

Jenseits des Tellerrands – Teil 7: Am Softwareentwickler-Stammtisch

Technology First

Softwareentwicklung ist weit mehr als Programmierung und Programmierung ist weit mehr als das bloße Beherrschen von Programmiersprachen und Werkzeugen. Die Artikel dieser Serie greifen wichtige Themen, aktuelle Fragen und auch grundsätzliche Probleme auf, beleuchten diese und stellen „die gängigen Antworten“ immer wieder in Frage. Subjektivität ist dabei keineswegs verpönt, sondern das notwendige Salz in der technischen Einheitsuppe.

Nach dem Ausflug in die Welt des Krimis, gibt es diesmal eine Reise in die Selbstorganisation einer Gruppe. Ausgangspunkt für die folgende Aufzeichnung einer Instant-Messaging-Konferenz war die Frage: Warum setzen wir all die guten und Erfolg versprechenden Praktiken der Softwareentwicklung – von Unit-Tests bis Design Patterns – nur sehr sporadisch um? Tatsächlich wurden jedoch im Laufe der anderthalbstündigen Diskussion eine ganze Reihe (hoffentlich) interessanter Themen gestreift. Teilnehmer am Experiment (in alphabetischer Reihenfolge) waren:

- Achim Bangert (AOK-Systems)
- Dierk König (Canoo)
- Eberhard Wolff (Saxonia)
- Henning Wolf (it-agile)
- Johannes Link (andrena objects)
- Martin Lippert (it-agile)
- Stefan Rook (it-agile)

Das Protokoll wurde nachträglich gemeinsam manipuliert und sortiert, dennoch stellt das Skript an manchen Stellen ein Puzzle dar, dessen Beiträge der Leser in eine logische Reihenfolge bringen muss.

Johannes Link
(john.link@gmx.net)



Johannes Link: Schön, dass ihr alle „anwesend“ seid. Wenn ich mir die Welt der Softwareentwicklung betrachte, könnte ich manchmal verzweifeln: Wir haben in den letzten Jahren scheinbar so viel über besseres Vorgehen gelernt, z.B. über den Nutzen von Design Patterns und Unit-Tests, und dennoch wursteln viele Teams und IT-Firmen weiter, als wäre nichts gewesen: neue Technologien – ja; neues Vorgehen – nein. Beobachtet ihr dieses Phänomen auch und wenn ja, was sind denn für euch die Gründe?

Eberhard Wolff: Ich glaube, ein Problem ist, dass es eben eine Technologie-Branche ist, in der wir arbeiten ... Zumindest ist das die Wahrnehmung. Daher schaut man eben auf Technologien und nicht so sehr auf die Prinzipien.

Achim Bangert: Johannes, du sprichst von „Wir haben gelernt ...“. Werden die Praktiken denn auch von Organisationen, Teams, Personen nicht befolgt, die sie kennen und daran glauben?

Stefan Rook: „Kennen“ bedeutet ja nicht „können“.

Eberhard: ... und vor allem nicht, daran zu glauben.

Johannes: Vielleicht ist das ja das allererste Problem: Glauben denn so viele an die Dinge, die wir für Lösungen halten?

Henning Wolf: Steckt die Antwort nicht schon in der Frage? Solange wir uns dauernd mit neuen Technologien beschäftigen müssen, kommen wir zu nichts anderem. Ich glaube eben nur, dass wir uns gar nicht so viel mit (neuen) Technologien beschäftigen sollten.

Martin Lippert: Viele Kollegen, die ich in Projekten kennen gelernt habe, finden Technologien einfach spannender und es interessanter, mit neuen Sachen Dinge auszuprobieren und zum Laufen zu bekommen. Das echte Problem des Kunden zu lösen oder „einfach nur“ qualitativ hochwertige Software abzuliefern (ist für mich was anderes, als „zum Laufen zu bringen und mal auszuprobieren“) ist da eben nicht so spannend, sondern lästige Arbeit. Manchmal ertappe ich mich auch selbst dabei.

Henning: Hauptsache, die Stimmung ist gut?

Eberhard: :)

Stefan: Gute Stimmung unter den Entwicklern ist aber nicht das, was der Kunde bestellt hat.

Jenseits des Tellerrands

Was bisher geschah und Sie zukünftig erwartet

- Teil 1: Das nächste Java?
- Teil 2: Typisierung in Java und darüber hinaus
- Teil 3: Warum gibt es eigentlich Softwareentwicklungsprojekte?
- Teil 4: Softwarearchitektur: eine kritische Bestandsaufnahme
- Teil 5: Textuelle domänenspezifische Sprachen
- Teil 6: Moderne System- und Abnahmetests
- Teil 7: Warum nutzen wir die Mittel zur Softwareverbesserung nicht, obwohl wir sie kennen? Eine Diskussion
- Teil 8: Java-Bashing: Unsere liebste Plattform ist zu kompliziert geworden
- Teil 9: Rich Client vs. Web Client
- Teil 10: Wie viel Abstraktion brauchen wir?
- Teil 11: Berufsethos, Qualität, Rollenverständnis
- Teil 12: Time's Arrow: ein Projekt von hinten

Johannes: Es ist also ein „People Problem“. Wie alles. [1]

Henning: Klar, Entwickler stehen tendenziell eher auf Technologien, Kunden sind schwierig zu verstehen, Manager denken nur an Zeit und Geld. ;)

Achim: Nun ja, vielleicht auch ein Prozessproblem; wenn die Zusammenarbeit mit den Kunden hinreichend schwierig ist, schafft man lieber erst mal ein Framework, das die konkreten Kundenprobleme als einfache Spezialisierung enthält. ;)

Stefan: Ist also die Antwort auf die Frage, warum wir schlechter sind, als wir sein könnten: Weil wir es so wollen?

Achim: Weil vielleicht die Kommunikation mit den Kunden aufgrund der Organisation, dem Vorgehensmodell etc. frustrierend ist?

Eberhard: Es kann aber auch bei Technologien passieren, dass die Bedingungen so sind, dass man keine Lust hat, zu programmieren.

Johannes: Da kämpfen aber nur wir mit unserem Computer. Das ist vergleichsweise einfach.

Vom Testen zur Projektorganisation

Eberhard: Wir haben ja einige Test-Gurus unter uns. Glaubt ihr eigentlich an das „Test Infected – Programmers love writing tests“ [Anm.: Der erste Artikel über JUnit hatte diesen Titel]? Ist JUnit bzw. Test-Driven Development (TDD) etwas, das sich selbst verkauft? Und wenn ja, was unterscheidet es von anderen Vorgehensweisen?

Johannes: Was ich in den Projekten erlebe, ist, dass es eine relativ kleine Menge von Entwicklern gibt, die sich tatsächlich infizieren lassen. Typischerweise scheinen das die gleichen zu sein, die auch sonst Verantwortung in Projekten übernehmen.

Stefan: Ich glaube, viele Entwickler mögen TDD sehr früh und sind dann frustriert, wenn Probleme auftreten. Von selbst schaffen Sie die Hürden à la Mock Objects häufig nicht. Sie wollen also, aber wissen nicht so recht, wie. Schlimmer wird es noch, wenn zusätzlich Projektdruck da ist. In einem unserer Projekte war es z.B. so, dass viele Entwickler testen wollten,

dann aber vor den technischen Schwierigkeiten kapituliert haben. Erst nach massiven Schulungen und Workshops hat sich das deutlich gebessert – und nachdem die Projektleitung explizit die Zeit für Tests auch erlaubt/gewünscht hat.

Achim: Das scheint mir in die Richtung meines Eingangsgedankens zu gehen.

Dierk König: Mocks haben in meinen Augen Technologiecharakter ...

Achim: Es ist meiner Einschätzung nach kein Wunder, dass die Vorgehensmodelle dann nicht benutzt werden, wenn man sie nicht kennt, man sie für den eigenen Fall für ungeeignet hält, die Organisation, der Kunde, das Preismodell es angeblich nicht zulässt usw. Spannend

Viele Entwickler mögen TDD sehr früh und sind dann frustriert, wenn plötzlich Probleme auftreten.

finde ich den Fall, wo es trotz guter Vorsätze und prinzipieller Bereitschaft nicht klappt. Wo liegen da die Ursachen?

Henning: Da werden doch gerne Projektumstände als Entschuldigung verwendet. Gute Vorsätze würde ich immer unterstellen. Ich befürchte eher, dass die Technologiewechsel so früh erfolgen und so häufig, dass selten Technologie so weit beherrscht wird, dass man sich den ganzen anderen (auch spannenden und wichtigen) Themen rund um das Vorgehen widmen kann.

Johannes: Muss man die Technologie beherrschen, um den Prozess anzugehen? Ich beherrsche immer weniger davon, gerade deshalb konzentriere ich mich auf den Prozess, um die Softwareentwicklung trotzdem im Griff halten zu können.

Henning: Nein, muss man nicht, aber es ist eine Frage der Rolle und des Selbstverständnisses. Ich komme auch mit recht wenig Technologie klar, habe aber eben auch meinen Fokus bei Prozessen und Projektmanagement. Unsere Entwickler haben jedoch einen Entwicklungs- und damit Technologiefokus.

Stefan: Wenn mir ständig mein Handwerkszeug auf die Füße fällt, werde ich kaum viel Energie in den Prozess inves-

Anzeige

tieren. Das gilt jedenfalls im JUnit-Beispiel: Wenn ich JUnit und die zugehörigen Bibliotheken nicht beherrsche, läuft der TDD-Prozess nicht reibungslos.

Veränderungen managen

Eberhard: Nun ja, aber was ich spannend finde ist, dass bei dem JUnit-Beispiel die Änderung *aktiv* gemanagt werden musste. Das bedeutet doch, dass Verbesserungen sich nicht ad hoc ergeben, sondern aktiv herbeigeführt werden müssen. Wenn das Management dann Technologien entscheidet und keine Prozesse oder andere Techniken, hat man ein Problem.

Die Neugier auf neue Technologien sollten wir erhalten, nur gehört sie eben nicht in den Projektalltag.

Stefan: Guter Punkt, Eberhard: Wir begleiten die Einführung unserer Verbesserungen nicht aktiv genug. Häufig gibt es eine kurze Schulung und dann sollen die Entwickler selbst sehen, wie sie damit zu recht kommen.

Johannes: Das liegt aber wiederum am Management. Ich sage immer sehr deutlich, dass ich diese 1/2/3-Tage-Schulung für bei weitem nicht ausreichend halte.

Achim: Führt nicht jede Änderung des etablierten Prozesses zuerst in Chaos und lassen sich erst nach einer gewissen Orientierungsphase allmählich Verbesserungen erkennen?

Henning: Wie auch bei jeder neuen Technologie.

Johannes: Sind wir also zu ungeduldig? Beziehungsweise ist es das Management?

Stefan: Ja, wir sind zu ungeduldig und zu geizig und naiv – wir wollen einfach nicht glauben, dass die Einführung von so einer Kleinigkeit wie JUnit > 100.000 Euro kosten kann.

Henning: Vielleicht auch, aber wir hoffen zu sehr, dass die eierlegende Wollmilchsau in Form einer Technologie oder eines Prozesses auftaucht ...

Martin: Bedeutet das, dass im Prinzip alles nur ein Management-Problem ist? Ich stelle mir gerade in meinem aktuellen Projekt die Situation vor, dass alle Tech-

nologien fest vorgegeben sind, alle diese Technologien beherrschen und einsetzen können ... klingt ja erst mal prima ... Und schwups, wären die Leute gelangweilt ...?!?

Stefan: Sie hätten dann Zeit für die fachlichen Dinge!

Henning: Reichen die fachlichen Herausforderungen nicht?

Johannes: Aber die will doch keiner.

Henning: Doch, ich!

Martin: Nicht wollen kann man auch nicht sagen, denke ich.

Eberhard: Also rammen wir die Technologien einmal in den Boden, lassen sie so und sind glücklich?

Stefan: Nein, aber wir gehen nur in kleinen behutsamen Schritten weiter zur näch-

sten Technologie. Wir müssen nicht alles machen oder können, was es so an Hypes gibt.

Technologieneugier

Henning: Ist denn „Hypes“ ein neues Framework? ;)

Stefan: Ja, das gibt es für Java, Ruby und Groovy: JHypes, RHypes, GHypes

Dierk: Die Neugier auf neue Technologien sollten wir erhalten, nur gehört sie eben nicht in den Projektalltag.

Johannes: Wohin dann?

Henning: Also SLACK zum Austoben des Technologiespieltriebs?

Achim: Ich komme gerade ins terminologische Schwimmen ... Was ist jetzt Technologie und was nicht? Für gewöhnlich wird so was doch zu Beginn des Projekts festgeklopft, z.B. Tomcat, Hibernate etc.

Dierk: Die Idee mit den *Gold Cards* [Anm.: Entwickler können Gold Cards in der Iterationsplanung ausspielen, um sich ein oder zwei Tage mit projektfremden Technologien o.Ä. zu beschäftigen] oder der Google-Ansatz der x-prozentigen Freistellung der Entwickler zum Ausprobieren neuer Technologien geht in diese Richtung.

Eberhard: Genau. Zumal JUnit eben Technologie *und* Vorgehen ist.

Dierk: So kann man den Spieltrieb produktiv ausnutzen und stört die Projekte weniger.

Agilität und Überforderung

Johannes: Apropos „festklopfen“. Ich sehe immer häufiger, dass unter dem Schlagwort „Agilität“ nichts mehr festgelegt wird. Weder Prozess, noch Anforderungen, noch Technologie. Ist das nicht zu viel Agilität?

Henning: Klar, Durchwurschteln wird jetzt hoffähig, heißt jetzt Agilität.

Johannes: Da bieten wir Schulungen an.

Dierk: :) LOL!

Stefan: Ich finde daher den Begriff Lean Software Development so schön. Der ist weiter gefasst als eine konkrete Methode wie eXtreme Programming, aber deutlich präziser als Agilität.

Eberhard: Nein, das Problem ist universell. Unabhängig von Agilität.

Martin: Denke ich auch!

Dierk: „Festklopfen“ ohne Freiheit ist dumm. Freiheit ohne Verantwortung für die gemeinsamen Ziele ist unproduktiv.

Stefan: Ich fasse mal zusammen, was ich bisher verstanden habe: 1. Entwickler interessieren sich stärker für Technologien als für Fachlichkeit. 2. Manager investieren nicht ernsthaft in Verbesserungen. 3. Die Neugier auf Technologien sollte kanalisiert werden (z.B. Gold Cards).

Johannes: Ich würde gerne noch mal auf Achims Punkt von vorhin zurückkommen. Manchmal scheitert es trotz des guten Willens der Beteiligten. Sind viele unserer „guten“ Ansätze vielleicht zu kompliziert für den „durchschnittlichen“ Entwickler?

Stefan: Zum Beispiel?

Johannes: Zum Beispiel TDD.

Henning: Nein, wir sind nur oft zu schnell und zu ungeduldig.

Eberhard: OK, anderer Versuch. Die Vorgehensweisen bringen nicht schnell genug und einfach genug einen Pay-off, als dass man sie alleine umsetzen könnte. Das wäre bei TDD ja der Fall ...

Coaching

Eberhard: Was dann aber zu der Frage führt, warum ich TDD gut finde, obwohl ich nicht darin gecoacht worden bin.

Johannes: Ich bin auch nie gecoacht wor-

den. Bei ausreichend intrinsischer Motivation schafft man vieles.

Eberhard: Kann man denn Motivation durch Coaching ersetzen?

Johannes: Coaching sollte meiner Meinung nach vor allem motivieren.

Eberhard: Coaching vermittelt also keine Inhalte, sondern nur Motivation?

Johannes: Manchmal Inhalte *als* Motivation.

Eberhard: Ein motiviertes Team braucht also kein Coaching. Das ist eine interessante Ansicht! Die Idee ist mir noch nie gekommen.

Henning: Vielleicht nutzt Coaching der Motivation und motivierte Teams können Coaching inhaltlich nutzen.

Johannes: Die meisten technischen Probleme können wir heutzutage doch ohne Coach lösen, oder?

Henning: Aber nicht alle.

Eberhard: Und Prozess- oder Vorgehensprobleme können wir nicht selbst lösen?

Stefan: Bei Prozess- und Vorgehensproblemen ist die Beziehung zwischen Ursache und Wirkung viel komplexer.

Henning: Och, ich kenne da Frameworks ... ;-)

Johannes: Jetzt ist der Wein auf der Tastatur gelandet.

Stefan: In welchem Verzeichnis landet der?

Johannes: In `/dev/mouth/`.

Henning: Wie sind wir eigentlich auf Coaching gekommen? Gab es da die Hoffnung einer Lösung, die wir zerschlagen wollten?

Dierk: Was heißt „Erfolg“ beim Coaching? Einen Anschlussauftrag zu bekommen?

Stefan: Erfolg heißt, dass der Kunde das sich selbst gesetzte Coaching-Ziel erreicht.

An die eigene Nase fassen

Martin: Mal eine kurze Frage: Deutet die Diskussion darauf hin, dass wir mit der Eingangsfrage, warum wir nicht so gut sind, wie wir sein könnten, nur die nicht gecoachten oder falsch gecoachten Entwickler und Teams meinen und die Coaches selbst schon so gut sind, wie sie sein könnten?

Henning: Wie wird dann wohl eine lange Frage aussehen?

Martin: Ich versuche nur gerade, wieder den Bogen vom Coaching zur Ausgangsfrage zu bekommen ... ;)

Stefan: Gute Frage von Martin: Bisher haben wir nur über andere gesprochen. Was fühlen wir denn, wenn wir uns an die eigene Nase fassen?

Johannes: Ich selbst mache manchmal bestimmte Dinge nicht, weil ich einfach keine Lust dazu habe. Obwohl ich sie für richtig halte.

Martin: Mir geht es so, dass ich mir immer mal wieder selbst sagen muss, dass ich hier (wenn ich in einem Projekt entwickle) ein *Produkt* baue, mit dem viele andere jeden Tag arbeiten müssen. Denen ist es eben total egal, ob ich denn Spaß bei der Konstruktion hatte oder dabei eine tolle Technologie oder Ähnliches ausprobiert habe. Manchmal vergesse ich das, bei meiner Neugier ...

Henning: Möchte einer von euch aufhören, besser zu werden?

Stefan: Nein, das wäre ja langweilig.

Eberhard: Nein.

Achim: Ich weiß nicht sicher, in welcher Richtung „besser“ liegt ...

Johannes: Und ich verstehe die Frage nicht.

Henning: Die Frage war ja, ob wir uns persönlich für gut (oder gut genug) befinden. Und es gibt immer noch was zu verbessern. Das heißt umgekehrt auch, dass es immer noch eine gewisse Unzufriedenheit gibt mit dem Status quo. Die scheint viele von uns ja auch anzutreiben.

Eberhard: Klar. Innovation geht eben immer weiter ...

Henning: (Hmm, ob Woody Allen diese verqueren Texte von Leuten, die aneinander vorbeireden, auch per Skype entwickelt?)

Eberhard: Die Frage war aber, warum wir das, *was wir heute sicher wissen*, nicht umsetzen.

Johannes: Vielleicht wissen wir gar nichts.

Henning: Weiß nicht.

Stefan: Oh ha, jetzt wird's philosophisch.

Eberhard: Ist das das Schlusswort?

Achim: So abwegig ist das vielleicht gar nicht: Ein Beispiel: Neulich wurde mir Cut & Paste als Vorgehen empfohlen ...

Das dicke Ende (The Long Goodbye)

Johannes: Erzähl! Als Schluss.

Achim: Ich wandte ein: Und wenn man mal was ändern muss? Antwort: Da wird

Anzeige

nie was dran geändert ... Kann ich das als Unsinn verdammen? Oder kennt er die Zusammenhänge tatsächlich besser?

Henning: „Was wir heute sicher wissen“, ist mehr, als wir verkraften können, und widerspricht sich teilweise.

Eberhard: ... und das meine ich. Wie kann man DRY [Anm.: Don't Repeat Yourself] so verletzen? Und sich dann noch wohl fühlen? Das verstehe ich nicht.

Stefan: Cut & Paste kann vollkommen O.K. sein. Man muss halt immer nur wissen, worauf man sich einlässt. Ich habe da auch noch eine Geschichte: Auf einer Fachtagung hat mal jemand von der Softwareentwicklung für spezielle Broker in einer Bank erzählt. Die hatten ständig kurzfristige spezielle Deals zu machen und genau dafür haben die Software bauen lassen. Die wurde einmal für den Deal benutzt und anschließend weggeworfen. Da muss man sich über DRY nicht so viele Gedanken machen. Das Zeug muss korrekt sein und rechtzeitig verfügbar sein, Wartbarkeit ist völlig unerheblich.

Eberhard: Ist das Ergebnis des Gesprächs, dass wir nicht wissen, was gut ist? Noch nicht mal bei DRY? Wäre zwar eine valide Antwort, aber ich empfinde es als suboptimal.

Johannes: „Quality is value to some person.“ [2]

Henning: Ich bin zu lange Berater, um nicht mit „it depends“ zu antworten.

Eberhard: Da stimmt was nicht ...

Stefan: Ich weiß, was gut ist, in den Kontexten, in denen ich damit Erfolg hatte. Es gibt eine Chance, aber keine Gewissheit, dass es in anderen Kontexten auch funktioniert.

Eberhard: Auf der einen Seite heißt es „Die Leute machen willkürlich irgendeinen Technologiekram“, ohne auf Kunden zu achten. Auf der anderen Seite heißt es jetzt „Wenn der Kunde will, bauen wir Schrott ohne Rücksicht auf unsere eigene Meinung.“ Eins kann nur wahr sein, oder?

Johannes: Vielleicht ist es das Problem, dass wir als seriöse Berater nie sagen: Es wird hundertprozentig funktionieren.

Henning: Ich kann sogar mittlerweile anerkennen, dass es mehr als eine richtige Lösung gibt.

Johannes: Was will der Kunde wirklich? Manchmal weiß er das eben nicht selbst.

Stefan: Ich stimme dem auch nicht zu. Ich habe eine eigene Vorstellung von meiner Arbeit. Wenn der Kunde was ganz anderes will, ist das vollkommen O.K. Aber er muss es dann ohne mich machen.

Johannes: Wartbarkeit ist eben erst in Release 2 wichtig.

Stefan: Cooler Spruch!

Eberhard: Das heißt, Code wird nur einmal pro Release geschrieben und ist dann stabil?

Johannes: So denken sich das viele Kunden

Henning: Da fällt mir der Schiffsname „Unsinkbar 2“ ein.

Achim: Was wir jetzt sicher wissen ... Vielleicht gibt's ja nicht mal ein Release 1. ;)

Eberhard: Noch eine Bemerkung: ...

Johannes: Zum Schluss?

Eberhard: Ich finde es auch eine komische, aber verbreitete Meinung, dass „besser = aufwendiger“ gilt.

Henning: Das ist ein Psychoeffekt ... Wir kennen ja mehr gute Sachen, als wir in einem Projekt einsetzen könnten, oder?

Eberhard: Ist das dann das Problem? Wir kennen zu viele gute Sachen? Kann sein.

Henning: Es gibt Untersuchungen, die zeigen, dass in Diskussionen die Leute mit den aufwendigeren Modellen im Kopf gewinnen ... Leider.

Eberhard: Und wer hat hier dann gewonnen ...?

Martin: Noch ein Zitat zum Schluss gefällig?

Johannes: Aber ein allerletztes!

Martin: „Perfection is achieved, not when there is nothing more to add, but when there is nothing more to take away.“ – Antoine de Saint-Exupéry.

Johannes: Ich danke euch. Wie ich daraus einen Artikel machen soll, weiß der Himmel ... ■

■ Links & Literatur

[1] Gerald M. Weinberg: Secrets of Consulting: A Guide to Giving and Getting Advice Successfully, Dorset House Publishing, 1986

[2] Gerald M. Weinberg: Quality Software Management, Band 1–4, Dorset House Publishing, 1997

[3] Tim Mackinnon. Innovation and Sustainability with Gold Cards: www.frankwestphal.de/xp2001/TimMackinnon.html