Integrationsstrategien für Cyber-Physikalische Systeme

Mario Winter
Technische Hochschule Köln
mario.winter@th-koeln.de

Abstrakt: Cyber-Physikalische Systeme (cyber-physical systems, CPS) bestehen aus einem Verbund mechatronischer Systeme, die auf unterschiedliche Arten gekoppelt sind und über Kommunikationsinfrastrukturen wie z. B. das Internet miteinander kommunizieren. IoT & Industrie 4.0 repräsentieren hinsichtlich des Spektrums der Granularität ihrer Teilsysteme charakteristische Beispiele für CPS. Bei beiden spielen Integration und Integrationstest der Teilsysteme eine wichtige Rolle. Der Beitrag kategorisiert nach einem Literaturüberblick zunächst die unterschiedlichen Arten der Kopplung und Interaktion mechatronischer Systeme und ihrer (eingebetteten) Software-Komponenten sowie Möglichkeiten zu ihrer Modellierung. Danach werden unterschiedlichen Strategien von Integrationstests für CPS sowie ihre Voraussetzungen und Ziele skizziert.

Schlüsselworte: Softwaretest, Cyber-Physikalische Systeme, CPS, Kopplung, Abhängigkeiten, SysML, Integrationstest, Integrationsstrategie, Optimierung

1 Einleitung

Integration und Integrationstest spielen bei CPS eine herausragende Rolle, da diese oft aus vielen, heterogenen und unterschiedlich granularen eingebetteten Systemen bestehen, welche überdies auf unterschiedliche Arten gekoppelt sind und über Kommunikationsinfrastrukturen wie z. B. das Internet miteinander kommunizieren. Bei CPS als Verbund vervielfachen sich somit die Test-Herausforderungen eingebetteter Systeme ([Tr07], [Gi10]).

Auch die Softwaretest-Umfrage 2015/16 sieht zukünftigen Forschungsbedarf für die QS in diesen Bereichen. Mit 84% ist dort zwar Safety/Security eindeutiger Spitzenreiter, direkt gefolgt jedoch von dem Internet der Dinge/Industrie 4.0 mit 73% und Künstlicher Intelligenz mit ebenfalls 73%. Knapp dahinter folgen Smart Living/Digitale Lebenswelten (68%) und Eingebettete Systeme (67%) sowie Big Data (60%) [Wi17].

Der Beitrag erläutert nach einem Literaturüberblick zunächst die unterschiedlichen Arten der Kopplung und Interaktion mechatronischer Systeme und ihrer (eingebetteten) Software-Komponenten sowie Möglichkeiten zu ihrer Modellierung. Danach werden unterschiedlichen Strategien von Integrationstests für CPS sowie ihre Voraussetzungen und Ziele skizziert.

2 Beiträge zu Integration und Integrationstest für CPS

In [Bo16] beschreiben die Mitglieder des EU-AMADEUS-Projekts ihre Erkenntnisse zu „Systems-of-Systems“ (SoS). Hierbei liegt das Augenmerk auf den Schnittstellen zwischen den vielen Bausteinen solcher Systeme und dem sich daraus ergebenden ermergenten Verhalten. Sie schreiben „*Systems-of-systems become alive by exchanging information and control between their constituent systems and the physical environment via interfaces. […] The most fascinating and disturbing phenomenon in systems-of-systems is emergence: Behavior or properties that only become active or visible when the constituent systems start cooperating.*“

In [Wi12] behandeln die Autoren Grundlagen, Methoden und Techniken für den Software-Integrationstest. Kernthema sind dabei Abhängigkeiten zwischen Software-Bausteinen. Einen relativ aktuellen Überblick über modellbasierte Integrationstests geben Häser et al. in [Hä14].

Sztipanovits et al. schlagen in [Sz12] eine Theorie der Integration für CPS vor, die sich auf Stabilität (im regelungstechnischen Sinn) konzentriert. Kern der Arbeit ist ein Entwurfsansatz, der die Stabilität von den Zeit-Unsicherheiten entkoppelt, die durch Vernetzung und Berechnung verursacht werden. Darüber hinaus beschreiben sie domänenübergreifende Abstraktionen, die eine effektive Lösung für modellbasierte, vollautomatische Softwaresynthese und Leistungsanalysen bieten.

Ruchkin formuliert Herausforderungen und Forschungs-richtungen für die Integration komplexer Systeme. Er plädiert für eine Erweiterung der Sicht von Komponenten und einzelnen Modellierungstechniken hin zu Modellierungsmethoden, Datenerhebungen und menschlichen Faktoren [Ru16].

Abbaspour et al. geben einen Überblick über Testverfahren für CPS [Ab15]. Sie unterscheiden sechs Test-Ebenen und positionieren den Integrationstest herkömmlich als vorletzte Testebene vor den Systemtest und zitieren hierzu einige eher Testtechnik-fokussierende Veröffentlichungen.

Boumen et al. beschreiben ein Verfahren zur Berechnung optimaler Integrations- und Testreihenfolgen für komplexe Systeme [Tr07][Bo09]. Eingabe sind die geschätzte Entwicklungsdauer der Komponenten, die Dauer und erforderlichen Komponenten für die Tests sowie die (paarweisen) Schnittstellen zwischen den Komponenten. Optimierungskriterium ist die minimale Dauer für Entwicklung, Integration und Test. Algorithmus ist eine abgewandelte Tiefensuche auf And/OR-Bäumen.

Da Silva et al. umreißen eine Methode zur Erzeugung von dynamischen Plänen für den Integrationstest selbst-adaptierender Systeme [dS11]. Eingabe ist eine Architekturbeschreibung sowie eine generische Prozesschablone für den Integrationsatest, Ausgabe ein konkreter Integrations-Testplan. Der konkrete Algorithmus zur Reihenfolgeberechnung wird nicht erläutert, die Autoren schreiben lediglich: „*The outcome of this activity is an abstract architectural model with its associated integration order and necessary stubs.*“

[Wi13] gibt ein ganzzahliges lineares Optimierungsverfahren zur Ermittlung optimaler Integrationsreihenfolgen an, welches als Zielkriterium die Kosten zur Erstellung von Test-Stellvertretern (stubs) und -Treibern (driver) minimiert.

3 Bausteine und Abhängigkeiten in CPS

CPS setzen sich aus heterogenen und unterschiedlich granularen eingebetteten Bausteinen zusammen: Vom einfachen „Single-Chip-Thing“ im Internet of Things über komplexere Sensor-Prozessor-Aktuator-Elemente bis hin zu autonomen, KI-gestützten (Teil-)Systemen mit *Monitor–Analyze–Plan–Execute* (MAPE)-Kontroll-Schleife. Diese interagieren und kommunizieren über entsprechend

heterogene Kopplungs-Arten und Kommunikations-Infrastrukturen.

Die Kopplung der Bausteine in CPS lässt sich grob in physikalische (mechanisch, elektromagnetisch), informationstechnische (Hardware, Software) und organisatorische Kopplungsarten kategorisieren (s. u.A. [Gi10], [Wi12], [Bo16]):

- Mechanisch (Kraft, Wärme, (Material-) Fluss, …)
- Elektromagnetisch (Strom(versorgung), Signale, Wert-/Zeit-Diskret / -Kontinuierlich, …)
- Hardware (Speicher, Netzwerke, …)
- Software (Syntaktisch, Semantisch, Indirekt, …)
- Organisatorisch (Projekt-Zeitpläne, Risiken/Kritikalität von Bausteinen, Lieferantenbeziehungen, …)

In der SysML [OMG15] werden solche Kopplungen wie in Abb. 1 gezeigt i.d.R. über Ports der Bausteine (Blöcke) modelliert, welche mittels Interface-Blöcken typisiert und mit Verhaltensmodellen weiter spezifiziert werden können. Die Dynamik der physikalischen Kopplungen wird oft über Simulationsmodelle spezifiziert.

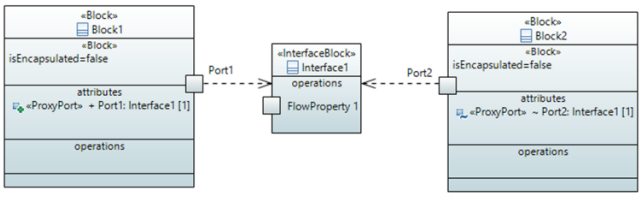

Abb. 1. Modellierung von Kopplungen in SysML

Sind zwei Bausteine auf eine oder mehrere Arten gekoppelt, so lässt sich für jede Kopplung einer der beiden als abhängiger und einer als unabhängiger Baustein identifizieren. Der Integrationstest fokussiert nun genau auf solche Abhängigkeiten zwischen Bausteinen. Abstrahiert man von den einzelnen Kopplungen, dann existieren zwischen je zwei Bausteinen maximal zwei Abhängigkeiten.

4 Heuristische und optimale Integrationsstrategien

Die Integrationsstrategie legt fest, wie viele Bausteine in einem Integrationsschritt zur bereits integrierten Menge von Bausteinen hinzugefügt werden und in welcher Reihenfolge die Bausteine zum Gesamtsystem zusammengesetzt werden [Wi12]. Erfolgt dies aufgrund der Abhängigkeiten der Bausteine, spricht man von strukturabhängigen Integrations-strategien. Meistens wird die Integrationsstrategie in der Praxis heuristisch manuell festgelegt, obwohl viele Ansätze für ihre Optimierung bekannt sind.

Heuristische Vorgehensweisen gehen bezüglich der Abhängigkeiten oft Top-Down oder Bottom-Up vor. Top-Down erfordert es hauptsächlich stubs, während Bottom-Up mehr driver benötigt werden. Angelehnt an eine systemische Betrachtungsweise („Mehre die Möglichkeiten!“) kann folgende Heuristik helfen, während der Integration flexibel auf Änderungen im Projekt reagieren zu können:

Ist der Median des Fan-Out (Anzahl ausgehender Abhängigkeiten) der Bausteine größer als der Median ihres Fan-In (Anzahl eingehender Abhängigkeiten), dann integriere Top-Down, sonst Bottom-Up.

Optimale Integrationsstrategien hängen von der Art des Optimierungszieles wie z.B. Integrationszeit [Bo09] oder Kosten für Stubs und Driver [Wi13] sowie von den dabei betrachteten Produkt- und Projektcharakteristika wie z.B. Struktur, Komplexität, Risiken oder Lieferzeitpunkte ab.

Experimente mit dem strukturbasierten Optimierungs-verfahren aus [Wi13] für die 41 abstrakten Teilsysteme des SysML-Modells einer realen Bahn-Triebzug-Plattform mit 277 Abhängigkeiten ergaben eine Einsparung der grob geschätzten Kosten für driver und stubs gegenüber einer Bottom-Up-Vorgehensweise um 20%.

In laufenden und zukünftigen Arbeiten wird das Optimierungsverfahren zunächst für organisatorische Abhängigkeiten erweitert und dann mittels tiefergehender Analysen weiterer Aspekte des SysML-Modells präzisiert.

5 Literaturverzeichnis

[Ab15] Abbaspour Asadollah, S.; Inam, R. & Hansson, H.: A Survey on Testing for Cyber Physical System. In: Testing Software and Systems: 27th IFIP WG 6.1 Int. Conf. ICTSS, 2015, S. 194-207

[Bo16] Bondavalli, A.; Bouchenak, S.; Kopetz, H. (Eds.) : Cyber-Physical Systems of Systems. Springer International Publishing, 2016

[Bo09] Boumen, R.; de Jong, I.; Mestrom, J.; van de Mortel-Fronczak, J. & Rooda, J.: Integration and Test Sequencing for Complex Systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2009, S. 177 -187

[dS11] da Silva, C. E. & de Lemos, R.: Dynamic plans for integration testing of self-adaptive software systems. Proc. 6th Int. Symp. on SE for Adaptive and Self-Managing Systems. ACM, 2011, S. 148-157

[Gi10] Giese et al.: Model-Based Integration. Proc. Int. Dagstuhl Conference on Model-based Engineering of Embedded Real-time Systems. LNCS 6100, Springer, 2010, S. 17-54

[Hä14] Häser, F.; Felderer, M. & Breu, R.: Software Paradigms, Assessment Types and Non-functional Requirements in Model-based Integration Testing: A Systematic Literature Review. Proc. 18th Int. Conf. on Evaluation and Assessment in Software Engineering, ACM, 2014, S. 1-29

[OMG15] Object Management Group: SysML Version 1.4 specification. OMG document formal/2015-06-03, 2015

[Ru16] Ruchkin, I.: Integration Beyond Components and Models: Research Challenges and Directions 2016. Proc. Architecture-Centric Virtual Integration (ACVI), 2016, S. 8-11

[Sz12] Sztipanovits, J.; Koutsoukos, X.; Karsai, G.; Kottenstette, N.; Antsaklis, P.; Gupta, V.; Goodwine, B.; Baras, J. & Wang, S. Toward a Science of Cyber-Physical System Integration Proceedings of the IEEE, 2012, 100, 29-44

[Tr07] Tretmans, J. (Ed.) Tangram: Model-based integration and testing of complex high-tech systems. Embedded Systems Institute, 2007

[Wi12] Winter, M.; Ekssir-Monfared, M.; Sneed, H. M.; Seidl, R. & Borner, L.: Der Integrationstest. Carl Hanser Fachbuchverlag, München, 2012

[Wi13] Winter, M.: Optimale Integrationsreihenfolgen. Proc. SE 2013, GI-LNI, Vol. P-213, 2013, S. 259-270

[Wi17] Winter, M.; Spillner, A.; Vosseberg, K.: Softwaretest in der Forschung – Ergebnisse der Umfrage 2015/16. 40. TAV-Workshop der GI-FG "Test, Analyse und Verifikation von Software", Softwaretechnik-Trends, Vol. 37, Nr. 1, Februar 2017

Danksagung: Der Autor bedankt sich bei Siemens Corporate Technology und Siemens Mobility für die Kooperation.