VTT Technical Research Centre of Finland

Using model checking for interlocking software verification

Pakonen, Antti; Turunen, Juha

Published in:
SIGNAL+DRAHT

Published: 01/09/2023

Document Version
Publisher's final version

Verwendung der Modellprüfung für die Verifikation von Stellwerkssoftware

Using model checking for interlocking software verification

Antti Pakonen | Juha Turunen

Die Anwendung verschiedener Verifikationsmethoden ist ein wichtiger Bestandteil im Entwicklungsprozess sicherheitsrelevanter Systeme. Einige Methoden eignen sich für die frühen, andere für die späteren Phasen des Lebenszyklus. Das Wichtigste ist, dass alle Phasen des Lebenszyklus in ausreichender Weise und mit der für den jeweiligen Zweck am besten geeigneten Methode überprüft werden können. Bei Mipro haben wir festgestellt, dass die formale Verifikation die Vielfalt erhöht, um die bestmöglichen Ergebnisse durch die Verifikation zu erzielen und die Qualität und Sicherheit der Lieferungen zu verbessern.

1 Verwendung der Modellprüfung für die Verifikation von Stellwerkssoftware

1.1 Formale Verifikation in CENELEC-Sicherheitsnormen

Der Entwicklungsprozess sicherheitskritischer Systeme wird in der Regel in einem Lebenszyklusmodell dargestellt, wobei die Definition des Lebenszyklus ein einfaches Top-down-Linienmodell, ein V-Modell oder eine angepasste Kombination beider Modelle ist. Das Prinzip ist leicht zu verstehen. Der Lebenszyklus beginnt mit dem High-Level-Konzept und geht weiter zu einer detaillierteren Systemdefinition unter Berücksichtigung der Risikoanalyse und der Anforderungsspezifikationen, gefolgt von Entwurf und Implementierung einschließlich der erforderlichen Testphasen und endet mit der Validierung der Anlage oder des Systems. Um die Integrität des Prozesses und den korrekten Betrieb des implementierten Systems zu gewährleisten, müssen wir alle Ergebnisse der verschiedenen Lebenszyklusphasen überprüfen. Dies bedeutet, dass die Verifikation, wie sie in den Normen definiert ist, *„ein Prozess der Prüfung ist, gefolgt von einer Beurteilung auf der Grundlage von Nachweisen, dass die Output-Elemente (Prozess, Dokumentation, Software oder Anwendung) einer bestimmten Entwicklungsphase die Anforderungen dieser Phase in Bezug auf Vollständigkeit, Korrektheit und Konsistenz erfüllen“* [1] und somit ein entscheidender Teil des Entwurfs- und Implementierungsprozesses, der durch Prüfung und Erbringung eines objektiven Nachweises bestätigt, dass die Anforderungen erfüllt wurden.

Da der Rahmen für die funktionale Sicherheit das Modell für das Lebenszyklusmanagement mit einer Vielzahl von Anforderungen vorgibt, definieren die Normen auch eine umfassende Liste verschiedener Techniken und Maßnahmen für den Umgang mit systematischen Fehlern, die in jeder Lebenszyklusphase des Entwicklungsprozesses von Menschen verursacht werden können. Das bedeutet nicht, dass alle vorgestellten Techniken ausgewählt werden müssen, sondern dass eine Kombination aus ihnen erforderlich ist. Je nach geforderter Sicherheitsintegritätsstufe wird die geeignete Kombination verschiedener Methoden in den Tabellen der Techniken und

The application of different verification methods is a prominent part of the development process for safety related systems. Some methods are suitable for the early lifecycle phases, while others are best used later. In the end, the most significant thing is that all the lifecycle phases can be verified in a sufficient manner and using the method best suited for the purpose. At Mipro, we have found that formal verification increases diversity in order to achieve the best possible verification results and improve the quality and safety of the deliveries.

1 Using model checking for interlocking software verification

1.1 Formal verification in CENELEC signalling standards

The development process for safety critical systems is typically presented in a lifecycle model where the definition of the lifecycle is a straightforward top-down line model or V-model or a tailored combination of the two. The principle is easy to understand. The lifecycle starts from the high-level concept and moves forward to more detailed system definition that considers the risk analysis and requirement specifications, followed by the design and implementation, including all the necessary test phases, and ending with the validated equipment or system. We need to verify all the outcomes from the different lifecycle phases in order to ensure the integrity of the process and the correct operation of the implemented system. This means that verification, as defined in the norms, “*is a process of examination followed by a judgment based on evidence that the output items (process, documentation, software or application) of a specific development phase fulfil the requirements of that phase with regard to completeness, correctness and consistency*” [1] and as such is a crucial part of the design and implementation process that provides confirmation through examination and objective evidence as to the fact that the requirements have been met.

Given that the functional safety framework presents the lifecycle management model with a lot of requirements, the standards also define a comprehensive list of different techniques and measures for dealing with systemic failures that might be introduced into any lifecycle phase of the development process by human beings. This does not mean that all the presented techniques will be selected, but that a combination of them is required. The adequate combination of the different methods is listed in the tables of techniques and measures depending on the required safety integrity level. These

Maßnahmen aufgeführt. Diese sind mit Buchstaben gekennzeichnet, die angeben, ob die Implementierung der Methode obligatorisch (M), dringend empfohlen (HR) oder empfohlen (R) ist. Es ist auch möglich, dass einige Methoden nicht empfohlen werden (NR). Die Kombination angemessener und geeigneter Techniken besteht in der Regel aus obligatorischen und / oder dringend empfohlenen Methoden.

Eine Reihe von Techniken sind formale Methoden. Die formalen Methoden sind ein übergeordnetes Konzept, das sich auf *„mathematisch strenge Techniken und Werkzeuge für die Spezifikation, den Entwurf und die Verifikation von Software- und Hardwaresystemen“* bezieht [1]. Die formalen Methoden sind keine obligatorischen Techniken, werden aber dennoch dringend empfohlen, wenn die höchste Sicherheitsintegrität (Safety Integrity Level, SIL) erforderlich ist (SIL3 und SIL4). Bei niedrigeren Sicherheitsintegritätsstufen (SIL1 und SIL2) werden diese Techniken als empfohlen definiert. Für die Softwareentwicklung verweist die Norm EN 50128 für mehrere Lebenszyklusphasen auf den Einsatz formaler Methoden, legt aber nicht im Detail fest, welche formale Methodentechnik verwendet werden soll. Stattdessen werden im Anhang mehrere Techniken unter dem Oberbegriff der formalen Methode aufgeführt.

In diesem Beitrag gehen wir näher darauf ein, wie Mipro die formale Verifikationstechnik eingesetzt und genutzt hat. Mipro hat sich zur Unterstützung des Produktentwicklungsprozesses für den Modellprüfungsdienst von VTT entschieden. Anlass für den Einsatz einer Modellprüfung war die Suche nach einer neuen, vielseitigen Verifikationstechnik, um verborgene Entwurfsprobleme in wiederverwendbaren Softwaremodulen aufzudecken und eine schnelle und einfache Lösungsiteration zur Erhöhung der Produktzuverlässigkeit zu erreichen.

1.2 Die formale Verifikation führt zu logischen Nachweisen

Formale Sprachen und Methoden bieten einen systematischen Ansatz für die Spezifikation und den Entwurf von Anforderungen. Mathematisch strenge Techniken ebnen formalen Methoden den Weg für Klarheit und Eindeutigkeit, ermöglichen aber auch die formale Verifikation – den logischen Nachweis der Korrektheit.

Model Checking [2] ist eine formale Verifikationsmethode, bei der ein Computerwerkzeug, ein sogenannter Model Checker, verwendet wird, um zu überprüfen, ob eine formale Eigenschaft für ein gegebenes Systemmodell gilt (Bild 1). Das Ergebnis ist entweder ein mathematischer Beweis, dass die Eigenschaft immer gilt, oder ein Gegenbeispiel – ein Modellausführungspfad, der die Eigenschaft verletzt. Das Gegenbeispiel kann ein Designproblem im ursprünglichen System oder einen Fehler des Analysten bei der Modellierung des Systems oder der Spezifikation der Eigenschaft aufzeigen, was bedeutet, dass die Methode selbstkorrigierend ist.

are indicated with letters showing whether the implementation of the method is mandatory (M), highly recommended (HR) or recommended (R). Some methods may also be not recommended (NR). The combination of adequate and suitable techniques typically consists of mandatory and / or highly recommended methods.

One set of techniques involves formal methods. Formal methods are a high-level concept that refers to "*mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems*" [1]. The formal methods are not mandatory techniques, but are still highly recommended when the highest safety integrity is required (SIL3 and SIL4). Lower safety integrity levels (SIL1 and SIL2) define these techniques as recommended. The EN 50128 standard refers to several lifecycle phases in software development for the use of formal methods, but does not specify in detail which formal method technique should be used. Instead, the Appendix lists several techniques under the formal method umbrella.

In this article, we focus in more detail on how Mipro has used and utilised the formal verification technique. Mipro has selected VTT's model checking service to support its product development process. The motivation behind this decision to use model checking was an attempt to find a new diverse verification technique to reveal any hidden design issues in reusable software modules and gain fast and easy solution iteration to increase product reliability.

1.2 Formal verification results in logical proofs

Formal languages and methods provide a systematic approach to requirement specification and design. Formal methods pave the way for clarity and unambiguity through the use of mathematically rigorous techniques, but also enable formal verification, i.e. logical proof of correctness.

Model checking [2] is a formal verification method, where a computer tool called a model checker is used to verify that a formal property holds for a given system model (fig. 1). The result is either a mathematical proof showing that the property always holds or a counter-example, i.e. a model execution path that violates the property. The counter-example can reveal a design issue in the original system or an error made by the analyst when modelling the system or specifying the property meaning that the method is self-correcting.

The properties are specified using temporal logic languages such as linear temporal logic (LTL), computational tree logic (CTL), property specification language (PSL) or their different variants. Notably, the properties can also express the ab-

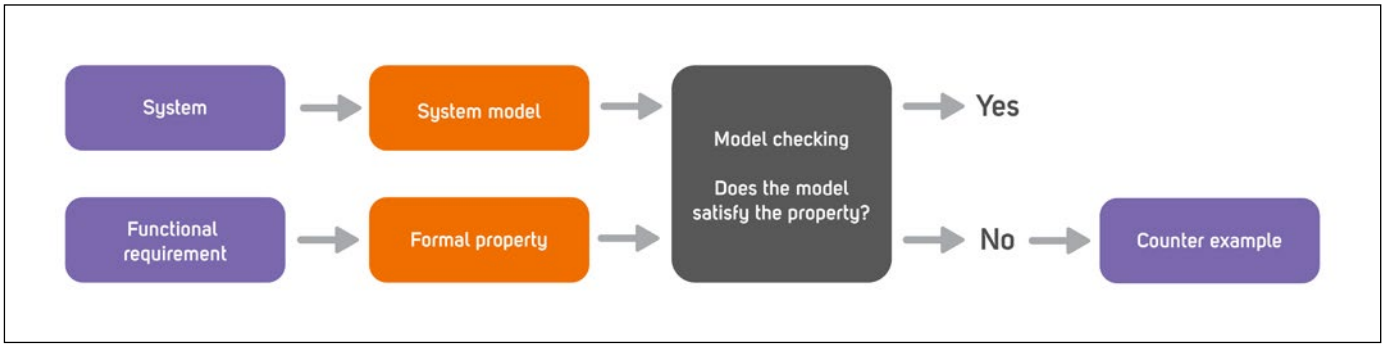

Bild 1: Beim Model Checking basiert die Analyse auf der formalen Spezifikation der funktionalen Anforderungen.
Fig. 1: In model checking, the analysis is based on the formal specification of functional requirements.

Die Eigenschaften werden mit Sprachen der temporalen Logik wie der Linearen temporalen Logik (LTL), der Computation Tree Logic (CTL), der Property Specification Language (PSL) oder ihren verschiedenen Varianten spezifiziert. Insbesondere können die Eigenschaften auch die Abwesenheit unerwünschter Funktionen ausdrücken (z. B. dass widersprüchliche oder unerwünschte Signale / Zustände niemals auftreten dürfen). In der Praxis, insbesondere wenn die Modellprüfung auf Entwürfe angewandt wird, welche bereits getestet wurden, werden Entwurfsprobleme häufig durch die Prüfung auf solche „negativen" Eigenschaften aufgedeckt.
Ein praktischer Vorbehalt ist, dass die Beschreibungssprachen für Eigenschaften nicht mit anderen Sprachen vergleichbar sind, die üblicherweise in der IuK-Technik verwendet werden. Bei den Versuchen, benutzerfreundliche (oft grafische) Sprachen zu entwickeln, war es schwierig, einen vernünftigen Kompromiss zwischen Ausdruckskraft und Benutzerfreundlichkeit zu finden. Sprachen wie LTL sind schwer zu beherrschen, aber die selbstkorrigierende Natur der Methode hilft dem Analytiker, die richtige Formulierung für jede Eigenschaft zu finden.

1.3 Iterativer und kooperativer Verifikationsprozess

VTT wendet das Model Checking seit 2008 in praktischen Projekten an, vor allem in der finnischen Atomindustrie, wo Dutzende von Konstruktionsmängeln bei verschiedenen Neubau- und IuK-Erneuerungsprojekten aufgedeckt wurden [3]. Zu diesem Zweck haben VTT und das Energieunternehmen Fortum ein grafisches Tool namens MODCHK (Bild 2) für die Verifikation von IuK-Logiken entwickelt. Als Frontend für die Model Checker NuSMV und nuXmv (entwickelt von der Fondazione Bruno Kessler in Italien) unterstützt MODCHK die benutzerfreundliche Analyse von Funktionsbausteinsprachen. MODCHK visualisiert z. B. die Gegenbeispiele, indem es das Blockdiagramm animiert. Die elementaren Funktionsblöcke werden manuell modelliert, sodass neben der Standardlogik der IEC

sence of any unwanted functionalities (e.g. stating that contradictory or spurious signals / states will never occur). In practice, especially if model checking is applied to designs that have already been subjected to testing, design issues are often revealed though the checks for such "negative" properties. One practical caveat is that the property specification languages do not resemble the other languages commonly used in I&C engineering. The attempts to develop user-friendly (often graphic) languages have struggled to find a reasonable trade-off between expressiveness and ease of use. Languages like LTL are hard to master, but the self-correcting nature of the method assists the analyst in arriving at the correct formulation of each property.

1.3 The iterative and cooperative verification process

VTT has applied model checking to practical projects since 2008, mostly in the Finnish nuclear industry, where tens of design issues have been detected in different new-build and I&C renewal projects [3]. To that end, VTT and the Fortum energy company have developed a graphic tool called MODCHK (fig. 2) for verifying I&C logics. Serving as the frontend for the NuSMV and nuXmv model checkers (developed by Fondazione Bruno Kessler in Italy), MODCHK supports a user-friendly analysis of function block diagrams. For example, MODCHK visualises the counter-examples by animating the block diagram. The elementary function blocks are modelled manually, allowing the analysis of non-standard, vendor specific block diagrams (common in nuclear applications) in addition to standard IEC 61131-3 logic. Hierarchical logics can be constructed by modelling composite function block types. VTT creates models using MODCHK in its projects for Mipro with SILworX block diagrams as the reference. VTT then writes the formal properties based on the requirement specifi-

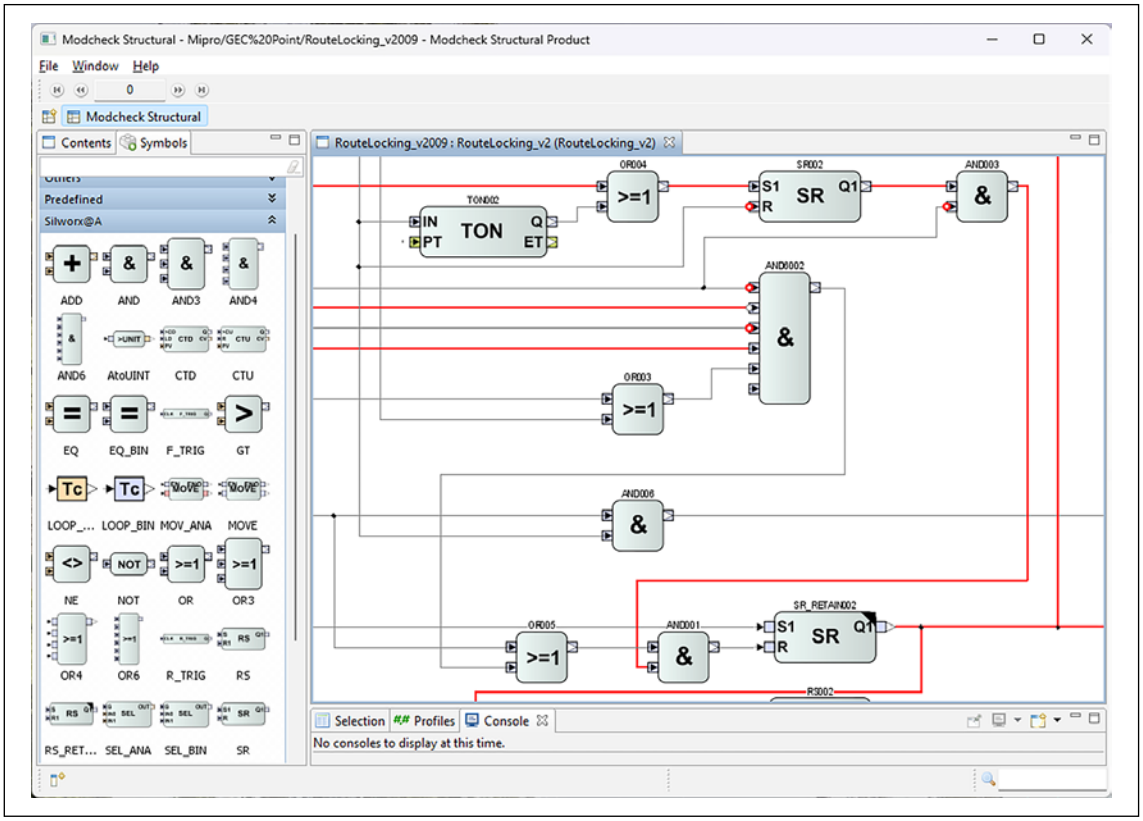

Bild 2: MODCHK ist ein grafisches Werkzeug für die Überprüfung von Funktionsbausteinsprachen.
Fig. 2: MODCHK is a graphic tool for the verification of function block diagrams.

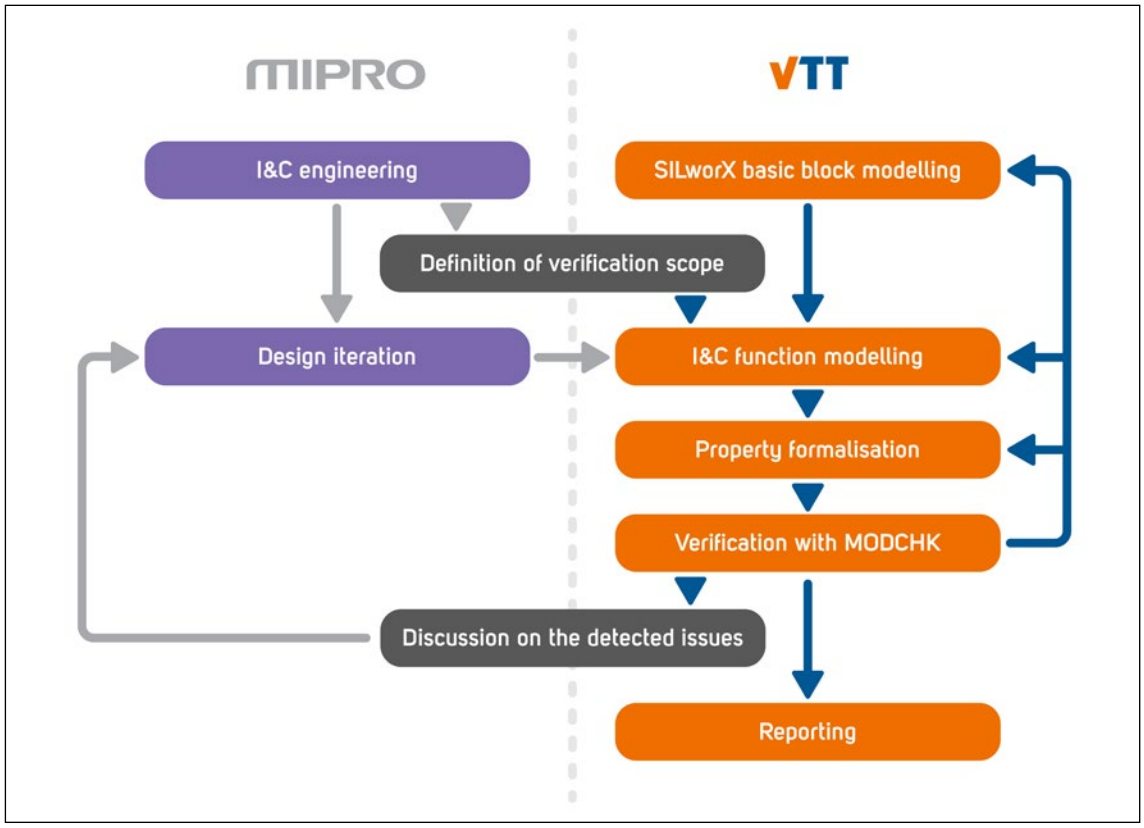

Bild 3: Iterativer Arbeitsprozess zwischen Mipro und VTT
Fig. 3: The iterative work process between Mipro and VTT

61131-3 auch nicht standardisierte, herstellerspezifische Blockdiagramme (wie sie in der Kerntechnik üblich sind) analysiert werden können. Hierarchische Logiken können durch Modellierung zusammengesetzter Funktionsblocktypen konstruiert werden.
In den Projekten für Mipro erstellt VTT Modelle mit MODCHK, wobei SILworX-Blockdiagramme als Referenz dienen. VTT schreibt dann die formalen Eigenschaften auf Grundlage von Anforderungsspezifikationsdokumenten, die von Mipro in natürlicher Sprache verfasst werden. Da das Gegenbeispiel einen Fehler aufzeigen kann, den der Analytiker bei (1) der Spezifizierung der Logik eines grundlegenden Funktionsblocks, (2) dem Zeichnen des Blockdiagramms oder (3) der Spezifizierung der formalen Eigenschaften gemacht hat, wird der Prozess so lange iteriert, bis alle Eigenschaften zutreffen oder ein Gegenbeispiel ein potenziell relevantes Konstruktionsproblem aufzeigt (Bild 3).
VTT meldet dann die entdeckten Probleme an Mipro, sodass die Mipro-Experten ihre sicherheitstechnische Bedeutung analysieren können. Wenn Mipro-Designer Änderungen an der Logik (oder den Spezifikationen) vorschlagen, können die entsprechenden Änderungen schnell in MODCHK repliziert werden, und die erneute Verifikation verläuft dann sehr schnell. Im Idealfall wäre die Modellprüfung natürlich direkt in die Tools der Designer integriert.

1.4 Anwendung der Modellprüfung bei Mipro

Die von VTT geprüften Funktionen sind Teil der Stellwerkssoftware TCS-O von Mipro, einer generischen Anwendung, die für die finnischen Eisenbahn-Sicherungsanlagen entwickelt wurde. Darin enthalten sind die Kernfunktionen, die für die Zugstrecke und die Gleisgeometrie zuständig sind, sowie der Object Controller (OC), der für die Steuerung der Weichen verantwortlich ist. (Siehe Bild 4 für einen Überblick über die Softwarearchitektur). Model Checking wurde auf die komplexesten Softwarekomponenten erfolgreich angewandt.

cation documents authored by Mipro using natural language. Given that the counter-example can reveal an error that the analyst has made in (1) specifying the logic of a basic function block, (2) drawing the block diagram or (3) specifying the formal properties, the process is iterated until all the properties hold or a counter-example demonstrates a potentially relevant design issue (fig. 3).
VTT then reports the detected issues to Mipro, so that the Mipro experts can analyse their safety significance. If the Mipro designers suggest changes to the logic (or the specifications), the corresponding changes can be quickly replicated in MODCHK and the re-verification is then very fast. Ideally, model checking would, of course, be directly integrated into the designers' tools.

1.4 The application of model checking at Mipro

The functions verified by VTT are part of Mipro's TCS-O interlocking software, a generic application developed for Finnish railway signalling projects. It includes the core functions responsible for the train route and track geometry and the Object Controller (OC) responsible for controlling the points. (See fig. 4 for an overall view of the software architecture.) Model checking has been applied to the most complex software components.
Model checking has proven very useful, as it is hard to achieve full coverage with more conventional testing and verification activities. To date, VTT has found several possible weaknesses and inconsistencies in the application logics. Recurring features in the detected issues include aspects not necessarily accounted for in testing, e.g.:

- a fault or a reset occurs while the points are being set,
- the programmable logic controller (PLC) goes offline and then restarts,

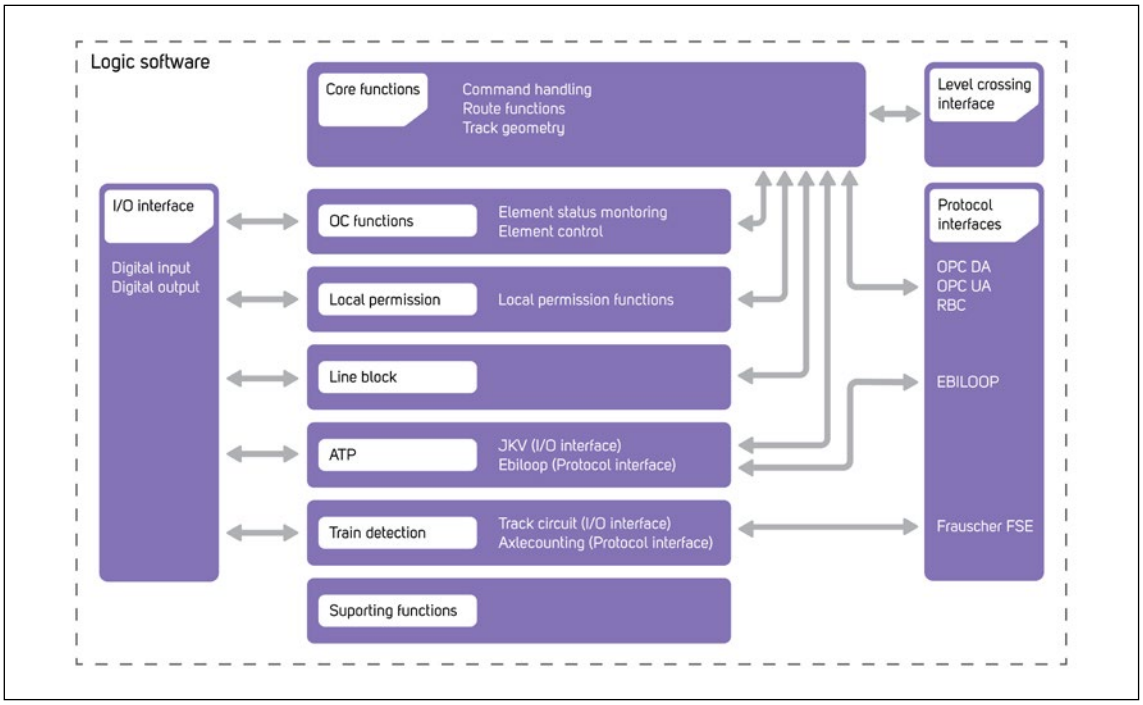

Bild 4: Gesamtansicht der Stellwerks-softwarearchitektur TCS-O von Mipro
Fig. 4: An overall view of Mipro's TCS-O interlocking software architecture

Die Modellprüfung hat sich als sehr nützlich erwiesen, da es schwierig ist, mit herkömmlichen Test- und Verifikationsmaßnahmen eine vollständige Abdeckung zu erreichen. Bislang hat VTT mehrere mögliche Schwachstellen und Unstimmigkeiten in den Anwendungslogiken entdeckt. Zu den wiederkehrenden Merkmalen der festgestellten Probleme gehören Aspekte, die bei den Tests nicht unbedingt berücksichtigt wurden, z. B.:

- tritt eine Störung oder ein Reset auf, während die Weiche gestellt wird
- geht die speicherprogrammierbare Steuerung (SPS) offline und startet dann neu
- erfolgen unterschiedliche Signalwechsel im gleichen SPS-Zyklus
- wechseln die Richtungsbefehle sehr schnell hin und her
- kommt die erwartete Rückmeldung von einem verbundenen System nicht innerhalb der erwarteten Zeit
- wird der Punkt manuell auf eine andere Position verschoben.

Die Funde waren zwar nicht sicherheitskritisch, hätten aber zu einem unerwünschten momentanen Systemverhalten führen können. Auf der Grundlage der Funde konnte Mipro die Qualität und Zuverlässigkeit der Software verbessern, komplexe logische Strukturen vereinfachen und dadurch den Speicherverbrauch und die Zykluszeit der Steuerungsanwendung reduzieren.
Während des Modellprüfungsprozesses hat sich auch die Qualität der Softwareanforderungen verbessert. Die Anforderungen sind nun präziser und beschreiben deutlicher die Art von Situationen, in welche die Software niemals geraten darf.
Die anfängliche Konstruktion des Modells und Formalisierung der Eigenschaften erforderte einen hohen Arbeitsaufwand, aber später in der Entwurfsiteration war es schnell und einfach, das Modell bei Änderungen im Software-Design auf dem neuesten Stand zu halten. Wenn Mipro Änderungen vorschlägt, überprüft VTT die Modelle erneut, um sicherzustellen, dass die Änderungen keine unerwarteten Fehler verursachen. Insgesamt sind automatisierte Regressionstests und Verifikationswerkzeuge wie die Modellprüfung eine hervorragende Möglichkeit, um in jeder Lebenszyklusphase des Systems sichere und zuverlässige Software zu erreichen und zu erhalten.

- different signal changes happen on the exact same PLC cycle,
- the direction commands change back-and-forth very rapidly,
- the expected feedback from a related system does not arrive within the expected time, or
- the points are moved to another position manually.

The findings have not been critical to safety, but could have resulted in unwanted momentary system behaviour. Based on these findings, Mipro has been able to improve the quality and reliability of the software, simplify the complex logic structures and thereby reduce the control application's memory consumption and cycle time.
The quality of the software requirements has also improved during the model checking process. The requirements are now more precise and describe more explicitly the kinds of situations the software is required to never end up in.
The initial construction of the model and the formalisation of the properties required a lot of work, but it subsequently became fast and easy to keep the model up to date with changes in software design later in the design iteration. When Mipro proposed changes, VTT re-verified the models to ensure that the changes caused no unexpected errors. Overall, automated regression testing and verification tools like model checking are excellent ways to achieve and maintain safe and reliable software during every lifecycle phase of the system.

1.5 A practical example of a detected issue

Fig. 4 depicts a simplified example of a design issue revealed in Mipro's systems. The example comes from the route setting logic for a set of points and the intended function is to lock the points at either end position (A or B) once the requested position has been reached. If requests to both end position A and B are active at the same time, the points will be locked in their current position. When the conflict is resolved, the points will then be allowed to automatically move to the requested position. The latter requirement can be formalised (for position A) in LTL by stating:
G ((At_A **&** Request_to_A **&** Request_to_B)
& X (At_A **& !** Request_to_A **&** Request_to_B)
-> **X !** Lock)

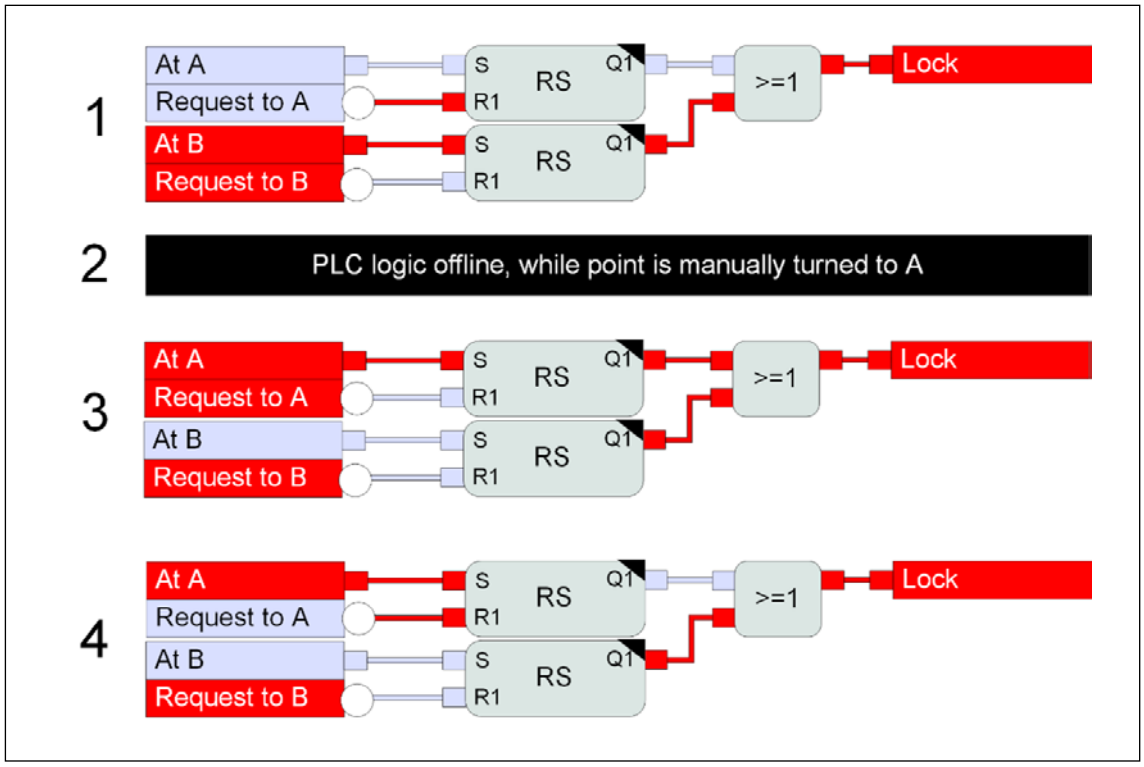

Bild 5: Ein praktisches Designproblem, das durch Modellprüfung aufgedeckt wurde
Fig. 5: A practical design issue revealed by model checking

1.5 Ein praktisches Beispiel für ein entdecktes Problem

Bild 5 zeigt ein vereinfachtes Beispiel für ein Designproblem, das bei den Systemen von Mipro auftritt. Das Beispiel stammt aus der Stellbetriebslogik für eine Weiche, und die beabsichtigte Funktion besteht darin, die Weiche in einer der beiden Endpositionen (A oder B) festzuhalten, wenn die gewünschte Position erreicht ist. Werden die Endpositionen A und B gleichzeitig angefordert, wird die Weiche in ihrer aktuellen Position festgehalten. Wenn der Konflikt gelöst ist, kann die Weiche dann automatisch in die gewünschte Position umgestellt werden. Die letztgenannte Anforderung kann durch folgende Aussage formuliert werden (für die Position A) in LTL:

G ((At_A **&** Request_to_A **&** Request_to_B)
& X (At_A **& !** Request_to_A **&** Request_to_B)
-> **X !** Lock **)**

Mit anderen Worten, es gilt immer **(G)**, wenn die Weiche in der Endposition A ist, beide Anforderungen aktiv sind und im nächsten Zustand **(X)** die Anforderung an A zurückgesetzt wird, dann muss die Sperre zurückgesetzt werden.
Der Model Checker liefert ein Gegenbeispiel, das in Bild 5 dargestellt ist. Wir beginnen in Zustand 1, in dem die Weiche in der gewünschten Endposition B festgestellt ist. Als nächstes, in Zustand 2, ist die SPS aus irgendeinem Grund offline. Während die SPS offline ist, wird die Weiche manuell in die andere Endposition A gestellt. In Zustand 3 wird die SPS dann neu gestartet. Da es sich bei dem Speicherbaustein SILworX RS um einen speziellen „Retain"-Typ handelt (gekennzeichnet durch das schwarze Dreieck in der Ecke), wird der Ausgangszustand bzw. der „B"-Speicher vom letzten Betrieb der SPS abgerufen. Daher bleibt die Weiche in Zustand 4 in der Position A festgestellt, obwohl die Anforderung an B der einzige aktive Befehl ist.
Wir zeigen hier nur drei Funktionsblöcke, aber das ursprüngliche Modell war sehr umfangreich und enthielt 143 Inputs, 244 Outputs und insgesamt 788 Funktionsblöcke in einem Diagramm, das sechs

In other words, it will always (G) hold that if the points are in end position A, both requests are active, and the request to A is reset in the next state (X), then the lock will be reset.
The model checker returned the counter-example depicted in fig 5. We begin in state 1, where the points are locked in the requested end position B. The PLC then goes offline in state 2 for some reason. While the PLC is offline, the points are manually moved to the other end position A. The PLC is then restarted in state 3. Since the SILworX RS memory function block is of a specific "retain" type (indicated by the black corner triangle), the initial state of the "B" memory is recalled from when the PLC was last running. Therefore, in state 4, the points remain locked in position A, even though the request to move to B is the only active command.
We have only shown three function blocks here, but the original model was large consisting of 143 inputs, 244 outputs and a total of 788 function blocks in a diagram that made use of six composite block types. Due to the size of the model, VTT used a technique called bounded model checking (BMC), thereby allowing NuSMV to disprove the property and generate the counter-example in just six seconds (recorded on an Intel i5-1235U running at 1.30 GHz).
The counter-example is representative of the kinds of issues often found with model checking, as it requires something very improbable and counterintuitive to take place. When specifying test cases, such a sequence of events may not come to mind or even necessarily be that easy to replicate in a test suite. With model checking, it is sufficient to formalise the functional requirement and let the algorithms find the counter-example. ■

zusammengesetzte Blocktypen verwendet. Aufgrund der Größe des Modells verwendete VTT eine Technik namens Bounded Model Checking (BMC), die es NuSMV ermöglichte, die Eigenschaft zu widerlegen und das Gegenbeispiel in nur sechs Sekunden zu erzeugen (aufgezeichnet auf einem Intel i5-1235U mit 1,30 GHz).

Das Gegenbeispiel ist repräsentativ für die Art von Problemen, die häufig bei der Modellprüfung auftreten, da etwas sehr Unwahrscheinliches und Kontraintuitives passieren muss. Bei der Spezifikation von Testfällen wird eine solche Abfolge von Ereignissen vielleicht nicht vorgesehen oder ist sogar in Tests nicht so leicht zu reproduzieren. Bei der Modellprüfung reicht es aus, die funktionale Anforderung zu formulieren und die Algorithmen das Gegenbeispiel finden zu lassen. ■

LITERATUR | LITERATURE

[1] Europäische Norm EN 50128:2011, Bahnanwendungen – Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme – Software für Eisenbahnsteuerungs- und Überwachungssysteme

[2] Clarke, E.; Grumber, O.; Peled, D.: Model Checking. Cambridge, Massachusetts, US: MIT press, 1999

[3] Pakonen, A.; Buzhinsky, I.; Björkman, K.: „Model checking reveals design issues leading to spurious actuation of nuclear instrumentation and control systems", Reliability Engineering & System Safety, vol. 205, 107237, 2021. https://doi.org/10.1016/j.ress.2020.107237

AUTOREN | AUTHORS

Dipl.-Ing. Antti Pakonen
Senior Scientist, Project Manager
VTT
Anschrift / *Address*: Tekniikantie 21, FI-02150 Espoo
E-Mail: antti.pakonen@vtt.fi

Dipl.-Ing. Juha Turunen
Safety Manager
Mipro Oy
Anschrift / *Address*: Bertel Jungin aukio 1, FI-02600 Espoo
E-Mail: juha.turunen@mipro.fi