Häufig gestellte Fragen zum Chrona Validator

Was ist konzeptionell der Chrona Validator?

Eine Simulation eines Reglers (= *Controller*), bei der zusätzlich zur Funktionalität das Zeitverhalten des Reglers berücksichtigt wird. Die Simulation kann sowohl auf einem offenen Regelkreis als auch auf einem geschlossenen Regelkreis basieren. Bei einem offenen Regelkreis ist kein Modell der zu regelnden Entität (= *Plant*) nötig, sondern es genügen aufgenommene Sensor-Werte oder manuelle Eingaben. Der Validator erlaubt darüberhinaus einen geschlossenen Regelkreis von *Controller* und *Plant* in Form einer Co-Simulation. Sowohl bei offenem als auch bei geschlossenem Regelkreis wird der tatsächliche Quell-Code des Reglers ausgeführt.

Wie funktioniert der Chrona Validator?

Der Chrona Validator beruht auf einer zusätzlich zum Regler-Code hinzugefügten "Schaltzentrale", die an bestimmten Stellen bei der Ausführung entscheidet, ob eine Funktion (= *Task*) unterbrochen wird, wie mit einer Unterbrechung (= *Interrupt*) umgegangen wird, etc. Diese Schaltzentrale ist während der Simulation eine zusätzliche Komponente im Regler, die als diskrete Ereignissimulation implementiert ist. Im eigentlichen Regler-Code werden an Stellen, wo Entscheidungen bei der Ausführung auf einer Plattform vom Betriebssystem getroffen werden, Rücksprünge zur Schaltzentrale eingefügt. Die Schaltzentrale entscheidet dann aufgrund eines separat definierten Plattformmodells, was in der konkreten Situation zu tun ist. Je detaillierter das Plattformmodell ist, umso genauer entspricht die simulierte Ausführung der Ausführung auf einer Plattform.

Wieviel Aufwand ist es, ein System für eine Validator-Simulation vorzubereiten?

Die Vorbereitung des C-Codes ist weitgehend automatisiert: Hilfswerkzeuge analysieren den Quelltext, um zum Beispiel globale Variablen und Aufrufe von Betriebssystem-Funktionen zu finden. An diesen Stellen wird jeweils ein Rücksprung zur diskreten Ereignissimulation des Validators, also zur "Schaltzentrale" eingefügt. Die Annotierung des Quellcodes mit Ausführungszeiten erfolgt auch automatisch. Die Ausführungszeiten können durch statische Code-Analyse oder durch Messungen ermittelt werden. Zusätzlich zur Vorbereitung des C-Codes des Reglers ist die Ausführungsplattform zu modellieren. Dazu bietet der Chrona Validator das visuell/interaktive Werkzeug Ovid zusammen mit Bibliotheken für typische Plattformen an, wie zum Beispiel für das Betriebssystem OSEK oder den CAN-Bus.

Welche Simulationsumgebungen und welche Programmiersprachen werden unterstützt?

Der Regler muss als C-Code vorliegen. Das Modell der zu regelnden Entität (= *Plant*-Modell) muss in der derzeitigen Version des Validators ein MATLAB®/Simulink®-Modell sein. Das *Plant*-Modell ist nur für einen geschlossenen Regelkreis nötig. Wenn der Regler oder Teile davon in MATLAB®/Simulink® entwickelt wurden, muss in der aktuellen Validator-Version

noch für diese Teile C-Code, zum Beispiel mit dem Embedded Coder® oder Simulink Coder®, erzeugt werden.

Wie modelliere ich die Ausführungsplattform?

Mit einem visuell/interaktiven Hilfswerkzeug, das wir Ovid genannt haben. Typische Plattform-Aspekte wie das Betriebssystem oder Busverbindungen können mit den vorgefertigten Bibliotheksbausteinen rasch modelliert werden. Zusätzlich kann mit Ovid komfortabel die Betriebssystem-Konfiguration editiert werden, damit in der Validator-Simulation mit verschiedenen Konfigurationsvarianten experimentiert werden kann. Beispielsweise könnte die Aufruffrequenz einer Funktion verändert werden.

Wie bekomme ich die Ausführungszeiten von Code-Teilen?

Mit Analysewerkzeugen oder durch Messungen, zum Beispiel auf einem *Integrated Circuit Emulator*.

Wie genau ist eine Validator-Simulation?

Die Genauigkeit hängt von der Genauigkeit der Ausführungszeiten und dem Detailierungsgrad der Plattformmodellierung ab. Beispielsweise kann bei der Plattformmodellierung zunächst die CAN-Kommunikation mit anderen Steuergeräten vernachlässigt werden und erst später hinzumodelliert werden.

Erfordert eine Validator-Simulation ein *Plant*-Modell?

Nein, im Fall eines offenen Regelkreises. Ja, im Fall eines geschlossenen Regelkreises, und zwar in der derzeitigen Version des Validators als MATLAB®/Simulink®-Modell.

Kann ich den Chrona Validator verwenden, wenn ich den Regler neu in MATLAB®/Simulink® entwickelt habe?

Ja, allerdings muss in der aktuellen Validator-Version noch für diese Teile C-Code, zum Beispiel mit dem Embedded Coder® oder Simulink Coder®, erzeugt werden.

Benötige ich den Validator überhaupt, wenn ich ein "sauberes" Scheduling definiert habe?

Ja, da auch andere Aspekte für die Korrektheit wichtig sind. Beispielsweise hängt bei sicherheitskritischen Systemen das Verhalten davon ab, wann und wie kritische Unterbrechungen (wie etwa bei einer Motorsteuerung die von der Kurbelwelle ausgelösten Unterbrechungen) behandelt werden.

Die Validierung durch Testen, wie sie der Validator unterstützt, wird nur dann überflüssig, wenn ein sogenannter C*orrect-by-Construction*-Ansatz verwendet wird. Beispielsweise garantiert Chronas *Timing Definition Language* (TDL) formal, dass das modellierte Zeitverhalten auch auf der jeweiligen Plattform unter allen Umständen eingehalten wird.

Wozu benötige ich Co-Simulation? Genügt es nicht, den C-Code in eine MATLAB®/Simulink®-S-Funktion zu geben?

Erfahrungsgemäß kann ein bestehender C-Code meist nicht in eine MATLAB®/Simulink®-S-Funktion gegeben werden, da er nicht compiliert werden kann. Der Grund dafür ist, dass in MATLAB®/Simulink® die entsprechenden Compiler-Optionen nicht einstellbar sind.

Selbst wenn das Compilieren möglich wäre, könnte damit noch keine zeitgetreue Simulation des Gesamtsystems durchgeführt werden, da keine "Schaltzentrale", die eine Kernkomponente des Validators darstellt, sowie die zugehörigen Zeitannotationen und die Plattformmodellierung vorhanden sind.

Welche Vorteile bietet der Validator gegenüber anderen Werkzeugen mit ähnlicher Zielrichtung, was sind die Nachteile?

Die Vorteile von Chronas Validator sind signifikant und können durch zwei Charakteristika zusammengefasst werden: eine wesentlich genauere Simulation, sodass Fehler tatsächlich beim Testen gefunden werden, sowie nahtlose Integration in die gängigen Entwicklungswerkzeuge MATLAB®/Simulink® und Eclipse. Etwas ausführlicher lassen sich die Vorteile so beschreiben:

1) der Kontrollfluss muss nicht separat modelliert werden, weil der vorliegende C-Code ausgeführt wird; dadurch entspricht das simulierte Verhalten auch dem Verhalten bei Ausführung auf der Zielplattform (= *Target*)

2) die Validierung ist mit dem Validator wesentlich effektiver, weil zusätzlich zu 1) auch

    a) die Ausführungszeiten fein-granular definierbar sind, wohingegen bei anderen Werkzeugen diese nur grob-granular für eine Task festgelegt werden und

    b) weil mit dem Validator sowohl eine Simulation mit offenen Regelkreis als auch mit geschlossenem Regelkreis möglich ist.

3) Der Validator ist nahtlos in MATLAB®/Simulink® und in Eclipse integriert.

Zur Zeit unterstützt der Validator noch nicht den Import von UML-Diagrammen mit annotiertem Zeitverhalten.

Der Chrona Validator kann um Werkzeuge ergänzt werden, die die Verwaltung von Testfällen ermöglichen, die aber keine Co-Simulation wie der Validator unterstützen.

Unterstützt der Validator die Erfüllung von ISO 26 26 2?

Ja, sogar optimal. Der Validator bietet von allen am Markt erhältlichen *Software-in-the-Loop* (SiL)-Werkzeugen die effektivste Validierung von Funktionalität und Echtzeitverhalten und trägt somit wesentlich zur Erfüllung des ISO 26262 Standards bei.