Vergessen lernen:
Applikationswissen für nachhaltigen Nutzen organisieren

Elke Mittendorf
Trivadis AG, Business Integration Services/Requirements Engineering
Europastrasse 5, 8152 Glattbrugg
elke.mittendorf@trivadis.com

1. Kurzfassung

Requirements Management Prozesse werden definiert, um Applikationswissen zu bewahren, in Form von Dokumentation oder durch Wissensträger. Wir zeigen an Erfahrungsbeispielen wie eine Balance aus „Wissen bewahren" und „Wissen vergessen", Blockaden bei der Weiterentwicklung der Anwendung lösen konnte. Ein in der Praxis entwickeltes Vorgehen, das beide Perspektiven – Bewahren und Vergessen – ermöglicht konnte Entwicklungsteams dabei helfen, ihr Vorgehen zielführend zu verbessern. Wir stellen dieses Vorgehen zur Prozessverbesserung vor und diskutieren es kritisch.

2. Einleitung

Wenn Applikationsverantwortliche und Entwicklungsteams ihr Wissen organisieren, dokumentieren sie mit dem Ziel „Wissen zu bewahren". Requirements Management sollte-das ist die These dieser Arbeit-nicht nur bewahren sondern auch Freiraum für Neues schaffen, in diesem Zusammenhang unterschätzen wir das „Vergessen". Entwicklungsteams sollten vergessen, damit sie und ihre Produkte angemessen auf neue Herausforderungen reagieren können.

Wenn in einer fortgeschrittenen Phase des Produktlebenszyklus-in der Wartung, beim Vorbereiten eines Out- oder Insourcing oder bei einer Modernisierung-Unsicherheiten auftreten, ist es üblich, mangelnde oder schlechte Dokumentation oder den Abgang von Wissensträgern für Probleme verantwortlich zu machen. Bei genauer Situationsanalyse merkt man allerdings, dass nicht nur ein „zu wenig" sondern auch ein „zu viel" von Dokumentation und Wissen schadet. Beispiele dafür sind vielfältig:

- Zeit und Energie von Business Analysten, die sie in die aufwändige Wartung von vergangenheitsorientierten, selten genutzten Dokumenten stecken, geht verloren;
- Beteiligte, die mit unnötigen oder unnötig komplexen Spezifikationen, Testfällen oder Code arbeiten müssen, verlieren den Fokus;
- langjährige Wissensträger halten an unwichtig gewordenen Spezialfällen fest oder an Arbeitsstrukturen, die früher erfolgreich waren.

N. Luhmann verortet „*Die Hauptfunktion des Gedächtnisses" „letztlich im Vergessen, im Verhindern der Selbstblockierung des Systems durch ein Gerinnen der Resultate früherer Beobachtungen" [1].* Die Erwartungen an Requirements Management haben viel Ähnlichkeiten mit den Aufgaben eines Gedächtnisses der Anwendung im Entwicklungsprozess: Veraltete Funktionalitäten, überholte Spezifikationen, aber auch unpassend gewordene Dokumentationstechniken und Prozesse sind im Sinne von Luhmann „geronnen" und blockieren die Entwicklung.

Unsere Vision ist, dass Requirements Management als „Gedächtnis" einer Anwendungsentwicklung die Kommunikation und Dokumentation so organisiert, das „Bewahren" und „Vergessen" im Gleichgewicht sind.

Im Folgenden illustrieren wir an Erfahrungsbeispielen, wo ein „Vergessen" für die Evolution eines Systems zuträglich war. Dann möchten wir ein Vorgehen vorstellen, welches in unserer Praxis entstanden ist und Entwicklungsteams dabei helfen konnte, Wissen zu fokussieren und Blockaden zu lösen. Zum Abschluss betrachten wir das Vorgehen kritisch und überlegen, wie wir es evaluieren und verbessern können.

3. Lähmungen und Blockaden durch „Vergessen" überwinden

Die folgenden Beispiele illustrieren, wie das einseitige Festhalten an Wissen Blockaden und Lähmungen verursacht hat und wie „Vergessen" helfen konnte.

Beispiel 1: Ein System, das Geschäftskorrespondenz für einige hundert Geschäftsvorfälle aus einer Hostapplikation generiert, basierte auf Code aus den 1980 -1990er Jahren. Ein Team sollte diese Applikation modernisieren. Anforderungsspezifikationen waren seit Jahrzehnten sorgfältig gepflegt, Testfälle und Code waren zur Spezifikation verfolgbar. Da das Team Inkonsistenzen von Spezifikationen zu Projektzielen entdeckte, stellte es bei ungefähr der Hälfte der Geschäftsvorfälle die Frage: „sind die Fälle noch nötig?". Die Fachseite bestand weiterhin auf die Umsetzung dieser Fälle. Bei einer automatischen Codeanalyse entdeckte das Team u.a. eine Kontradiktion (falls a<2 und a>=2), welche sogar dokumentiert war. So konnten die Analysten nachweisen, dass das System Zehntausende von Codezeilen seit mehreren Jahren in Produktion nicht durchlief. Die dazugehörigen Anforderungsspezifikationen, Testfälle und Testdaten waren dennoch jahrelang sorgfältig gepflegt worden. Trotz dieser Erkenntnis fiel es den Fachverantwortlichen schwer zu entscheiden, diese in Frage gestellten Geschäftsvorfälle abzuschaffen. Ein „Vergessen" der Geschäftsvorfälle hätte über Jahre Ressourcen gespart und eine Sanierung erst ermöglicht. **Beispiel 2:** Ein Modul eines CRM für die Eröffnung einer Kundenbeziehung implementierte komplexe Regeln, um u.a. Hypothesen zur Steuerpflicht der Kunden in verschiedenen Ländern zu erstellen. Ein fachseitiger Produktverantwortlicher und ein

Requirements Engineer (nennen wir sie „das Duo“) kannten die komplexen Funktionen und Fähigkeiten wie ihre Westentasche. Die Dokumentation erstellten die beiden zwar gemäss der Richtlinien, andere Beteiligte fanden diese aber unverständlich. Vorhaben, das Modul an neue Steuergesetze anzupassen, scheiterten, da das Duo viele Spezialfälle aus der Vergangenheit fand, die eine Umsetzung nicht realistisch erscheinen liess. Andere Beteiligte durschauten die komplexen Regeln nicht. Aus Altersgründen begann das Duo loszulassen. Gemeinsam mit ihren Nachfolgern analysierten sie die komplexen Regeln neu aus der Perspektive zukünftiger Gesetze. Eine neue Priorisierung der Anforderungen–vor allem ein neues „Schneiden“ der Regeln–konnte die Komplexität der Spezifikationen reduzieren und führte zu einer drastischen Reduktion des Codes und der Anzahl Regressionstests. Hier half ein Perspektivenwechsel, alte Beobachtungen und Handlungsstrategien zu vergessen, ein System zu vereinfachen und Weiterentwicklung zu ermöglichen.

Die Muster, die im Requirements Management zu Lähmungen und Blockaden durch „*das Gerinnen der Resultate früherer Beobachtungen*“ führen, sind vielfältig. Diese Beispiele illustrieren zwei unterschiedliche Muster aus vielen möglichen. Eine Ursachenanalyse hilft in diesen Situationen nicht, da sie meist Interpunktionsprobleme darstellen und zu Schuldzuweisungen (an Personen, Teams oder Dokumentationstechniken) führen, aber nur selten Lösungsideen hervor bringen. Wir beschreiben daher einen–aus Praxissituationen heraus–entstandenen Weg, um Blockaden zu beseitigen und das Verhältnis zwischen „Wissen bewahren“ und „Vergessen“ zu regeln.

4. „Vergessen lernen“ in vier Phasen

Die folgenden vier Phasen beschreiben wie die Teams „Vergessen lernten“.

Phase 1 (Ziele und Wissensfluss): Zuerst sammelten und identifizierten wir die aktuellen Ziele und Herausforderungen für die Anwendung und dokumentierten die Informations- und Wissensflüsse. Neben Artefakt basierten Flüssen fanden wir im Idealfall auch informelle Wissensflüsse. Am wichtigsten waren dabei die Flüsse zwischen den folgenden drei Aspekten:

a) Strategische Aspekte, d.h. langfristige wie Vision und Roadmap,
b) verändernde Aspekte, d.h. projekt- oder releaseorientierte wie agile Backlogs, CRs etc.) und
c) bewahrende Aspekte (produktorientierte wie Code, Regressionstests oder auch Use Case Spezifikationen).

Phase 2 (Checkliste entwickeln): Anhand dieser Wissensflüsse entwickelten wir gemeinsam mit wenigen Teammitgliedern eine „Checkliste“. Diese Fragen gaben den Teams einerseits Sicherheit, nichts Wichtiges zu vergessen und prüften andererseits Dokumentationen und Artefakte auf Nutzen und Notwendigkeit. Es gab zwei Designkriterien für die Fragen:

a) Der Leser der Checkliste soll möglichst viele Perspektiven einnehmen müssen (Funktion-Verhalten-Struktur, Nutzer-Entwickler-Tester, Projekt-Wartung, Überblick-Detail, beteiligte Systeme/Prozesse etc.) und
b) die Checkliste besteht aus ca. zehn Fragen.

Einer der Auftraggeber führte diese Beschränkung der Fragenzahl ein. Eine anspruchsvolle Reduktion, die sich als wichtig erwiesen hat.

Phase 3 (Doppel-Workshops): Mit Einzelpersonen oder kleinen Teams bearbeiteten wir in jeweils zwei aufeinanderfolgenden Workshops die Checkliste. Im ersten Workshop regten die Fragen zum Perspektivenwechsel an. Im zweiten Workshop generierten die Beteiligten beim Beantworten der Fragen Ideen für Massnahmen: wie z.B. einfachere Templates, andere Dokumentationen, bis hin zur Stilllegung einer Systemkomponente.

Phase 4 (Massnahmen bewerten): Die nun gesammelten Ideen bewerteten wir im Hinblick auf die in Phase 1 formulierten Ziele und Herausforderungen. Wir betrachten den Weg „Vergessen lernen“ jeweils als Erfolg, wenn die Teams Massnahmen identifizierten, welche Komplexität reduzierten und Blockaden lösten, aber auch wenn die Teams zu bewahrendes Wissen besser dokumentieren konnten.

5. Kritische Betrachtung des Vorgehens

Die Struktur des Vorgehens hat sich in der Praxis heraus kristallisiert. In Coaching-Mandaten war das Vorgehen stark durch die Bedürfnisse des jeweiligen Auftraggebers geprägt und daher uneinheitlich. Diese starke Anpassung einerseits und der Wunsch mancher Teams nach vertraulicher Behandlung der Ergebnisse andererseits machen eine systematische Evaluation nach wissenschaftlichen Kriterien schwierig. Folgende Aussagen können wir dennoch treffen:

- Wir konnten sowohl in agilen Teams als auch bei klassischen Vorgehen Massnahmen zum „Vergessen lernen“ identifizieren. Bei agilen Teams fiel es leichter als bei klassischen Vorgehen.
- Es gab Fälle, wo wir in Phase 1 die Ziele nicht identifizieren und das Diagramm nicht erstellen konnten und das Vorhaben der Prozessverbesserung abbrachen.
- Die These des Vorgehens, dass weniger manchmal mehr ist, polarisiert Beteiligte.

Gerne möchten wir wissenschaftlich Folgendes evaluieren: Nutzen und Qualität der Checklistenfragen und verschiedener Perspektivenwechsel, Nutzen und Qualität der Wissensflussdiagramme und deren Verbesserung, z.B. Cod e- und Dokumentengeneratoren einbeziehen, da sie oft eine eigenwillige Dynamik verursachen. Geplant ist, in Teams das Vorgehen anzuwenden und nach einem halben Jahr zu wiederholen und dabei auch die in der ersten Runde identifizierten Massnahmen zu evaluieren.

6. Referenzen

[1] Luhmann, Niklas (1997): Die Gesellschaft der Gesellschaft, S. 579.