Inhaltsverzeichnis

KEYNOTE

MODELLIERUNG

ARCHITEKTUR

ECHTZEIT & MULTICORE

IMPLEMENTIERUNG

OPEN SOURCE

SAFETY

SECURITY

TEST & QUALITÄT

AUTOMATION

MACHINE LEARNING

AUTOMOTIVE

INTERNET OF THINGS

SOFTWARE ENGINEERING MANAGEMENT

MENSCH – MANAGEMENT – TEAM

TIPPS – TRICKS – LÖSUNGEN

2011-2021: Was hat der Umstieg auf Agilität verändert?

Aus der Perspektive von Entwickler*innen, Tester*innen, Projektleiter*innen und Testmanager*innen.

Prof. Dr. Andreas Spillner (i.R.), Hochschule Bremen
Prof. Dr. Karin Vosseberg, Hochschule Bremerhaven
Prof. Dr. Mario Winter, Technische Hochschule Köln

Seit 2011 wird die Umfrage „Softwaretest in Praxis und Forschung“ alle fünf Jahre ausgerichtet. In den letzten Jahren werden immer mehr Projekte agil durchgeführt. Was hat sich verändert? Gibt es Branchen, die agiler sind als andere? Wird frühzeitiger, intensiver und systematischer getestet? Ist der Grad der Automatisierung von Unit Tests gestiegen? Wie ist das Verhältnis zwischen funktionalen Tests und nicht-funktionalen Tests? Welche Testverfahren werden eingesetzt? Testen Entwickler*innen anders als Tester*innen und anders als früher? Ist die Software qualitativ besser geworden, gibt es weniger Fehler?
Im Folgenden wird diesen Fragen nachgegangen mit dem Schwerpunkt der Gegenüberstellung der Sichten von Entwickler*innen und Tester*innen ebenso die Sichten von Projektleiter*innen und Testmanager*innen. Alle Fragen und Antworten der drei Umfragen, sowie zusammenfassende Broschüren und ausführliche Technische Berichte sind zum Download auf der Internetseite der Umfrage (www.softwaretest-umfrage.de) zu finden.

Ausgangssituation

Die Umfragen zu „Softwaretest in Praxis und Forschung“ wurden in den Jahren 2011, 2016 und 2020 durchgeführt. Die wissenschaftliche Leitung oblag im Wechsel Prof. Dr.-Ing. Andreas Spillner (Hochschule Bremen), Prof. Dr.-Ing. Karin Vosseberg (Hochschule Bremerhaven) sowie Prof. Dr. Mario Winter (Technischen Hochschule Köln). Finanziert wurden die Umfragen überwiegend vom German Testing Board (GTB e.V.) mit Beteiligung der Testing Boards aus Österreich (ATB) und der Schweiz (STB). Die Umfrage 2011 wurde von der Firma Anecon Software Design und Beratung G.m.b.H mitfinanziert. Unterstützt wurden die Umfragen von der Fachgruppe TAV (Test, Analyse und Verifikation von Software) der Gesellschaft für Informatik, dem ASQF (Arbeitskreis für Software-Qualität und -Fortbildung), den Softwareforen Leipzig, dem dpunkt.verlag und dem bitkom. Sie haben für die Umfrage geworben, um möglichst viele Personen zur Beantwortung der Fragen zu bewegen.
An den drei Umfragen nahmen zwischen 1.200 und 1.600 Personen teil, wovon im Schnitt die Hälfte auch den jeweiligen Fragebogen bis zum Ende ausgefüllt hat. Ein Großteil der Antwortenden kommt aus dem Umfeld der Qualitätssicherung, allerdings wurde auch auf dem Heise-Ticker für die Umfrage geworben und damit viele Entwickler*innen motiviert, teilzunehmen. Dies trifft besonders auf die Umfrage 2020 zu, hier lag der Anteil der Entwickler*innen bei über 25% (2016 15%, 2011 19%).

Agile Vorgehensweise

Als im Jahr 2000 das Agile Manifest publiziert wurde, war nicht abzusehen, wie weit die agile Vorgehensweise und die agilen Praktiken Einzug in die Praxis finden. Bei den drei Umfragen wurde nach dem verwendeten Vorgehensmodell gefragt und dabei zwischen phasen-/planorientiertem, agilem und keinem expliziten Vorgehensmodell unterschieden (s. Abb. 1).

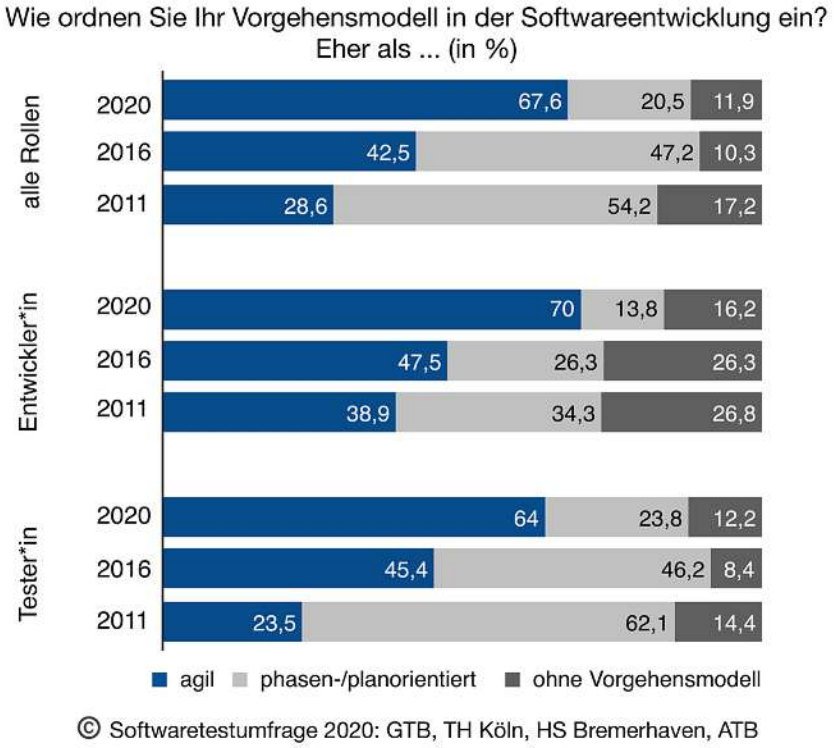

Abbildung 1: Veränderungen in der Nutzung von Vorgehensmodellen

In 2011 richten sich noch doppelt so viele Projekte nach einem phasen-/planorientierten Vorgehen wie nach agilen Modellen. In 2020 hat sich das Bild mehr als gedreht: mehr als dreimal so viele Projekte werden agil als phasen-/planorientiert durchgeführt. Eher konstant geblieben ist der Anteil der Projekte, die keinem expliziten Vorgehensmodell folgen.
Werden die Entwickler*innen und Tester*innen separat betrachtet, zeigt sich, dass die Entwickler*innen agiler sind. Sie liegen bei allen drei Umfragen über den Werten aller Befragten. Sind die Entwickler*innen die Treiber der Agilität? Es scheint so zu sein. Aber der Anteil der Entwickler*innen, die ohne ein Vorgehensmodell auskommen, liegt weit über dem Durchschnitt aller Befragten. Der Umstieg zur Agilität geht insgesamt auf Kosten der phasen-/planorientierten Vorgehensmodelle. Ein ähnliches Bild ergibt sich bei den Testmanager*innen und Projektleiter*innen allerdings mit einem höheren Anteil an phasen-/planorientierten Vorgehen.

Agile Branchen

In der differenzierten Betrachtung der Branchen wurden im Jahr 2016 58% der Projekte im Bereich Konsumgüter & Handel bereits agil durchgeführt. 2020 geht der erste Platz

mit 75% an die Branche Gesundheits- & Sozialwesen, gefolgt vom ersten Platz aus 2016 (2020: 73%). Alle Angaben zu den agil realisierten Projekten variieren in den Branchen zwischen 61% und 75% bei einem Median bei 70%. 61% werden in der Medizintechnik erreicht, die damit die Branche ist, deren Projekte am wenigsten agil durchgeführt werden.
Eine große Veränderung im Vergleich zu 2016 hat die Automotive-Branche vollzogen: Von ehemals 37% werden aktuell fast doppelt so viele Projekte agil bearbeitet (66%). In der Luft- & Raumfahrt gibt es mit 26% noch den größten Anteil an phasen-/planorientiert durchgeführten Projekten, Medien & Entertainment sind hier das Schlusslicht mit 14%. Der Median über alle Branchen liegt bei den phasen-/planorientierten Modellen bei knapp 20%.
Unter der Prämisse, dass Vorgehensmodelle im Allgemeinen Grundlage für eine Steuerung von Qualität sind, ist es beruhigend, dass in der Branche Gesundheits- & Sozialwesen nur knapp 8% der Projekte ohne jedes Vorgehensmodell auskommen – allerdings sind es noch doppelt so viele (16%) in der Medizintechnik. Auch hier werden mit 23% noch vergleichsweise viele Projekte phasen-/planorientiert realisiert.

Agile Rollen

Mit der Agilität sind bestimmte Rollen verbunden. In 2020 wurde gefragt, welcher Rolle sich die Befragten mit ihrer Hauptaufgabe zuordnen. Die meist genannte agile Rolle ist der Product Owner, allerdings von nur 4,9% – auf den Scrum Master entfielen nicht einmal 1%. Ihre Hauptaufgabe im agile Team tätig zu sein, sahen ebenfalls nur 2,3%. Die etablierten Rollen der Softwareentwicklung prägen weiterhin die Aufgabenteilung in den Teams, scheinbar unbeeinflusst vom Vorgehensmodell: Entwickler*in 27,1%, Tester*in 17,7%, Testmanager*in 15,9%, Projektleiter*in/Teamleiter*in 9,2%.

Agile Praktiken

Bei der Frage nach der Bedeutung der agilen Praktiken im Hinblick auf die Qualitätssicherung ergibt sich über die Jahre eine deutliche Zunahme mit folgender Reihenfolge (s. Tab. 1).

Praktik	2020		2016		2011
Refactoring	63,6 %	↖	60,8 %	↖	39,9 %
Collective Code Ownership	45,1 %	↖	35,1 %	↖	22,1 %
Pair Programming	44,1 %	↖	40,3 %	↖	29,8 %
Zentrale Storyboards	38,3 %	↖	36,3 %	↖	24,4 %

Tabelle 1: Bedeutung der Praktiken für die Qualitätssicherung

An Bedeutung verloren hat die gemeinsame Aufwandsschätzung der Arbeitspakete (2020 49,2%, 2016 59,2%, 2011 60,9%). Und auch bei der Testgetriebenen Entwicklung (2020 57,7%, 2016 67,2%, 2011 49,2%) ist nach einem Anstieg in 2016 ein

Rückgang in 2020 zu verzeichnen. Nach Rollen aufgeteilt ergibt sich hierfür folgendes Bild: Entwickler*innen und Tester*innen messen in den letzten fünf Jahren der Testgetriebenen Entwicklung keine so große Bedeutung für die Qualitätssicherung mehr zu (s. Tab. 2). Ist das Vorgehen in der Praxis zu aufwändig und/oder durch Testautomatisierung und Continuous Integration abgelöst bzw. unnötig geworden?

Rolle	2020	2016	Differenz (%-Punkte)
alle Rollen	57,7 %	67,2 %	-9,5
Projektleiter*in	62,6 %	63,4 %	-0,8
Testmanager*in	58,1 %	62,7 %	-4,6
Entwickler*in	58,6 %	71,6 %	-13,0
Tester*in	47,9 %	58,1 %	-10,2

Tabelle 2: Bedeutung der Testgetriebenen Entwicklung

Alle weiteren Praktiken (User Stories, Retrospektive, Productbacklog/Sprintbacklog, Stand-up Meeting, Auswertung der Effektivität) haben über die Jahre kaum Veränderungen in der Einschätzung erfahren.
Entwicklungsnahe Praktiken wie Testautomatisierung (82,2%), Code Reviews (78,6%), Continuous Integration (75,2%) und Clean Code (73%) haben aktuell für die Befragten eine sehr hohe Bedeutung für die Qualitätssicherung. Dabei ist zu berücksichtigen, dass die vier Praktiken erst in 2020 separat abgefragt wurden und bei der aktuellen Umfrage der Anteil an Entwickler*innen vergleichsweise hoch war.

Frühzeitiges Testen

„Zuerst die Testfälle erstellen, …“ hat an Bedeutung verloren. Werden dann Reviews als erste Aktivität zur Qualitätssicherung genutzt? Wie sieht es bei den Reviews von Anforderungen/User Stories aus? Optimistisch sind die Projektleiter*innen mit einem Abstand gefolgt von den Testmanager*innen, weniger überzeugt sind die beiden operativ tätigen Rollen (s. Tab. 3).

Rolle	2020	2016
alle Rollen	61,2 %	61,9 %
Projektleiter*in	80,6 %	67,7 %
Testmanager*in	63,6 %	61,7 %
Entwickler*in	49,8 %	52,6 %
Tester*in	55,2 %	67,1 %

Tabelle 3: Werden Anforderungen immer oder meist einem Review unterzogen?

Ein ähnliches Bild ergibt sich bei den anderen Artefakten (Architektur, Schnittstellen, Code, Testfälle), die einem Review unterzogen werden: Das Management ist meist optimistischer als die Personen, die Reviews durchführen. Die größten Zuwächse gab es beim Code-Review (s. Tab. 4). Ein „Shift-left“ (frühzeitiges Testen) kann mit den Zahlen der Umfrage allerdings nicht nachgewiesen werden.

Rolle	2020		2016
alle Rollen	66,8 %	↖	51,5 %
Projektleiter*in	75,0 %	↖	60,6 %
Testmanager*in	63,3 %	↖	47,2 %
Entwickler*in	69,8 %	↖	57,3 %
Tester*in	61,6 %	←	60,8 %

Tabelle 4: Werden Code.Reviews immer oder meist durchgeführt?

Testverfahren

Das meistgenannte Testverfahren ist der Anwendungsfallbasierte Test. 70,6% der Befragten setzten ihn immer oder meist ein, gefolgt vom Explorativen Testen ohne Charta (33,2%), der Grenzwertanalyse (27,9%), der Äquivalenzklassenbildung (24,6%), dem Zustandsbasierten Test (20,6%) und dem Explorativen Testen mit Charta (17,5%). Alle „klassischen Testverfahren“ haben im Vergleich zu 2016 erheblich an Bedeutung verloren, sogar bei Tester*innen und Testmanager*innen (beispielhaft s. Tab. 5).

Rolle	2020		2016
alle Rollen	24,6 %	↙	38 %
Projektleiter*in	20,3 %	↙	38,3 %
Testmanager*in	36,6 %	↙	45,7 %
Entwickler*in	12,0 %	↙	17,1 %
Tester*in	32,2 %	↙	47,6 %

Tabelle 5: Immer oder meist wird die Äquivalenzklassenbildung eingesetzt

Einhergehend mit der Agilität hat Exploratives Testen an Bedeutung gewonnen. Lässt sich diese Behauptung belegen? (s. Tab 6)

Rolle	Expl.Test ohne Charta			Expl.Test mit Charta		
	2020		2016	2020		2016
alle Rollen	33,2 %	←	34,8 %	17,5 %	←	17,6 %
Projektleiter*in	17,8 %	↙	24,2 %	18,9 %	↖	13,3 %
Testmanager*in	40,9 %	←	44,1 %	22,5 %	←	19,0 %
Entwickler*in	21,3 %	←	24,9 %	9,7 %	←	7,9 %
Tester*in	52,6 %	↖	35,8 %	21,8 %	←	20,5 %

Tabelle 6: Immer oder meist wird Exploratives Testen (ohne/mit Charta) eingesetzt

Beim Explorativen Testen mit Charta gibt es leichte Anstiege und die Tester*innen scheinen von dem systematischen Vorgehen bei der Testfallerstellung hin zum Explorativen Test ohne Charta gewechselt zu haben.
Bei der Nutzung aller Testverfahren ergibt es keine einheitliche Einschätzung zwischen Management und operativ Tätigen, mal sieht das Management eine höhere Nutzung, mal die operativ Tätigen.

Testautomatisierung

Das die Testautomatisierung 2020 einen hohen Stellenwert einnimmt, wurde bei den Praktiken bereist deutlich. Wie hoch ist der Grad der Automatisierung auf den jeweiligen Teststufen und wie sieht es das Management bzw. die operativ Tätigen? Auf der untersten Teststufe, dem Unit-Test, ist ein deutlicher Anstieg der vollständigen Automatisierung (100%) zur Umfrage 2016 gegeben und wird wie folgt von den jeweiligen Rollen gesehen (s. Tab. 7).

Rolle	2020		2016
alle Rollen	47,2 %	↖	28,0 %
Projektleiter*in	57,9 %	↖	32,8 %
Testmanager*in	38,6 %	↖	21,6 %
Entwickler*in	53,3 %	↖	44,4 %
Tester*in	40,9 %	↖	26,0 %

Tabelle 7: 100% Automatisierung beim Unit-Test

Es wäre zu erwarten, dass bei einem agilen Vorgehen, der Grad der Automatisierung der Unit-Testfälle die 100% erreicht oder zumindest angestrebt wird, was (noch) nicht der Fall ist. Bei den höheren Teststufen (Integrationstest, Systemtest, Abnahme-/Akzeptanztest, Systemintegrationstest) nimmt der Anteil der zu 100% automatisiert durchgeführten Tests erwartungsgemäß ab. Beispielhaft sind die Zahlen für den Systemtest

aufgeführt, bei dem – wie bei den anderen Teststufen – eher Schwankungen und kein klarer Trend zu erkennen ist (s. Tab. 8).

Rolle	2020		2016
alle Rollen	8,7 %	←	7,6 %
Projektleiter*in	15,6 %	↖	10,0 %
Testmanager*in	3,6 %	↙	7,2 %
Entwickler*in	11,5 %	↖	10,1 %
Tester*in	3,5 %	↙	7,5 %

Tabelle 8: 100% Automatisierung beim Systemtest

Nicht-funktionale Testarten

Über die Jahre werden die nicht-funktionalen Testarten (z.B., Migrationstest, Sicherheits/Penetrationstest) immer weniger intensiv berücksichtig, So verliert beispielsweise der Last-/Performanztest von 2011 mit 48,2% über 2016 mit 40,1% hin zu 2020 mit 26,5% fast die Hälfte an Zustimmung. Möglicher Weise ist eine Erklärung die Zunahme der Agilität: Für übergreifende Tests fühlt sich das einzelne agile Team nicht wirklich zuständig.

Fazit

Die agilen Vorgehensweisen haben sich etabliert, aber die Umsetzung scheint sehr zu variieren. Auffällig ist, dass die agilen Rollen wie Product Owner oder Scrum Master kaum von den Teilnehmenden als „ihre“ Rolle angesehen werden. Das bisherige Rollenverständnis ist noch verbreitet.
Auch die Bedeutung der agilen Praktiken für die Qualitätssicherung hat nicht in dem Maße zugenommen, wie es die Verbreitung der agilen Vorgehensweise nahelegt. Die Testgetriebene Entwicklung ist rückläufig und auch Reviews werden nicht für die frühen Dokumente der Softwareentwicklung zur Qualitätssicherung genutzt. Code-Reviews haben zugelegt.
Die systematische Herleitung von Testfällen unter Nutzung von Testverfahren hat deutlich abgenommen, sogar bei den Tester*innen, mit Ausnahme von Explorativen Testen ohne Charta, da gibt es eine deutliche Steigerung. Ebenso kann Exploratives Testen mit Charta für die Rollen Projektleiter*in, Testmanager*in, Entwickler*in und Tester*in auf niedrigem Niveau zulegen. Im Mittel über alle Rollen verteilt ist der Wert jedoch gleichgeblieben.
Obwohl die Testautomatisierung 2020 einen hohen Stellenwert einnimmt, lässt der Grad der Automatisierung auf allen Teststufen noch reichlich Platz zur Optimierung.
Konnte durch den Umstieg auf Agilität die Qualität der Software gesteigert werden? Die Frage, ob nach Auslieferung der Produkte schwerwiegende Fehler auftreten beantworten die Befragten 2020 (50% einige oder zu viele Fehler der Funktionalität werden

ausgeliefert) deutlich optimistischer als noch 2016 (62%) und 2011 (78%). Bei einer differenzierten Betrachtung nach Rollen wird jedoch deutlich, dass Projektleiter*innen (34% einige oder zu viele Fehler der Funktionalität werden ausgeliefert) optimistischer sind als Entwickler*innen (53%), Testmanager*innen (51%) und Tester*innen (55%).

Autoren

Prof. Dr. Andreas Spillner war Professor für Informatik an der Hochschule Bremen und ist Mitglied im ASQF-Präsidium. Er war Gründer der Fachgruppe „Test, Analyse und Verifikation von Software“ der Gesellschaft für Informatik e.V. (GI). Er ist GI-Fellow und Ehrenmitglied im German Testing Board e.V.
E-Mail: Andreas.Spillner@hs-bremen.de

Prof. Dr. Karin Vosseberg ist Hochschullehrerin im Studienbereich Informatik der HS Bremerhaven. Ihre Schwerpunkte sind IT-Systemintegration und Software Engineering mit dem Fokus auf Qualität von Software. Seit 2015 ist sie Präsidiumsmitglied im ASQF.
E-Mail: Karin.Vosseberg@hs-bremerhaven.de
Internet: https://informatik.hs-bremerhaven.de/kvosseberg/

Prof. Dr. Mario Winter ist Professor am Institut für Informatik der TH Köln, Gründungsmitglied des German Testing Board e.V. und Sprecher des Arbeitskreises „Testen und KI“ der GI-Fachgruppe „Test, Analyse und Verifikation von Software“.
E-Mail: mario.winter@th-koeln.de
Internet: http://www.gm.fh-koeln.de/~winter/

Model-Driven Safety für Echtzeitanwendungen

Mit MechatronicUML sichere Rekonfigurationen gewährleisten

Prof. Dr. Steffen Becker, ISTE SQA, Universität Stuttgart

Software, die sicherheitskritische Echtzeitsysteme antreibt, muss strenge Anforderungen an die Zeitschranken einhalten, um nicht in katastrophalem Verhalten zu enden. Dies gilt umso mehr, wenn man moderne Architekturen betrachtet, die es erlauben, die Komponenteninstanzen der Software zur Laufzeit zur rekonfigurieren. Solche Rekonfigurationen können dabei das Instanziieren und Zerstören von Komponenteninstanzen oder Kommunikationskanälen, sowie die Änderung der Kommunikationswege sein. Damit entsprechende Rekonfigurationen in sicherheitskritischen Echtzeitanwendungen überhaupt eingesetzt werden können, müssen sie die sogenannten ACI-T Eigenschaften erfüllen, also atomar, konsistent und isoliert sein, sowie Zeitschranken während der Rekonfiguration nicht verletzen. In diesem Paper wird ein Ansatz auf Basis von MechatronicUML, einem aus der Forschung stammenden Komponentenmodell, skizziert. Dieser garantiert die Sicherstellung der ACI-T Eigenschaften durch Generatoren.

Sicherheitskritische Echtzeitsysteme sind durch strikte Echtzeiteigenschaften charakterisiert, die oft aus der physikalischen Domäne der Anwendung entspringen. Als Beispiel nehmen wir in diesem Artikel ein RailCab, ein autonomes Schienenfahrzeug, das in der Lage ist, durch Convoybildung mit anderen RailCabs energieeffizient zu fahren. Für ein solches System ergibt sich, dass ein Bremsmanöver eines solchen RailCabs durch eine zeitliche Oberschranke gekennzeichnet ist, da es ansonsten zu einem Unfall kommt. Ebenso ist das Manöver zur Bildung eines Convoys zeitkritisch, damit das einscherende RailCab nicht auf das vorausfahrende RailCab auffährt.

In modernen Architekturen kann man verschiedene Betriebsmodi durch Rekonfigurationen der unterliegenden Komponenten- und Kommunikationskanalinstanzen umsetzen. Dabei werden sowohl Komponenteninstanzen erzeugt und wieder frei gegeben, als auch Kommunikationsverbindungen angelegt und wieder aufgelöst. Bei der Convoyfahrt kann also zum Beispiel der Vorgang des Einscherens als Wechsel der Architekturkonfiguration von der Normalfahrt zur Convoyfahrt (und später umgekehrt beim Ausscheren) angesehen werden.

Bei Echtzeitsystemen ist nun die Herausforderung, solche Architekturrekonfigurationen funktional sicher umzusetzen. Für den hier diskutierten Ansatz bedeutet dies, dass die sogenannten ACI-T Eigenschaften für die Rekonfigurationen gegeben sein müssen. Rekonfigurationen müssen daher

(1) Atomar (atomic) sein, das bedeutet, sie müssen entweder vollständig oder gar nicht durchgeführt worden sein
(2) Konsistent (consistency) sein, was bedeutet, dass vor und nach Rekonfigurationen keine ungültigen Architekturkonfigurationen vorliegen dürfen
(3) Isoliert (isolated) sein, was bedeutet, das nebenläufige Rekonfigurationen durchgeführt werden können, ohne sich gegenseitig zu stören

(4) Und die Zeitschranken (timing) einhalten, damit die Rekonfiguration beendet ist, bevor ein katastrophaler Zustand (wie ein Auffahrunfall) eintritt.

Im Folgenden skizziere ich einen akademischen Ansatz auf Basis der MechatronicUML Modelliersprache [2], der es erlaubt Rekonfigurationen zu spezifizieren, ihre Umsetzung zu generieren und dabei die ACI-T Eigenschaften beweisbar zu belegen. Der Ansatz wurde von C. Heinzemann in seiner Dissertation vorgestellt und ist im Detail unter [1] beschrieben.

Der Ansatz

Der hier skizzierte Ansatz besteht aus drei Teilen: (1) Einer Spezifikationssprache, die es ermöglicht, Rekonfigurationen zu beschreiben, (2) einem Generator, der die Spezifikationen in ein klassisches MechatronicUML Modell umsetzen kann, und (3) einem Model Checker, der zusichert, dass die Spezifikation die ACI-T Eigenschaften erfüllt.

Spezifikationssprache

Die Spezifikationssprache ermöglicht es, Komponenten des Systems eine Rekonfigurationsschnittstelle hinzu zu fügen. Dabei handelt es sich um spezielle Schnittstellen, über die man eine Rekonfiguration aus einer Liste von bekannten Rekonfigurationen auslösen kann. Gleichzeitig wird sie genutzt, um Rekonfigurationen entlang der Komponentenhierarchie so zu dekomponieren und zu koordinieren, dass die ACI-T Eigenschaften eingehalten werden. Dazu wird unter anderem beschrieben, welche Rekonfigurationen es gibt, was sie tun, wie lange sie maximal brauchen dürfen, ob sie nach unten oder oben in der Komponentenhierarchie propagiert werden müssen, wieviel Zeit zur Planung maximal vergehen darf, welche Komponenten und Kommunikationskanäle angelegt bzw. entfernt werden, und ähnliches. Dabei ist die Spezifikation so reichhaltig und gleichzeitig so formal, dass später Modelle für die Rekonfigurationen generiert werden können, die exakt die beschriebenen Eigenschaften besitzen.

Codegenerator

Der Codegenerator legt aus den Spezifikationen eine Implementierung an, die für jede Komponente, die potentiell an einer Rekonfiguration beteiligt ist, eine Rekonfigurationsschnittstelle und jeweils ein Manager und einen Executor anlegt. Der Manager koordiniert dabei die Planung und Ausführung von Rekonfigurationen, während der Executor diese Rekonfigurationen tatsächlich umsetzt. Der Manager nutzt dabei für die Umsetzung der ACI-T Eigenschaften einen Ansatz, der an das 2-Phase-Commit Protokoll angelehnt ist. Vor einer Rekonfiguration werden alle beteiligten Komponenten entlang der Komponentenhierarchie gefragt, ob sie bereit für die Rekonfiguration sind und ob sie die vorgegebene Zeitschranke einhalten können. Nur wenn alle Antworten positiv ausfallen, dann kann die Rekonfiguration ausgeführt werden. Dies wird dann durch den Executor realisiert, der die Komponenten nach den Hierarchieebenen umbaut.

Model Checker

Anschließend kann der Model Checker auf Basis der Spezifikation der Rekonfigurationen überprüfen, ob die ACI Eigenschaften und insbesondere auch die T-Eigenschaft

gilt. Dabei wird das vom Codegenerator ausgegebene Modell so umgewandelt, so dass es mittels des Uppaal Model Checkers [3] überprüft werden kann. Verläuft die Prüfung erfolgreich, so kann man sicher sein, dass die Rekonfigurationen die ACI-T Eigenschaften gemäß der Spezifikation einhalten (ausgenommen sind hierbei allerdings Hardwareausfälle). Falls die ACI-T Eigenschaften nicht gelten, so bekommt man ein Gegenbeispiel, welches bei der Fehlersuche hilft.

Fazit

Mittels des skizzierten Ansatzes wurde das einleitende RailCab Beispiel modelliert und umgesetzt. Dabei zeigte sich, dass der Ansatz grundsätzlich funktioniert und eine Modellierkomplexität handhabbar macht, die anderenfalls nur schwer in den Griff zu bekommen wäre. Dennoch muss man berücksichtigen, dass der Ansatz ein wissenschaftlicher Prototyp ist, der primär dafür geeignet ist, die Machbarkeit zu demonstrieren und nicht ohne wissenschaftliche Begleitung in die Praxis übernommen werden kann.

Literatur- und Quellenverzeichnis

[1] Heinzemann, C., Becker, S., & Volk, A. (2019). Transactional execution of hierarchical reconfigurations in cyber-physical systems. Softw. Syst. Model., 18(1), 157--189
[2] MechatronicUML website. http://www.mechatronicuml.org/de/index.html
[3] Uppaal. https://uppaal.org/

Autor

Steffen Becker ist Leiter der Abteilung Softwarequalität und -architektur (SQA) am Institut für Software Engineering (ISTE) der Universität Stuttgart. Seine Forschung betrift den Zusammenhang zwischen Softwarequalitätseigenschaften wie Performance, Zuverlässigkeit oder funkt. Sicherheit und der Architektur. Besonders moderne Architekturen, wie Microservice-basierte Architekturen oder rekonfigurierbare Architekturen, stehen dabei im Fokus.

Kontakt

Internet: www.iste.uni-stuttgart.de/sqa
E-Mail: steffen.becker@iste.uni-stuttgart.de