Im Dialog zum Projekterfolg

Selbstversuch mit Hundefutter

Operation misslungen. Patient lebt. So lautet das Fazit eines eher ungewöhnlichen Projekts, in dem Ralf Westphal viel über Anforderungen, Qualitätssicherung und Vorgehensweisen gelernt hat.

Auf einen Blick

Ralf Westphal (www.ralfw.de) ist freier Softwaretechnologievermittler im Bereich .NET-Softwarearchitektur. Er arbeitet als Fachautor, Coach/Berater und als Referent auf Entwicklerevents. Ralf Westphal ist Mitbegründer des Professional Developer College (www.prodevcollege.de) und Microsoft Visual Developer Solution Architect MVP.

Inhalt

- Faktura-Lösung mit Excel und Dropbox soll verbessert werden.
- Scheitern an API-Problemen und ungenauen Anforderungen.
- Schlussendlich Erfolg durch Nutzen für den Kunden.

dnpCode
A1205Kolumne

Sein eigenes Hundefutter zu essen („to eat one's own dogfood") ist eine Methode, um schnell Feedback zu den eigenen Produkten zu bekommen und dabei auch noch Glaubwürdigkeit aufzubauen. Wenn Sie hören, dass Microsoft, Telerik oder JetBrains intern mit der selbst entwickelten Software arbeiten, steigt Ihre Bereitschaft, sich auch darauf einzulassen.

Wer ein Angebot macht – sei das etwas Greifbares wie ein Rasierapparat, etwas Flüchtiges wie eine Dienstleistung oder etwas Virtuelles wie Software –, der stellt im Grunde eine Hypothese auf. Die lautet: „Mein Angebot ist nützlich."

Ein Unternehmen ist mithin ein Forschungslabor, das Experimente durchführt. Versuchsfeld ist der Markt, Versuchskaninchen zur Überprüfung der Hypothese sind die Kunden. Nehmen sie die Produktangebote des Unternehmens an? Wie vertragen sie sie? Kommen sie wieder für mehr?

Dogfooding ist durch diese Brille betrachtet ein Selbstversuch. Statt andere zu Versuchskaninchen zu machen, testet der Forscher seine Hypothese an sich selbst.

So einen Selbstversuch habe ich neulich unternommen. Ich wollte meinen Ansatz Spinning für agiles Vorgehen [1] außerhalb von Seminar und Beratung an einem Softwareprojekt selbst anwenden. Dadurch versprach ich mir unmittelbareres Feedback als durch die Beobachtung von anderen, die danach vorgehen.

Das Projekt

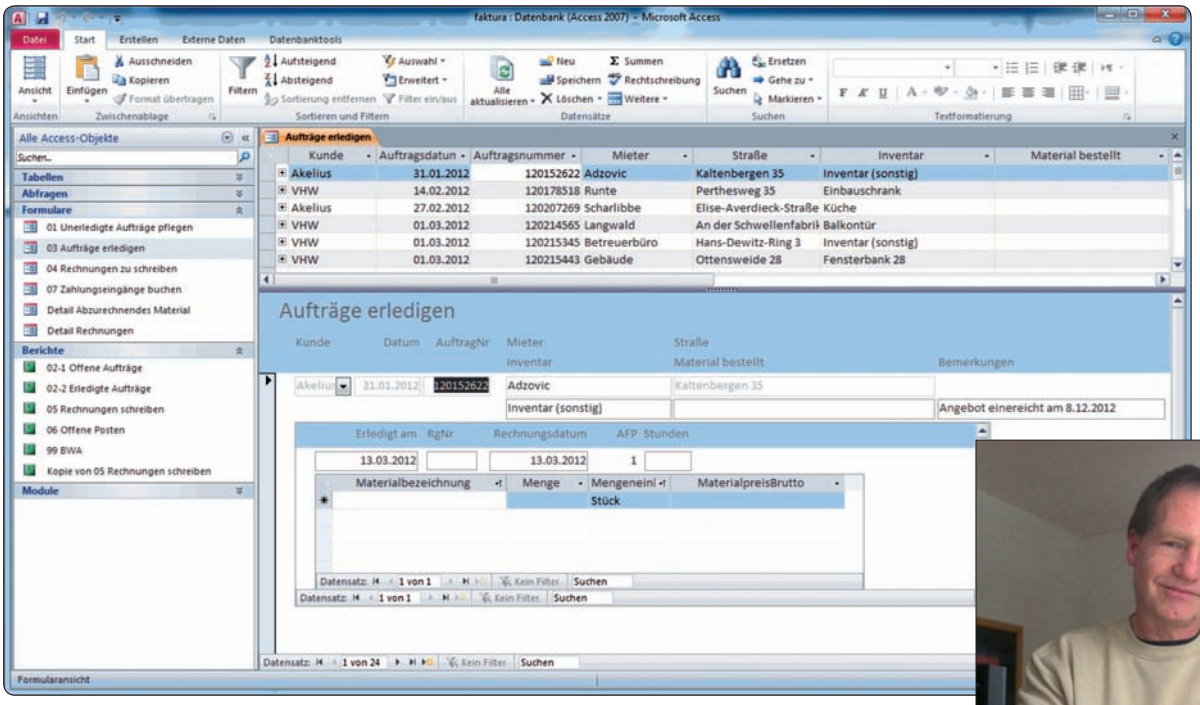
Eigentlich bin ich selbst ja nicht im Projektgeschäft tätig, sondern ich helfe als Trainer und Berater Teams bei ihren Projekten. Deshalb musste ich etwas suchen, um ein Projekt zu finden, das ich selbst durchführen konnte. Schließlich aber bot sich über meinen Kontakt zu Professional Organizer Andrea Kaden **(Abbildung 1)** von Zeitgewinn Hamburg [2] eine passende Gelegenheit.

Sie war bei ihrem Kunden Tischlermeister Florian Staeter [3] **(Abbildung 2)** unter anderem mit dem Streamlining der Rechnungslegung betraut. Der hatte bisher die Aufträge der Hausverwaltungen, für die er arbeitet, ganz traditionell fakturiert und versank dadurch in Papierbergen.

Die zu vermeiden war Andrea Kadens erstes Ziel gewesen. Dazu hatte sie mit einfachsten Mitteln einen Workflow mit Excel und Dropbox [4] aufgesetzt. Etwas vereinfacht sah der so aus:

[Abb. 1] Projektvermittlerin Andrea Kaden von Zeitgewinn Hamburg.

[Abb. 2] Kunde Tischlermeister Staeter.



[Abb. 3] Die MS-Access-Lösung.

[Abb. 4] Florian Staeter bei der Fakturierung mit der Access-Lösung.

1. Als PDF per E-Mail eingehende Aufträge in einem Dropbox-Ordner ablegen.
2. PDF-Auftrag in der Excel-Tabelle *Offene Aufträge* erfassen.
3. *Offene Aufträge* terminieren.
4. Nach Erledigung die Aufträge in die Excel-Tabelle *Erledigte Aufträge* verschieben und Angaben zu verbrauchtem Material, Zeit, Fahrtkosten ergänzen.
5. Erledigte Aufträge abrechnen durch Kopieren der Daten in eine Excel-Tabelle *Rechnung*. Je Auftrag eine Rechnung schreiben.
6. Zahlungseingänge pflegen in Excel-Tabelle *Erledigte Aufträge*.

Wenn Sie das lesen, sträuben sich Ihnen natürlich die Haare ob der ganzen Kopiererei. Ich habe auch mit den Augen gerollt. Aber ich habe dann ganz schnell etwas gelernt: Es kommt nicht auf Eleganz an, sondern auf Schnelligkeit und Nutzen.

Andrea Kaden hat ihrem Kunden aus dem Stand mit Standardwerkzeugen, die zur Hand waren, eine große Erleichterung verschafft. Sie hat auf eine Verbesserung statt auf eine ultimative Lösung gesetzt; relative Vereinfachung statt maximaler Einfachheit. Nach Aussage von Tischler Staeter hat sie ihm mit dem kleinen Prozess ermöglicht, das Auftragsvolumen, das er pro Woche schafft, deutlich zu steigern. Rechnungen schreiben wurde für ihn von einer Last zu einer Lust.

Diese Situation fand ich vor und hatte die Idee, sie durch etwas .NET-Magie weiter zu verbessern. Tischler Staeter war dafür offen. Er hatte Blut geleckt, was die Vereinfachung seiner Abrechnung durch einen Softwareeinsatz anging.

Die bisherige Lösung wollte ich allerdings nicht umkrempeln und durch eine neu entwickelte .NET-Faktura-Anwendung ersetzen. Nein, Bekanntes und Bewährtes sollte erhalten bleiben; Überflüssiges und Lästiges sollte gestutzt werden. Konkret bedeutet das:

- Excel als Frontend/Erfassungswerkzeug sollte erhalten bleiben. Damit kam Tischler Staeter gut zurecht.
- Datendopplungen durch Kopieren sollten entfallen.
- Übersichtslisten, die in Auszügen auch den Auftraggebern geschickt wurden, und Rechnungen sollten auf Knopfdruck erstellt werden können.

Als Lösung schwebten mir kleine .NET-Batchprogramme vor, die mit den Excel-Daten arbeiten. Sie würden eventuell noch nötige Kopieroperationen übernehmen und Ausdrucke anfertigen.

Das klang ideal für meinen Zweck: Da war ein williger Kunde. Die Anforderungen lagen auf der Hand. Die Technologie war mir bekannt.

Programmierung mit Visual Basic for Applications (VBA) beziehungsweise .NET-Add-ins für Excel wären auch möglich gewesen, doch davor bin ich zurückgeschreckt. Zu lange ist mein Kontakt mit VBA her und dementsprechend eingerostet sind die Erfahrungen. Zu wenige Kenntnisse habe ich von Excels-Objektmodell und dem Deployment von .NET-Add-ins.

Die Ausführung

Dieser Artikel würde kaum lohnen, wenn dann alles glattgegangen wäre. Schöne Ideen hatte ich – nur leider kollidierten sie mit der Realität. Das heißt nicht, ich hätte sie nicht umsetzen können. Aber eine Kursabweichung schien am Ende angemessener. Mit .NET habe ich mich eine Zeit lang geplagt. Die Bibliotheken für den Zugriff auf Excel waren mir am Ende aber zu umständlich. Die eine machte Probleme mit Datumswerten in Zellen, die andere konnte nur .xls-Dateien statt .xlsx. Dazu kam die unschöne Fummelei mit den Schemata der Excel-Dateien.

Von Excel abzurücken schien allerdings nicht möglich. Denn aufgrund des Erfolgs der bisherigen Lösung wünschte sich Tischler Staeter die Möglichkeit der Datenerfassung auf dem iPad. Das schien mit Dropbox und Office² HD [5] kein Problem; das Frontend wäre wie beim PC eine Tabellenkalkulation. iOS-Programmierung bliebe mir erspart.

Die technischen Misserfolge und andere Verpflichtungen zogen dann die Um-

setzung meiner Ideen zeitlich sehr in die Länge. Von Spinning keine Spur. Dabei war das doch der eigentliche Anlass für das Projekt.

Am Ende musste es kommen, wie es immer kommt: Der Druck, nein, eher das schlechte Gewissen wurde so groß, dass irgendwie einfach eine Lösung hermusste. In meiner Verzweiflung griff ich dann zur bewährten Allzweckwaffe MS Access. Und siehe da, eine Lösung war in zwei Tagen gezaubert **(Abbildung 3)**.

Ohne viel Brimborium kann Tischler Staeter damit nun seine Aufträge fakturieren und bekommt sogar eine kleine betriebswirtschaftliche Auswertung. Der Aufwand ist gegenüber der Excel-Lösung noch einmal drastisch gesunken.

Der Umgang mit der Access-Datenbank ist recht krude. Es gibt keine besonderen Formulare und keinen zusammenfassenden Anwendungsdialog. Auch meine Access-Kenntnisse waren eingerostet. Ich habe mich durchgewurschtelt. Zum Glück hat das gereicht. Die Lösung ist gut genug, das heißt wieder nur eine relative Verbesserung, kein Optimum.

Tischler Staeter ist damit sehr zufrieden **(Abbildung 4)**. Und schon stellen sich Wünsche ein. Einige werde ich noch mit der Access-Datenbank realisieren. Der sinnvollste nächste Schritt besteht jedoch in der Auswahl einer Standardsoftware. Jetzt ist Tischler Staeter dafür bereit. Hätte Andrea Kaden ihm dazu gleich am Anfang geraten, wäre das wahrscheinlich ein zu großer Schritt gewesen. So aber konnten alle Beteiligten in kleinen Schritten lernen. Und zu jedem Schritt gehörte und gehört eine andere Lösung. Die Lösungen wachsen sozusagen mit.

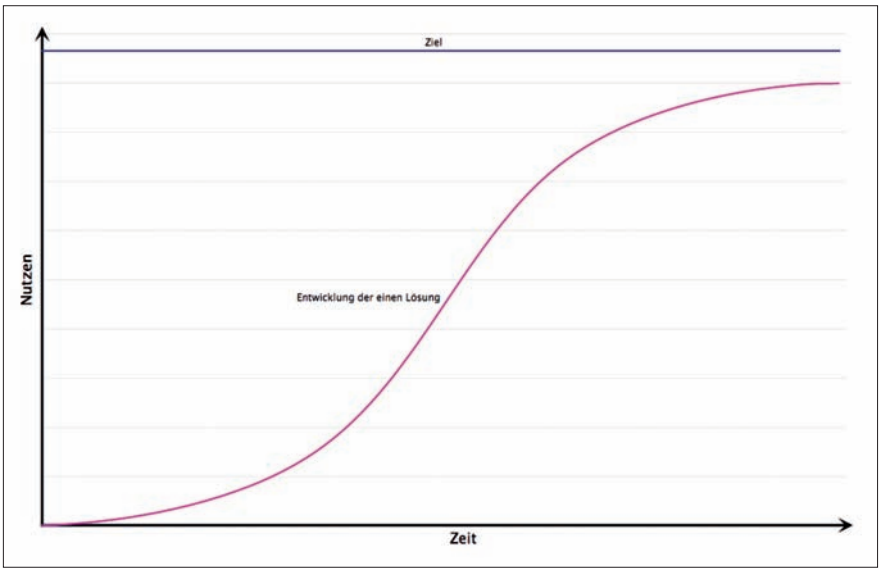
Für Tischler Staeter ist das Projekt damit ein Erfolg. Mein Ziel habe ich hingegen nicht erreicht. Zeit für eine Reflexion.

Lessons learned

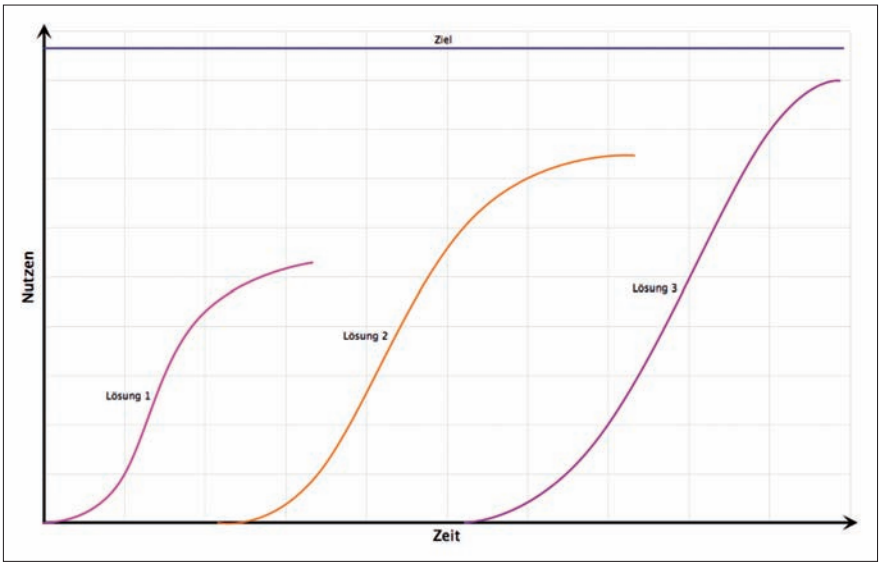
Besser wird es nur, wenn man sich genau anschaut, was schiefgegangen ist – und das beim nächsten Mal versucht zu vermeiden. Deliberate practice.

Nach außen ist nicht viel schiefgegangen. Tischler Staeter ist pragmatisch. Den interessiert am Ende nicht, ob ich eine Lösung mit .NET gestrickt habe oder mit MS Access. Hauptsache, sie ist nützlich. Und das ist sie.

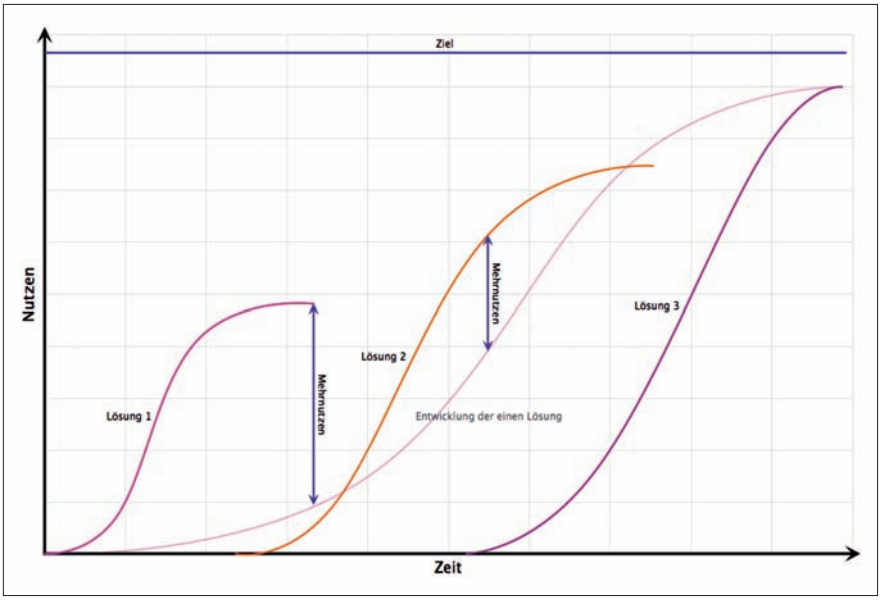
Gegenüber meinem eigenen Anspruch bin ich allerdings zurückgeblieben. Ich habe im Spinning-Modus arbeiten wollen – aber es ist ganz anders gekommen. War-



[Abb. 5] **Dem Kunden Nutzen liefern mit einer Lösung.**



[Abb. 6] **Dem Kunden mit einer Folge von Lösungen Nutzen liefern.**



[Abb. 7] **Schneller mehr Nutzen mit verschiedenen Lösungen.**

um? Was kann ich daraus für mich und für Sie lernen? Hier meine Erkenntnisse in loser Reihenfolge. Ich fange einfach mal irgendwo an:

1. Unsicherheiten ausräumen und explorieren

Ich war mir zu früh zu sicher, schon eine Lösung zu haben. So ein bisschen mit Excel-Dateien hantieren, sollte mit .NET doch kein Problem sein. Aber woher hatte ich meine Sicherheit? Sie hat sich als Illusion herausgestellt. Nicht, dass es gar nicht funktionieren würde, was ich mir ausgedacht hatte – doch auch ich hatte mir nur einen gewissen Gesamtaufwand zugestanden, der schnell überschritten wurde, weil ich mit Excel-APIs herumexperimentieren musste.

Vermeintliche Sicherheiten lauern überall. Glaubenssätze, mit denen man in ein Projekt startet. Vorurteile, vorgefasste Meinungen, Überzeugungen oder auch nur durch Extrapolation und vom Hörensagen Angenommenes.

Ich hatte geglaubt, der Excel-Dateizugriff sei ganz einfach. Fehlanzeige. Ich hatte geglaubt, Excel zu erhalten sei wirklich wichtig, weil Tischler Staeter sich so daran gewöhnt hatte. Fehlanzeige. Ich hatte geglaubt, das iPad als zusätzliches Frontend sei unverrückbar gesetzt und dringend zu unterstützen. Fehlanzeige.

Diese Annahmen und andere haben mich viel Kraft gekostet und die Realisierung verzögert. Durch den Frust mit den Excel-APIs ist nicht mal eine Teillösung auf die Straße gekommen; ich habe immer wieder die Arbeit aufgeschoben, weil sie einfach nicht den Erfolg gebracht hat, den ich mir vorgestellt hatte.

Dass Office² HD auf dem iPad wirklich ein gutes Frontend abgeben würde, schien mir irgendwann auch nicht mehr sicher. War es wirklich so eine gute Idee, überhaupt mit verschiedenen Devices auf Excel-Dateien zu arbeiten? Oder wäre Google Docs besser? Aber wie darauf aus .NET zugreifen? Dieses Gedankenkarussell hat mich stumm gemacht und für weitere Verzögerung gesorgt.

Im Nachhinein zeigt sich, dass ich mich selbst unter Druck gesetzt habe. Mein Anspruch war, sicher zu sein als Dienstleister. War mein Job nicht, sofort eine Lösung zu haben? Sehen, lösen, umsetzen, fertig?

Die erste Lehre, die ich aus dem Projekt ziehe: Mehr Zeit für Exploration. Alle Beteiligten sind viel unsicherer, als sie gern wären oder zugeben zu sein. Viel weniger, als man denkt, ist wirklich, wirklich gesetzt. Das oberste Gebot lautet mithin, bei jeder Aussage zu fragen: „Ist das tatsächlich so? Woher weißt du/wissen Sie, dass es so ist?“ Das gilt für Aussagen anderer wie für meine eigenen. Und es gilt auch für alles, was man vorfindet. Denn was da ist, ist sozusagen eine implizite Aussage: So ist es und sollte auch so sein.

Softwareentwicklung ist mithin zunächst einmal eine Kunst des Hinterfragens. Das macht sie ja auch für Kunden so unangenehm. Wer will schon ständig und alles hinterfragt bekommen? Aber es hilft nichts. Fragen müssen sein, noch mehr Fragen müssen sein, wenn nicht unnötig Aufwand betrieben werden soll.

Und Exploration muss sein. Wenn Fragen Unsicherheiten oder Freiheitsgrade aufdecken, dann sollten die erkundet werden, bevor man sich entscheidet. Das braucht auch Zeit, doch die ist gut angelegt. Exploration kostet weniger, als sich in eine Sackgasse zu verrennen oder erst Suboptimales zu liefern und dann später nachzubessern.

2. Zügig Nutzen liefern

Meine zweite Erkenntnis scheint im Widerspruch zur ersten zu stehen. Sie lautet: Dem Kunden sollte schnell Nutzen geliefert werden. Nicht lange nach der perfekten Lösung suchen, sondern zügig einen ersten Schritt tun und dann einen nächsten und so weiter.

Das ist ganz im Sinne agilen Denkens im Allgemeinen und auch auf der Linie von Spinning. Warum also diese Betonung? Weil ich diese Erkenntnis nicht auf Fortschritte innerhalb einer Lösung beziehe, sondern auf Fortschritt mit verschiedenen Lösungen.

Als Lösung bezeichne ich hier eine Sammlung von Werkzeugen und Prozessen, um einen Nutzen herzustellen. Der liegt bei Tischler Staeter darin, seine Faktura schneller durchzuführen.

Normalerweise ist das Ziel, den Nutzen einmalig mit einer Lösung möglichst komplett herzustellen. Das geschieht dann im Rahmen einer S-Kurve, bei der der Nutzen über die Zeit wächst, je vollständiger sie wird **(Abbildung 5)**. Die Entwicklung der Kurve wird zwar iterativ vorangetrieben. Doch es gibt eben nur eine Kurve, eine Lösung.

Bei Tischler Staeter ist es aber anders gelaufen. Das hat mir gefallen. Dort sieht die Entwicklung eher wie in **Abbildung 6** aus. Andrea Kaden hat Lösung 1 in Form der Excel-Dropbox-Kombination implementiert. Die ist nicht ultimativ vollständig, aber hat schon großen Nutzen gebracht.

Während diese Lösung in Betrieb war, habe ich Lösung 2 mit MS Access aufgesetzt. Die ist immer noch nicht ultimativ vollständig, aber liefert einen größeren Nutzen als Lösung 1. Auf die baut sie allerdings eben nicht auf. Das war mein erster Gedanke, ich wollte etwas aus Lösung 1 retten. Wie sich herausstellte, war es aber einfacher, eine ganz neue Lösung zu entwickeln.

Lösung 3 steht nun in den Startlöchern. Während Lösung 2 in Betrieb ist, eruiert Andrea Kaden entspannt, wie der Nutzen noch weiter gesteigert werden könnte (Lösung 3).

Das Schöne bei diesem Vorgehen ist, dass von Lösung zu Lösung ganz fundamental gelernt werden kann. Es gibt keinen Lock-in durch schon geschaffene Fakten.

Andererseits bedingt dieses Vorgehen natürlich, dass zeitweise zwei Lösungen in Benutzung sind. Das ist eine größere Belastung für die Anwender. Und es leistet Dateninkonsistenz Vorschub. Wie sich aber herausstellte, war der Gewinn durch größere Flexibilität höher als der Verlust durch die Fallstricke des Parallelbetriebs. Wir waren einfach nicht durch die Grenzen eines Lösungswerkzeugs eingeschränkt.

Andrea Kaden und ich konnten auf diese Weise schneller mehr Nutzen liefern, wie uns scheint **(Abbildung 7)**. Und wir konnten auf dem Weg Gelerntes berücksichtigen; Tischler Staeter musste sich ja auch erst mal einfinden beim Umstieg auf eine softwaregestützte Faktura. Statt einer One-Size-fits-all-Lösung gab es also eine wachsende und sich wandelnde Lösungsfolge.

3. Der Kunde muss ziehen

Im Zentrum des Spinning steht die hohe Drehzahl von einer Iteration pro Tag. Jeden Tag ein wenig Nutzen liefern. Ziel des Projekts war, das zu (er)leben. Und genau das ist nicht passiert. In diesem Sinn war das Projekt ein Misserfolg.

Ich glaube allerdings, etwas retten zu können, indem ich nach der Ursache frage. Warum hatten meine Iterationen keine solch hohe Frequenz? Warum hat sich die Realisierung über Wochen gestreckt, ohne dass der Kunde Staeter etwas von mir zu sehen bekam?

Ich denke, die Ursache ist ganz einfach. Es fehlte die Voraussetzung. Es ist ein Kunde, der zieht. Spinning liefert nicht

einfach jeden Tag – das wäre ein Push-Modus. Beim Spinning muss vielmehr gezogen werden; Spinning funktioniert nur im Pull-Modus. Das bedeutet, der Kunde (beziehungsweise sein Stellvertreter) muss jeden Tag auf der Matte stehen und fragen: Wo ist das Inkrement, zu dem ich Feedback geben soll?

Das ist nicht geschehen. Weder Tischler Staeter noch Andrea Kaden haben sich täglich bei mir gemeldet, um Feedback zu geben. Das hat mich dazu verführt, beim Excel-API rumzudrucksen, zwischendurch andere Projekte voranzubringen, die weniger frustrierend waren, oder immer neue Lösungsideen zu prüfen, statt die ursprüngliche voranzutreiben.

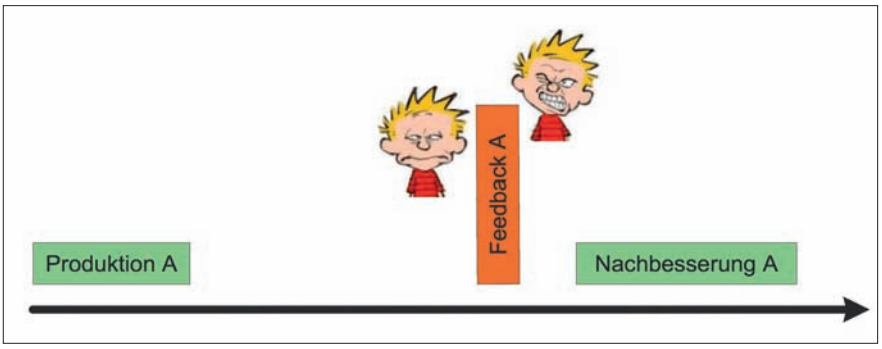
Und warum hat niemand auf meiner Matte gestanden? Weil ich diese Notwendigkeit nicht wirklich kommuniziert hatte. Ich habe schlicht versäumt, alle Beteiligten an einen Tisch zu holen, um ihnen den Prozess zu erläutern und sie darauf einzuschwören. Irgendwie war ich im Push-Modus gefangen und habe gedacht, ich muss nur von mir aus versuchen, jeden Tag zu liefern. Aber Pustekuchen … daraus ist nichts geworden. Da habe ich zu viel von mir verlangt. Mein Job war es, Nutzen zu liefern, nicht den Prozess zu steuern. Ich habe das Single Responsibility Principle nicht gelebt. Es gab sozusagen eine Ämterhäufung.

Richtig wäre gewesen, Andrea Kaden als Stellvertreterin des Kunden (und Anwenders) Tischler Staeter als Product Owner (PO) einzusetzen mit der Pflicht, jeden Abend bei mir nach dem Inkrement zu fragen und den mit dem nächsten Inkrement zu liefernden Nutzen zu besprechen.

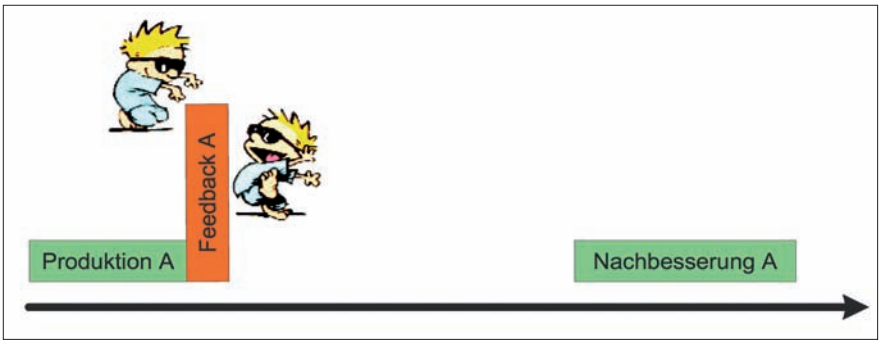
Dadurch wäre viel schneller ins Bewusstsein gekommen, dass irgendetwas hakt, dass es noch Unsicherheiten gibt. Dann hätten wir alle zusammen gegensteuern können.

Hätte Andrea Kaden das leisten können? Ich denke schon. Sie hatte ja schon eng mit Tischler Staeter gearbeitet und kannte dessen Anforderungen. Zeit und Lust hätte sie auch gehabt. Diszipliniert genug ist sie auch. Sie wäre als PO tough und genau die Richtige gewesen.

Nur habe ich diese Rolle mit ihr nie ernsthaft besprochen. Warum? Meine Vermutung: Ich hatte schon Unsicherheiten gespürt und Angst, die anzusprechen. Sie hätten mich als weniger kompetent darstellen können. Es wäre vielleicht herausgekommen, dass der Realisierungszeitraum weniger klar war als gedacht.



[Abb. 8] **Spätes Feedback ist frustrierend und kontraproduktiv.**



[Abb. 9] **Zügiges Feedback macht Spaß.**

Das sind natürlich alles nur Rationalisierungen. Denn am Ende ist ja genau das auf die eine oder andere Weise doch herausgekommen. Ich musste irgendwann gestehen, in einer Sackgasse zu stecken.

Was habe ich daraus gelernt? Nur schonungslose Offenheit hilft. Denn dann kann Exploration mit in den „Projektplan“ hineingenommen werden. Und Exploration war nötig. Nutzenlieferung und Exploration können sich auch abwechseln. Während der Exploration ist dann kein täglicher Rapport nötig; in Kontakt mit dem PO muss man natürlich trotzdem sein.

Ich hätte mit dem Versuch täglicher Lieferung starten können. Wäre der gescheitert, hätten wir auf Exploration umschalten können. Auf ausreichende Exploration wäre wieder tägliche Lieferung gefolgt und so weiter. Je mehr Unsicherheit, desto mehr/längere Phasen der Exploration. Je mehr Sicherheit, desto längere Phasen täglicher Lieferung.

Gelernt habe ich also, dass Spinning nicht vom Team getrieben werden kann, sondern wirklich und wahrhaftig der Kunde ziehen muss. Zweitens funktioniert Spinning nur, wenn und wo zumindest technologisch hohe Sicherheit vorhanden ist. In der Domäne kann ruhig Unsicherheit herrschen; die auszuräumen ist Aufgabe des Spinning. Spinning selbst ist sozusagen Exploration.

4. Schnelles Feedback

Beim Spinning laufen Produktion und Feedback in schneller Folge ab. Wenn nun aber Spinning mit täglichem Feedback nicht möglich ist, was dann?

Meine Erkenntnis aus diesem Experiment: Feedback muss in jedem Fall schnell kommen, egal wie lange es dann dauert, bis an dem Projekt wieder etwas getan wird.

Egal wie lange Sie brauchen, um etwas Vorzeigbares zu produzieren, wenn Sie am Ende eine Woche oder länger auf Feedback warten, ist das unbefriedigend und kontraproduktiv **(Abbildung 8)**. Der Kontext, in dem Sie produziert haben, ist einfach aus Ihrem Kopf verschwunden. Sie haben das Thema innerlich abgehakt; es bei negativem Feedback wieder aufzumachen, ist frustrierend und kostet Mühe. Kritik ist einfacher anzunehmen, wenn sie unmittelbar geäußert wird.

Wenn Sie nicht kontinuierlich an einem Projekt arbeiten können oder der Kunde nicht ständig verfügbar ist, dann richten Sie es deshalb besser so ein: Nehmen Sie sich die nötige Zeit zur Produktion von etwas Vorzeigbarem. Holen Sie unmittelbar Feedback ein – und lassen Sie dann die Arbeit ruhen **(Abbildung 9)**. Schreiben Sie beim Feedback auf, welche Konsequenzen sich daraus ergeben, wie Sie nachbessern müssen. Machen Sie sich ein kleines Backlog. Sobald Sie dann wieder Zeit ha-

ben, arbeiten Sie es ab und holen sich sofort wieder Feedback.

5. Am Engpass ansetzen

Behindert haben mich nicht nur Schwierigkeiten mit den Excel-APIs, sondern auch eine Unkonzentriertheit. Ich habe mich verwirren lassen durch die Idee, dass eine mobile Datenerfassung mit dem iPad auch eine coole Sache wäre. Immer wieder sind meine Gedanken dahin abgeschweift, weil das natürlich technologisch eine Herausforderung ist.

Und als ich dann gemerkt habe, dass mir durch iPad-Visionen Kraft für die Verbesserung der Desktop-Faktura verloren ging, da habe ich nicht gewagt, das dem Kunden gegenüber anzusprechen. Die Idee der mobilen Datenerfassung hatte ich im Überschwang eingebracht und Tischler Staeter war sofort darauf angesprungen. Ein iPad wurde angeschafft. Er war in den Startlöchern. Da schien es mir unmöglich, ihm zu sagen, dass dieser Aspekt vielleicht doch nicht so wichtig sei, das iPad gar voreilig angeschafft worden war.

Am Ende hat mich aus diesem Konflikt erlöst, dass ich mit MS Access eine Lösung gebastelt habe, die Tischler Staeter überzeugte. Er selbst war dann gar nicht mehr so versessen auf die mobile Datenerfassung. Anderes wurde ihm dafür wichtiger. Außerdem hatte er sich mit dem iPad für andere Zwecke angefreundet. Seine Anschaffung hatte also auch Sinn ohne Nutzung für die Faktura.

Meine Erkenntnis daraus ist, dass Ideen eine schöne Sache sind – aber sie sollten sich auf den Engpass des Problems konzentrieren. Wo drückt der Schuh am meisten? Eine mobile Datenerfassung wäre nett für Tischler Staeter, doch nicht wirklich nötig. Damit würde er nicht so viel Zeit sparen wie mit weniger Kopiererei zwischen Excel-Tabellen.

Als Entwickler tun wir also gut daran, uns nicht von coolen Ideen oder auch Erfahrungen davontragen zu lassen. Wir müssen vor die Füße des Kunden schauen. Nicht weiter. Welcher nächste Schritt bringt am meisten?

Das ist eine der wohl schwierigsten Fragen. Aber es hilft nichts. Bei begrenzten Ressourcen muss der Fokus gesetzt werden. Kurzfristig mag es wehtun, dem Kunden hier und da eine Absage zu erteilen, um sich auf eine Sache zu konzentrieren. Langfristig bringt das mehr Erfolg.

In meinem Projekt habe ich erfahren, dass die Angst vor den Folgen, eine übereilte Empfehlung einzugestehen, viel zu groß war. Besser ein offenes Wort als ein offenes Bein. Der Fokus, etwas Nützliches zu liefern, versöhnt den Kunden mit dem Frust eines dadurch korrigierten Fehlers.

6. Abschied von der Wunschliste

In meinem Projekt-Backlog stand über längere Zeit „Ein Formular zum Nachbearbeiten von Rechnung“. Das hatte sich nicht der Kunde gewünscht, sondern ich hatte es als Anforderung eingetragen, sozusagen eine Erinnerung an mich.

Die Access-Lösung enthielt dieses Feature allerdings nicht. War das ein Problem? Nein. Der Kunde ist glücklich ohne. Und selbst wenn es einmal eine Gelegenheit geben sollte, bei der so ein Feature nützlich wäre, ist die Frage, ob er deshalb dafür Geld ausgeben will. Vielleicht gibt es für diesen seltenen Fall ja einen Workaround, der unterm Strich kostengünstiger ist.

Warum habe ich mir also diese Idee für eine Anforderung notiert? Weil mir meine Erfahrung gesagt hat, dass so etwas bestimmt nötig sei. Wie die Realität gezeigt hat, war das aber Quatsch. Ich habe damit nur das Backlog verlängert und subtilen Druck auf mich erzeugt.

Noch schlimmer wäre es gewesen, ich hätte mir sogar die Mühe gemacht, das Feature zu implementieren. Eine Verschwendung von Ressourcen wäre das gewesen.

Und so ist meine Erkenntnis: Längere Wunschlisten tun nicht gut. Was nicht unmittelbar umgesetzt werden kann, sollte nicht in einem Backlog notiert werden. Vor allem sollte kein Aufwand zu seiner Beschreibung getrieben werden.

Wenn dem Kunden etwas wirklich wichtig ist – das gilt auch fürs Bugfixing –, dann wird er sich dazu so lange melden, bis er es bekommt. Wichtig ist nicht ein ausführliches Backlog, sondern schnelle Reaktionszeit. Und damit ist Evolvierbarkeit zentral, weil ohne sie keine schnelle Reaktion möglich ist.

7. Nur die Abnahme zählt

Alle Anforderungsanalyse, alle Tests durch Entwickler und auch eine Qualitätssicherung (QS) sind am Ende ohne Bedeutung. Sie sind nicht nutzlos, aber sie sagen nichts darüber aus, ob die Software tatsächlich fehlerfrei oder gar fertig ist.

Ich habe zum Beispiel bis zur Übergabe der Software an Tischler Staeter die Umwandlung eines Auftrags in eine Rechnung immer mit Angabe von verbrauchtem Material getestet. Das hat funktioniert. Aber es war eben nicht genug, weil es auch Rechnungen ohne Material gibt. Das ist selten, aber möglich. Mir lagen dafür keine Beispiele vor. Bei der Anforderungsanalyse ist sowohl Andrea Kaden wie mir dieser Fall durch die Lappen gegangen. Kann passieren. Mag hier offensichtlich sein und Sie schütteln den Kopf. Doch auch und gerade wenn es komplizierter wird, fallen Akzeptanzkriterien durch den Rost. Kaum hatte Tischler Staeter jedoch meine Access-Lösung in der Hand, hat er sofort eine Rechnung ohne Material erfasst und ist auf die Nase gefallen. Als hätte er es geahnt …

Damit ist die Grundannahme hinter Spinning für mich bestätigt: Nur die Abnahme zählt. Sie darf nicht verzögert werden, sie muss viel, viel häufiger als üblicherweise stattfinden. Nur sie bringt Sicherheit, dass das, was produziert wurde, etwas taugt. Die QS mag dabei eine wichtige Instanz sein. Letztlich zählt jedoch nur das Wort des Kunden. Verlassen Sie sich auf nichts anderes.

Fazit

Operation misslungen, Patient lebt. Glück gehabt. Es hätte anders ausgehen können. Am Ende haben Pragmatik und ein offenes Wort die Situation gerettet. Andrea Kaden und ich konnten dem Kunden vermitteln, dass ein Umstieg von Excel auf Access mehr bringen würde, als zwanghaft um Excel etwas herumzustricken. Der Kunde war offen dafür – das Ergebnis gibt uns recht.

Was sagt das aus über mein eigenes Hundefutter-Spinning? Funktioniert es nicht, weil ich es nicht hinbekommen habe?

Ich denke, Kernelemente von Spinning haben sich bestätigt: Zügiges Feedback und Forderung durch den Kunden sind und bleiben nötig. Aber Spinning hat eine Einstiegshürde. Um Spinning herum habe ich einiges gelernt. Beim nächsten Mal will ich das besser machen. Und ich denke, auch Sie können das eine oder andere mitnehmen, selbst wenn Sie nur nach Scrum arbeiten oder bisher ganz ohne systematisches Vorgehensmodell. **[tib]**

[1] Ralf Westphal, Get into the flow with Spinning, www.dotnetpro.de/ SLA1205Architektur1
[2] http://zeitgewinn-hamburg.de/
[3] http://servicetischler-hamburg.de/
[4] www.dropbox.com/
[5] http://itunes.apple.com/de/app/id364361728?mt=8