Zustandsautomaten-Origami

Effiziente Embedded-Software durch interaktive Statecharts

Dr. Klaus Birken und Axel Terfloth, itemis AG

Zustandsautomaten haben sich in vielen Projekten zur Implementierung von Embedded-Software bewährt. Damit lässt sich das Verhalten von Komponenten grafisch beschreiben und effizienter Code in C oder C++ generieren. Die Interaktion der Statecharts mit ihrer Umgebung wird über Konzepte wie Events und formale Schnittstellen beschrieben. Das Open-Source-Projekt Franca erlaubt die Modellierung solcher Schnittstellen, speziell ihrer Semantik (d.h. erlaubte Abfolgen von Events). In den meisten Projekten wird diese Information nur informell dokumentiert – mit Franca sind die erlaubten Abläufe maschinenlesbar und damit automatisiert prüfbar. Tools zur Erstellung von Zustandsautomaten können sich diese Semantik-Information zunutze machen und den Entwickler interaktiv unterstützen, z.B. durch Hinweise auf zu erwartende oder zu sendende Events sowie auf nicht erreichbare Zustände. Dies stellt die Einhaltung aller Schnittstellen-Contracts sicher und sichert letztlich die Qualität des Codes.

Schnittstellen und ihre Dynamik beschreiben mit Franca

Bei der Embedded-Entwicklung sollten Schnittstellen zwischen Komponenten oder Subsystemen nicht direkt als C-Header-Dateien entworfen und repräsentiert werden, sondern auf einer höheren Abstraktionsebene, die von der Programmiersprache unabhängig ist. Dies wird durch eine Beschreibungssprache für Schnittstellen oder *IDL* (kurz für: *Interface Definition Language*) möglich [1]. Diese reduziert die Syntax auf genau das Vokabular, das für die inhaltliche Definition von Schnittstellen gebraucht wird. Die Abbildung auf eine Programmiersprache geschieht typischerweise durch Codegenerierung.

Eine IDL erlauben es, Schnittstellen unabhängig von der Zielplattform und -sprache formal zu beschreiben. Damit können dieselben Schnittstellen zur Integration über Teilsysteme, Programmiersprachen, Controller und Steuergeräte hinweg eingesetzt werden. Schnittstellen werden so ein übergreifendes Werkzeug für Architekten und Entwickler. Das Open-Source-Projekt *Franca* [2] bietet eine solche IDL und verschiedene Tools, um Schnittstellen zu erstellen, zu bearbeiten oder daraus Programmcode zu generieren. Franca wurde zunächst im Rahmen des GENIVI-Konsortiums [3] entwickelt und ist derzeit auf dem Weg, ein offizielles Projekt unter dem Dach der Eclipse Foundation zu werden. Auf früheren ESE-Konferenzen wurden die verschiedenen Aspekte von Franca vorgestellt (z.B. [1], [4], [6]).

Der typische Sprachumfang einer IDL deckt die *statischen* Anteile von Schnittstellen ab, d.h. Datentypen, Attribute und Funktionsaufrufe. Dies ist zur Dokumentation der Schnittstellen und auch zur Codegenerierung zunächst ausreichend, da in Programmiersprachen ebenfalls nur statische Anteile von Schnittstellen beschrieben werden (C-Header oder C++-Klassen enthalten keine Festlegung z.B. von Aufrufreihenfolgen).

Die *dynamischen* Anteile von Schnittstellen werden heute meist erst in der Implementierung berücksichtigt. Dies sind erlaubte Aufrufreihenfolgen (also Sequenzen) und deren Parameterwerte. Ebenfalls gehören z.B. nicht-funktionale Eigenschaften wie Timing-Restriktionen zu den dynamischen Anteilen. Gerade für komplexere Systeme ist es sinnvoll, die dynamischen Anteile bereits als Teil der Schnittstellendefinition festzulegen, denn der Schnittstellen-Designer hat die genaueste Vorstellung davon, wie seine Schnittstelle zu verwenden ist [4]. Über die formale Festlegung der Dynamik kann er dies den nachfolgenden Implementierern übermitteln. Dies steigert die Qualität des resultierenden Softwaresystems.

Franca unterstützt genau diese Beschreibung des dynamischen Verhaltens von Schnittstellen. Dazu bietet Franca IDL die Möglichkeit, Verträge (sog. *contracts*) von Schnittstellen in Form von Protokoll-Zustandsautomaten zu spezifizieren. Dies sind Zustandsautomaten, die auf der Verbindung zwischen den beiden kommunizierenden Softwareteilen (z.B. Komponenten) angesiedelt sind und auf Kommunikationsereignisse in beiden Richtungen reagieren. Beispielsweise geschieht ein Zustandsübergang, falls die Client-Seite eine Remote-Methode aufruft. Ein weiterer Zustandsübergang wird ausgelöst, wenn die Server-Seite die Antwort-Nachricht zu dieser Methode sendet. In jedem Zustand des Protokoll-Automaten sind nur die Ereignisse erlaubt, die durch ausgehende Transitionen beschrieben sind. Alle anderen Ereignisse wären Protokollfehler und damit Verletzungen des Vertrags.

Mit Franca werden die Zustandsautomaten als Teil der IDL textuell modelliert. In einem Vorjahresbeitrag wurde detailliert gezeigt, wie solche Verhaltensdefinitionen verwendet werden können, um Trace-Daten aus einem realen Embedded-System zu validieren [4]. Im vorliegenden Artikel wird nun beschrieben, wie solche Dynamik-Festlegungen an der Schnittstelle bereits *während der Implementierung* eines Embedded-Systems verwendet werden können, sofern diese Implementierung mittels eines Statechart-Modellierungswerkzeugs umgesetzt wird.

Embedded-Software mittels Statecharts erstellen

Zustandsautomaten sind ein verständlicher Formalismus, der sich zur Spezifikation und Implementierung des reaktiven Verhaltens von Systemen bewährt hat. Es gibt eine Reihe von Ansätzen, wie Zustandsautomaten in Programmiersprachen wie C und C++ effizient implementiert werden können. Wie bei IDLs hat sich aber auch bei Zustandsautomaten die Beschreibung auf einer höheren Abstraktionsebene in Form von Modellen bewährt. Meist werden Zustandsautomaten graphisch in Form von Statecharts spezifiziert. Diese von der Programmiersprache unabhängigen Modelle können dann ebenfalls mittels Codegenerierung in C, C++ oder Code anderer Programmiersprachen übersetzt und in die Software eingebunden werden. Da Zustandsautomaten das dynamische Verhalten von Systemen beschreiben, sind bei der Entwicklung insbesondere Hilfsmittel zur Ausführung der Zustandsautomaten in der Modellierungsumgebung z.B. mit Hilfe von Simulatoren sehr hilfreich.

Das Open-Source-Project *YAKINDU Statechart Tools (YSCT)* [5] stellt eine Werkzeugumgebung zur Modellierung, Simulation sowie Code-Generierung für Statecharts bereit.

Statecharts mit Franca-Schnittstellen – Integration von Tools und Modellen

Softwarekomponenten, die mit Hilfe von Statecharts implementiert werden, interagieren in der Regel mit anderen Softwarekomponenten. Dies erfordert wohldefinierte Schnittstellen. Damit liegt es nahe, die Schnittstellen der Softwarekomponenten mit einer IDL zu beschreiben. Eine solche Integration gibt es zwischen Franca und YSCT. Die Statecharts bieten die Möglichkeit Ports zu definieren, die zur Anbindung passender Softwarekomponenten dienen. Für jeden Port kann spezifiziert werden, ob über diesen ein in Franca definiertes Interface bereitgestellt (*provide*) oder genutzt (*require*) werden soll. Die in dem Interface definierten Ereignisse, Methoden sowie Datenelemente können im Statechart dann direkt verwendet werden. In Abhängigkeit davon, ob das Statechart als Nutzer oder Anbieter einer Schnittstelle agiert, stehen jeweils passende Elemente zur Verfügung.

Softwarekomponenten verfügen oftmals über mehr als eine Schnittstelle und müssen auch im Hinblick auf ihr dynamisches Verhalten Schnittstellen-konform sein. Die Komplexität der Schnittstellen multipliziert sich allerdings potentiell in der Implementierung der Komponenten, so dass ohne weitere Hilfsmittel die Validierungs- und Testaufwände sehr

groß werden können. Im nachfolgenden Beispiel wird nun dargestellt, wie durch die Nutzung von Protokoll-Zustandsautomaten für die Schnittstellen die Statechart-basierten Implementierungs-Zustandsautomaten automatisch validiert werden können.

Beispielprojekt: Roboterarm-Steuerung

Als Beispiel für die beschriebene Integration der Open-Source-Tools YSCT und Franca dient eine Steuerungssoftware für einen Roboter-Arm. Die Roboter-Hardware ist ein AL5D-Kit von Lynxmotion mit SSC32-Steuerplatine. Abbildung 1 zeigt den Roboterarm; dieser besteht aus fünf Achsen und einer Greifeinheit.

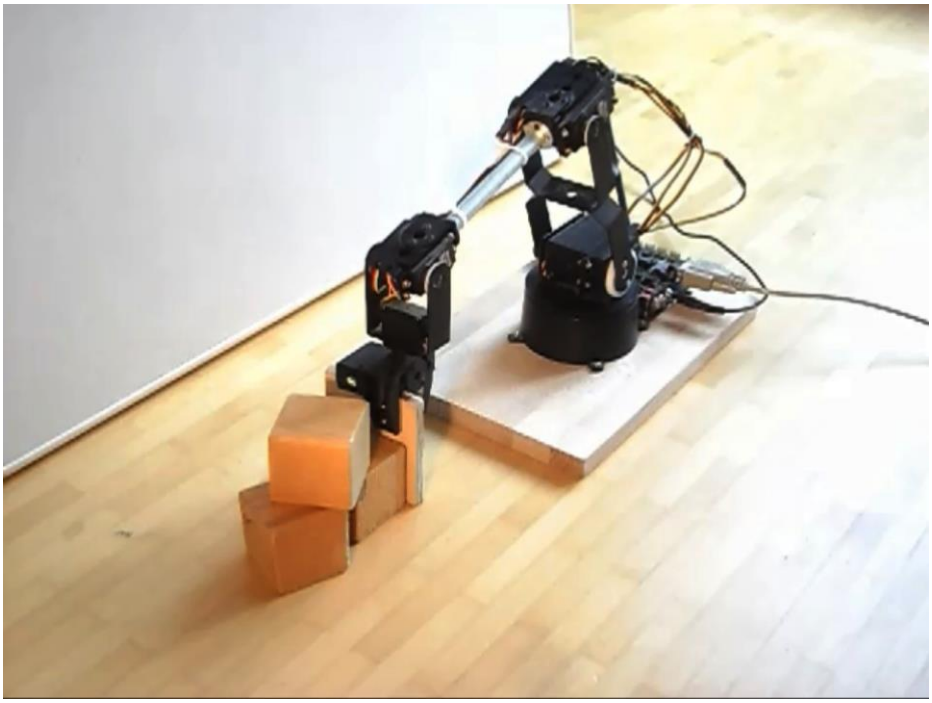

Abb. 1: Roboterarm-Steuerung als Beispiel für ein Embedded-Software-System.

Für unser Beispiel modellieren wir das Verhalten an der Applikationsschnittstelle der Robotersteuerung; diese erlaubt die Kontrolle folgender Funktionen:

- Der Greifer kann Objekte festhalten und loslassen (Operationen *grab* und *release*).
- Der Arm kann den Greifer an eine bestimmte kartesische Position bewegen (Operation *move*).

Die statischen Elemente der Schnittstelle sind also die Methoden *move*, *grab* und *release*. Jede Ausführung einer dieser Methoden besteht aus zwei Ereignissen: Zuerst ruft die Client-Seite der Schnittstelle die Methode auf; danach sendet die Server-Seite eine positive oder negative Antwort. Zusätzlich zu dieser elementaren Protokollfestlegung gibt es fachliche Einschränkungen der erlaubten Abläufe. Beispielsweise ist eine *release*-Operation ohne eine vorherige *grab*-Operation nicht sinnvoll. Insgesamt lassen sich alle erlaubten Sequenzen durch den Protokoll-Zustandsautomaten aus Abbildung 2 definieren.

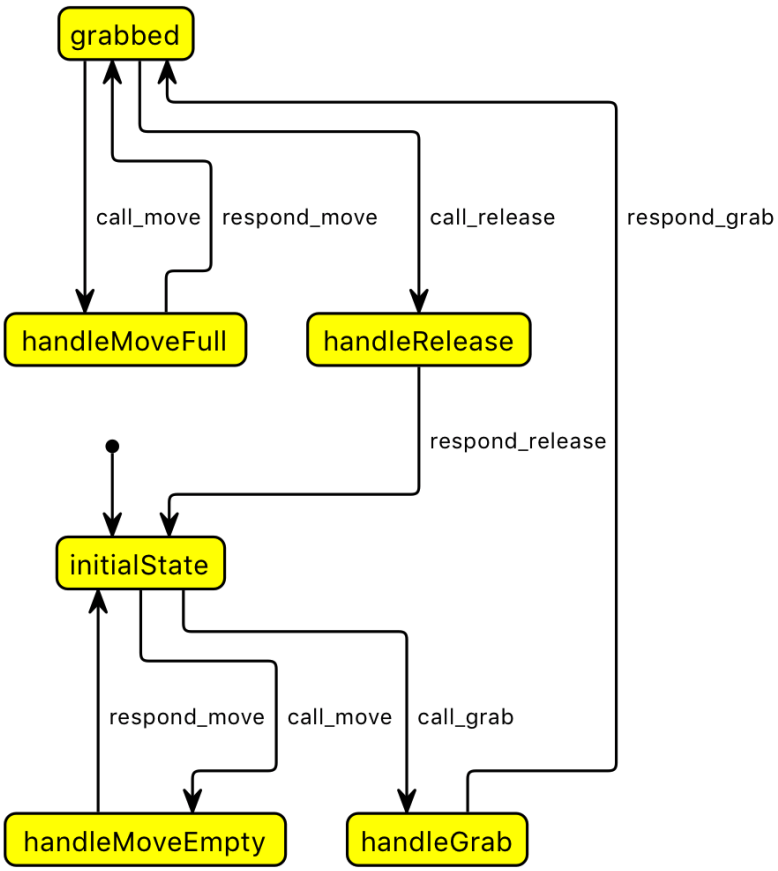

Abb. 2: Der Protokoll-Zustandsautomat für die applikative Schnittstelle der RobotArm-Steuerung.

Ein Protokoll-Zustandsautomat wie in Abbildung 2 wird als Bestandteil der Schnittstellendefinition mittels Franca IDL formuliert. Wichtig ist, dass dieser nur die Semantik der Schnittstelle beschreibt und nicht die eigentliche Implementierung der Steuerungskomponente. Diese wird durch einen weiteren, viel detaillierteren Zustandsautomaten mit dem Yakindu Statecharts-Tool erstellt.

Interaktives Feedback bei der Statechart-Modellierung

In einem früheren ESE-Beitrag wurde gezeigt, wie aufgezeichnete Trace-Daten automatisiert gegen einen oder mehrere Protokoll-Zustandsautomaten validiert werden können [4]. Dies erhöht die Qualität des resultierenden Produkts, indem Fehler in der Implementierung beim Testen gezielt aufgedeckt und behoben werden können. Besser ist es jedoch, die Fehler bereits während der Implementierung zu vermeiden. Dies wird durch die Integration von YSCT und Franca ermöglicht: Die Implementierungs-Statecharts werden hierbei während der Entwicklung ständig gegen die Protokoll-Zustandsautomaten der beteiligten Franca-Schnittstellen geprüft. Der Entwickler wird durch interaktives Feedback angeleitet.

Abbildung 3 zeigt für das oben beschriebene RobotArm-Beispiel anhand eines Screenshots, welche Informationen aus den Schnittstellendefinitionen abgeleitet und wie diese für den Nutzer dargestellt werden. Der Entwickler hat gerade begonnen, den Zustandsautomat für die RobotArm-Steuerungskomponente zu bauen. Diese Komponente implementiert die Schnittstelle RobotArm mit dem Verhalten aus Abbildung 2.

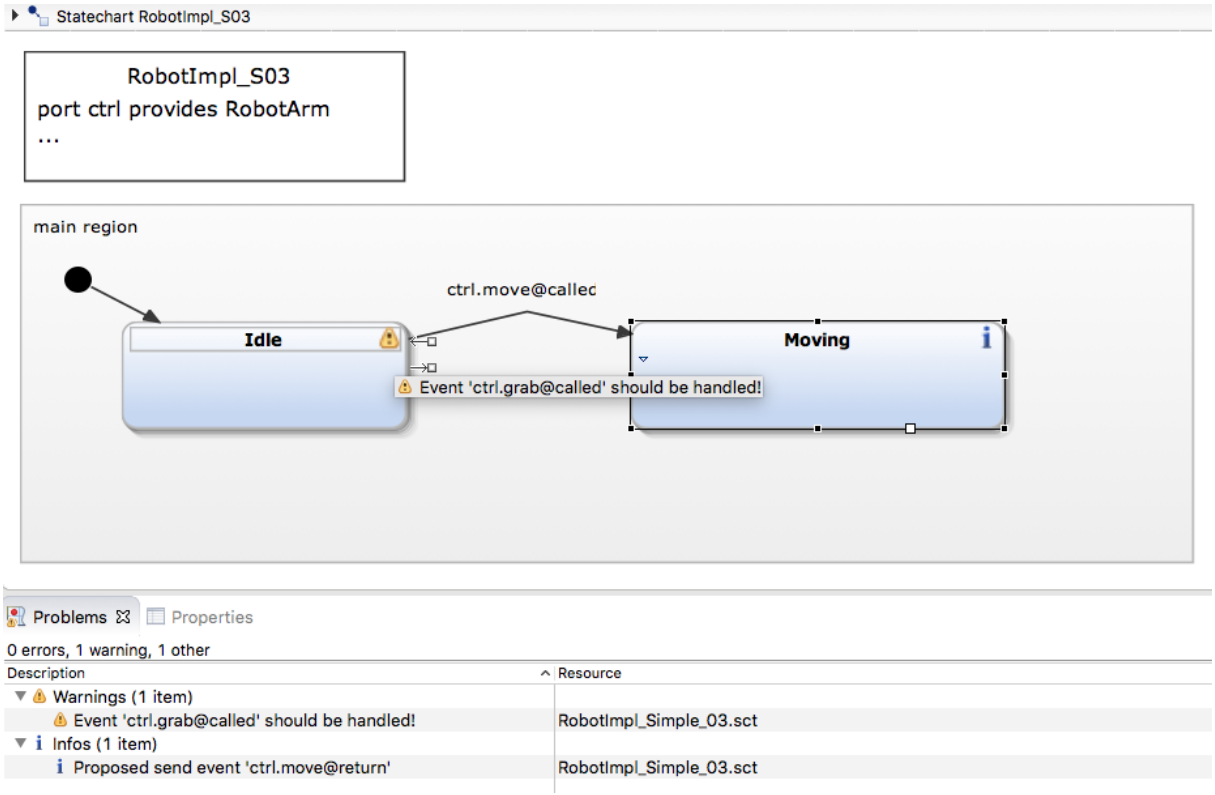

Abb. 3: Interaktives Feedback im Statechart-Editor. Dargestellt ist der halbfertige Zustandsautomat für die RobotArm-Steuerung zu Beginn der Entwicklung.

Aus der Schnittstellendefinition werden folgende Informationen abgeleitet:

- Im Zustand *Idle* wird zwar der Aufruf der Operation *move* behandelt (siehe Transition *Idle→Moving*), die Behandlung der Operation *grab* fehlt jedoch. Dies wird durch eine Warnung dargestellt. Der Entwickler wird dadurch angeleitet, eine weitere Transition (und einen weiteren Zustand) zur Behandlung des *grab*-Aufrufs zu ergänzen.
- Der Zustand *Moving* wird durch einen *move*-Aufruf erreicht. Die Antwort auf die *move*-Operation steht jedoch noch aus, dies wird durch eine Information dargestellt („Proposed send...“). Der Entwickler wird somit angeleitet, die *move*-Antwort zu implementieren.

Durch die weitere Arbeit am Zustandsautomaten werden interaktiv weitere Warnungen und Informationen generiert und durch erneute Entwickler-Aktion bearbeitet. Am Ende entsteht ein Zustandsautomat, der keine Warnungen mehr enthält. Dies zeigt, dass die Vorgaben des Protokoll-Zustandsautomaten eingehalten sind. Bei komplexen Komponenten mit mehr als einer Schnittstelle verstärkt sich der positive Effekt, weil der Entwickler durch mehr Informationen noch besser angeleitet werden kann.

Im nächsten Schritt werden zu jeder Warnung oder Information des Tool zusätzlich Vorschläge angeboten, wie der Hinweis durch einen Mausklick direkt zu beheben bzw. umzusetzen ist (sog. *Quickfixes*). Der Entwickler kann damit komplette Teile des Zustandsautomaten automatisch erzeugen lassen und gleichzeitig vorhandene Warnungen beheben.

Vorteile für die Embedded-Praxis

Werkzeuge zur Modellierung von Zustandsautomaten und Codegenerierung werden allenthalben in Embedded-Projekten benutzt. In diesem Beitrag wurde daran anknüpfend gezeigt, wie die innovative Berücksichtigung von Schnittstellen-Semantik Entwickler dazu anleiten kann, dass das Verhalten der erstellten Softwarekomponenten konform zu vorgegebenen Schnittstellen ist. Der Entwickler wird automatisch möglichst früh auf mögliche Fehler hingewiesen und dadurch entlastet. Implementierungsfehler werden frühzeitig entdeckt und behoben, somit wird die Qualität der Software erhöht. Die teure Behebung von Integrationsfehlern und Fehlern in der Serie wird soweit wie möglich vermieden. Durch die oben beschriebenen automatisierten Vorschläge des Tools wird dem Entwickler zusätzlich das Editieren des Diagramms erleichtert, was seine Arbeit insgesamt beschleunigt.

Generell ist die modellbasierte Definition von Schnittstellen gerade bei verteilten Anwendungen notwendig und sinnvoll. Durch Franca IDL können die Schnittstellen formal definiert werden und dienen dann als maschinenlesbarer „Vertrag“ für die beteiligten Parteien und Systeme. Franca ist unter [1] für jedermann verfügbar und kann inklusive Dokumentation frei heruntergeladen und direkt eingesetzt werden. Ebenso kann Yakindu Statecharts ohne Lizenzkosten für die Implementierung von Embedded Systems eingesetzt werden. Die Integration beider Werkzeuge bringt darüberhinaus einen zusätzlichen Nutzen für Entwickler und Architekten.

Referenzen

[1] K. Birken: *Schnittstellen voll im Griff – Modellbasierte Ansätze mit dem Open-Source-Werkzeug Franca.* In: Tagungsband – Embedded Software Engineering Kongress 2012, Sindelfingen, 2012.

[2] http://code.google.com/a/eclipselabs.org/p/franca/

[3] http://www.genivi.org

[4] K. Birken: *Interfaces dynamisch beschreiben mit Franca.* In: Tagungsband – Embedded Software Engineering Kongress 2013, Sindelfingen, 2013.

[5] Yakindu Statechart Tools (http://statecharts.org)

[6] T. Szabo, K. Birken: *Modellbasiert im Internet of Things.* In: Tagungsband – Embedded Software Engineering Kongress 2014, Sindelfingen, 2014.

Autoren

Dr. Klaus Birken studierte Informatik in Erlangen und promovierte zum Thema Parallelisierung an der Univ. Stuttgart. In zahlreichen Embedded-Projekten brachte er sein Wissen über Architekturen und Entwicklungsmethodik ein. Dr. Birken ist Vortragender auf internationalen Konferenzen und Autor (z.B. Buch "Basiswissen Softwarearchitektur", 3. Auflage). Nach fast einem Jahrzehnt als Infotainment-Architekt bei Harman arbeitet Klaus Birken seit 2012 als Principal Engineer bei der itemis AG.

Kontakt
Internet: www.itemis.de
Email: klaus.birken@itemis.de,
Xing: www.xing.com/profile/Klaus_Birken

Axel Terfloth studierte Informatik an der Universität Dortmund. Seit 2007 arbeitet er bei der itemis AG und leitet dort den Bereich Forschung und Entwicklung für eingebettete Systeme. Er beschäftigt sich in verschiedenen Industrie- und Forschungsprojekten schwerpunktmäßig mit Software-Architekturen für eingebettete Systeme sowie mit Methoden und Werkzeugen zur modellgetriebenen Entwicklung. Er ist als Autor von Fachpublikationen und Referent auf Fachkongressen aktiv und engagiert sich in Open-Source-Projekten.

Kontakt
Internet: www.itemis.de
Email: axel.terfloth@itemis.de