Die richtigen Rahmenbedingungen für ein Entwicklungsteam

Editorial

Liebe Manager!

Heute möchte ich eine Lanze für unsere Entwickler brechen. In der Beratung werde ich sehr oft in kritische Situationen geholt: Zugesagte Termine werden nicht eingehalten, das Produkt hat trotz jahrelanger Entwicklungsarbeiten immer noch viele Fehler und wichtige Features sind nicht fertig.

Wir anaylsieren dann immer sehr genau, was die Ursachen für die missliche Lage sind[1]. Immer wieder stellen wir dabei fest, dass die Ursache für die Misere nicht bei schlecht qualifizierten oder unmotivierten Entwicklern liegt. Natürlich kommt das auch vor. Aber sehr oft liegt die Ursache darin, dass die Entwickler (und Tester, Architekten etc.) gar nicht die Chance haben, gute Arbeit zu leisten. Sie müssen dermaßen viel Zeit und Energie aufwenden, um mit den gegebenen Rahmenbedingungen zurecht zu kommen, dass oft kaum mehr Zeit für die eigentliche Entwicklung bleibt.

Liebe Manager, Geschäftsführer, Teamleiter! Dieser Knowledge Letter ist ein Appell an uns Führungskräfte, einmal innezuhalten und die Welt mit den Augen unserer Teams zu betrachten. Haben unsere Team wirklich alles, was sie brauchen, um gut arbeiten zu können? Was habe ich im letzten Jahr dafür getan? Wie sieht das aus Sicht der Teams aus? Und könnte ich unter diesen Umständen produktiv sein und ein tolles Produkt bauen? Lassen Sie uns auf den folgenden Seiten diesen Fragen gemeinsam nachgehen…

Markus Unterauer
Berater, Trainer

[1] Mit unserem Produkt „Potenzialanalyse" zeigen wir schnell und effizient Probleme auf und geben Empfehlungen zu Verbesserung.

Damit Softwareentwicklung gut läuft, braucht es zwei Dinge: Das richtige Team und die richtigen Rahmenbedingungen. Passen die Rahmenbedingungen nicht, kann auch das beste Team nichts Ordentliches produzieren. Was aber gehört zu diesen Rahmenbedingungen dazu? Im folgenden Artikel geben wir konkrete Antworten.

Um erfolgfreich zu sein braucht es den richtigen Rahmen

Softwareentwickler können im Schnitt nur 12-50% für tatsächliche Softwareentwicklung aufwenden [1]. Der Rest geht für Kommunikation, Meetings, Organisation und vor allem dem Nachlaufen nach fehlenden Informationen drauf. Fast zwei Drittel dieses ohnehin sehr geringen Anteils arbeitet das Team an Features, welche die Kunden gar nicht oder anders brauchen [2]. Tatsächlich produziert ein Entwicklungsteam also nur in 7-30% seiner Zeit echten Wert für den Kunden. Sieht man sich das an, so muss eines der wichtigsten Ziele jedes Managers sein, dass 1) Entwickler mehr Zeit für tatsächliche Softwareentwicklung bekommen und 2) in dieser Zeit die richtigen Sachen entwickelt werden.

Die Ursachen für die Schieflage in der Produktivität liegen nun oft nicht beim Team selbst. Fehlende oder falsche Rahmenbedingungen und Vorgaben verhindern, dass das Team produktiver ist.

$$Erfolg = Team + Rahmenbedingungen$$

Viele Management-Ratgeber greifen das auf und postulieren, dass man als Manager zuerst einmal die richtigen Rahmenbedingungen herstellen soll. Auch in den Prinzipen agiler Software-Entwicklung ist das verankert. Prinzip Nummer 5 beschreibt das so: „Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen." [3] Als Manager muss ich also dem Team das Umfeld und die Unterstützung geben, das sie benötigt und sie dann in Ruhe arbeiten lassen. Aber was bedeutet das? Was gehört alles zu diesem Umfeld, zu diesen Rahmenbedingungen? Was braucht das Team, um produktiv arbeiten zu können und was muss ich als Manager alles gestalten? Die wohl einfachste Art herauszufinden, was das Team benötigt ist, es einfach zu fragen. Wer weiß besser was er braucht als die jeweilige Person bzw. das Team selbst? Oft liefert die Frage nach dem „Was braucht ihr?" aber überraschend wenige Antworten. Vielfach nennt das Team nur die 2-3 Dinge, die in dem Moment gerade am meisten stören oder fehlen. Aber was ist mit dem Rest?

Notwendige Rahmenbedingungen

Folgende Checkliste zeigt, was ein Software-Team braucht, um produktiv arbeiten zu können:

1. **Die richtigen Mitglieder**: Als Manager muss ich dafür sorgen, dass mein Team die richtigen und ausreichend Mitglieder hat. Selten aber doch muss man auch Mitglieder aus dem Team nehmen, wenn sie die Produktivität nach unten ziehen. Fallen Teammitglieder aus, so muss rechtzeitig für Ersatz gesorgt werden. Können Aufgaben nicht vom Team selbst erledigt werden, muss ich als Manager externe Unterstützung organisieren, damit das Team seine Ziele wieder erreichen kann.
2. **Ausreichend Zeit und Fokus**: Zum einen braucht das Team ausreichend Zeit, um qualitativ hochwertige Produkte mit geringer technischer Schuld produzieren zu können. Zum anderen braucht jedes Teammitglied ausreichend Zeit für das Projekt. Als Manager muss ich sicherstellen, dass die Teammitglieder ausreichend Zeit für das Projekt bekommen, sich darauf fokussieren können und für das Projekt insgesamt ausreichend Zeit geplant wird.
3. **Skills in Software Entwicklung**: Als Manager muss ich gemeinsam mit dem Team analysieren, welche Fähigkeiten benötigt werden und wie diese erlangt werden können. Dazu gehört eine solide Ausbildung in den relevanten Programmiersprachen und modernen Programmiermethoden, gutes Grundwissen in Architektur und Design, Kenntnisse aktueller Software-Engineering Techniken wie Continuous Integration, Branching/Merging, Code-Reviews etc. und ein Basiswissen über manuelles und automatisiertes Testen von Software.

   Als Manager muss ich dafür sorgen, dass die entsprechenden Schulungen, Bücher, Videos, Coaches, Trainer und Berater zur Verfügung gestellt werden. Die Software Quality Lab Academy bietet z.B. in allen relevanten Bereichen des Software-Engineerings eine solide Grundausbildung, sowie Vertiefungs- und Expertentrainings an. [4]
4. **Domänenwissen über das Business**: Als Manager muss ich dem Team möglichst viel Gelegenheit verschaffen, etwas über das Business und die Fachdomäne zu lernen. Dazu kann ich Experten einladen, das Team zum Kunden rausschicken, Schulungen organisieren etc.
5. **Wissen über Legacy-Systeme**: Soll das Team bestehende Legacy-Systeme weiter pflegen, muss ich als Manager dafür sorgen, dass entsprechendes Know-How aufgebaut wird. Auch hier kann ich Experten holen, Schulungen organisieren und dem Team ausreichend Zeit einräumen, sich die alten Systeme anzusehen und „Software Archäöologie" zu betreiben.
6. **Die Möglichkeit, irgendwo Experte zu werden**: Jeder möchte irgendwo richtig gut sein und als Experte geachtet werden. Gemeinsam mit dem Team wird geplant, was das Spezialgebiet des Teams und jeder Person darin ist. Natürlich bedeutet das nicht, dass man nichts anderes mehr machen kann. Aber als Manager muss ich ermöglichen, dass sich das Team in ein Gebiet richtig vertieft und dort Experten-Know-How aufbaut.
7. **Geeignete Arbeitsmittel**: Als Manager muss ich dem Team alle notwendigen Arbeitsmittel zur Verfügung stellen. Diese müssen auch hochwertig und performant sein. Kein Holzfäller kann mit einer stumpfen Axt produktiv arbeiten, genauso wenig kann ein Entwickler mit einem langsamen PC, einer instabilen Internetverbindung oder einem langsamem Build Server effizient arbeiten. Die wesentlichen Arbeitsmittel für ein Softwareteam sind:

   - Hardware (Rechner, Monitor, Server, W-LAN, …)
   - Arbeitsplatz (ein gemeinsames Büro für das gesamte Team, Tisch, Sessel, Regale, Küche, …)
   - Moderationsmaterialien für Meetings und Meetingräume mit Whiteboard und Flipchart
   - Entwicklungsumgebung und Entwicklungstools (Source-Control, CI, Statische Analyse, Build System, Testautomatisierung, …)

Themenbereich	Aufgabe	Skill	Product Owner	Entwickler	SW-Architekt	Tester
Projektmanagement	Portfolio Management	Roadmap Planung	*			
Projektmanagement	Portfolio Management	Priorisierungstechniken (projektübergreifend)	*			
Projektmanagement	Projektleitung	Projektvorgehensmodelle	*	*	*	*
Projektmanagement	Projektleitung	Teamzusammenstellung	*			
Projektmanagement	Projektleitung	Projektplanung	**	*	*	*
Projektmanagement	Projektleitung	Aufgabenmanagement	*	*	*	*
Projektmanagement	Projektleitung	Controlling	**	*		*
Projektmanagement	Projektleitung	Reporting	**	*		*
Projektmanagement	Projektleitung	Priorisierungstechniken (im Projektes)	****	*	*	*
Projektmanagement	Projektleitung	Koordination von externen Partnern	***			
Fachwissen	Applikationen	Fachliches Applikations-Know-How	***	**	***	****
Fachwissen	Applikationen	Technologische Möglichkeiten der Applikation	**	****	****	**
Qualitätssicherung	Testmanagement	Testprozess / Testplanung	*	*		***
Qualitätssicherung	Testmanagement	Aufgabenmanagement (Testaufgaben)		*		***
Qualitätssicherung	Testdurchführung	Testfallentwurf (Entwurfsverfahren)	*	*	*	****
Qualitätssicherung	Testdurchführung	Manuelles Testen	**	**		****
Qualitätssicherung	Testdurchführung	Testautomatisierung		**	*	***
Qualitätssicherung	Reviews	Spezifikations-Reviews		**	**	***
Qualitätssicherung	Code-Qualität	Code Reviews		**	****	*
Qualitätssicherung	Code-Qualität	Statische Analyse		**	****	*

Abbildung 1 - Eine Skill-Matrix zeigt die im Team notwendigen Fähigkeiten

- Anforderungs- und Testmanagement-Tools
- Aufgabenmanagement- und Zeiterfassungssystem
- Kommunikationstools (äußerst wichtig bei verteilten Teams!), Wiki

8. **Support für die Infrastruktur**: Wenn mal etwas kaputtgeht oder ein Teil der Infrastruktur nicht funktioniert, braucht das Team Ansprechpartner, die ihnen schnell weiterhelfen. Dies reicht von Aufsperren von gesperrten Benutzerkonten bis zum Aufsetzen neuer Testserver.

9. **Passende Technologien, Komponenten und Frameworks**: In vielen Unternehmen ist ein grundsätzlicher Technologie Stack bereits vorgegeben (Java vs. .NET, Windows vs. Linux, Oracle vs. SQL Server, etc.). Dieser Technologie Stack muss zur Aufgabenstellung im Projekt passen. Während der Entwicklung werden dann vielfach zugekaufte Frameworks und Komponenten verbaut. Diese müssen ausgewählt und gekauft werden. Als Manager muss ich für eine schnelle Beschaffung sorgen und sicherstellen, dass die gewählten Komponenten und Frameworks den Unternehmensrichtlinien entsprechen.

10. **Ein Ziel (Scope, Zeit, Qualität)**: Nichts ist so motivierend wie ein gemeinsames Ziel. Zum Ziel gehört die inhaltliche Dimension (Vision, Scope), die zeitliche Dimension (Meilensteine, Release-Termine) und die Qualitätsdimension (gefordertes Qualitätsniveau). Als Manager muss ich sicherstellen, dass das Team immer ein Ziel hat und alle gemeinsam darauf hinarbeiten. Das Team muss den Zweck kennen, warum es da ist und was sein Beitrag zum Unternehmenserfolg ist. Wichtig ist auch, dass das Team nicht einfach nur eine Story nach der anderen aus einem endlosen Backlog abarbeitet, sondern dass es immer das Big Picture kennt, also die großen Themen auf der Roadmap, auf die gerade hingearbeitet wird.

11. **Gute Anforderungen**: Schlechte Anforderungen sind die häufigste Ursache für Probleme in der Softwareentwicklung [1]. Als Manager muss ich sicherstellen, dass das Team rechtzeitig ausreichend Anforderungen in hoher Qualität bekommt. Bei Fragen müssen Ansprechpersonen jederzeit erreichbar sein und sich dann auch tatsächlich darum kümmern, Antworten zu liefern. Zu den Anforderungen gehören fachliche Anforderungen aus dem Business aber auch Anforderungen der Organisation selbst und rechtliche Anforderungen (Gesetze, Normen). Fehlen diese Vorgaben, senkt dies die Produktivität deutlich.

12. **Schnelle und klare Entscheidungen**: Viele Entwicklungsteams müssen ständig auf Entscheidungen von außen warten. Seien es fachliche Entscheidungen, wie einzelne Features aussehen sollen, Infrastrukturentscheidungen, technische Entscheidungen zu Umsystemen, Entscheidungen zu Prozessen und Organisation, etc. Als Manager muss ich dafür sorgen, dass Entscheidungen entweder von außen schnell und klar getroffen werden, oder dem Team die Kompetenz geben, selbst zu entscheiden. Achtung! Soll das Team selbst entscheiden, braucht es dafür das entsprechende technische und fachliche Know-How (siehe oben). Wer was entscheidet, kann dann beispielsweise mithilfe der „7 Levels of Autority" von Jurgen Appelo ausgemacht und beschrieben werden [5].

13. **Klare Zuständigkeiten**: Immer wieder müssen Teams viel Zeit investieren, um herauszufinden, wer für ein bestimmtes Thema, z.B. ein Produkt oder einen Fachbereich, zuständig ist. Vielfach verläuft diese Suche ergebnislos, niemand ist zuständig. In diesen Fällen sind die Entwickler gezwungen, selbst zu entscheiden, wenn sie überhaupt vorankommen wollen, auch wenn ihnen eigentlich das notwendige Wissen fehlt. Als Manager muss ich sicherstellen, dass es für alle relevanten technischen, fachlichen und organisatorischen Themen klare Zuständigkeiten entweder im Team selbst oder außerhalb des Teams gibt. Diese zuständigen Personen müssen ihre Verantwortung auch wahrnehmen.

14. **Abstimmung mit anderen**: In größeren Organisationen ist es wichtig, dass das Team sich mit anderen Teams und den Stakeholdern abstimmen darf. Dies muss koordiniert werden, beispielsweise indem regelmäßige Meetings oder Sprechstunden definiert werden.

15. **Transparenz**: Als Manager muss ich dafür sorgen, dass das Team jederzeit den aktuellen Status der Zielerreichung und andere relevante Dinge sehen kann. Am Besten geht dies, wenn man Metriken für inhaltlichen Fortschritt (z.B. Earned Value, Burndown), zeitlichen Fortschritt (z.B. Meilensteinplan) und Qualität (z.B. Bug-Backlog-Size, Escaped Defects, Bugs-per-Severity) definiert und jederzeit sichtbar macht.

	Manager decides						Team decides
Decision	**1. Tell**	**2. Sell**	**3. Consult**	**4. Agree**	**5. Advice**	**6. Inquire**	**7. Delegate**
Define salary and bonus	x						
Hire / Fire team members			x				
Select tools					x		
Select features					x		
Create schedule						x	
Set budget	x						
Document standards							x

Abbildung 2 - Wer entscheidet was? 7 Levels of Authority

16. **Stabile Prioritäten**: Als Manager muss ich schauen, dass das Team begonnene Sachen auch fertig machen darf. Tägliche ändernde Prioritäten, die das Team zwingen, ständig halbfertige Sachen liegenzulassen und etwas anderes anzufangen, killen nicht nur die Produktivität des Teams, sondern auch die Qualität des Produktes.

17. **Gleichmäßiger Arbeitstakt**: Im Sinne der Nachhaltigkeit und Berechenbarkeit braucht es eine gleichmäßige Auslastung, einen berechenbaren Arbeitstakt, z.B. in Form von Sprints und eine stabile Release-Cadence. Zu einem gleichmäßigen Arbeitstakt gehört auch, dass alle Teammitglieder möglichst große Überschneidungen in den Anwesenheitszeiten haben.

18. **In Ruhe arbeiten können**: Als Manager muss ich dem Team ermöglichen, ruhig und fokussiert zu arbeiten. Ich muss das Team vor Störungen schützen. Typische Störungen sind Bugs und Supportaufgaben, Zwischenfragen von anderen Teams, Aufwandsschätzungen und Konzepte für

zukünftige Features oder eine ständige Unruhe wie sie in Großraumbüros herrscht. Vielfach erlebe ich, dass Manager selbst das Team oft unterbrechen, Teammitglieder für Sonderprojekte heranziehen, Meetings einberufen etc. Das ist nicht gut und senkt Produktivität und Qualität.

19. **Jemand, der sich um sie kümmert**: Jedes Team braucht jemanden, der sich um das Team kümmert und der sich echt bemüht, dass es gut arbeiten kann. Dieser „Kümmerer" hat immer ein offenes Ohr für alle Sorgen und Anliegen des Teams. In agilen Organisationen teilen sich diese „Kümmerer" Rolle oft Scrum Master und Manager.

20. **Ein positives Arbeitsumfeld**: Dies beginnt bei einer offenen Unternehmenskultur, wo sich jeder freundlich grüßt, umfasst gemeinsame Rituale wie zusammen Mittagessen und geht bis zu kleinen Goodies, wie Getränke, Kaffee und Obst, die das Unternehmen zur Verfügung stellt. Wichtig ist auch, dass jeder über wichtige Veränderungen im Unternehmen informiert ist. All das kann und muss ich als Manager aktiv gestalten.

21. **Feedback**: Als Manager muss ich dem Team und jedem einzelnen Feedback geben. Sowohl zur fachlichen Arbeit, als auch zur persönlichen Entwicklung, Fähigkeiten und Eigenschaften. Ich muss auch dafür sorgen, dass das Team möglichst direktes Feedback von den Stakeholdern, vor allem von Kunden und Anwendern, erhält.

22. **Sicherheit**: Als Manager muss ich dafür sorgen, dass sich niemand verletzt und dass die Arbeitsplätze so gestaltet sind, dass niemand davon krank wird, z.B. durch schlechte Möbel oder trockene Luft. Datenschutz und Informationssicherheit sind ebenfalls Themen, auf die ich ein Auge haben muss. Als Manager muss ich deshalb im Sinne der nachhaltigen Erhaltung der Arbeitsleistung immer darauf achten, dass meine Mitarbeiter nicht überfordert sind und in ein Burnout rutschen. Manche Mitarbeiter sind so begeistert von ihrem Job, dass sie sich selbst überfordern. Sie müssen vor sich selbst geschützt werden. Interessanterweise erlebe ich in Unternehmen immer wieder auch das Gegenteil, nämlich dass ein Team nichts zu tun hat. Beispielsweise weil zu Beginn eines Releases das Produktmanagement noch keine Features spezifiziert hat. Es entsteht unproduktiver Leerlauf, der bei längerer Dauer zu einem Boreout führen kann. Als Manager muss ich das Team sowohl vor permanenter Überlastung als auch vor Unterlast schützen. Zuletzt muss ich das Team noch vor eskalierenden Konflikten, die bis hin zu Mobbing gehen können, schützen.

23. **Ein ordentliches Gehalt**: Geld ist zwar nur ein Hygienefaktor, trotzdem muss das Gehalt passen. Mitarbeiter, die sich unterbezahlt fühlen, schrauben ihre Leistung zurück, leider oft ohne etwas zu sagen. Als Manager muss man daher immer wieder mit den Mitarbeitern sprechen und für eine angemessene Entlohnung sorgen.

24. **Karrierepfad**: Als Manager muss ich meinen Mitarbeitern eine Perspektive geben, wo sie sich inhaltlich und finanziell hin entwickeln können.

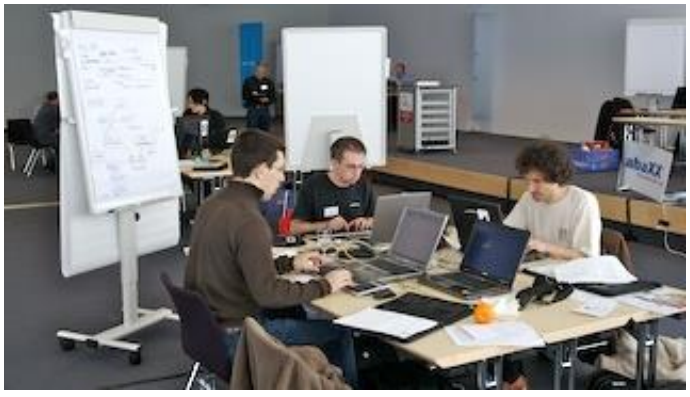
	Level 1	Level 2	Level 3	Level 4
Development Path	Developer	Senior Developer	Architect	Enterprise Architect
Quality Path	Tester	Senior Tester	Test Manager	Enterprise Quality Coach
Product Path	Requirements Engineer	Product Owner	Product Manager	Portfolio Owner
Coaching Path	Scrum Master	Senior Scrum Master	Agile Coach	Enterprise Agile Coach

Abbildung 3 - Gleichwertige Karrierepfade für alle Disziplinen

25. **Vertrauen und Freiheit**: Ist das Team richtig zusammengesetzt und hat es alles was es braucht, so wird es gute Arbeit leisten. Darauf muss man als Manager vertrauen. Das bedeutet, das Team in Ruhe arbeiten lassen, nicht vorschreiben, wie es seine Aufgaben zu erledigen hat und seine eigenen Entscheidungen treffen lassen. Gerade in agilen Organisationen ist Micro-Management ein absolutes No-Go.

26. **Erfolge**: Vielleicht kennen Sie den Spruch *„Erfolg macht sexy"*. Jeder möchte stolz auf etwas sein. Als Manager muss ich dafür sorgen, dass das Team Erfolge hat und diese Erfolge sichtbar werden, sowohl nach innen als auch nach außen. Wenn das Team einen schönen Erfolg verbuchen konnte, sorgen Sie dafür, dass dies auch gebührend gefeiert wird. Jeder Release ist ein Erfolg, entsprechend muss ich als Manager sicherstellen, dass nicht jahrelang dahinentwickelt wird, ohne dass etwas ausgeliefert wird, sondern dass rasch und häufig Release an den Kunden gehen und auch tatsächlich eingesetzt werden.

Wichtig ist es, bei all diesen Punkten so individuell wie möglich auf das Team einzugehen und so wenig wie möglich mit Unternehmensstandards drüberzufahren. Dies ist natürlich nicht immer möglich, aber man sollte es immer zuerst versuchen. Alle diese Rahmenbedingungen müssen einmal initial hergestellt, laufend überwacht und bei Abweichungen wiederhergestellt werden. Dieses Herstellen, Überwachen und Wiederherstellen ist somit eine permanente Aufgabe für das Management. Nur so ist sichergestellt, dass ein Team jederzeit alles hat, was es braucht, um produktiv zu arbeiten und gute Software zu entwickeln.



Self-Assessment Checkliste für Rahmenbedingungen

Rahmenbedingung für Team / Projekt _____	**Nicht erfüllt** (0 Pkte)	**Kaum erfüllt** (1 Pkt)	**Eher wenig erfüllt** (2 Pkte)	**Eher erfüllt** (3 Pkte)	**Fast erfüllt** (4 Pkte)	**Voll erfüllt** (5 Pkte)
Die richtigen Mitglieder						
Ausreichend Zeit (Projekt und Mitglieder)						
Vertrauen und Freiheit						
Skills in Software Entwicklung						
Domänenwissen über das Business						
Wissen über Legacy-Systeme						
Geeignete Arbeitsmittel						
Support für die Infrastruktur						
Passende Technologien, Komponenten und Frameworks						
Ein Ziel (Big Picture, Zeit, Qualität)						
Gute Anforderungen						
Schnelle und klare Entscheidungen						
Klare Zuständigkeiten im und außerhalb des Teams						
Abstimmung mit anderen Teams und Stakeholdern						
Transparenz						
Stabile Prioritäten						
Gleichmäßiger Arbeitstakt						
Die Möglichkeit, Experte zu werden						
In Ruhe arbeiten können						
Jemand, der sich um sie kümmert						
Ein positives Arbeitsumfeld						
Feedback						
Ein ordentliches Gehalt						
Karrierepfad						
Sicherheit						
Erfolg						
Gesamtpunkte (Maximum = 130 Pkte)						

Literatur & Links

[1] Dr. Dirk Stelzer, **Möglichkeiten und Grenzen des prozessorientierten Qualitätsmanagements**, 2016, Köln

[2] Technova, **What Are the Consequences of Poor Requirements Specifications?**

[3] Beck et. Al., **Prinzipien hinter dem Agilen Manifest**, 2011

[4] Software Quality Lab Academy, **Academyprogramm 2019**, https://www.software-quality-lab.com/leistungen/academy/

[5] J. Appelo, **Managing for Happiness: Games, Tools, and Practices to Motivate Any Team**, 2016.

Zitate

"Schreibe nichts der Unfähigkeit des Teams zu, was durch falsche Rahmenbedingungen hinreichend erklärbar ist."

Ein weiser Berater in Anlehnung an Hanlon's Razor

"Frage nicht, was Dein Team für Dich tut, sondern frage erst, ob Du alles notwendige für Dein Team getan hast!"

Ein weiser Berater inspiriert von einer Rede von John F. Kennedy

Autor

Markus Unterauer

Berater & Trainer bei Software Quality Lab

Impressum

Der Quality Knowledge Letter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit dem Schwerpunkt SW-Qualität.

Weitere Infos zu diesem und anderen Themen finden Sie auf www.software-quality-lab.com

Leistungen von Software Quality Lab

Die Experten von Software Quality Lab unterstützen durch folgende Leistungen:

- Potenzialanayse zum finden Identifizieren von Problemen und Aufzeigen von Verbesserungspotenzialen
- Workshop „Agile Leadership & Management" zur Reflexion, Standortbestimmung und Finden von Möglichkeiten zur weiteren Entwicklung in ihrem Management Team
- Begleitung von Führungskräften in Softwareentwicklungs-Organisationen als Coach und „Sparring Partner"
- Unterstützung bei Auswahl und Einführung der geeigneten agilen Methoden (zB Scrum, Kanban)
- Begleitung bei der Einführung agiler Methoden als externer Coach
- Impulsvorträge zu Spezialthemen und Vertiefung (zB Agiles Requirements Engineering, Branching Strategien, Testautomatisierung, BDD/TDD, etc.)
- Analyse und Optimierung der begleitenden Prozesse wie Produktmanagement, Risikomanagement, Change- und Configuration-Management, etc.
- Tool Evaluation Center zur Auswahl von passenden Tools sowie Unterstützung bei der Tool-Einführung
- Integration von verschiedenen Tools zur besser Automatisierung und Effizienzsteigerung des Entwicklungsprozesses
- Seminare & Trainings für Scrum Master, Entwickler, Tester, Architekten, Produktmanager und Anwender sowohl öffentlich als auch inhouse

Detailliertere Infos zu den Leistungsbereichen von Software Quality Lab sowie weitere Knowledge finden Sie auf der Website:

www.software -quality-lab.com