Modellsicht auf die Logik eines generischen Stellwerkssystems

Serhat Cakaloglu, Hans-Jürgen Nollau, Murat Sahingöz

Siemens AG, Industry Sector, Mobility Division
Ackerstraße 22, 38126 Braunschweig
{Serhat.Cakaloglu, Hans-Juergen.Nollau, Murat.Sahingöz}@Siemens.com

Zusammenfassung: In diesem Beitrag werden Methoden und Werkzeuge dargestellt, mit denen eine Modellsicht auf eine vorhandene Stellwerkslogik erstellt werden kann. Auf die besonderen Anforderungen eines generischen Systems wird eingegangen und die Einbindung in den Verifikations- und Validierungsprozess dargestellt. Ein Ausblick auf eine Anwendung für generalisierende Testaussagen wird gegeben.

1 Einführung

Bei dem elektronischen Stellwerk vom Typ Simis D (Simis - Sicheres Mikrocomputersystem von Siemens) handelt es sich um ein generisches System für die Anwendung bei der DB AG. Unser Team erstellt hierfür die Stellwerkslogik, die Softwarekomponente, die die sicherheitskritischen Abhängigkeiten beispielsweise zwischen Weichen und Signalen herstellt. Die Entwicklung der Stellwerkslogik beruht auf dem Konzept der generischen Komponenten (Fahrwegelemente). Jedes Fahrwegelement wird dabei isoliert betrachtet und seine Reaktionen auf eintreffende Ereignisse mittels GRACE-Modellen (GRACE - Graphical Requirements Analysis and design method in a CENELEC based Engineering process) spezifiziert. Die Elemente kommunizieren über definierte Kanäle – sogenannte Spuren.

Das Überführen ins Zielsystem erfolgt automatisiert mit der zugelassenen Werkzeugkette GRACE.

Von den Elementen werden im Rahmen der Projektierung Instanzen gebildet und mit Spuren nach definierten Regeln verknüpft. Die Instanzen der Elemente können auf verschiedenen Rechnern eines Rechnernetzwerkes liegen.

Das Funktionsverhalten eines konkreten Stellwerkes wird nicht nur durch die in den Elementen implementierte Logik bestimmt, sondern auch durch die Kommunikation der Elementinstanzen untereinander.

2 Warum Modellsichten auf eine vorhandene Logik anwenden?

Zu Entwicklungsbeginn im Jahr 2000 stand uns kein stabiles Werkzeug für eine modellbasierte Entwicklung eines komplexen generischen Systems zur Verfügung, das in unsere zugelassene Werkzeugkette integrierbar gewesen wäre.

Der Zwang zur Abstraktion in der Entwurfsphase war aufgrund der Komplexität der Steuerungsaufgabe gegeben und wurde beispielsweise durch Spezifikation von Funktionen mit Statemate, Animation und Tests der Modelle beantwortet. Es erfolgte eine manuelle Transformation in die zugelassene Entwicklungsumgebung GRACE.

Für eine gewachsene Stellwerkslogik ist das Einführen einer modellbasierten Entwicklung mit hohen Risiken verbunden. Für unser Team rückten folgende Fragestellungen in den Vordergrund:

1. Wie wird sich das System nach einer Änderung verhalten?

2. Wie interagieren die Elementinstanzen einer bestimmten topologischen Struktur für eine bestimmte Funktionalität?

3. Wie wird sich das System verhalten, wenn Elementinstanzen aufgrund topologischer Anforderungen anders angeordnet werden?

Um diese und verwandte Fragen effektiv beantworten zu können, erschienen insbesondere Sequenzdiagramme für einfach zu konfigurierende Elementanordnungen als Lösung geeignet. Diese Aufgabe wurde bei uns tätigen Studenten übertragen. Sie nutzten Zwischenresultate eines von der Siemens AG geförderten Promotionsprojektes GRACE-SIM (SIM - Simulation). GRACE-SIM basiert auf einem Codegenerator und einer domänenspezifischen Testsprache. Diese Testsprache dient der Konfiguration des aus den GRACE-Modellen erzeugten Codes und ermöglicht eine entwicklungsbegleitende Validierung der Systemfunktionen.

3 Entwicklungsbegleitende Validierung mittels GRACE-SIM

3.1 Datenbasis

Die Stellwerkslogik von Simis D wird für externe Werkzeuge in XML-Dateien abgebildet. Diese XML-Dateien sind als offene Schnittstellen zu GRACE-Quellen zu verstehen. Sie verfügen über alle Deklarationen und Telegramm-Empfangsmethoden und sind plattformunabhängig. Um dynamische Aspekte in einem Modell repräsentieren zu können (z. B. in einem Sequenzdiagramm) war es erforderlich, einen lauffähigen Code zu erzeugen und die Eigenschaften des Zielsystems nachzubilden.

3.2 Codegenerator

Der Codegenerator von GRACE-SIM nutzt die XML-Repräsentation der Stellwerkslogik. Er erzeugt einen Java-Code mit offen gelegter API (Application Programming Interface). Mit dieser API ist es möglich, das System beliebig zu konfigurieren und zu stimulieren. Die Konfiguration entspricht einer Vernetzung der Fahrwegelemente. Die Stimuli sind die Telegramme, die die Aufrufe von Empfangsmethoden in Fahrwegelementen auslösen.

Die Ablaufumgebung und die funktionalen Eigenschaften des Zielsystems werden mitgeneriert. GRACE-SIM behandelt Telegramme genau so, wie sie das Zielsystem umsetzen würde.

3.3 Definition des Testobjektes und Testsprache

Bestandteil von GRACE-SIM ist ein Editor zum Herstellen von Elementinstanzen und deren Verbindung mit anderen Elementinstanzen. Mit diesem Editor können beliebige komplexe topologische Strukturen abgebildet werden.

Die Artefakte der entwickelten Testsprache definieren die Konfiguration des Prüflings und können mehrere Testfälle enthalten. Die Konfiguration erlaubt beliebige Abbildungen komplexer topologischer Strukturen darzustellen, indem Instanzen von Fahrwegelementen und deren Kommunikationsverbindungen zu einander deklariert werden. Der interne Zustand von Instanzen kann initialisiert werden.

Die Testfälle definieren, welche Telegramme in welcher Reihenfolge an die Elementinstanzen gesendet werden. Das Senden von Telegrammen erfolgt erst nach dem Einnehmen eines stabilen Zustandes des zuvor gesendeten Testobjekttelegramms. Ein stabiler Zustand bedeutet, dass keine Kommunikation zwischen den Elementinstanzen stattfindet.

Es können auch Soll-Daten definiert werden, die nach dem Senden eines Telegramms auftreten müssen. Die Kommunikationen zwischen Elementinstanzen können ebenfalls als Soll-Daten genutzt werden. Um Timer und zyklische Ereignisse aus der Stellwerkslogik berücksichtigen zu können, wurde die Option Wartezeiten für Testfälle vorgesehen.

Die Testfälle werden durch einen für diese Testsprache entwickelten Parser auf dem generierten Code ausgeführt. Am Ende eines Testlaufs liegen die Ergebnisse als Sequenzdiagramme
(mit integrierten Berichten über durchlaufene/abgedeckte Empfangsmethoden oder Funktionen) vor.

Nicht relevante Datenflüsse können durch Anweisungen der Testfalldefinition ausgeblendet werden. Eine hierarchische Strukturierung von Testobjekten ist möglich. Komplexe topologische Strukturen können abstrahiert und wiederverwendet werden.

Die aufbereiteten Modellsichten werden verwendet, um elementübergreifende Systemverhalten zur Entwicklungszeit zu beobachten und auf deren Basis eine frühzeitige Validierung zu ermöglichen.

3.4 Anwendungsergebnisse

Erfolgreiche Anwendungsergebnisse liegen bei

- der Verifikation und Validierung komplexer Teilfunktionen
- der Fehlersuche
- der Schulung neuer Mitarbeiter

vor.

3.5 Weitere Zielsetzungen

Zurzeit werden die Testläufe an einer konkreten topologischen Struktur (Testbahnhof) durchgeführt. Aufgrund der beobachteten Soll-Daten kann für diese Struktur ein Gut-Schlecht-Entscheid ausgeführt werden. Auf Basis der repräsentativen Struktur dieser Testbahnhöfe wird auf die Soll-Funktionserfüllung in den anderen zur Anwendung freigegebenen Strukturen geschlossen.

Für das Definieren zulässiger topologischer Strukturen wurde an der TU Braunschweig das Beschreibungsmittel LpSpec (LpSpec - Lageplan-Spezifikation) entwickelt. LpSpec beherrscht das Darstellen von Häufigkeiten von Elementinstanzen und das Verschieben von Elementinstanzen in einer toplogischen Struktur.

Es liegt nahe, eine zulässige Ableitung der beschriebenen topologischen Struktur manuell zu untersuchen und weitere zulässige Ableitungen zu automatisieren. Neben der Schnittstelle zwischen LpSpec und GRACE-SIM bleiben eine optimale Suchstrategie und ein Ende-Kriterium für den Testprozess weiterhin offen.