qualityminds

Stephan Merkel

Der effektive Requirements Manager

Der Management 3.0 Weg des Requirements Engineers

Der effektive Requirements Manager

Der Management 3.0 Weg des Requirements Engineers

Stephan Merkel

Dieses Buch können Sie hier kaufen http://leanpub.com/erm

Diese Version wurde auf 2014-10-14 veröffentlicht

Twittern Sie dieses Buch!

Bitte unterstützen Sie Stephan Merkel, indem Sie dieses Buch auf Twitter weiterempfehlen!

Hier ein Vorschlag für einen Tweet:

Gerade von Leanpub runtergeladen: Der effektive Requirements Manager

Vorschlag: Verwenden Sie den folgenden Hashtag, wenn Sie über dieses Buch twittern: #WegdeseRM.

Was sagen Andere über dieses Buch? Klicken Sie hier, um nach diesem Hashtag auf Twitter zu suchen:

https://twitter.com/search?q=#WegdeseRM

Ebenfalls von Stephan Merkel

Wir wollen bauen!

Inhaltsverzeichnis

Effektives Requirements Management **1**
Professionalität fordert Werte, Handwerkszeug und bewusste Auseinandersetzung 1
Wen meinen wir mit "Requirements Manager"? . . . 2
Requirements Engineer, Requirements Manager oder Requirements Developer? 5
Kompatibilität mit Prozessen und Methodiken 6
Vorbilder 6
Was hat es mit Management 3.0 zu tun? 8
FAQ 9

Effektives Requirements Management

In der Ausgabe 2014-03 des RE-Magazine des IREB[1] ist der Leitartikel tituliert: “Project Value Delivered - The True Measure of Requirements Quality”. Genau darum geht es uns beim effektiven Requirements Management. Das Qualitätskriterium, das uns interessiert, ist der Effekt, also die Wirkung, die die geschaffene Software entfaltet, und der Wert, den sie damit stiftet. Die Nabelschau, ob die Anforderungen korrekt, vollständig, konsistent usw. sind, greift uns als Kriterium für gute Arbeit zu kurz.

Professionalität fordert Werte, Handwerkszeug und bewusste Auseinandersetzung

> Softwareentwicklung braucht Profis.

Das ist ein Zitat von der Clean Code Developer Startseite[2].

Der Grundgedanke, dass zu einer professionellen Arbeitsweise ein **Wertesystem** und die **bewusste Auseinandersetzung**

[1] http://re-magazine.ireb.org/issues/2014-3-gaining-height/

[2] http://clean-code-developer.de/

in Form von gezielter, regelmäßiger Reflexion gehören, ist auf das Requirements Engineering übertragbar – ja, auf alle Disziplinen.

- Die bewusste Auseinandersetzung bedeutet, ungeachtet der Bestätigung von außen, immer zu hinterfragen und besser werden zu wollen – und dafür gezielt etwas zu unternehmen.
- Das Wertesystem ist der Kompass, der uns dabei leitet. Ich zitiere wieder Stefan Lieser und Ralf Westphal[3]:

  > Gegen dieses Wertesystem prüft [der Softwareentwickler] seine Ergebnisse und Handlungen. Nur, wenn seine Arbeit diesem Wertesystem entspricht, empfindet er sie als gut getan, als professionell. Sein Streben ist es daher, auch unter widrigen Umständen, unter Druck von Kunden oder Herstellern, diesem Wertesystem treu zu sein.

Mit diesen Seiten möchten wir das Pendant zu CCD für das Requirements Engineering zur Verfügung stellen.

Wen meinen wir mit “Requirements Manager”?

Es gibt schon so viele Definitionen von Requirements, Requirements Engineering und Requirements Management. Wir wagen dennoch eine weitere. Denn die spröden IEEE und DIN Definitionen eignen sich nicht um auszudrücken, was uns eint.

[3] http://clean-code-developer.de/Team.ashx

Die Definitionen enthalten Vorwärtsreferenzen – eine Sünde in Spezifikationen. Es handelt sich um Referenzen auf die Bücher, die weiter unten im Abschnitt "Vorbilder" erwähnt werden. Die Frage aus Sicht des Lesers "Bin ich gemeint?" finden wir wichtiger als die Frage, wer uns als Vorbild dient. Also ein Konflikt zwischen *Empfängerorientierung* (das Wichtigste zuerst) und *Keine Vorwärtsreferenzen*. Wir geben der Empfängerorientierung Vorrang.

Anforderungen (Requirements)
: sind Vorstellungen, die Menschen vom Softwaresystem haben. Sie haben sie schon, bevor das System existiert, und sie werden vom System beeinflusst, sobald es da ist. Artikulierte Anforderungen, gesprochen oder aufgeschrieben, sind präskriptive Aussagen darüber, wie das System sein und wie es die Anwendungsdomäne beeinflussen soll. Wenn das System schließlich da ist, kann es den Vorstellungen auch zuwiderlaufen oder nicht die gewünschte Wirkung entfalten. Das heißt, die Vorstellung ist immer stärker als das System selbst und die Dokumentation. Anforderungen gelten als veränderlich, im Gegensatz zu den Rahmenbedingungen (Contraints), die als unveränderlich gelten.

Das "System"
: ist übrigens das Softwaresystem, wenn nicht ausdrücklich von etwas anderem die Rede ist. (Das "System" im Sinn von van Lamsweerde (2009) nennen wir gerne "die Welt".)

Anforderungsspezifikation (Requirements Specification)
ist, wenn sie üppig ausfällt, mehr oder weniger die Sammlung der in der Volere[4] Vorlage aufgeführten Inhalte. Grob gesagt gehören dazu Analyseergebnisse wie die Ist-Situation, die Ziele und die Anforderungen im engeren Sinn, sowie Lösungsansätze und Modelle der zukünftigen Welt. Sie muss nicht als geschlossenes Dokument vorliegen, vollständig sein oder auf einmal erstellt werden. In irgendeiner Form gibt es sie aber in aller Regel. Neben Volere gibt es weitere Referenzmodelle, zum Beispiel die Pyramide in Gottesdiener (2005) oder AMDiRE[5]. Jedes hat seine Berechtigung. Die genaue Abgrenzung ist nicht so wichtig. Wenn die Anforderungsspezifikation schlank ausfällt, ist sie vielleicht eine Sammlung von User Stories oder von Use-Case 2.0 Kernelementen.

Requirements Engineer
ist jeder, der Anforderungen für Andere artikuliert, und dabei systematisch vorgeht, wie es sich für einen Ingenieur gehört. Die "Anderen" nennt man Stakeholder, weil die Software, die entwickelt werden soll, sie etwas angeht. Die Stakeholder im weiteren Sinn sind die Zielgruppe, für die der RE arbeitet. Manche von ihnen möchten in seiner Arbeit ihre Willenserklärungen wiederfinden, andere, wie Entwickler und Tester, erwarten von ihm Vorgaben für ihre eigene Arbeit. Je nach Schwerpunkt der Tätigkeit spricht man neben Requirements Engineer auch vom Business Analyst, Anforderungsmanager, Systemanalyst, Program Manager[6] usw.

[4] http://www.volere.co.uk

Systemspezifikation (System Specification)
: sind die Artefakte, die schon stark lösungsorientiert sind und den Entwicklern als Vorlage dienen. Auch diese Artefakte sind noch im Scope des eRM.

Requirements Engineer, Requirements Manager oder Requirements Developer?

Wir sprechen vom effektiven Requirements *Manager*. Zum Einen aus dem weiter unten zum Thema Management genannten Grund. Noch mehr aber, weil uns der Aspekt *Feedback* im Sinne der Effektivität so wichtig ist. Wer nur 180° Feedback erhält, nämlich nur vom Analyse-Umfeld, aber weder von den Tekkies noch von den Anwendern, hat nur eingeschränkte Möglichkeiten zu lernen. Requirements *Engineering* klingt uns zu sehr nach Einbahnstraße. Da wir uns mit Entwicklern und Testern verbrüdern, sind wir vielleicht Requirements Developers. Aber "Developer" klingt nach Black Box und für Kritiker der agilen Softwareentwicklung nach "going off and doing vague stuff with promises that it'll be done when it's done" (Martin Fowler auf ThoughtWorks: The Purpose of Estimation[7]).

Bleiben wir also beim effektiven Requirements Manager. Das sind ohnehin eher philosophische Überlegungen, die man nicht überbewerten sollte. Wer sich Requirements Engineer oder Business Analyst nennt, muss daran nichts ändern. (Die Architekten

[5] http://www4.in.tum.de/~mendezfe/openspace.shtml

[6] So nennt Joel Spolsky die Rolle in Painless Functional Specifications - Part 3, was aus seiner Zeit bei Microsoft stammt.

[7] http://www.thoughtworks.com/de/insights/blog/purpose-estimation

und Entwickler haben es hier einfacher, kann das sein?)

Kompatibilität mit Prozessen und Methodiken

Damit ist auch gesagt, dass wir ein Umfeld suchen, in dem 360° Feedback möglich ist, zum Beispiel agile Teams. Agile Softwareentwicklung ist aber keine Voraussetzung. Wir meinen, dass die Werte und Prinzipien nicht so stark davon abhängen, wie das Team vorgeht. Manches setzt allerdings Iterationen voraus, die höchstens ein paar Monate dauern.

Im Unterschied zu Vorgehensmodellen und Methodiken sagen wir nicht, welche Arbeit getan werden muss. Wir gehen davon aus, dass das im Projekt klar und in der Literatur ausreichend beschrieben ist, und dass es getan wird. Uns beschäftigt, *wie* wir die Arbeit machen wollen. Das, was wir unter gut getaner Arbeit verstehen, drückt sich vor allem in den Ergebnissen aus. Die Prinzipien sind etwas wie die 11 Gebote des eRM, die wir uns selbst auferlegen. Es gibt keine feste Reihenfolge, nur einen Vorschlag, um sie Schritt für Schritt zu vertiefen. Jede andere zweckmäßige Reihenfolge kann gewählt werden.

Vorbilder

CCD ist unser Vorbild dafür, wie man selbst-gesteuertes Lernen in Organisationen implementiert. Die Struktur des Leitfadens und auch einige Praktiken haben wir uns dort abgeschaut.

Inhaltlich ist es schwieriger. Es gibt viel gute Literatur zum RE. Leider gibt es nicht das *eine* Buch. Einige Quellen für Prinzipien, Kriterien und Praktiken sind (nicht abschließend):

- Ivar Jacobson, Ian Spence und Kurt Bittner. *Use-Case 2.0.* @WWW. – Mehrere Parallelen zu unserem Ansatz, insbesondere geleitet von Prinzipien und iterativem Vorgehen, fokussiert auf Geschäftswert und integriert mit dem Test, jedoch ist eRM weiter gefasst und weniger invasiv. Man kann eRM auch ohne UC 2.0 machen.
- Suzanne Robertson und James Robertson. *Mastering the Requirements Process.* – Dieses Buch mögen wir besonders gern, weil es starke Werte transportiert und einige Kriterien und Praktiken zum eRM beigetragen hat.
- Ellen Gottesdienner. *The Software Requirements Memory Jogger.* GOAL/QPC, 2005. – Eines der besten Nachschlagewerke.
- Klaus Pohl und Chris Rupp. *Basiswissen Requirements Engineering.*
- Chris Rupp und die SOPHISTen. *Requirements Engineering und -Management.*
- Axel van Lamsweerde. *Requirements Engineering.* Wiley, 2009. – Grandios in Präzision und Vollständigkeit – und schwere Kost.
- Dean Leffingwell. *Agile Software Requirements.* Addison-Wesley, 2011. – Eines der ersten Bücher, die das Thema nicht nur als Randerscheingung behandeln, und eines, das uns gut gefällt.
- Michael Jackson. *Software Requirements and Specifications.* Addison-Wesley, 1995. – In unserer Sammlung das einzige Buch, das ein starkes Fundament für das Streben nach Einfachheit, Präzision und Problemanalyse vor Lösung liefert.

Was hat es mit Management 3.0 zu tun?

Wenn Sie Management 3.0 nicht kennen, kann das so bleiben. Es ist für unsere Sache nicht so wichtig.

Wenn Sie Management 3.0 kennen, sich aber nicht angesprochen fühlen, ist das auch kein Problem.

Wenn Sie Management 3.0 aber kennen und etwas Positives damit verbinden, dann werden Sie eRM mühelos verstehen.

Bei Management 3.0[8] geht es darum, bei sich selbst damit zu beginnen, die Art der Zusammenarbeit zu verändern. Jetzt gleich. Und genau das wollen wir erreichen, bezogen darauf, wie das Requirements Management im Team gelebt wird.

Es gehören zwei dazu: Der Requirements Manager, der es praktiziert; und die Organisation, die es ihn praktizieren *lässt*. Denn die Zeit für die Reflexion muss zur Verfügung stehen.

Jurgen Appelo schreibt in seinem Buch Management 3.0 Workouts[9]:

> A management practice is a good practice when:
>
> 1. It engages people and their interactions;
> 2. It enables them to improve the system;
> 3. It helps to delight all clients.

Die Idee, dass es funktioniert, wenn wir dem Einzelnen einen Kanon von Werten, Prinzipien und Praktiken geben, stammt nicht von uns. Die Erfinder von Clean Code Developer kamen schon viel früher darauf. Danke!

[8] http://www.management30.com/

[9] http://www.management30.com/workouts/

Wir wollen mit diesen Seiten diejenigen ansprechen, die RE praktizieren, nicht die Manager. Wenn Sie Manager sind, möchten wir Ihnen Mut machen, dem Team den Raum für effektives Requirements Management zu geben. Wir können sogar Argumente liefern, allerdings tun wir das nicht hier, eben weil sich diese Seiten an die Praktizierenden richten.

Wir sehen es aber auch so: Wer den Schritt tut, eRM zu praktizieren, trifft eine Entscheidung, wie er mit seiner Zeit umgehen will. Und das ist Management!

FAQ

Braucht man zum Auftakt eine Schulung?

Nein. Sie können morgen damit anfangen. Bei Gruppen hilft es, einen Organisator zu haben, der die Themen für die gemeinsame Reflexion auswählt und jemanden für die Vorbereitung gewinnt.

Gibt es eine Zertifizierung?

Nein. Bei einer Zertifizierung wird eine Momentaufnahme vom Wissen gemacht. Bei den meisten Zertifizierungen genügt es, auf die Prüfung Sachen auswendig zu lernen. Man braucht die Inhalte nur zu kennen. Wir wollen sie aber verinnerlichen. Wir setzen die Ausbildung voraus, bzw. wir verlassen uns darauf,

dass die Reflexion für selbst-gesteuertes Lernen genutzt wird. Interessant ist dann die *Umsetzung im Alltag*. Auf die kommt es uns an. Und darauf, nie stehen zu bleiben. Das Gefühl, mit der Ausbildung fertig zu sein, das eine Zertifizierung vermittelt, halten wir gerade nicht für förderlich. (Ob man sich das Wissen bescheinigen lässt, wenn man es denn schon hat, da mischen wir uns natürlich nicht ein.)

Wie verhält sich eRM zum Wissensschatz von IREB oder zum BABOK?

Im Großen und Ganzen wollen wir einen Weg aufzeigen, wie man als Einzelner oder im Team diese Wissensschätze durchstreift, um sie zu verinnerlichen. Wir sehen uns nicht als Konkurrenz zu IREB & Co, sondern als Katalysator. Wir meinen, dass es nur Schätze sind, wenn man sie ausgräbt. Das passiert aber zu wenig, wenn man es dem Zufall überlässt.

Wir erheben weder den Anspruch, diese Wissensschätze vollständig kartographiert zu haben, noch in allen Punkten gleicher Meinung zu sein. Das eRM ist auch geprägt von unserem persönlichen Geschmack. Und es steht jedem frei, es nach seinem persönlichen Geschmack zu verändern. In den Wissensschätzen sind die Praktiken etwas untergewichtet, weil viele davon Soft Skills erfordern und jeder seinen persönlichen Stil finden muss. Allerdings gehen wir schon davon aus, dass man nach einem oder zwei Durchläufen die CPRE Prüfungen mühelos besteht. Sollten Sie hier auf einen blinden Fleck stoßen, teilen Sie es uns bitte mit!

Wie kann ich das ganze Team dazu verpflichten?

Gar nicht. Es funktioniert nur als Selbstverpflichtung. Die Erfahrung mit CCD hat aber gezeigt, dass neue Teammitglieder nicht außen vor bleiben wollen. Wenn die anderen es praktizieren, wirkt es in der Regel ansteckend.

Und wann ist man fertig?

Nie. Wenn man alle Grade durchlaufen hat, beginnt man von Vorne. Es wird Neues zu lernen und zu entdecken geben. Im Lichte der neuen Erfahrungen werden wir anderen Dingen größere Aufmerksamkeit schenken. Die Teamzusammensetzung wird sich ändern, wir werden Anderes von anderen Menschen lernen. Das ist ja das Schöne.

Was, wenn ich bestimmte Kriterien oder Praktiken schlecht finde?

Weglassen, natürlich. Es gibt keine Dogmen. Am besten uns eine Email schreiben um uns mitzuteilen, warum wir uns an der Stelle geirrt haben.

Wer hat es schon erfolgreich eingeführt? Gibt es schon Erkenntnisse zum RoI?

Okay, vielleicht sollten Sie doch das Buch von Appelo lesen… Das effektive Requirements Management ist ja noch ganz neu. Aber wenn es Sie beruhigt: CCD ist inzwischen weit verbreitet. Ein Qualitätsmanager bei Siemens Healthcare bat mich dereinst,

es der auf der ganze Welt verteilten Entwicklungsgemeinde vorzustellen. Von Erfahrungen mit eRM werden wir jedenfalls berichten!