Nutzung von Architekturinformationen zur Beherrschung der Komplexität im Software-Integrationstest

Frank Elberzhager, Matthias Naab
Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern
{frank.elberzhager, matthias.naab}@iese.fraunhofer.de

Zusammenfassung:
Architekturdefinition und Integrationstesten sind wesentliche Aktivitäten in der Softwareentwicklung. Häufig wird empfohlen, Architekturinformationen als Grundlage für den Integrationstest zu nutzen, wobei oftmals jedoch keine Unterstützung bei der konkreten Umsetzung bereitgestellt wird. Wir wollen in diesem Artikel Konzepte für die stärkere Nutzung von konkretem Architekturwissen im Integrationstest vorstellen und diskutieren, wie der Integrationstest profitieren kann. Damit soll letztlich die zugrundeliegende Software verbessert werden, sowie die oftmals hohe Komplexität bei der Erstellung und Qualitätssicherung von Software besser handhabbar sein.
Schlüsselworte: Architektur, Integrationstest, Integrationsbaum

1. Einleitung und Motivation

Software wird mehr und mehr zum Teil unseres täglichen Lebens, beispielsweise in Smartphones, in Smart Homes oder im Automobil. Im Businessumfeld wird Software zum beherrschenden Treiber für Innovation. Dabei wachsen zunehmend unterschiedliche Bereiche zusammen, insbesondere eingebettete und Informationssysteme sowie mobile Geräte, welche dann häufig als smart ecosystems oder system-of-systems betrachtet werden.

Durch die zunehmende Integration unterschiedlicher Komponenten, Geräte und Software steigt gleichzeitig die Komplexität in der Entwicklung solcher Systeme. Die Softwareentwicklung hat sich in den vergangenen Jahrzehnten deutlich weiter entwickelt und an Reife gewonnen. Unabhängig von konkreten Entwicklungsmodellen – wie wasserfallartiger oder agiler Entwicklung – sind weiterhin typische Aktivitäten wie die Anforderungserhebung, Design, Codierung sowie Qualitätssicherung zu finden. Konkrete Ziele an die Softwareentwicklung sind in den letzten Jahren jedoch deutlich herausfordernder geworden, beispielsweise durch den Anspruch einer hohen Qualität der Software, die mit einer geringen time-to-market ausgeliefert werden soll, und dabei effizient entwickelt werden muss, bei zunehmender Komplexität der zu entwickelnden Software.

Qualitätssicherung zählt zu den wesentlichen Disziplinen während der Softwareentwicklung. Generell stellt man aber häufig fest, dass entweder die Qualität ausgelieferter Software oder Apps unzureichend ist, der Aufwand für die Qualitätssicherung zu hoch ist, oder beides zutrifft. Der Integrationstest, der wie zu Beginn motiviert, eine zunehmende Rolle bei immer stärker integrativen Systemen spielt, ist in der Literatur eher stiefmütterlich behandelt. Auch in der Praxis trifft man selten auf eine dedizierte Integrationstestphase, häufig wird der Integrationstest zusammen mit dem Unittest oder dem Systemtest durchgeführt, ohne jedoch den eigentlichen Zweck des Integrationstest, nämlich die explizite Prüfung von Schnittstellen bzw. der Kommunikation über Schnittstellen, explizit zu adressieren.

Um den genannten Herausforderungen begegnen zu können und die Komplexität in der Entwicklungskette zu reduzieren, möchten wir hier den Fokus auf eine bessere Steuerung und Durchführung des Integrationstests basierend auf Architekturinformationen legen. Die Architekturdefinition zerlegt ein System, welches in der Integration wiederum zusammengesetzt wird; der Integrationstest prüft dabei das Zusammenspiel der einzelnen Teile, wozu Architekturinformationen helfen sollen.

Der Integrationstest ist eine der wesentlichen Aktivitäten in der Qualitätssicherung, welcher die Architekturdokumentation als Testbasis nutzen kann (also als Grundlage für die Testfallableitung). Allerdings stellt man ebenfalls fest, dass häufig nur wenig methodische Unterstützung für die konkrete Durchführung eines Integrationstests in der Literatur angeboten wird. Aus diesem Grund möchten wir in diesem Beitrag beleuchten, wie Architekturinformationen stärker genutzt werden können, um den Integrationstest besser zu planen, durchzuführen und zu steuern. Langfristiges Ziel dabei ist eine bessere Integration der Architekturdefinitions- und Integrationstestphase, eine höhere zu erwartende Qualität der resultierenden Software, ein stärkerer Automatisierungsgrad sowie eine bessere Beherrschung der Komplexität. Insbesondere für aktuelle Themen wie continuous integration oder die bereits genannten smart ecosystems kann damit ein substantieller Beitrag für die zuvor genannten Ziele geliefert werden. Nachfolgend

werden Grundlagen für dessen Erreichung dargelegt und diskutiert.

2. Hintergrund

Architektur und die Integrationstest sind feste Bestandteile in der Softwareentwicklung. Architektur umfasst dabei eine Strukturbeschreibung, welche die Menge aller Systeme und Subsysteme, Komponenten und Schnittstellen beschreibt. Der Integrationstest auf der anderen Seite hat die Hauptaufgabe, diese Schnittstellen sowie die Kommunikation zwischen Komponenten zu prüfen. Die Architektur wird dabei häufig als Testbasis, also als Grundlage für die Ableitung von Testfällen, empfohlen.

Jedoch ist eine konkrete Unterstützung, wie Testfälle basierend auf Architekturinformationen abgeleitet, sowie eine Teststrategie gefunden werden kann, selten gegeben. Typische Testliteratur (z.B. [1-3]) nennt zwar den Integrationstest, behandelt diesen aber sehr kurz. Es wird in der Regel auf hierarchische Strategien (wie top-down Integration) oder den big-bang Ansatz eingegangen, teilweise werden zudem prozedurale und objektorientierte Besonderheiten für den Integrationstest beleuchtet. Lediglich Winter et al. [4] bietet umfangreichere Informationen zum Integrationstest, welche etabliertes Wissen zusammenfassen. Auch in der Forschung wird die engere Verzahnung von Architektur und dem Integrationstest untersucht (z.B. vom SEI[1]), die Ergebnisse sind jedoch als wenig ausgereift einzustufen.

Somit bleibt festzuhalten, dass die Wissenschaft und Praxis den Integrationstest als wesentliche Säule während der Qualitätssicherung ansieht, jedoch wenig konkrete Unterstützung für die Umsetzung verfügbar ist.

3. Verzahnung von Architektur und Integrationstest

Zunächst wollen wir auf die relevanten Rollen eingehen, den Integrationsbaum als Hilfsmittel zur Unterstützung einführen sowie die Zusammenhänge zwischen Architekturinformationen und dem Integrationstest darlegen sowie auf mögliche Integrationsstrategien eingehen. Im Anschluss diskutieren wir das dargelegte.

3.1 Rollen

In den Aktivitäten Architekturdefinition sowie Integrationstest sind zunächst die Rolle des Architekten sowie des Testers (wir gehen im Folgenden immer davon aus, dass es sich um den Tester für den Integrationstest handelt) relevant.

Der Tester sollte frühzeitig vom Architekten, welcher die Architektur definiert und verfeinert bzw. zerlegt, eingebunden werden. Der Tester erhält somit frühzeitig Informationen über die Architektur und kann diese für die Ableitung der Integrationsteststrategie nutzen; zusätzlich kann der Tester frühzeitig Feedback hinsichtlich der Testbarkeit geben, welche ggf. zu angepassten Architekturentscheidungen führt.

Darüber hinaus sollte auch ein Projektleiter mit eingespannt werden, um die Ressourcen (z.B. verfügbare Mitarbeiter, benötigte Infrastruktur) und zeitliche Aspekte abstimmen zu können.

Essentiell ist, dass in diesem Schritt erfahrene Personen die Rollen ausfüllen.

3.2 Integrationsbaum

Um eine Übersicht von Architekturkomponenten zu erhalten, sowie eine Zuordnung zu Teams, welche Komponenten entwickeln, darzustellen, führen wir den sogenannten Integrationsbaum ein. Dieser ist initial durch den Tester zu erstellen, basierend auf dem Input eines Architekten bzw. einer Architekturdokumentation. Für jedes neue Release wird die Aktualität geprüft und der Baum ggf. angepasst.

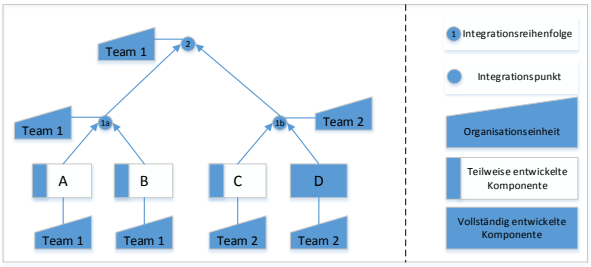

Abbildung 1: Integrationsbaum

Abbildung 1 zeigt einen einfachen Integrationsbaum, welcher aus 4 Komponenten besteht. Die Blätter beschreiben die zu entwickelnden Komponenten, die Knoten die Integration von Komponenten. Zwei Teams sind den Komponenten zugeordnet. Komponenten sind entweder vollständig entwickelt, oder auch nur partiell, jedoch soweit, dass sie integrationsfähig sind (beispielsweise klare Definition und Implementierung der Interfaces). Eine Reihenfolge zur Integration kann ebenfalls festgehalten werden, im Beispiel werden zunächst die beiden Teilbäume am Integrationspunkt 1a bzw. 1b integriert, bevor diese in einem zweiten Schritt an der Wurzel zusammen geführt werden.

3.3 Architekturelemente und ihre Bedeutung im Integrationstest

Der Integrationsbaum beschreibt lediglich Komponenten auf der Blattebene, welche jedoch unterschiedliche Architekturkonzepte abbilden können (bzw. eine Komponente kann wiederum selber ein Baum sein).

[1] https://insights.sei.cmu.edu/sei_blog/2011/08/improving-testing-outcomes-through-software-architecture.html

Zunächst kann eine Komponente eine konkrete Codeklasse oder bereits ein Zusammenschluss aus Codeklassen darstellen, die eng miteinander interagieren. Weiterhin kann es Konnektoren geben, die unterschiedliche Komponenten miteinander verbinden und dadurch Kommunikation explizit unterstützen (beispielsweise ein Enterprise Service BUS); Konnektoren können also selbst als Komponente modelliert und umgesetzt werden, so dass sie entsprechend im Integrationstest geprüft werden müssen. Häufig werden zur besseren Strukturierung einer Architektur Cluster genutzt, die zum Beispiel eine Funktionalität zusammenfassen. In diesem Fall ist es denkbar, dass Komponenten aus unterschiedlichen hierarchischen Systemebenen involviert sind. Weiterhin gibt es häufig unterschiedliche Schichten in einer Architektur, die über explizite Kanäle miteinander kommunizieren. Kommunikation sollte dabei ausschließlich über klar definierte Interfaces erlaubt sein. Ausnahmen davon mag es geben, erschweren aber unter Umständen erheblich den Integrationstest und bilden ein hohes Risiko für nicht entdeckte Fehler.

Neben den hier beschriebenen Elementen kann es weitere geben, die hinsichtlich der Berücksichtigung in einem Integrationstest analysiert werden müssen.

3.4 Ableitung einer Integrationsteststrategie

Grundsätzlich gibt es unterschiedliche Faktoren, die bei der Ableitung einer Integrationsstrategie, d.h. bei der Festlegung der Reihenfolge bei der Integration der Komponenten, unterstützen kann.

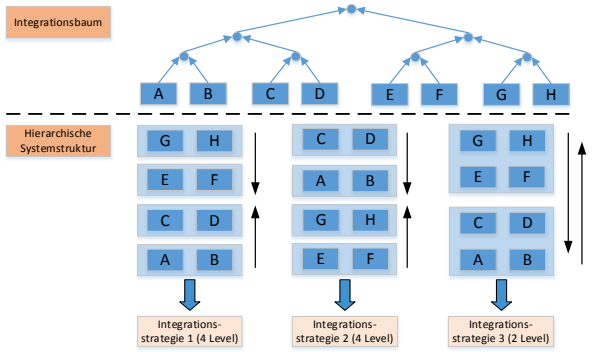

Abbildung 2: Beispiele für Integrationsstrategien

Zunächst kann der in Abschnitt 3.2 eingeführte Integrationsbaum genutzt werden. Vereinfacht gehen wir davon aus, dass nur ein Team zur Entwicklung und Integration vorliegt. Abbildung 2 zeigt einen Integrationsbaum mit 8 Komponenten, sowie drei unterschiedliche hierarchische Strukturen. Bei Strategie 1 und 2 sind jeweils 4 Schichten mit einer unterschiedlichen Anordnung zu sehen, bei Strategie 3 erkennt man ein System bestehend aus 2 Schichten (z.B. ein Front- und ein Backend). Beispielsweise könnte Strategie 1 dabei zunächst die Komponenten AB, CD, GH und EF integrieren, danach ABCD und EFGH, und danach die beiden Teilbäume (inkl. Testaktivitäten bei der jeweiligen Integration). Strategie 1 und 2 entsprechen einer outside-in Sandwich Strategie, Strategie 3 könnte bottom-up oder top-down durchgeführt werden (je nach Teilintegration der Komponenten innerhalb der beiden Schichten, worüber keine Informationen direkt dargestellt sind, siehe auch die Pfeile an den Seiten). Die konkrete Auswahl der Integrationsteststrategie kann wiederum von weiteren Faktoren abhängen, beispielsweise der Entwicklungsreihenfolge einzelner Komponenten, verfügbarer Ressourcen, oder des funktionalen Schnitts. In der Regel sollte die Festlegung der Integrationsstrategie zusammen mit der Teststrategie passieren. Es gibt also unterschiedliche Möglichkeiten bei der konkreten Ableitung einer Strategie. Allerdings wäre eine inside-out Sandwich Strategie hier weniger naheliegend, da die Komponenten CDEF zunächst nicht integriert werden.

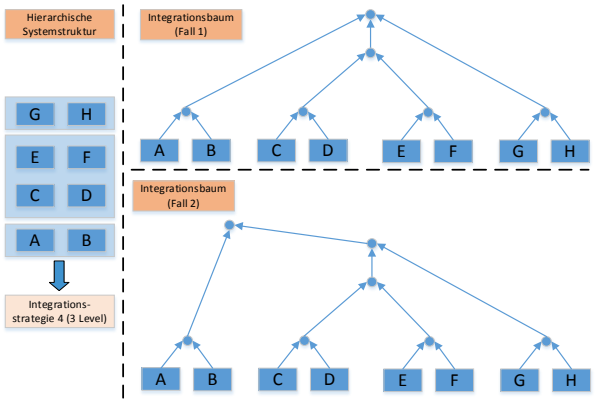

Abbildung 3: Verschiedene Integrationsbäume für eine Integrationsstrategie bei hierarchischen Strukturen

Abbildung 3 zeigt diese Möglichkeit auf und gibt ein Beispiel, wie unterschiedliche Integrationsbäume bei der Ableitung einer Integrationsstrategie genutzt werden können bzw. wie dieselbe Systemstruktur zu unterschiedlichen Strategien führen kann. Strategie 4, welche auf einer Hierarchie von drei Schichten basiert, wäre so nicht direkt mit dem Integrationsbaum wie zuvor in Abbildung 2 vereinbar, wohl aber mit den beiden unterschiedlichen Integrationsbäumen aus Abbildung 3. Fall 1 würde auch die bereits zuvor erwähnte Sandwichstrategie ermöglichen (allerdings eher einem inside-out oder einem „big-bang“ der drei Schichten), während Fall 2 eine top-down Strategie nahelegt.

Natürlich können nicht nur Informationen über hierarchisch aufgebaute Architekturen zur Ableitung einer Integrationsstrategie genutzt werden. Eine weitere verbreitete Möglichkeit für die Durchführung eines Integrationstest ist die Nutzung von Anwendungsfällen, wie in Abbildung 4 dargestellt. Hier wurden drei Anwendungsfälle (Use cases) für die Integration der Systemkomponenten genutzt, zunächst Use-case 1, welcher die

Komponenten A-D integriert, danach Use-case 2, welcher zusätzlich Komponenten E und F benötigt und integriert, zuletzt die restlichen Komponenten durch Use-case 3; der Integrationstest findet jeweils bei der Integration statt.

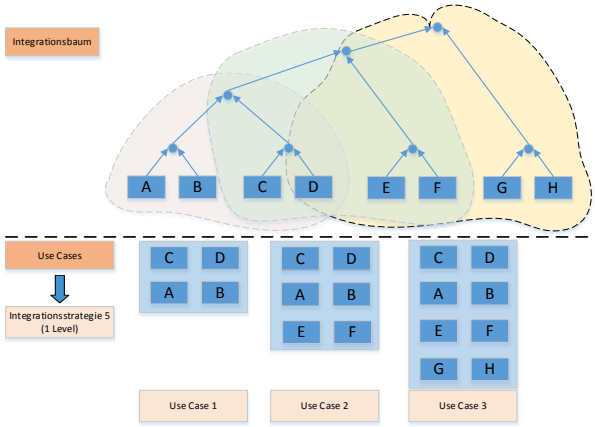

Abbildung 4: Use-case basierte Integrationsstrategie

Bei allen Strategien werden in der Regel Dummies benötigt (Stubs und Driver), diese sollten jedoch so gering wie möglich gehalten werden. Zwischen dem Integrationsbaum und der ausgewählten Integrationsstrategie gibt es keine strikte 1:1 Abbildung, diese sollte aber so intuitiv wie möglich sein. Sie liefert aber eine erhebliche Unterstützung bei der Ableitung der Integrationsteststrategie durch den einfachen Überblick aller relevanter Architekturkomponenten sowie möglicher Teamstrukturen, die es zu berücksichtigen gilt.

Neben Strukturinformationen bei der Ableitung einer Integrationsteststrategie sowie dem zugehörigen Prüfen gibt es weitere Faktoren, die eine Rolle spielen können. Dazu gehören u.a. Risiko, Datenflüsse sowie Datenkomplexität, die Topologie oder die funktionale Dekomposition. Entscheidend für die Ableitung der Strategie durch den Tester ist eine frühzeitige Berücksichtigung relevanter Faktoren mit Hilfe erfahrener Architekten bzw. ausreichend detaillierter Architekturdokumente.

3.5 Diskussion

Der eingeführte Integrationsbaum ist als Unterstützung zur Planung des Integrationstests zu verstehen. Er stellt auf kompakte Art und Weise alle relevanten Komponenten dar, so dass ein Integrationstester zunächst nicht eine Vielzahl von Architekturdiagrammen durchsehen muss (sofern überhaupt eine dokumentierte Architektur existiert). Somit wird auch eine frühe Auseinandersetzung mit dem Integrationstest unterstützt. Je größer die Architektur eines Systems wird, und je mehr Teams, die Komponenten entwickeln, koordiniert werden müssen, desto stärker spielt diese Darstellung ihre Vorteile aus. Jedoch gilt es zu berücksichtigen, dass es keine strikte 1:1 Abbildung zwischen dem Integrationsbaum und der Integrationsstrategie bzw. der Teststrategie gibt.

Der Integrationsbaum selber kann weiterhin dazu verwendet werden zu entscheiden, ob für ein Release nur Teile (also Teilbäume) oder der komplette Integrationsbaum „abgearbeitet“ werden soll. Darüber hinaus kann er unterstützend wirken bei Entscheidungen, welche Bereiche automatisiert geprüft werden sollen und in welcher Reihenfolge dies geschehen soll. Obwohl der Integrationsbaum eine wesentliche Unterstützung bieten kann, so ist der Freiheitsgrad für die Ableitung einer konkreten Integrationsstrategie immer noch recht hoch, wodurch auch eine Kostenabschätzung für den Integrationstest schwierig bleibt. Beides hängt, neben der Systemstruktur, von weiteren Faktoren ab, wie beispielsweise den verfügbaren Ressourcen und Zeit, Kritikalität von Komponenten oder auch der Qualität der Testbasis.

Hinsichtlich der relevanten Rollen sind frühzeitig der Tester und der Architekt zusammen zu bringen, unterstützt durch einen Projektleiter, der die Ressourcen verwaltet und ebenfalls Einfluss auf entsprechende Strategien haben kann. Obwohl wir empfehlen, wichtige Architekturentscheidungen zu dokumentieren, ersetzt eine solche Dokumentation nicht das Gespräch zwischen diesen Rollen. Der Tester bekommt frühzeitig Informationen über die Architektur und kann diese bei der Ableitung einer Teststrategie berücksichtigen bzw. zunächst beim Aufstellen des Integrationsbaums, jedoch auch Rückmeldung hinsichtlich der generellen Testbarkeit beisteuern, die frühzeitig vom Architekten genutzt wird.

4. Ausblick

Wir haben aufgezeigt, dass mit Hilfe von Architekturinformationen ein Integrationsbaum abgeleitet werden kann, der einen konkreten Integrationstest unterstützt. Nachdem wir in diesem Artikel grundsätzliche Konzepte eingeführt und beleuchtet haben, sind wir zukünftig daran interessiert, explizit Anleitung und Regeln vorzugeben, wie eine konkrete Integrationsstrategie abgeleitet werden kann. Diese soll dann in unterschiedlichen Umgebungen erprobt und schließlich weiter verbessert werden. Damit wollen wir zu einer stärkeren Berücksichtigung des Integrationstests und letztlich zu einer verbesserten Qualität von Softwareprodukten zu vertretbaren Kosten beitragen.

Referenzen

[1] P. Liggesmeyer, Software Qualität, Spektrum Akademischer Verlag, 2009

[2] I. Burnstein, Practical Software Testing, Springer, 2003

[3] A. Spillner, T. Linz, Basiswissen Softwaretest, dPunkt Verlag, 2012

[4] M. Winter, M. Ekssir-Monfared, H. Sneed, R. Seidl, L. Borner, Der Integrationstest, Carl Hanser Verlag, 2013