



360 Grad Projekt-Review für Softwareprojekte

1 Einleitung

Wie gut sind Ihre Softwareprojekte unterwegs? Gibt es Sanierungsbedarf? Wenn ja, welchen?

Oft sind Probleme im Projekt deutlich spürbar, aber die Sichtweisen darauf unterschiedlich. Wie damit umgehen? Wer hier auf schnelle Lösungen, sogenannte „Quick-Wins“, setzt, läuft Gefahr Symptombekämpfung zu betreiben, im schlimmsten Fall: „Leichen schminken“.

Ein 360 Grad Projekt-Review liefert Antworten auf diese Fragen, und schafft eine tragfähige Entscheidungsbasis für das weitere Vorgehen. 360 Grad heißt, dass dabei alle, für den Projekterfolg wesentlichen Faktoren, betrachtet werden. Neben dem Projektmanagement auch die technische Architektur, der Entwicklungsprozess, Skills und Kommunikation des Teams, sowie die Einbettung in die Organisation. Warum? Der TÜV prüft ja auch nicht nur die Lenkung!

Doch wie kann ein solches 360 Grad Review kostengünstig abgewickelt werden? Was ist dabei zu beachten? Wovon können sie ausgehen? Welche Kompetenzen braucht es dazu? Welche Artefakte sind zu reviewen? Und wie erzielen sie für unpopuläre Ergebnisse die notwendige Akzeptanz? Lesen sie weiter.

2 Zielsetzung eines 360 Grad Softwareprojekt-Reviews

Ein 360 Grad Projekt-Review liefert folgende Ergebnisse:

1. Aussage über die Erfolgsaussichten des Projektes und die zu erwartende Qualität der Software
2. Nachvollziehbare, vom Projektteam mehrheitlich verstandene Problemanalyse
3. Vom Projektteam mitgetragener Maßnahmenkatalog zur Verbesserung der Erfolgsaussichten
4. Kosten/Nutzen-Rechnung zu jeder Maßnahme

Es geht also nicht darum, einen Teilaspekt z.B.: die professionelle Anwendung von Projektmanagementmethodik zu reviewen, sondern ganzheitliche Aussagen zu treffen, und operationalisierbare Maßnahmen zu liefern. Deshalb ist ein 360 Grad Projektreview für das Projekt und den Projektauftraggeber besonders wertvoll.

Erfolg = Qualität x Akzeptanz!

Die Akzeptanz der Problemanalyse und die darauf basierenden Maßnahmenkatalogs ist unverzichtbar. Das erfordert ein dialogisches methodisches Vorgehen und kommunikatives Fingerspitzengefühl.

3 Was ist zu reviewen?

Bei einem 360 Grad Projektreview werden alle, für den Projekterfolg wesentlichen, Faktoren betrachtet. Aber welche sind das konkret? Auf Basis meiner Erfahrung konzentriere ich mich bei meinen Reviews auf folgende Aspekte:

- Projektmanagement
 - Projektmanagementmethodik
 - Changemanagement
 - Entwicklungsprozess
 - Projektumwelten und organisatorische Einbettung
- Technik
 - Software Architektur
 - Antipatterns und Code Smells
 - Qualitätssicherung
 - Konfigurationsmanagement und Infrastruktur
- Team
 - Kommunikationskultur
 - Ziele der Keyplayer
 - technische und fachliche Skills der Teammitglieder
 - Entscheidungskompetenzen
 - Workforce

Wichtig ist, immer alle Aspekte im Auge zu behalten, weil sie in ihrer Auswirkung auf den Projekterfolg starke Abhängigkeiten voneinander haben.

3.1 Projektmanagement

Durch Review des Projekthandbuchs lässt sich meist schon die Frage nach dem professionellen und adäquaten Einsatz der **Projektmanagementmethodik** klären – sozusagen das Ein-mal-Eins des PM. Das ist wichtig, da gerade in Softwareprojekten in der Praxis oft Techniker¹ oder Fachexperten Projektleitungsverantwortung übernehmen.

Changemanagement ist gerade in Softwareprojekten von besonderer Bedeutung. Der Einsatz einer Software in einer Organisation hat fast immer beachtliche Auswirkungen auf die Arbeitsprozesse, und damit auf die Organisation selbst – meist erwünschte Auswirkungen, deshalb setzt man

¹ Aus Gründen der besseren Lesbarkeit verwende ich vorwiegend eine neutrale oder männliche Form, möchte damit aber Frauen natürlich gleichermaßen ansprechen.



die Software ja ein. Daraus resultiert aber auch ein Mitgestaltungsbedarf aus der Organisation, der sich typischerweise in fachlichen Change Requests manifestiert.

Wesentlich ist hier auch, dass sich der Projektauftraggeber dieser Dynamik bewusst ist, und seine Budgetpolitik darauf einstellt.

Denn auch das technische Umfeld (neue Produkte, neue Versionen) ändert sich typischerweise mehrmals während der Laufzeit eines Softwareprojektes. Und anders als etwa bei Bauprojekten, wo der Austausch des fertigen Kellers, bei aufgesetzten Geschossen zu mindestens unüblich ist, ist der Austausch einer Basiskomponente, etwa im Zuge eines Refactorings, in Softwareprojekten durchaus häufig und auch oft sinnvoll.

Agile **Entwicklungsprozesse**, wie etwa SCRUM, XP oder RUP unterstützen durch iteratives Vorgehen die Einstellung des Projektes auf diese Dynamik. Für kleine Projekte eignet sich auch ein Vorgehen nach dem Wasserfallmodell. Ein Review-Aspekt ist jedenfalls, ob ein Entwicklungsprozess definiert, allen Involvierten bekannt, und auch gelebt wird.

Die **Projektumwelten und die organisatorische Einbettung** des Projekts definieren die Kräfte, welche von außen auf das Projekt wirken.

Insbesondere ist hier auf jene Interessen von Projektumwelten zu achten, welche dem Projektziel widersprechen.

Beispiel: Die Einführung eines Sales-Controlling-Tools wird von den Sales Mitarbeitern kritisch gesehen – bis hin zu kreativ-bedauerndem Boykott.

Häufig dienen Softwareprojekte als „internes Outsourcing von Organisationskonflikten“.

Beispiel: *Einführung einer Workflow-Software hat das Ziel, Unternehmensprozesse zu automatisieren. Dazu sind diese zuerst zu vereinheitlichen. Und das steht in Konflikt zu den Autonomiebestrebungen ausgelagerter Suborganisationen.*

Natürlich können diese Konflikte im Review nicht gelöst werden. Aufgabe eines 360 Grad Reviews ist es aber, diese Konflikte bewusst, und damit bearbeitbar zu machen.

3.2 Technik

Die **Software Architektur** ist vor allem eines: Ein Kostenhebel!

Die Softwarearchitektur bestimmt wesentlich folgende Kostenpositionen:

- Entwicklungsaufwand
- Wartungsaufwand
- Lizenzkosten für verwendete Produkte
- Die Skills des Entwicklungs- und Wartungsteams, und damit die Personalkosten

Insbesondere die Entwicklungs- und Wartungskosten (meist der Löwenanteil der Gesamtkosten) können durch eine geschickt gewählte Softwarearchitektur oft um Faktoren

gesenkt werden. Fehler in der Softwarearchitektur rächen sich immer, je später, desto schlimmer.

Antipatterns und **Code Smells** wie z.B.: „Duplicate Code“, „Spaghetti Code“ oder „Reinventing the Wheel“ haben immer deutlich negative Auswirkungen auf Systemgröße und Wartbarkeit einer Software, beeinflussen somit wesentlich die Kosten. Sowohl Entwicklungskosten, als auch Wartungskosten.

Schwächen in der Software Architektur, oder Antipatterns und Code Smells können durch Refactoring behoben werden. Die entsprechenden Schritte sind mit dem Entwicklungsteam im Workshop auszuarbeiten, und als Teil der Review-Ergebnisse zu präsentieren. Diese Maßnahme mit dem Entwicklungsteams auszuarbeiten ist unverzichtbar, da ein von außen verordnetes Refactoring erfahrungsgemäß keine Chance auf vollständige Umsetzung hat. Und ein teilweises, halbherziges Refactoring die Situation tendenziell sogar verschlechtert.

Ohne **Qualitätssicherung** ist die Qualität der Software nicht gesichert. Die Frage hier ist, welche Qualitätssicherungsmethoden, wie etwa Code-Reviews und Unit-Testing im Projekt tatsächlich gelebt werden. Ist ein Systemtestprozess definiert? Und wie sieht er aus?

In diesem Kontext lassen sich meist auch die Themen **Konfigurationsmanagement und Entwicklungsinfrastruktur** mit betrachten. Wesentlich sind hier ein funktionierender Prozess und adäquate Tools.

3.3 Team

Ohne Team kein Projekt!

Mit der **Kommunikationskultur** und der daraus resultierende Kommunikationsfähigkeit steht und fällt die Steuerbarkeit des Projekts. Und damit die Fähigkeit des Projekts, sich auf veränderte Rahmenbedingungen einzustellen, und mit Konflikten umzugehen. Dazu ein Zitat von Charles Darwin: „*It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change.*“.

Ein gut kooperierendes Team kann durchaus Schwächen in Projektmanagement und Entwicklungsprozess kompensieren. Umgekehrt führt mangelnde Konfliktkultur im Projektteam meist zu Schwächen in der Software Architektur – weil keine Einigung auf gemeinsame Basiskomponenten erfolgt. Daher ist die Kommunikation des Projektteams ein wesentlicher Review-Aspekt.

Die persönlichen **Ziele der Keyplayer** müssen im Einklang mit den Projektzielen stehen, ihnen zu mindestens nicht widersprechen. Davon hängen die Motivation der Projektteammitglieder und die daraus resultierende Kommunikationskultur ab. In Diesem Kontext fällt auch die Fragestellung



nach der gemeinsamen Sicht auf die Projektziele und dem Commitment dazu. Diese Ziele explizit abzufragen, hilft außerdem die internen politischen Verhältnisse des Projekts zu verstehen. Dieses Verständnis ist die Basis für eine anschlussfähige Formulierung der Review-Ergebnisse.

Die technischen und fachlichen Skills der Teammitglieder müssen für die Projektaufgaben ausreichend sein, sonst ist die Ergebnisqualität in Gefahr. Dieser Review-Aspekt erfordert naturgemäß das meiste kommunikative Fingerspitzengefühl. Statt oberlehrerhaft fehlende Skills im Projektteam zu beklagen, hat es sich bewährt z.B. auf eine fehlende Expertenrolle hinzuweisen (Software Architekt, Fachexperte, Testexperte, Projektcoach, etc.). Daraus lassen sich nämlich auch anschlussfähige, operationalisierbare Maßnahmen ableiten - etwa die Einbindung eines externen Beraters/Coaches, oder die Durchführung von Schulungen.

Neben den Skills ist zu überprüfen, ob die **Entscheidungskompetenzen der Projektteammitglieder** für die Bewältigung der Projektaufgabe ausreichen. Insbesondere die Einbindung von Anwendern (oder Anwendervertretern) ist für die Akzeptanz einer Softwarelösung entscheidend. Agile Softwareentwicklungsprozesse stellen das durch ein Rollenkonzept sicher (SCRUM: Product Owner, XP: On-Site Customer).

Schlussendlich müssen die Projektteammitglieder ausreichend **Workforce** dem Projekt planbar zur Verfügung stellen. Das klingt selbstverständlich. Insbesondere in den klassischen Linienorganisationen mit Ressourcenhöhe in der Linie, und Fachexperten, die ihr Tagesgeschäft zu erledigen haben, sieht die gelebte Praxis mitunter anders aus. Insbesondere kurzfristige Ausfälle von Experten, durch „Tagesgeschäft geht vor“, führen oft zu dramatischen Unterschieden zwischen Plan und Wirklichkeit.

4 Methodisches Vorgehen

Erfolg = Qualität x Akzeptanz!

Die Akzeptanz der Problemanalyse, und des darauf basierenden Maßnahmenkatalogs, ist die unverzichtbare Basis für dessen Umsetzung. Das erfordert ein dialogisches methodisches Vorgehen und kommunikatives Fingerspitzengefühl. Das heißt, dass die Review-Ergebnisse nicht durch Dokumenten-Review und Interviews allein gewonnen werden können, sondern mit dem Projektteam gemeinsam erarbeitet werden müssen. Dazu hat sich das folgende Vorgehen bewährt:

1. **Auftragsklärungsgespräch** Welche Ziele, und Hypothesen gibt es aus Sicht des Auftraggebers?
2. **Studium der Projektdokumentation:**
 - a. Projekthandbuch
 - b. Analyse
 - c. Test-Cases

- d. Dokumentation Entwicklungsprozess, Testprozess
 - e. Was immer sonst noch an Dokumentation im Projekt existiert
3. **Technische Analyse:** Eine Reihe von Tools werden zum Parsen des Source Codes eingesetzt. Damit können AntiPatterns, Code Smells und Metriken automatisiert ermittelt werden. Ergänzt durch punktuellen Code-Reading. Betrachtungsobjekte sind:
 - a. Software Architektur
 - b. Source Code
 - c. Technische Dokumentation
4. **Hypothesenbildung:** Vom Review der Projektdokumentation und technischer Analyse ausgehend sind meist schon qualifizierte Hypothesen über die Problembereiche im Projekt möglich. Der Reviewer muss zwar bereit sein, diese im Laufe des Reviews zu adaptieren, sie helfen aber bei der Vorbereitung des Interaktiven Review-Teiles. Der Reviewer sollte diese Hypothesen vorerst für sich behalten, um keine Abwehrhaltungen zu provozieren.
5. **Review-Start-Workshop** mit dem Projektteam: Hier ist es wichtig das Vertrauen des Teams zu gewinnen. Das gelingt erfahrungsgemäß durch eine persönliche Vorstellung, die durchaus umfangreich sein darf, sowie einer detaillierten Darstellung des Vorgehens beim Review. Den Projektteammitgliedern soll bewusst werden, dass die Review-Ergebnisse sich im Wesentlichen aus ihren Aussagen ableiten.
6. **Interviews mit den Projektteammitgliedern, Auftraggeber und ev. relevanten Projektumwelten** Diese Interviews können einzeln oder nach Interessengruppen eingeteilt erfolgen. Bei diesen Interviews muss zu jedem Review-Aspekt immer wieder die Frage gestellt werden, was gut läuft, und wo Probleme spürbar sind. Hier bringt der Reviewer seine Hypothesen als Fragen ein.
7. **Schwenk von der Analyse zur Lösungsorientierung.** In diesem Schritt stellt der Reviewer seine Problemanalyse auf Basis der Interviews bereits partiell zur Verfügung, und bittet die Teammitglieder einzeln (oder in sehr kleinen Gruppen) um Lösungsideen.
8. **Integration der Lösungsideen.** Der Reviewer vernetzt die Teammitglieder auf Basis ihrer Lösungsideen, und moderiert einen Prozess der kontinuierlichen Integration der Sichtweisen und Lösungsansätze. Dieser Prozess endet wenn ein Maßnahmenkatalog vorliegt, der vom Projektteam im Workshop optimaler Weise konsensual getragen wird. (Das ist für Maßnahmen, die personelle Änderungen im Projektteam erfordern, nicht immer möglich.) Zu jeder Maßnahme wird eine Kosten/Nutzen Schätzung angestellt.
9. **Präsentation der Ergebnisse.** Aufbereitung der inzwischen abgestimmten Review-Ergebnisse und Präsentations



tion an Auftraggeber, Projektteam und am Review beteiligte Projektumwelten. Hier ist die weitere Vorgehensweise zu vereinbaren.

5 Voraussetzungen

Zur Durchführung eines 360 Grad Projekt-Reviews ist umfangreiche praktische Erfahrung sowie methodische Kompetenz in folgenden Feldern nötig:

- Projektmanagementmethodik
- Softwareentwicklungsprozesse
- Software Architektur
- Software Design Patterns und Antipatterns
- Im Projekt eingesetzte Software Technologie
- Moderation, Interviewtechnik, Workshop-Design und Präsentation
- Kommunikatives Fingerspitzengefühl

6 Aufwand und Nutzen

Im optimalen Fall verfügt eine Person über alle diese Kompetenzen, dann kann ein 360 Grad Review für ein kleineres Projekt bereits in etwa 5 PT abgewickelt werden. Mit der Anzahl der Personen im Review-Team und der Projektgröße steigt der Aufwand entsprechend.

Erfahrungsgemäß übersteigt der durch die Umsetzung der Maßnahmen erzielbare Nutzen die Review-Kosten um ein Vielfaches.

Dazu kommt, dass positive Review-Ergebnisse, und umgesetzte Maßnahmen, Unsicherheiten hinsichtlich des Projekts in der Organisation beseitigen, und schon deshalb die Erfolgsaussichten signifikant erhöhen.

7 Literatur

[1] *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis* by William J. Brown, Raphael C. Malveau, SkipMcCormick, Thomas J. Mowbray