summarising, it is not difficult to imagine that there is a multitude of possible subsystem uses in testing; and the need to test all the system parts as well as the consideration of the different verification purposes in Section 1.1 indicate that several of these uses will actually be required in test campaigns. Implementing new versions for each use of the same subsystem would be laborious and involve redundant work, as parts of the subsystem IF and functionalities will remain the same. The idea of the test component (TC) presented in this article is to be more efficient by creating a software that can be flexibly configured, connected and controlled, so that it can be used in many different test set-ups.

da Teile der Teilsystemschnittstellen und -funktionalitäten dieselben wären. Die Idee der in diesem Beitrag vorgestellten Testkomponente (TK) ist effizienter: Hierbei wird eine Software erstellt, die flexibel konfiguriert, verbunden und gesteuert werden kann und die so in vielen verschiedenen Testanordnungen verwendet werden kann.

2 Zweck und Umfang der Testkomponente

2.1 Anwendungsfälle und Anforderungen

Die grundlegendste Anforderung an die TK ist, das nominelle Verhalten des/der Teilsystems/-systeme zu implementieren. Für standardisiertes Verhalten bedeutet das „wie spezifiziert"; für nicht standardisiertes Verhalten „Integration einer Simulation realistischen Eingabe-/Ausgabeverhaltens" (engl. input/output – I/O). Die Implementierung kann Verhalten der Rückfallebene umfassen; und sie beinhaltet, jedes externe und möglicherweise offene interne IF sowie wichtige Statusinformationen von außen zugänglich zu machen. Dies erlaubt bereits die Umsetzung einer Reihe von Anwendungsfällen für die TK:

- Nutzung als SUT oder als Teil des SUT, z. B. für frühzeitige Integrationstests. Allgemeiner betrifft dieser Anwendungsfall Situationen, in denen noch keine Teilsystemimplementierung verfügbar ist oder die verfügbaren keine ausreichenden Informationen/Einblicke für Debuggingzwecke bereitstellen.
- Nutzung als Referenz. Hierbei haben SUT und Referenz denselben Umfang und erhalten dieselben Eingaben; die Ausgaben des SUT werden dann mit denen der Referenz verglichen.
- Nutzung – evtl. mehrerer Instanzen – als Platzhalter („Dummy") in größeren Testanordnungen, z. B. beim (Software-)Systemtest. Dieser Anwendungsfall betrifft Situationen, in denen reale Produkte noch nicht fertiggestellt sind oder ihr Einsatz für den Test als zu teuer oder aufwendig betrachtet wird.

Da Bezeichner und Funktionalität von LST-Systemen üblicherweise konfigurierbar sind, wird von der TK vollständige Konfigurierbarkeit (entsprechend ihrer Spezifikation oder der verfügbaren Simulationsoptionen) gefordert. Dies stellt sicher, dass alle konfigurierbaren Funktionen auch tatsächlich getestet werden können oder im Dummy verfügbar sind und dass die TK an die Nutzung in existierenden Umgebungen angepasst werden kann.

Falls die TK mehrere Teilsysteme umfasst, sollte eine Option zur Abschaltung einzelner Teilsysteme – d. h. die Externalisierung interner IF – in Erwägung gezogen werden. Dies ermöglicht die wechselweise oder schrittweise Ersetzung von Teilsystemen der TK durch reale Teilsysteme und somit Tests beliebiger Integrationsstufen.

Zusätzlich sollte es dem Tester möglich sein, über das nominelle I/O Verhalten hinaus, normalerweise nicht mögliches Teilsystemverhalten auszulösen. Das kann die Manipulation von Funktionen (Fehlerinjektion) oder von nichtfunktionalen Eigenschaften (z. B. das Einfügen von Verzögerungen) sein. Hierdurch werden zusätzliche Anwendungsfälle wie das Testen der SUT-Reaktion auf eine fehlerhafte Umgebung ermöglicht.

2.2 Umfang und Funktionen

Als Umfang der TK wurden zwei Teilsysteme gewählt (s. dunkelblaue Kästen in Bild 2):

1. Weichencontroller mit IF und Funktionalität wie von EULYNX spezifiziert (s. Abschnitt 2.3 weiter unten), bis auf dass die spezielle „4-wire" Variante des P3 IF, die Option, ein IF zum EULYNX-Teilsystem „Maintenance and Data Management" zu nutzen und das „Standard Diagnostic IF" nicht umgesetzt wurden. Die wichtigsten Controllerfunktionen sind die Weiterleitung einer vom Stellwerk kommandierten Weichenlage an die Weichenantriebe (engl. point machines – PM) und andersherum die Benachrichtigung des Stell-

2 The purpose and scope of the test component

2.1 Use cases and requirements

The most basic requirement for the TC is the implementation of the nominal behaviour of the subsystem(s). For standardised behaviour, this means "as specified"; for non-standardised behaviour, this means "adding a simulation of realistic input/output (I/O) behaviour". This may include behaviour in degraded situations and it includes the provision of external access to every external and possibly open internal IF, as well as to important status information. This already allows a number of use cases to be realised for the TC:

- Use as SUT, or as part of the SUT, e.g. for early integration testing. More generally, this use case applies when no subsystem implementation is available yet or the available implementations do not provide sufficient information/insight for debugging purposes.
- Use as reference. This means that SUT and reference both have the same scope and receive the same inputs, and the SUT outputs are compared to the reference outputs.
- Use (possibly multiple instances) as replacement ("dummy") in larger test set-ups, e.g. during (software) system testing. This use case applies when real products are not yet ready or are considered too expensive/laborious to use in testing.

Given that the identity and functionality of signalling systems are usually configurable, the TC is likewise required to be fully configurable (according to the specification or to the available simulation options). This ensures that all the configurable functions can actually be tested or are available as part of the dummy and that the TC can be adapted for use in given environments.

If more than one subsystem is covered by the TC, an option to switch individual subsystems off (i.e. the externalisation of their internal IF) should be considered. This allows the TC subsystems to be alternately or gradually replaced with real subsystems, supporting testing of all kinds of integration stages.

Furthermore, the tester should be able to provoke impossible subsystem behaviour that goes beyond the nominal I/O behaviour. This may include the manipulation of functions (failure injection) or non-functional properties (e.g. the insertion of delays). This enables additional use cases such as testing the SUT reaction to a faulty environment.

2.2 Scope and features

The scope of the TC has been chosen to comprise two subsystems (see the dark blue boxes in fig. 2):

1. A point controller with IF and functionality as specified by EULYNX (see Section 2.3 below), with the exception that the special "4-wire" variant of the P3 IF, the option to use an IF to the EULYNX "Maintenance and Data Management" subsystem and the "Standard Diagnostic IF" have been excluded. The main controller functions involve relaying a point position commanded by the interlocking to the point machines (PM) and reporting an aggregated point position status and any failed point movements back to the interlocking.
2. A PM simulation with the IF to the point controller inherited from the EULYNX specification, but otherwise with self-specified functionality. The main PM simulation functions involve simulating the movement of the PM and reporting its status (position and ability to move) to the point controller.

This scope has the advantage of being self-contained in the sense that there is only one mandatory external IF (the standard

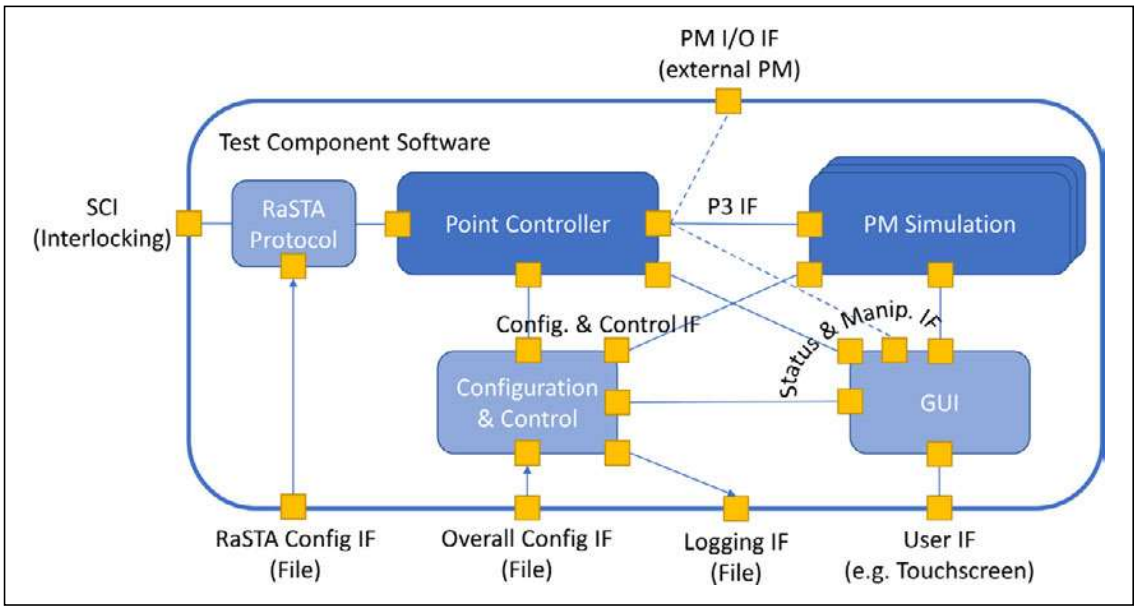

Bild 2: Überblick über Struktur und Schnittstellen der TK
Fig. 2: An overview of the TC structure and the IF

werks über den (über die PM aggregierten) Status der Weichenlage und fehlgeschlagene Weichenumläufe.

2. PM-Simulation mit aus der EULYNX-Spezifikation übernommenem IF zum Weichencontroller, ansonsten jedoch eigens spezifizierter Funktionalität. Die wichtigsten Funktionen der PM-Simulation sind die Simulation der PM-Bewegung und die Benachrichtigung des Weichencontrollers über den PM-Status (Position und Funktionsfähigkeit).

Der gewählte Umfang hat den Vorteil, dass er in sich abgeschlossen ist in dem Sinne, dass es lediglich ein für das funktionale Testen unverzichtbares externes IF zu einem anderen Teilsystem gibt (das standard communication interface, SCI, des Weichencontrollers zum Stellwerk), das außerdem vollständig standardisiert ist. Um Letzteres umzusetzen, wurde eine (existierende) DLR-Implementierung des dazu notwendigen RaSTA (Rail Safe Transport Application)-Protokolls [8] in die TK-Software integriert (s. linker hellblauer Kasten in Bild 2). Ein allgemeiner Vorteil der Wahl eines EULYNX-Teilsystems ist die existierende detaillierte Spezifikation nicht nur der IF, sondern auch des Teilsystemverhaltens sowie die (in einer ersten Version) existierenden Zertifizierungstestfälle.

Für die Initialisierung und Ausführung der TK wurden Konfigurations- und Steuerungsfunktionalität ebenso implementiert wie ein grafisches Nutzer-IF (engl. graphical user interface – GUI) zur Interaktion mit dem Tester (s. untere hellblaue Kästen in Bild 2). Das GUI besitzt vier Tabs, die den vier internen Status- und Manipulations-IF in Bild 2 entsprechen; ein Bild des ersten Tabs findet sich in Bild 3.

Die Konfigurierbarkeit der TK beginnt bereits auf der Ebene der Gesamt-TK: Durch einen Konfigurationsparameter des EULYNX-Weichencontrollers, der die Einstellung einer beliebigen Anzahl (≥1) von gesteuerten PM erlaubt ([17] erwähnt eine Begrenzung auf 52 PM, jedoch ohne jede Begründung oder Quelle), ist es möglich, mehrere Instanzen der PM-Simulation zu nutzen (in Bild 2 durch mehrere Kästen für die PM-Simulation angedeutet). Ebenso ist es für jede PM möglich, die integrierte Simulation abzuschalten und das PM-Verhalten einem externen Akteur (z. B. einer verbundenen physischen PM, oder dem mit dem GUI der TK interagierenden Nutzer – s. die gestrichelten Linien in Bild 2) zu überlassen.

communication interface, SCI, of the point controller) to another subsystem (the interlocking) that is important for functional testing, which moreover is completely standardised. Note that to achieve the latter, a (pre-existing) DLR implementation of the necessary RaSTA (Rail Safe Transport Application) protocol [8] has been integrated into the TC software (see the leftmost light blue box in fig. 2). A general advantage of choosing a EULYNX subsystem lies in the existing detailed specification not only of the IF, but also of the subsystem behaviour and the existing (first version of) certification test cases.

Configuration and control functionality has been added to set up and run the TC, as has a graphical user IF (GUI) for the interaction with the tester (see the bottom light blue boxes in fig. 2). The latter has four tabs corresponding to the four internal status and manipulation IF shown in fig. 2; a picture of the first tab is shown in fig. 3.

The TC configurability already starts at the overall TC level: induced by a configuration parameter for the EULYNX point controller that allows an arbitrary number (≥1) of controlled PM ([17] mentions a limit of 52 PM, but without any further justification or reference) it is possible to use several instances of the PM simulation (indicated by multiple boxes for PM Simulation in fig. 2). For each PM it is also possible to switch off the built-in simulation, thereby leaving the PM behaviour to an external actor (e.g. a connected physical PM or the TC user interacting with the GUI; see the dashed lines in fig. 2).

2.3 Specification analysis / creation

The point controller subsystem has been implemented according to the latest EULYNX specifications from Baseline 4 Release 2; this especially includes the requirement specifications [9, 10, 11], the SCI IF specifications [12, 13] and the architecture specification [14]. Generally, the tabular (SCI telegrams) and model-based (subsystem logical structure and behaviour) specifications define the subsystem very precisely and consistently. Overall, the specifications comprise 14 telegrams, ten interconnected SysML (Systems Modelling Language [15]) blocks containing one state diagram each and 15 configuration options relevant for the point controller. However, there are specification gaps, in particular since SysML (version 1.6) leaves the

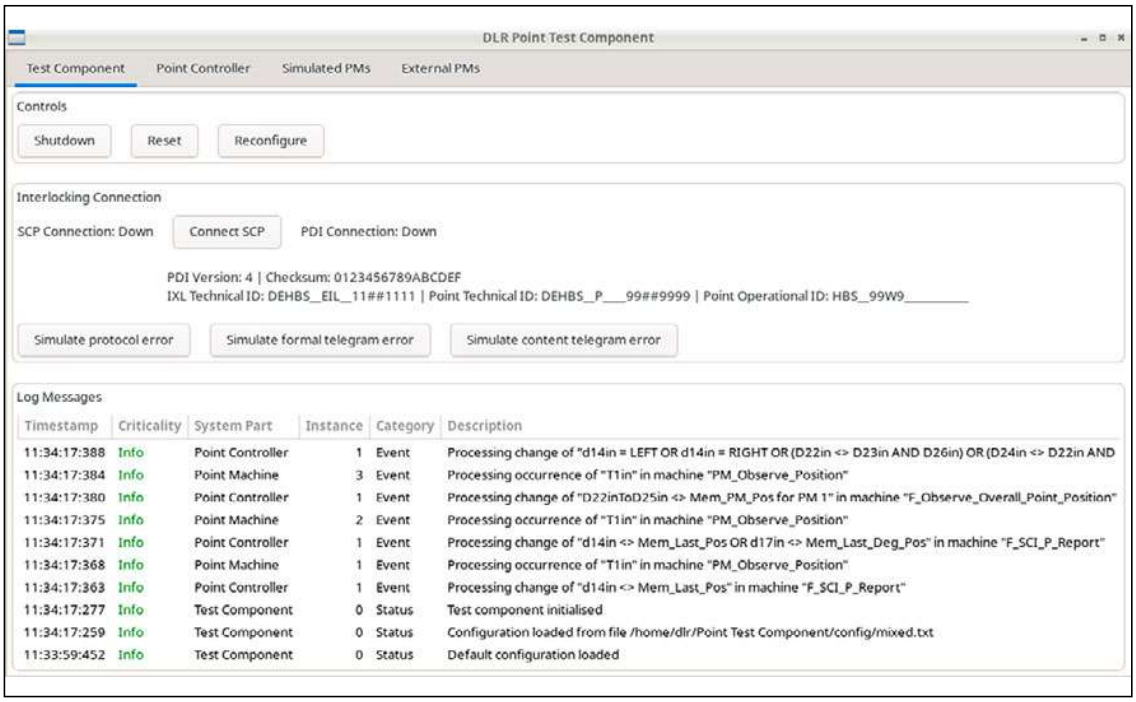

Bild 3: Bildschirmfoto des ersten Tabs des GUI der TK zur Überwachung und Steuerung der Gesamt-TK
Fig. 3: A screenshot of the first tab in the TC GUI for overall monitoring and control

2.3 Analyse / Erstellung der Spezifikation

Das Weichencontroller-Teilsystem ist gemäß den neuesten EULYNX-Spezifikationen aus Baseline 4 Release 2 implementiert; insbesondere gemäß der Anforderungsspezifikationen [9, 10, 11], der IF-Spezifikationen [12, 13] des SCI und der Architekturspezifikation [14]. Im Allgemeinen definieren die tabellarischen (SCI-Telegramme) und modellbasierten (logische Struktur und Verhalten des Teilsystems) Spezifikationen das Teilsystem sehr präzise und konsistent. Insgesamt beinhalten die Spezifikationen 14 Telegramme, zehn untereinander verbundene SysML (Systems Modeling Language [15])-Blöcke, die jeweils ein Zustandsdiagramm beinhalten, und 15 Konfigurationsoptionen von Bedeutung für den Weichencontroller. Dennoch gibt es auch Spezifikationslücken, insbesondere da SysML (Version 1.6) die Ausführungsreihenfolge von Ereignissen in Zustandsdiagrammen

execution order of events in state diagrams, as well as a few other details, up to implementation, and EULYNX has seemingly not closed these gaps either. In addition, behavioural ambiguities were detected in the concrete model during the analysis of the specifications and subsequently reported to and discussed with the EULYNX cluster in charge. Some working assumptions have been made to resolve these issues.

No standardised specification was known for the PM simulation subsystem. A subsystem was specified in the same style as the EULYNX specifications by building on the already defined P3 IF (fig. 2) and on some ideas of the physical processes. This amounted to three interconnected SysML blocks containing one state diagram each (one of which is shown in fig. 4), and two configuration options.

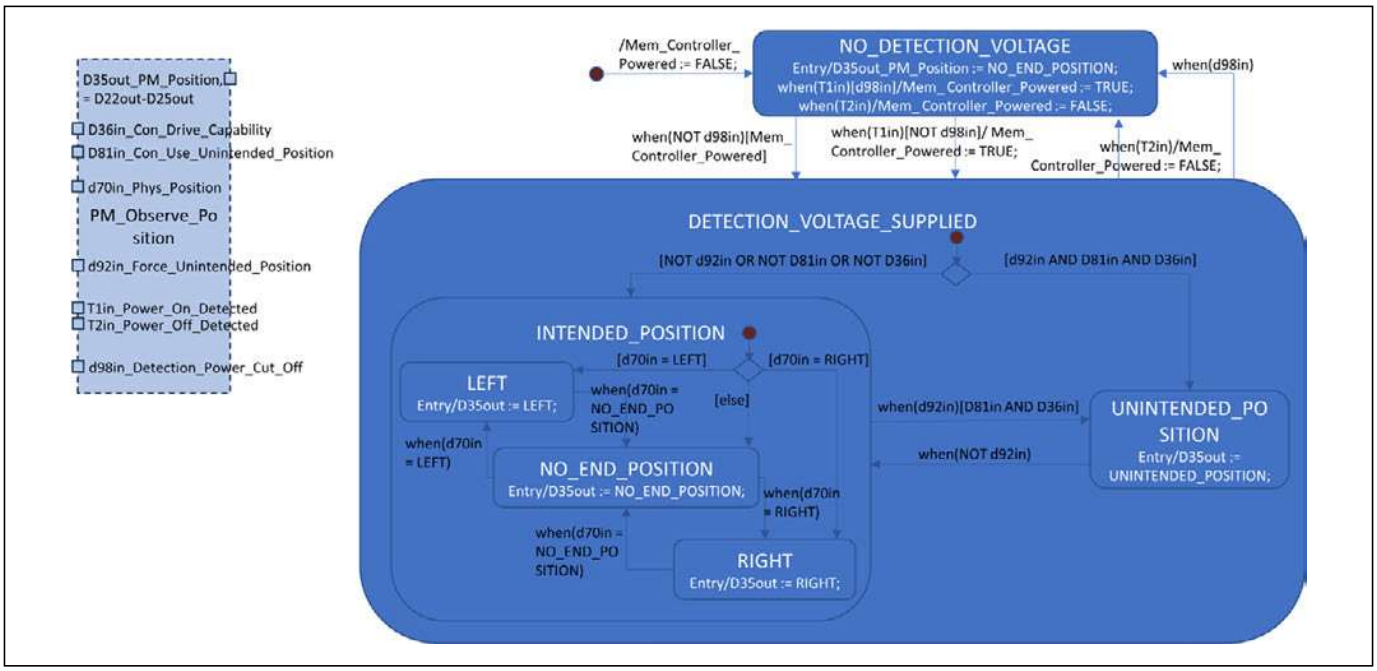

Bild 4: Zustandsautomat zur Beobachtung der PM-Position als Teil der Spezifikation der PM-Simulation
Fig. 4: A state diagram observing the PM position as part of the PM simulation specification

men ebenso wie wenige weitere Details der Implementierung überlässt, und auch EULYNX diese Lücken nicht zu schließen scheint. Zusätzlich wurden während der Analyse der Spezifikation Verhaltensmehrdeutigkeiten entdeckt, an das zuständige EULYNX-Cluster berichtet und mit diesem diskutiert. Für die weiteren Schritte wurden Annahmen getroffen, um die Probleme zu lösen.

Für das PM-Simulations-Teilsystem war keine standardisierte Spezifikation bekannt. Aufbauend auf das bereits definierte P3 IF (Bild 2) und auf eine gewisse Vorstellung der physischen Abläufe einer PM, wurde das Teilsystem im Stil der EULYNX-Spezifikationen spezifiziert. Daraus ergaben sich drei untereinander verbundene SysML-Blöcke, die jeweils ein Zustandsdiagramm beinhalten (eines davon ist in Bild 4 zu sehen), und zwei Konfigurationsoptionen.

3 Konzeption der Testkomponente

3.1 Weichencontroller

Aufgrund der vorhandenen strukturierten (verbundene SysML-Blöcke) und detaillierten EULYNX-Spezifikation für das Teilsystem Weiche und der Einschätzung, dass Leistungsoptimierung für Controller mit zeitlichen Anforderungen in der Größenordnung von 100 Millisekunden eher nachrangig ist, wurde entschieden, dass es am einfachsten ist, beim durch die Spezifikation gegebenen Design zu bleiben. Für einige wenige Teile, die absichtlich durch die Spezifikation offengelassen wurden (offene Ports von SysML-Blöcken), wurden meist entsprechende GUI-Elemente zur direkten Bedienung durch den Tester eingeführt (z. B. Buttons für Versorgungsspannung ein / aus).

3.2 Weichenantriebssimulation

Für den Entwurf der PM-Simulation wurden, neben dem durch EULYNX definierten P3 IF, weitere externe IF für Spannungsversorgung und zur physischen Weiche definiert, die über das GUI durch den Tester gesteuert werden können (z. B. Button für das Erzwingen einer unbeabsichtigten Position, was einem Ereignis wie dem Auffahren der Weiche entspricht). Außerdem kann die Zeit, die die PM

3 Designing the test component

3.1 The point controller

Given the availability of the structured (interconnected SysML blocks) and detailed EULYNX Subsystem Point specification, and considering the fact that performance optimisation is somewhat subordinate for controllers with timing requirements in the order of 100 milliseconds, the conclusion was reached that it was easiest to stick with the design given by the specification. For a few parts left intentionally open by the specification (open ports of SysML blocks), mostly corresponding GUI controls were designed for direct input by the tester (e.g. power on / off buttons).

3.2 The PM simulation

For the PM simulation design, in addition to the P3 IF defined by EULYNX, further external IF for the power supply and to the physical point were defined that can be controlled by the tester using the GUI (e.g. a button to force an unintended position representing an event such as the trailing of the point). Furthermore, the time it takes the PM to move from one end position to the other can be changed with the GUI.

The PM simulation subsystem has been structured into three logical components, each of which is responsible for a certain function of the PM simulation, as defined by a state diagram:

- The main function is the simulation of the PM movement based on inputs from the P3 IF, the physical point IF, on the drive power availability and the PM movement time. When designing it, it was necessary to define what happens in special situations such as conflicting inputs (e.g. move left AND move right). If a PM is configured as “no drive capability” (i.e. as a point detector only), the logical component responsible for the PM movement will not be instantiated.
- The second function is the detection of the PM position (see the corresponding state machine in fig. 4) based on the output of the PM movement simulation, forced unintended position and the availability of detection power. For the special

für die Bewegung von einer Endposition in die andere benötigt, per GUI geändert werden.
Die PM-Simulation wurde in drei logische Komponenten strukturiert, von denen jede für eine bestimmte Funktion der PM-Simulation verantwortlich ist, die durch ein Zustandsdiagramm definiert wurde:

- Die Hauptfunktion ist die Simulation der PM-Bewegung, basierend auf Eingaben vom P3 IF und dem IF zur physischen Weiche, auf der Verfügbarkeit der Antriebsspannung und der Laufzeit der PM. Für den Entwurf war es nötig zu definieren, was in speziellen Situationen wie konfligierenden Eingaben (z. B. Bewegung nach links UND Bewegung nach rechts) geschieht. Für den Fall einer als „nicht antriebsfähig" konfigurierten PM (d. h. die lediglich als Weichendetektor fungiert) wird die für die PM-Bewegung verantwortliche logische Komponente nicht instanziiert.
- Die zweite Funktion ist die Detektion der PM-Position (s. den entsprechenden Zustandsautomaten in Bild 4), basierend auf der Ausgabe der PM-Bewegungssimulation, dem Vorliegen einer erzwungenen unbeabsichtigten Position und der Verfügbarkeit der Detektionsspannung. Für den Spezialfall eines Weichendetektors wurde ein Design gewählt, das die Positionsinformation (d70in Eingabe) von den benachbarten antriebsfähigen PM ableitet.
- Die dritte Funktion ist die Detektion der Funktionsfähigkeit der PM, basierend auf der Verfügbarkeit der Antriebsspannung und darauf, ob die PM überhaupt für die Erkennung der Funktionsfähigkeit konfiguriert wurde.

3.3 Einbindung externer Weichen(antriebe)

Für die Einbindung einer externen PM (alternativ zur Nutzung der PM-Simulation) sieht das Design vor, dass das P3-IF als externes IF der TK bereitgestellt werden kann. Genauer gesagt wird es gleichzeitig zum Anschließen einer realen PM und auf dem GUI bereitgestellt – s. die zwei gestrichelten Linien in Bild 2. Das GUI dient dazu, den Status des IF dem Tester anzuzeigen, erlaubt es dem Tester jedoch zugleich, Eingaben zu ändern. Letzteres wurde zugelassen, um (a) die direkte Simulation wirklich jedes PM-Szenarios durch einen Tester zu ermöglichen, ohne ein reales Gerät anschließen zu müssen, und (b) das Erzwingen oder Auflösen spezieller Situationen durch den Tester zu ermöglichen (z. B. die einer unbeabsichtigten Position, ohne eine angeschlossene reale Weiche zu beschädigen).

3.4 Konfiguration und Manipulation

Es wurde ein einfaches textbasiertes Parameter/Wert-Konfigurationsdateiformat entworfen, um eine flexible Nutzung verschiedener Konfigurationen zu ermöglichen. Als erstes definiert es Festwert-Daten, die während eines Testlaufs nicht veränderbar sind: 15+2 Parameter für Weichencontroller- und PM-Konfiguration, wie z. B. zur Aktivierung der Redrive-Funktionalität des Controllers. Zweitens definiert das Format initiale Werte für Simulationsparameter, die per GUI während eines Testlaufs veränderbar sind: 8+12 Parameter für Weichencontroller und PM

case of detector only PM a design was chosen that derives the position information (d70in input) from the neighbouring PM with drive capability.

- The third function is the detection of the PM's ability to move based on the availability of drive power and on whether the PM has been configured to detect the inability to move at all.

3.3 Connecting external PM

For the connection of external PM (as an alternative to using the PM simulation), it has been foreseen by the design that the P3 IF can be exposed as an external TC IF. More precisely, it is made available both for connecting real PM and on the GUI in parallel: see the two dashed lines in fig. 2. The GUI is used to display the IF status to the tester, but at the same time it also allows the tester to change the inputs. The latter has been allowed (a) to enable the simulation of literally any PM scenario directly by a tester without the need for a connection to a real device and (b) to enable the tester to force or resolve special situations (e.g. force an unintended position without damaging a connected real point).

3.4 Configuration and manipulation

A simple text-based parameter/value configuration file format has been designed to allow for the flexible use of different configurations. Firstly, it defines fixed-value data that cannot be changed during a test run: 15+2 parameters for point controller and PM configuration, such as for enabling the controller's redrive functionality. Secondly, it also defines initial values for the simulation parameters that can be changed during a test run via the GUI: 8+12 parameters for point controller and PM such as whether the movement to the left is initially blocked for a PM. Thirdly, it also defines 9+9 initially displayed failure injection options for the GUI (see paragraph below).
A default configuration has been designed that allows the ad-hoc use of the TC. It is possible to reset or reconfigure the TC without shutting it down. A separate RaSTA configuration XML file is used (fig. 2) to enable successful RaSTA connections with any devices performing the role of an EULYNX interlocking.
Manifold manipulation capabilities via the GUI have been planned (but not yet fully implemented) for the behaviour of both the point controller and the PM simulation. Those functions with timing requirements specified by EULYNX can be artificially delayed for the controller, while the outputs, event processing and state diagram transitions can be manipulated for both subsystems by suppressing, inserting or delaying them etc.

4 Implementing the test component

4.1 Hardware

Important TC hardware requirements are

- support for connecting the SCI, which is normally done using an Ethernet cable;

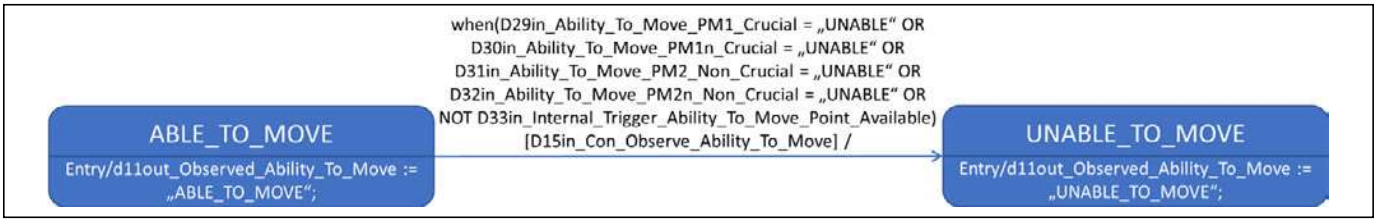

Bild 5: Beispieltransition eines Zustandsautomaten, neu gezeichnet nach [8]
Fig. 5: An example state diagram transition, redrawn from [8]

```
// Process change event involving D29in to D33in ports
void F_Observe_Ability_To_Move::processChangeEvent_D29inToD33in(int event)
{
  if (currentState == ABLE_TO_MOVE) {
    // Transition ABLE_TO_MOVE -> UNABLE_TO_MOVE
    if (event == 0) {
      if (Con_Observe_Ability_To_Move) {
        […]
        enterUnableToMove([…]);
        currentState = UNABLE_TO_MOVE;
      }
    }
  }
  […]
}
```

Bild 6: C++ Code für die Zustands-diagramm-Transition aus Bild 5; event 0 steht dabei für das change event „when(D29in_Ability …Point_Available)" der Transition
Fig. 6: The C++ code for the state diagram transition from fig. 5, where event 0 stands for the change event "when(D29in_Ability… Point_Available)" from the transition

wie z. B. ob Linksbewegungen initial für eine PM blockiert sind. Drittens definiert es 9+9 initial angezeigte Fehlerinjektionsoptionen für das GUI (s. Absatz weiter unten).
Eine Default-Konfiguration wurde entworfen, die eine sofortige Nutzung der TK erlaubt. Es ist möglich, die TK zurückzusetzen oder umzukonfigurieren, ohne sie herunterzufahren. Um erfolgreiche RaSTA-Verbindungen mit allen Geräten zu ermöglichen, die die Rolle eines EULYNX-Stellwerks übernehmen, wird eine eigene RaSTA-XML-Konfigurationsdatei genutzt (Bild 2).
Es wurden vielfältige Manipulationsmöglichkeiten per GUI geplant (noch nicht vollständig implementiert), sowohl für das Verhalten des Weichencontrollers als auch der PM-Simulation. Beim Controller können diejenigen Funktionen mit durch EULYNX spezifizierten Zeitanforderungen künstlich verzögert werden. Und bei beiden Teilsystemen können Ausgaben, Ereignisverarbeitung und Zustandsdiagramm-Übergänge durch Unterdrücken, Einfügen, Verzögerung usw. manipuliert werden.

4 Implementierung der Testkomponente

4.1 Hardware

Wichtige Anforderungen an die TK-Hardware sind

- Unterstützung für den Anschluss des SCI, der normalerweise per Ethernetkabel erfolgt,
- ein IF zur Anzeige des GUI und zur Bedienung der TK hierdurch,
- ein IF zur Einbindung externer PM,
- einfache Transportierbarkeit (als vielseitige TK) und
- geringe Kosten (um die Testkosten moderat zu halten, selbst wenn viele solcher TK genutzt werden).

Es wurde entschieden, einen Raspberry Pi Mikrocomputer (Modell 4B) mit dem Betriebssystem Raspberry Pi OS zu verwenden, eingebaut in ein stabiles Gehäuse und zusammen mit einem über HDMI verbundenen transportablen 10.1" Touchscreen. Dies genügt eindeutig den obigen Anforderungen (wobei das GPIO-IF für die externen PM verwendet wird) und bietet sogar weitere Möglichkeiten (z. B. WLAN als Alternative zum Ethernetkabel).

4.2 Software

Die TK-Software besteht größtenteils aus handgeschriebenem C++ Code und bindet den C Code der bereits zuvor existierenden RaSTA-Implementierung ein. Die logischen Komponenten der Teilsysteme mit ihren Zustandsautomaten wurden in strukturierter Weise in C++ Klassen überführt. Insbesondere wurden Klassenmethoden zur Evaluation geänderter Eingaben oder von Timern erstellt, die Ereignisse

- an IF to display the GUI and operate the TC through it;
- an IF for connecting external PM;
- easy transportability (as a versatile TC); and
- low cost (to keep testing costs moderate even if many such TC are used).

A decision was made to use a Raspberry Pi microcomputer (model 4B) running the Raspberry Pi OS, housed in a solid case and with a transportable 10.1" touchscreen connected by HDMI. This clearly fulfils the aforementioned requirements (using its GPIO IF for external PM) and even provides more options (e.g. using Wi-Fi as an alternative to the Ethernet cable).

4.2 Software

The TC software is mainly handwritten in C++, incorporating the pre-existing RaSTA implementation in C. The logical components of the subsystems and their state diagrams have been transformed into C++ classes in a structured manner. In particular, member functions have been created to evaluate changed inputs or timers that may generate events; further member functions have been created for the processing of these change or timeout events (see fig. 5 for an example transition that translates to the C++ code shown in fig. 6). The event scheduling is managed by a central control class based on one queue for each subsystem; the information at the internal and external IF is relayed by this control class as well. Version 4.0 of the gtkmm framework [16] has been used to create the GUI. gcc has been used on the Raspberry Pi to compile the software.

5 Validating the test component

5.1 The test environment

For a basic validation of the TC, it has been tested using the DLR RailSiTe lab. The core of the lab consists of a modular software for simulation and the testing of different signalling subsystems. In the current context, the lab executed the test cases and acted as the interlocking side of the SCI. The detailed definition of the SCI telegrams (SCI-P version 4) to be generated by the lab and those expected back from the point controller has been implemented in the lab (telegram structure) and given as part of the test cases (telegram values). The RailSiTe® includes a RaSTA protocol implementation and runs on a computer with an Ethernet port, so that a connection with the TC can be easily established after defining some connection parameters (IP address, UDP ports, RaSTA IDs) on both sides.

generieren können; weitere Klassenmethoden wurden für die Abarbeitung dieser Änderungs- oder Timeout-Ereignisse erstellt (Bild 5 für einen Beispiel-Zustandsübergang, dessen Übersetzung in C++ Code in Bild 6 zu finden ist). Das Ereignis-Scheduling wird von einer zentralen Steuerungsklasse übernommen, basierend auf einer Warteschlange für jedes Teilsystem; auch die Weiterleitung von Informationen an internen und externen IF ist Aufgabe dieser Steuerungsklasse. Für die Erstellung des GUI kam das gtkmm-Framework [16] in Version 4.0 zum Einsatz. gcc wurde auf dem Raspberry Pi verwendet, um die Software zu kompilieren.

5 Validierung der Testkomponente

5.1 Testumgebung

Als einfache Validierung der TK wurde diese im DLR- Labor RailSiTe getestet. Im Kern besteht das Labor aus einer modularen Software für die Simulation und den Test verschiedener signaltechnischer Teilsysteme. Im vorliegenden Kontext führte das Labor Testfälle aus und agierte als Stellwerksseite des SCI. Die detaillierte Definition der vom Labor zu generierenden und der vom Weichenkontroller erwarteten SCI-Telegramme (SCI-P Version 4) wurde im Labor implementiert (Telegrammstruktur) und als Teil der Testfälle spezifiziert (Telegramwerte). Das RailSiTe beinhaltet eine RaSTA-Protokoll-Implementierung und läuft auf einem Computer mit Ethernetport, sodass eine Verbindung mit der TK einfach hergestellt werden kann, sobald auf beiden Seiten einige Verbindungsparameter (IP-Adresse, UDP-Ports, RaSTA-IDs) definiert wurden.

5.2 The test cases and test execution

EULYNX Baseline 4 Release 1 has delivered a set of certification test cases for EULYNX point controllers. However, this is neither complete (see [17]) nor fully compatible with Baseline 4 Release 2 point controllers (e.g. using telegrams in version 3 of SCI-P), so the existing test cases have been adapted and completed for the purpose of TC validation. Five main test cases have been defined for use with the default TC configuration.

These test cases have been arranged into two test sequences that are executable by the lab, including additional prefix test cases for establishing a connection at the SCI. During test execution, the TC GUI was used by the tester to manipulate PM simulations so that the PM outputs were delayed or changed as required. The log messages recorded by the TC for both test sequences have been (automatically) saved in a text file (tab. 1) for a detailed evaluation of the test runs. The central block of five log entries in tab. 1 shows a test sequence where the second PM's drive voltage becomes insufficient (encoded as Boolean input d99in, see the bottom step of the block). The point controller is notified that the PM is unable to move (the following step upwards, cf. the transition from fig. 5), processes the change and finally sends a telegram informing the interlocking that the point is considered as unable to move (the uppermost step in the block).

5.3 Test results

The test process revealed some issues that were successively removed until all the test cases (see Section 5.2) were able to be executed successfully. Apart from some problems with lab con-

5.2 Testfälle und Testdurchführung

Mit EULYNX Baseline 4 Release 1 wurde ein Satz von Zertifizierungstestfällen für EULYNX-Weichencontroller bereitgestellt. Allerdings ist dieser weder vollständig (s. [17]) noch vollständig kompatibel mit Baseline 4 Release 2 Weichencontrollern (z. B. werden Telegramme in SCI-P Version 3 verwendet). Daher wurden zum Zweck der TK-Validierung existierende Testfälle angepasst und vervollständigt. Fünf wesentliche Testfälle wurden, zur Nutzung mit der Default-Konfiguration der TK, definiert. Diese Testfälle wurden in zwei Testsequenzen angeordnet, die im Labor ausführbar sind, inklusive zusätzlicher vorgeschalteter Testfälle für den Verbindungsaufbau am SCI. Während der Testdurchführung wurde das GUI vom Tester verwendet, um die PM-Simulationen so zu manipulieren, dass die PM-Ausgaben wie benötigt verzögert oder verändert wurden. Die durch die TK für beide Testsequenzen aufgezeichneten Log-Nachrichten wurden (automatisch) in eine Textdatei (Tab. 1) für die detaillierte Auswertung der Testläufe gespeichert. Der mittlere Block in Tab. 1 aus fünf Logeinträgen zeigt eine Testabfolge, in der die Antriebsspannung der zweiten PM unter die ausreichende Spannung fällt (kodiert als Bool'sche Eingabe d99in, siehe unterster Schritt des Blocks). Der Weichencontroller wird benachrichtigt, dass die PM nicht funktionsfähig ist (im folgenden Schritt darüber, vgl. die Transition aus Bild 5), verarbeitet diese Änderung und sendet schließlich ein Telegramm, das das Stellwerk informiert, dass die Weiche als nicht umlauffähig angesehen wird (oberster Schritt des Blocks).

5.3 Testergebnisse

Die im Rahmen des Testprozesses gefundenen Probleme wurden schrittweise behoben, bis alle Testfälle (s. Abschnitt 5.2) erfolgreich ausgeführt werden konnten. Abgesehen von einigen Problemen mit der Laborkonfiguration und der Testfallbeschreibung wurden die folgenden Probleme in der ersten Version der TK gefunden:

- ein Copy-and-Paste-Fehler
- eine vergessene Codezeile
- fehlerhafter Prüfsummen-(De-)Kodierungsalgorithmus
- zwei fehlerhaft strukturierte bedingte (if…then…else) Anweisungen (jeweils der gleiche Fehler).

Für eine handgeschriebene Software mit etwa 10 000 Codezeilen (exkl. RaSTA-Implementierung) sind das extrem wenige Fehler, von denen sich lediglich der letzte Aufzählungspunkt auf den Zustandsdiagramm-Code bezieht. Mögliche Erklärungen hierfür sind, dass (a) die fünf Testfälle den TK-Code nur teilweise abdecken, (b) die modellbasierte Spezifikation (SysML-Blöcke und Zustandsdiagramme) auf sehr systematische Weise in Code überführt werden konnten und (c) je zwei GUI-Fehler und zwei Fehler in der Zustandsdiagramminitialisierung bereits vor figuration and test case coding, the following issues were discovered in the first version of the TC:

- a copy-and-paste error;
- a forgotten line of code;
- a wrong checksum de-/encoding algorithm;
- two incorrectly structured conditional (if…then…else) statements (same error).

This constitutes an extremely small number of errors for a handwritten software with roughly 10,000 lines of code (excluding the RaSTA implementation), while only the last bullet point refers to state diagram code parts. Some explanations for that can be found in the fact that (a) the five test cases only partially cover the TC code, (b) the model-based specification (SysML blocks and state diagrams) was able to be transferred to the code in a very systematic manner and (c) a couple of GUI errors and state diagram initialisation errors (visible in the log) had been identified visually and fixed in advance of the testing. Nevertheless, the test results (in addition to validating the fact that the important functions work as expected) have given rise to the assumption that the TC code is of a high quality.

Furthermore, an issue with the EULYNX specifications has also been detected: They do not seem to constrain the scheduling of events for state diagrams that has been (intentionally) left open by SysML (cf. Section 2.3). This can lead to presumably undesired sequences of telegrams observed at the SCI during testing. Recorded log entries were used to evaluate the test runs. The interpretation of the logs was feasible, but sometimes a bit arduous, because no cause-effect relation between the log entries was recorded.

6 Summary, conclusions and future work

In summary, a versatile TC has been designed, implemented and validated. It comprises a point controller as specified by EULYNX and a PM simulation, as well as ample possibilities for configuration and interaction. Based on a Raspberry Pi microcomputer, the TC can be easily transported and connected using an Ethernet cable at the point controller's SCI. It can be used as part of many different test set-ups: i.e. as SUT or as part of the test environment, with an internally simulated or externally connected/simulated PM, with nominal behaviour or with manipulated behaviour (not yet fully implemented).

The conclusions from the work include the following:

Zeitstempel / Timestamp	Kritikalität / Criticality	Teilsystem / System Part	Instanz / Instance	Kategorie / Category	Beschreibung / Description
03:05:44:114	Info	IXL - Controller	0	Status	SCP connection closed
[...]					
03:04:01:805	Info	IXL - Controller	0	Data <-	Sent "Ability To Move Point(ability=UNABLE)" telegram
03:04:01:790	Info	Point Controller	1	Event	Processing change of "d11in = UNABLE_TO_MOVE" in machine "F_Control_Point"
03:04:01:776	Info	Point Controller	1	Event	Processing change of "d11in = UNABLE_TO_MOVE" in machine "F_SCI_P_Report"
03:04:01:761	Info	Point Controller	1	Event	Processing change of "D29in = UNABLE OR D30in = UNABLE OR D31in = UNABLE OR D32in = UNABLE OR NOT D33in" in machine "F_Observe_Ability_To_Move"
03:04:01:747	Info	Point Machine	2	Event	Processing change of "NOT d99in" in machine "PM_Observe_Ability_To_Move"
[...]					
03:03:49:202	Info	IXL - Controller	0	Status	SCP connection established successfully
[...]					
03:03:03:895	Info	Test Component	0	Status	Default configuration loaded

Tab. 1: Log-Ausschnitt einer Testausführung, von unten nach oben zu lesen; IXL = Stellwerk (engl. Interlocking)
Tab. 1: Log excerpt from a test run, to be read from the bottom up; IXL = interlocking

den Tests visuell identifiziert (im Log sichtbar) und behoben wurden. Trotzdem lassen die Testergebnisse – neben der Validierung, dass wichtige Funktionen sich wie erwartet verhalten – vermuten, dass der TK-Code von hoher Qualität ist.

Zusätzlich wurde ein Problem in den EULYNX-Spezifikationen entdeckt: Sie scheinen das Scheduling von Ereignissen für Zustandsdiagramme nicht weiter zu spezifizieren, das (absichtlich) durch SysML offengelassen wird (vgl. Abschnitt 2.3), was zu vermutlich ungewollten Telegrammabfolgen am SCI führen kann, die während der Tests beobachtet wurden.

Zur Auswertung der Testläufe wurden die aufgezeichneten Log-Einträge genutzt. Die Interpretation der Logs war machbar, manchmal jedoch etwas mühsam, da keine Ursache-Wirkungs-Zusammenhänge zwischen den Log-Einträgen aufgezeichnet wurden.

6 Zusammenfassung, Fazit und Ausblick

Zusammenfassend wurde eine vielseitige TK entworfen, implementiert und validiert. Sie umfasst einen Weichencontroller gemäß EULYNX-Spezifikationen und eine PM-Simulation und bietet eine Vielzahl von Möglichkeiten der Konfiguration und Interaktion. Als Software auf einem Raspberry Pi Mikrocomputer ist die TK einfach transportierbar und per Ethernetkabel am SCI des Weichencontrollers anschließbar. Die TK kann als Bestandteil vieler verschiedener Testanordnungen verwendet werden: als SUT oder als Teil der Testumgebung, mit intern simulierten oder extern verbundenen / simulierten PM, mit nominellem oder mit manipuliertem Verhalten (letzteres noch nicht vollständig implementiert).

- The detailed and clear design of the TC has been an important precondition for arriving at a high-quality software. This has included maintaining complete lists of IF parameters, configuration options, GUI controls and manipulation options; it has also included a detailed description of the PM behaviour using state machines and additional text in order to cope with each possible configuration and input combination.
- Transforming many state machines into C++ code is admittedly possible on the basis of a clear transformation scheme, yet laborious. It is estimated that if more than one EULYNX subsystem specification is to be implemented, it would be more efficient to first implement an automatic transformation and apply it to the exported XMI (XML Metadata Exchange) files of the specification models provided by EULYNX.
- Depending on the particular kind of manipulation, this can be easily integrated into the state-machine-based code structure (reading / setting information that is stored with or passed between state machines or the manipulation of telegrams or events) or requires major local (transition manipulation) or global (passing on additional delay values together with the information) changes.
- The detailed model-based specification leads to a surprisingly high ad-hoc quality of the implementation as witnessed by the results of the basic TC testing. However, the tests also showed that it is not trivial to come up with an appropriate solution for the parts (event scheduling) that remain unspecified by EULYNX and SysML.

Das folgende Fazit kann aus den Arbeiten gezogen werden:

- Die detaillierte und klare Konzeption der TK war eine wichtige Voraussetzung, um eine Software hoher Qualität zu erreichen. Dies schloss die Pflege einer vollständigen Liste an IF-Parametern, Konfigurationsoptionen, GUI-Steuerelementen und Manipulationsoptionen ein; ebenso gehörte hierzu eine detaillierte Beschreibung des PM-Verhaltens als Zustandsautomaten und zusätzlicher Text, um jede mögliche Konfiguration und jede Eingabenkombination zu behandeln.
- Die Übersetzung mehrerer Zustandsautomaten in C++ Code ist, auf Grundlage eines klaren Übersetzungsschemas, möglich, aber aufwendig. Schätzungsweise ist, sobald mehr als ein EULYNX-spezifiziertes Teilsystem implementiert werden soll, die Implementierung einer automatischen Übersetzung und deren anschließende Anwendung auf aus dem durch EULYNX erstellten Spezifikationsmodell exportierte XMI (XML Metadata Exchange) Dateien effizienter.
- Abhängig von der jeweiligen Art der Verhaltensmanipulation kann diese leicht in die zustandsautomatenbasierte Codestruktur integriert werden (Lesen / Schreiben von Informationen, die in den Zustandsautomaten gespeichert sind oder zwischen ihnen weitergegeben werden, oder Manipulation von Telegrammen oder Ereignissen) oder erfordert größere lokale (Manipulation von Zustandsübergängen) oder globale (Weitergeben zusätzlicher Verzögerungswerte zusammen mit einer Information) Änderungen.
- Die detaillierte modellbasierte Spezifikation führt zu überraschend hoher ad-hoc Qualität von Implementierungen, wie die Ergebnisse grundlegender Tests der TK zeigen. Die Tests zeigten jedoch ebenso, dass es nicht trivial ist, eine passende Lösung für diejenigen Teile (Ereignis-Scheduling) zu finden, die durch EULYNX und SysML unspezifiziert bleiben.

Eine erste Anwendung der TK als SUT ist im Zusammenhang mit einer neuen EULYNX-Prüfumgebung geplant, die aktuell von der Deutschen Bahn AG im Rahmen des europäischen R2DATO-Projekts aufgebaut wird. Eine andere Anwendung als Teil einer Testumgebung könnte die Integration in das RailSiTe®-Labor des DLR als Weichensimulation sein. Weitere Einsatzszenarien jenseits von EULYNX-Tests, wie für die Validierung von EULYNX-Laboren oder zur Ansteuerung realer Weichen im nicht-betrieblichen Kontext, sind vorstellbar.

Neben der Vervollständigung der Implementierung zählen zu den Ideen für die Weiterentwicklung die Ermöglichung von Standalone-PM-Simulationen und sich um die flexible Integration künftiger Versionen des Standards für das EULYNX-Weichen-Teilsystem zu kümmern. ■

Danksagung

Der überwiegende Teil der Arbeiten wurde im Rahmen des FP2 - R2DATO Projekts durchgeführt. Das Projekt wird von der Europäischen Union durch das Europe's Rail Joint Undertaking (JU) unter der Zuwendungsvereinbarung Nr. 101102001 gefördert. Eine erste Vorgängerversion der Software (C++ Kodierung von Weichencontroller-Zustandsautomaten gemäß EULYNX Baseline 4 Release 1) wurde während eines Praktikums von Hr. Tsogtbaatar Mendbayar beim DLR im Sommer 2022 erstellt.

Zum Ausdruck gebrachte Ansichten und Meinungen sind ausschließlich diejenigen des Autors und geben nicht notwendigerweise diejenigen der Europäischen Union oder des Europe's Rail JU wieder. Weder die Europäische Union noch das JU können hierfür haftbar gemacht werden.

AUTOR | AUTHOR

Dr. Daniel Schwencke
Verifikations- und Validierungsmethoden / *Verification and Validation Methods*
Deutsches Zentrum für Luft- und Raumfahrt e.V. / *German Aerospace Center*
Institut für Verkehrssystemtechnik / *Institute of Transportation Systems*
Anschrift / *Address*: Lilienthalplatz 7, D-38108 Braunschweig
E-Mail: daniel.schwencke@dlr.de

A first application of the TC as SUT is planned in connection with the new EULYNX test bed currently being developed by DB within the European R2DATO project. Another application as part of a test environment may involve integration into DLR's RailSiTe® lab as point simulation. Further uses beyond EULYNX testing are conceivable, such as the validation of EULYNX labs or controlling real points in non-operational settings.

In addition to the completion of the implementation, ideas for further development may include enabling standalone PM simulations and taking care of the flexible integration of future versions of the EULYNX Subsystem Point standards. ■

Acknowledgement

Most of the works reported have been conducted in the course of the FP2 - R2DATO project. The project is funded by the European Union through the Europe's Rail Joint Undertaking (JU) under Grant Agreement No. 101102001. A first precursor software (point controller state machine C++ encoding according to EULYNX Baseline 4 Release 1) has been created during an internship of Mr. Tsogtbaatar Mendbayar at DLR in summer 2022.

Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the Europe's Rail JU. Neither the European Union nor the JU can be held responsible for them.

LITERATUR | LITERATURE

[1] EULYNX Website, https://eulynx.eu, accessed on 03/05/2024 2:14 p.m.
[2] EUG and EULYNX partners (2022): RCA Architecture Poster, Preliminary issue, With OCORA contribution. RCA.Doc.40, version 0.4 (0.A) from 26/04/2022, published in Baseline 0 Release 4
[3] OCORA public Github repository, https://github.com/OCORA-Public/Publications, accessed on 03/05/2024 4:55 p.m.
[4] Europe's Rail System Pillar website, https://rail-research.europa.eu/system_pillar/system-pillar-architecture/, accessed on 03/05/2024 4:58 p.m.
[5] Salunkhe, S.; Berglehner, R.; Rasheeq, A. (2021): Automatic Transformation of SysML Model to Event-B Model for Railway CCS Application. In: Raschke, A.; Méry, D. (eds): Rigorous State-Based Methods. ABZ 2021. Lecture Notes in Computer Science, vol 12709. Springer, Cham. https://doi.org/10.1007/978-3-030-77543-8_14
[6] Bouwman, M.; van der Wal, D.; Luttik, B.; Stoelinga, M.; Rensink, A. (2023): A Case in Point: Verification and Testing of a EULYNX Interface. Form. Asp. Comput. 35, 1, Article 2. https://doi.org/10.1145/3528207
[7] EULYNX (2022): Certification plan. Eu.Proc.7, version 2B from 22/11/2022, published in Baseline 4 Release 1
[8] DIN VDE V 0831-200 (2015-06): Electric signalling systems for railways – Part 200: Safe transmission protocol according to DIN EN 50159 (VDE 0831-159)
[9] EULYNX (2023): Requirements specification for subsystem Point. Eu.Doc.36, version 4.3 (0.A) from 28/06/2023, published in Baseline 4 Release 2
[10] EULYNX (2023): Generic interface and subsystem requirements. Eu.Doc.20, version 4.0 (3.A) from 27/06/2023, published in Baseline 4 Release 2
[11] EULYNX (2023): Generic interface and subsystem requirements for SCI. Eu.Doc.119, version 1.0 (3.A) from 27/06/2023, published in Baseline 4 Release 2
[12] EULYNX (2023): Interface specification SCI Generic. Eu.Doc.93, version 3.2 (0.A) from 28/06/2023, published in Baseline 4 Release 2
[13] EULYNX (2023): Interface specification SCI-P. Eu.Doc.38, version 4.2 (0.A) from 27/06/2023, published in Baseline 4 Release 2
[14] EULYNX (2023): EULYNX System architecture specification. Eu.Doc.16, version 2.2. (0.A) from 27/06/2023, published in Baseline 4 Release 2
[15] OMG SysML Website, http://www.omgsysml.org/, accessed on 15/05/2024 11:51 a.m.
[16] gtkmm Reference Manual Website, https://gnome.pages.gitlab.gnome.org/gtkmm/, accessed on 15/05/2024 12:47 a.m.
[17] EULYNX (2022): Scope and coverage justification for EULYNX Certification test cases. Version 1.A from 18/11/2022, published in Baseline 4 Release 1