HANSER

Vorwort

Mario Winter, Mohsen Ekssir-Monfared, Harry M Sneed, Richard Seidl, Lars Borner

Der Integrationstest

Von Entwurf und Architektur zur Komponenten- und Systemintegration

ISBN (Buch): 978-3-446-42564-4

ISBN (E-Book): 978-3-446-42951-2

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-42564-4

sowie im Buchhandel.

Geleitwort

von Prof. Dr. Andreas Spillner, Hochschule Bremen

Die Autoren widmen sich einem bisher viel zu wenig beachteten und behandelten Thema im Bereich des Testens von Software-Systemen: dem Integrationstest. Für den Komponenten- und Systemtest gibt es eine Vielzahl von Verfahren und Methoden zur Aufdeckung von Fehlern und eine entsprechend vielfältige Auswahl an Büchern. In der Praxis habe ich schon oft folgende Argumentation zum Integrationstest gehört: „Wir wollen möglichst schnell durch den Integrationstest durch, um mehr Zeit für den Systemtest zu gewinnen. Außerdem ist der Integrationstest durch die Simulation der am Test nicht beteiligten Systemteile viel zu aufwendig. Die Fehler finden wir im Systemtest, wenn sie noch nicht beim Komponententest gefunden wurden."

In der Theorie bzw. Forschung, vor allem in Büchern, wird der Integrationstest allzu oft nur unter dem Gesichtspunkt der Integrationsstrategien behandelt. Ein methodisches Vorgehen zur Auswahl von Testfällen wird nur ganz selten diskutiert. Aber genau darum geht es beim Integrationstest.

Selbst im „Guide to the Software Engineering Body of Knowledge (SWEBOK)" sind unter dem Stichwort „Integrationstest" nur Strategien, aber keine Testentwurfsverfahren zur Prüfung der Schnittstellen zu finden:

> *„2.1.2. Integration testing: Integration testing is the process of verifying the interaction between software components. Classical integration testing strategies, such as top-down or bottom-up, are used with traditional, hierarchically structured software. Modern systematic integration strategies are rather architecture-driven, which implies integrating the software components or subsystems based on identified functional threads. Integration testing is a continuous activity, at each stage of which software engineers must abstract away lower-level perspectives and concentrate on the perspectives of the level they are integrating. Except for small, simple software, systematic, incremental integration testing strategies are usually preferred to putting all the components together at once, which is pictorially called ‚big bang' testing."*[1]

[1] http://www.computer.org/portal/web/swebok/html/ch5#Ref2.1.2 (Juli 2012)

Aber sind denn Fehler bei der Integration von (Teil-)Systemen aufzudecken? Ja, jedoch nur dann, wenn ein „wirklicher" Integrationstest durchgeführt wird. Mein „liebster Integrationstestfehler" soll dies belegen. 1999 ging die Sonde Mars Climate Orbiter beim Anflug auf den Mars verloren. Grund: Bei der Kommunikation zwischen zwei Kontrollteams wurden unterschiedliche Maßeinheiten verwendet:

> *„The peer review preliminary findings indicate that one team used English units (e.g., inches, feet and pounds) while the other used metric units for a key spacecraft operation. This information was critical to the maneuvers required to place the spacecraft in the proper Mars orbit."*[1]

Jedes Team hat sicherlich sein System gründlich getestet, aber halt immer nur sein System in seiner „Welt" – also in Zentimeter oder in Inch. Wahrscheinlich war auch nirgends spezifiziert, in welcher Maßeinheit zu rechnen ist, denn über „Trivialitäten" braucht man sich nicht auszutauschen oder zu reden. Ein Integrationstest mit den realisierten (also nicht den simulierten) Systemen hätte das Problem mit Sicherheit aufgezeigt.

An diesem Beispiel wird auch sehr schön deutlich, dass es eine ganze Reihe von unterschiedlichen Stufen der Integration und damit des Integrationstests gibt. Wird kein Integrationstest durchgeführt, wird die Möglichkeit vergeben, Schnittstellenfehler aufzudecken. Ein Systemtest kann nie so aufwendig sein, dass alle Schnittstellen zwischen den Systemteilen ausgiebig getestet werden können. Vom Problem der aufwendigen Fehlerlokalisierung, wenn ein Fehler aufgetreten ist, ganz abgesehen. Erschreckend ist, dass es bei einigen agilen Vorgehensweisen gar keinen Integrationstest zu geben scheint: Die Entwickler prüfen ihre Komponenten selbst, und nach der Freigabe führt der Kunde den Akzeptanztest (auf Systemebene) durch.

Die Autoren behandeln im Buch sehr umfassend alle Aspekte des Integrationstests. Ich möchte hier keine Zusammenfassung des Inhalts geben, sondern hervorheben, dass die unterschiedlichen Stufen der Integration vom Integrationstest innerhalb einer Klasse bis zum Integrationstest von Systemen behandelt werden. Dabei steht immer die methodische Vorgehensweise zur Ableitung der Testfälle im Mittelpunkt der Betrachtung. Beantwortet wird die Frage: Wie können Testfälle zur Prüfung der Schnittstellen entworfen werden? Fallbeispiele belegen die praktische Relevanz der Verfahren, aber verdeutlichen ebenso mögliche Probleme.

Das Buch schließt die Lücke zwischen dem systematischem Komponententest und dem Systemtest. Es zeigt aber auch den steinigen Weg der Integration und dass nicht immer die Möglichkeit der Umsetzung der reinen Lehre in die Praxis vorhanden ist. Den Autoren danke ich für die sehr gute Darstellung der Problematik und die vielen praxisnahen Lösungsmöglichkeiten. Den Lesern wünsche ich, dass sie die vielen Anregungen im Testalltag umsetzen und damit die Qualität der Produkte steigern können. Für uns alle hoffe ich, dass auch bei zunehmender Komplexität der Systeme ein systematischer Test auf allen Stufen ausreichend genug durchgeführt wird, denn wir sind alle abhängig vom Funktionieren der Systeme.

Apropos komplexe Systeme: Bauen wir wirklich große, komplexe Systeme? Auch hier möchte ich auf eine schon ältere, aber immer noch gültige Aussage von M. A. Hennel zum Bau von großen Systemen zurückgreifen:

---

[1] http://mars.jpl.nasa.gov/msp98/news/mco990930.html (Juli 2012)

„Is it really true that developers intentionally wrote a big system? In other words, did the system not go through preliminary development stages during which it was in smaller parts that were subsequently put together, and into which, it could, in principle, be decomposed at any time? Certainly, if a system cannot be broken down into meaningful chunks, and if it were not systematically analysed or tested as it was being built, than the owners of that system have a major problem. The system really would be better used as a monument to arrogance.“[1]

Andreas Spillner

Bremen, im August 2012

[1] Hennell, M. A.: How To Avoid Systematic Software Testing, The Journal of Software Testing, Verification and Reliability, Vol.1, No. 1, 1991, S. 23 – 30

Vorwort

Das Wesen des Integrationstests war schon immer schwer zu fassen und zu beschreiben. Als Aktivität liegt der Integrationstest zwischen dem Test der einzelnen Software-Bausteine (dem Unit- bzw. Komponententest) und dem Test des Ganzen (dem Systemtest). Somit deckt er einen ziemlich großen Raum ab, einen Raum, der angesichts der zunehmenden Komplexität moderner IT-Systeme immer größer wird. Es gibt exponentiell viele Möglichkeiten, Software-Bausteine wie Komponenten und Services zusammenzusetzen, und fast ebenso viele Möglichkeiten, sie falsch zusammenzusetzen. Wer wartet, bis das Ganze steht, um Fehler beim Zusammenspiel der Bausteine zu finden, tut sich schwer, die gefundenen Fehler zu lokalisieren. Abgesehen davon ist es dann oft zu spät, sie zu reparieren, denn die falsche Zusammensetzung ist nur schwer rückgängig zu machen.

Daher wäre es erforderlich, jede Zusammenfügung zweier Bausteine gleich zu erproben, um eine sofortige Rückkopplung zu erhalten, auch wenn dies zunächst mehr Aufwand bedeutet. Dass dies in der IT-Projektpraxis oft nicht geschieht, spricht nicht dagegen. Die Projektleitung scheut sich vor dem zusätzlichen Aufwand. Es ist so leicht zu glauben, man brauche nur alle Teile zusammenzubinden, und irgendwie werden sie dann zusammenpassen. Das ist der vielzitierte „Big Bang"-Ansatz. In den meisten Fällen stellt es sich dann heraus, dass irgendetwas doch nicht passt, und es ist Aufgabe der Tester, die Ursache herauszufinden. Der dann zu betreibende Aufwand, um die nicht passenden Stellen zu lokalisieren, übertrifft bei Weitem den Aufwand, jede Zusammensetzung einzeln gezielt zu testen. Aber der Mensch ist Spieler und lässt es gerne darauf ankommen. Vielleicht hat man Glück und es passt doch alles zusammen. Dann spart man den großen Aufwand für den schrittweisen Integrationstest. Viele Entwickler meinen, der Big-Bang-Ansatz sei am Ende billiger, weil die Wahrscheinlichkeit, dass etwas nicht zusammenpasst, gering sei und man diese wenigen Fehler immer noch zurechtbiegen könne.

Leider fehlen empirische Daten, um zu belegen was wirklich billiger ist: es darauf ankommen zu lassen oder gleich jede Zusammenfügung zu testen. Weitaus schwerer wiegt jedoch die Tatsache, dass manches gravierende Zusammensetzungsproblem im Systemtest unbemerkt bleibt und erst in der Produktion auftaucht, z. B. wenn bei einer Datenübergabe Daten ab einer gewissen Menge abgeschnitten werden. Durch den Integrationstest werden solche Probleme gezielt angegangen, beim Systemtest dagegen gehen sie meistens in der Menge der zu testenden Eigenschaften unter. Deshalb ist es auf jeden Fall sicherer, jede Zusammenfügung zweier Software-Bausteine, ob Klassen, Komponenten, Teilsysteme oder Systeme, gleich zu testen.

In welcher Reihenfolge das zu tun ist, ist der Gegenstand zahlreicher Bücher und Artikel. Schon 1978 hat Glenford Myers verschiedene Integrationsstrategien vorgeschlagen, darunter Top-Down, Bottom-Up, Inside-Out, Outside-In und Kreuzweise. Jede Strategie hat Vor- und Nachteile. Welche letztendlich gewählt wird, hängt von der Architektur und der Umgebung des Systems sowie von der Art der Applikation und der verfügbaren Zeit ab. Seitdem haben viele Autoren auf dem Buch von Myers aufgebaut und neue Einsichten zugefügt. Im deutschsprachigen Raum haben sich besonders Spillner und Liggesmeyer mit ihren Veröffentlichungen hervorgetan. Jedenfalls sind zahlreiche Beiträge zum Thema Integrationstest in der Literatur vorhanden. Warum dann dieses Buch? Erstens, weil die bisherigen Beiträge lediglich einzelne Integrationsstufen behandeln, meistens die unteren. Zweitens, weil der Integrationstest im Vergleich zu den anderen Teststufen gewöhnlich eher stiefmütterlich besprochen wird. Und drittens, weil kein vorhandenes Buch zum Test die Auswirkungen neuerer Technologien wie z. B. der dienstorientierten Architektur (SOA) auf den Integrationstest behandelt.

Außerdem fehlt ein umfassendes, aktuelles Werk, das alle Stufen der Software-Integration – von der Integration einzelner Funktionen und Klassen bzw. Modulen bis hin zur Enterprise-Applikations-Integration – abdeckt. Da so viele Schichten mit unterschiedlichen Implementierungstechnologien dazwischen liegen, ist dies kein leichtes Unterfangen. Mit diesem Buch stellen sich die Autoren dieser Herausforderung. Das Buch geht auf alle Integrationsstufen ein, von der Integration der Member-Funktionen bzw. -Variablen einzelner Klassen bzw. Module bis hin zur Systemintegration ganzer Unternehmensanwendungen z. B. in einer Cloud. Jede Integrationsstufe wird ausführlich behandelt und mit Beispielen aus der Praxis illustriert. Neben den Integrationsstufen, -strategien und -methoden werden unterstützende Techniken und der Aufbau der Testumgebung besprochen. Die theoretischen Grundlagen der Software-Integration werden anhand der Graphentheorie dargelegt. Auch die Fehlerarten, die durch die Interaktion diverser Software-Bauteile zustande kommen, werden behandelt. Diese zu kennen, ist für einen gezielten Integrationstest unerlässlich. Schließlich wird die Rolle der Automation beim Integrationstest behandelt: Welche Arten von Testwerkzeugen werden gebraucht, um den Integrationstest zu unterstützen?

Das nun endlich vorliegende Buch ist somit ausschließlich dem Integrationstest gewidmet. Da dieser bislang nur in wissenschaftlichen Einzelarbeiten betrachtet wurde, bedingten die vielen Diskussionen zu Inhalt und Aufbau des Buches der Integration und vielen Integrationstests auch der einzelnen Kapitel. Letztendlich war das Buchprojekt auch für uns Autoren, die wir aus vier verschiedenen Herkunftsländern und beruflichen Umfeldern stammen, eine Art Integrationstest. Daher bitten wir um Verständnis, wenn der Aufbau des Buches zu wünschen übrig lässt, Einzeldarstellungen nicht ausreichend gewürdigt wurden oder einige unserer Erläuterungen zu allgemein oder zu speziell sind. Besonders freuen würden wir uns über konstruktive Kritik und insbesondere über Ihre Fragen, die uns helfen, unsere Gedanken und die vielen Einzelquellen zum Integrationstest noch verständlicher darzustellen und zu integrieren.

Wir hoffen, mit diesem Buch eine Lücke in der Testliteratur zu schließen. Ob es uns gelungen ist, dieses breite Feld wirklich ausreichend abzudecken, muss letztendlich der Leser entscheiden.

Mario Winter, Mohsen Ekssir-Monfared, Harry M. Sneed, Richard Seidl, Lars Borner

Danksagung

Zuallererst danken wir Andreas Spillner, der mit seiner Dissertationsschrift zum Integrationstest sowie seinen vielen weiteren Beiträgen zu Softwaretest und Qualitätssicherung das Feld bereits seit über zwanzig Jahren bestellt und gepflegt und nun unser Buch mit seinem Geleitwort dort bestens eingebettet hat. Ganz besonderer Dank gebührt unseren Reviewern, insbesondere Holger Hanisch, dem wir unzählige Präzisierungen und Ergänzungen verdanken, sowie Tilo Linz (Präsident GTB e. V.), der trotz eigenem Buchprojekt die Zeit und viele Verbesserungsvorschläge für unser Manuskript fand. Großen Dank natürlich auch an die „Probeleser" Stephan Weißleder und Ina Schieferdecker.

Ebenso danken wir der Software Data Services GmbH, einer Tochtergesellschaft der T-Systems Österreich, für ihren Beitrag über den Integrationstest des GEOS Banking Systems. Ohne solche Beiträge von der Industrie ist es nicht möglich, die Testpraxis wirklichkeitsnah zu schildern. Darum unser besonderer Dank für Ihre Bereitschaft, die Erfahrungen mit unseren Lesern zu teilen.

Dem Hanser Verlag, in persona Frau Margarete Metzger, danken wir insbesondere dafür, dass sie unser Projekt über all die Jahre immer wieder befürwortet und unterstützt hat. Dank auch an Frau Brigitte Bauer-Schiewek und Frau Irene Weilhart, die uns in der letzten Phase der Buchentstehung zur Seite standen, und Herrn Jürgen Dubau, der im Copy-Editing noch viele Fehler ausbügelte, die uns entgangen waren.

Persönlicher Dank von Mario Winter

Meinen Co-Autoren und dem Hanser Verlag danke ich für ihr Verständnis und die große Geduld während der langen Unterbrechung des Buchprojekts aus familiären Gründen. Mein persönlicher Dank gilt auch der Fachhochschule Köln, Fakultät 10, Informatik und Ingenieurwissenschaften, sowie den Kolleginnen und Kollegen des Instituts für Informatik, welche die Fertigstellung des Buches durch Gewährung und Unterstützung meines Forschungsfreisemesters ermöglichten.

Orpierre, Frankreich, August 2012

Die Autoren

Mario Winter

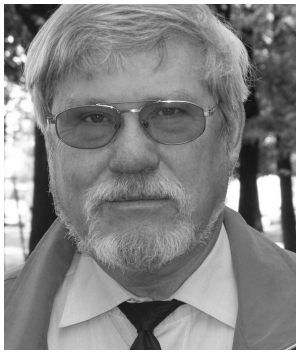
Mario Winter ist seit 1983 in der IT tätig, zunächst als Software-Entwickler, dann in der Leitung kommerzieller und forschungsbezogener Software-Projekte. Ab 1994 konzentrierte er sich zunehmend auf den Bereich der Software-Qualitätssicherung und promovierte 1999 auf diesem Gebiet. Seit 2002 ist er Professor am Institut für Informatik der Fachhochschule Köln und dort Mitglied des Forschungsschwerpunkts „Software-Qualität". Er ist Gründungsmitglied des German Testing Board e. V. und war von 2003 bis 2011 Sprecher der Fachgruppe „Test, Analyse und Verifikation von Software" (TAV) der Gesellschaft für Informatik e. V. (GI). Seine Lehr- und Forschungsschwerpunkte sind Software-Entwicklung und Projektmanagement, insbesondere die modellbasierte Entwicklung und Qualitätssicherung von Software. Er ist Autor und Mitautor zahlreicher Publikationen im Bereich Software-Entwicklung und Software-Tests.

Mohsen Ekssir-Monfared

Mohsen Ekssir-Monfared ist Bereichsleiter Software-Test und Qualitätssicherung bei BDC EDV-Consulting. Besonderes Augenmerk gilt bei seiner Arbeit der Entwicklung und dem Einsatz der Test Services wie Testkonzeption, -metrik, -automatisierung, -management und Testprozess-Assessment. Die Zertifizierung zum ISTQB® Certified Tester, Full Advanced Level und Quality Assurance Management Professional (QAMP) erlangte er 2010. Außerdem ist er seit 2012 iNTACS™ Certified ISO/IEC 15504 Provisional Assessor TestSPICE.

Er leitet seit 2010 die regionale ASQF-Fachgruppe Software-Test Österreich und unterrichtet zusätzlich an der Fachhochschule Wiener Neustadt Software Qualitätsmanagement. Er war nach mehreren Jahren Tätigkeit als Test Manager und Senior Systems Integrator bei Siemens AG vom Oktober 2008 bis Juli 2012 bei ANECON Software Design und Beratung tätig.

Harry M. Sneed



Harry M. Sneed arbeitet seit 2003 als Tester für die Firma ANECON Software Design und Beratung GmbH. Unter anderem wirkte er dort in einigen großen Testprojekten mit und war Qualitätssicherungsbeauftragter bei einer Bundesbehörde in Deutschland. Neben seiner Projektarbeit entwickelt er Testwerkzeuge für die Unterstützung einzelner Testaktivitäten wie die Analyse der Anforderungen, die Gewinnung von Testfällen, die Generierung und Validierung von Testdaten, den Test von Web Services und die Dokumentation und Messung der Testdurchläufe.

Er hat zusätzlich Lehraufträge für Software-Engineering an der Universität Regensburg, für Software-Test an der Universität Koblenz und für Software-Evolution an der Fachhochschule Hagenberg in Oberösterreich. Außerdem lehrt er als Gastdozent für Software-Metrik an der Universität Szeged in Ungarn.

Harry M. Sneed gehört zu den Pionieren der Software-Testtechnologie, wurde 1996 von der IEEE ausgezeichnet und ist seit 2005 GI-Fellow der Deutschen Gesellschaft für Informatik. 2009 erhielt er von der IEEE den Stevens Award für seine Errungenschaften auf dem Gebiet der Software-Reverse- und Reengineering.

Als Autor hat Sneed über 400 Fachartikel in deutscher und englischer Sprache veröffentlicht und 19 Bücher verfasst.

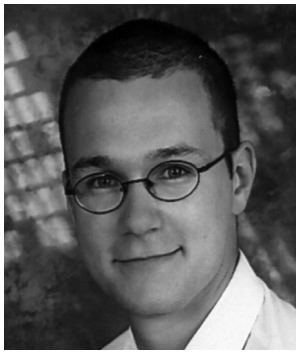
Richard Seidl

Richard Seidl ist Testmanager und Testspezialist bei GETEMED Medizin- und Informationstechnik AG. Die Konzeption und Implementierung von strukturierten und automatisierten Software-Tests bilden den Schwerpunkt seiner Arbeit.

Von 2005 bis 2010 war Richard Seidl als Testmanager bei ANECON Software Design und Beratung GmbH tätig. Die Zertifizierung zum ISTQB® Certified Tester (Full Advanced Level) schloss er Mitte 2006 ab. In diesem Bereich arbeitet er seit 2008 als Trainer. Ende 2007 erlangte er durch die Ausbildung zum IREB Certified Professional for Requirements Engineering die Zertifizierung zum Quality Assurance Management Professional (QAMP).

Gemeinsam mit Harry Sneed und Manfred Baumgartner veröffentlichte Richard Seidl die Fachbücher „Der Systemtest – Anforderungsbasiertes Testen von Software Systemen“ und „Software in Zahlen – Die Vermessung von Applikationen“ und ist auf internationalen Konferenzen mittlerweile ein gefragter Redner.

Lars Borner



Nach seinem Abschluss als Diplom-Informatiker arbeitete Lars Borner als wissenschaftlicher Angestellter am Lehrstuhl für Software-Systeme an der Universität in Heidelberg. In dieser Zeit unterstützte er die Ausbildung der Studenten auf dem Gebiet des Software-Engineerings mit dem Schwerpunkt auf Anforderungsmanagement und Qualitätssicherung. Während der Zeit an der Universität verfasste er seine Promotionsarbeit mit dem Titel „Integrationstest – Testprozess, Testfokus und Integrationsreihenfolge“.

Seit Mai 2010 ist er bei der Datev eG in Nürnberg beschäftigt und unterstützt die dortigen Software-Entwicklungsprojekte hinsichtlich Software-Test- und Software-Entwicklungsprozess, Testmethodik und Werkzeugeinsatz.