Feature-Teams oder Komponententeams? Oder doch ganz anders?

Agile Methoden propagieren das Konzept der cross-functional Featureteams. D.h. Teams, die ein Feature vom User Interface bis zum Datenbank Backend vollständig und eigenständig umsetzen können. Was in der Theorie super klingt, ist in der Praxis aber oft sehr schwer umzusetzen und auch nicht immer sinnvoll. Wie immer kommt es auf eine ganze Reihe von Faktoren an, welche Teamzusammensetzung in einer Organisation am besten geeignet ist.

Die richtige Teamzusammenstellung ist Voraussetzung für effiziente Entwicklung

Bei der Einführung agiler Methoden, der Umstrukturierung von Unternehmen oder der Entwicklung komplett neuer Produkte werden oft auch die Entwicklungsteams neu zusammengestellt. Im Lichte agiler Vorgehensmodelle herrscht recht schnell Einigkeit darüber, dass neben den Entwicklern auch Tester und Dokumentierer im Team drin sein sollen. Das ist meist einfach machbar und liefert schnell bessere Effizienz und höhere Qualität. Schwieriger zu lösen ist die Frage, ob man die Teams nach Produkten, nach technischen Komponenten, nach Schichten oder doch lieber gleich Featureteams, die alles können, zusammenstellen soll.

Diese Zusammenstellung der Teams ist wohl neben der Auswahl der Entwicklungsmethodik eine der wichtigsten Entscheidungen in einer Entwicklungsorganisation. Sie sollte daher gut vorbereitet und strukturiert umgesetzt werden. Im Folgenden werden einige Möglichkeiten dargestellt, wie man Entwicklungsteams zusammenstellen kann:

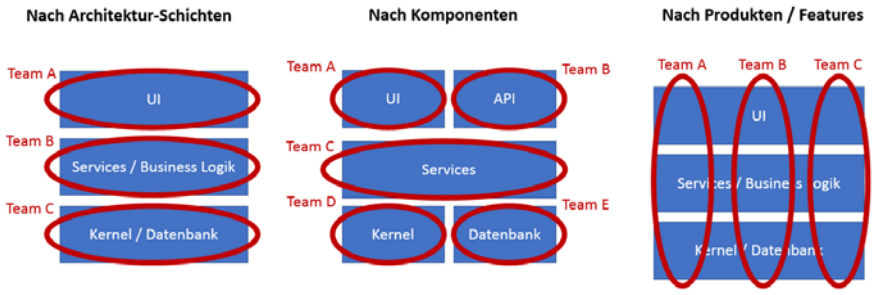

Abbildung 1 – Möglichkeiten der Zusammenstellung von Teams

In Entwicklungsorganisationen, die nach klassisch plangetriebenen Methoden wie dem V-Modell oder nach Wasserfall vorgehen, findet man meist Komponententeams oder nach Schichten getrennte Teams. Dies funktioniert lange Zeit gut. Werden die Produkte aber größer und komplexer, so entstehen häufig Engpässe bei Komponenten oder Schichten, die zentral im Produkt verankert sind. Immer wieder kommen Kunden zu Software Quality Lab, weil beispielsweise das zentrale Kernel-Team völlig überlastet ist und die Application und UI-Teams immer wieder warten müssen. Gemeinsam entwerfen wir dann eine bessere Teamaufstellung, die einen effizienteren Durchfluss ohne Engstellen und Wartezeiten ermöglicht.

Auf den folgenden Seiten diskutieren wir solche Möglichkeiten, Teams zu gliedern. Wir beleuchten die Vor- und Nachteile und Kriterien, wann einzelne Varianten sinnvoll sind und wann nicht.

High-Performance Teams

Gute Teams, die das Maximum aus sich herausholen, sind einer der zentralen Erfolgsfaktoren für ein Software-Entwicklungsunternehmen. In der agilen Welt spricht man oft von „*High-Performance Teams*“. Wiese und Ricci nennen 10 Eigenschaften solcher High-Performang Teams [1]:

Vertrauen. Die Möglichkeit, Ideen und Gefühle frei auszudrücken.

Gemeinsame Ziele. Jeder arbeitet auf dieselben Ziele hin.

Klare Zusammenarbeit. Jeder weiß, wie man zusammenarbeitet und Aufgaben erledigt.

Klare Erwartungen. Jeder kennt die Erwartungen ans Team und ihn selbst.

Spannungen aktiv beseitigen. Konflikte werden aktiv angegangen und konstruktiv gelöst.

Jeder beteiligt sich. In Diskussionen bekommt jeder Platz für seine Meinung.

Konstruktive Kritik. Konflikte und Kritik werden als positiv und Chance wahrgenommen.

Klare Entscheidungen. Entscheidung werden gemeinsam getroffen. Wenn das nicht möglich ist, entscheidet der Team Lead oder Sponsor.

Respekt. Jeder kennt seine Verantwortung und respektiert die vereinbarten Prozesse.

Abwechselnde Führung. Von Zeit zu Zeit wechselt die Führung des Teams.

Schon bei der Teambildung sollten diese 10 Eigenschaften beachtet werden. Sind die Teams dann gebildet, hat das Management eine Coaching Aufgabe, um aus guten Teams, High-Performance Teams zu machen.

Markus Unterauer

Head of Consulting, Trainer

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

Mögliche Gliederungen von Teams

In der Software-Engineering Literatur wird meist nur zwischen (klassischen) Komponententeams und (agilen) Featureteams unterschieden. In der Praxis findet man aber deutlich mehr Varianten und oft auch Mischformen:

- Produktteams
- Komponententeams
- Schichtenteams
- Featureteams
- Mischformen

Komponenten- und Schichtenteams findet man eher in Organisationen, die nach klassisch plangetriebenen Modellen wie V-Modell, Wasserfall u.Ä. vorgehen. Agile Organisationen versuchen eher Feature- oder Produktteams zu bilden. Jede dieser Varianten hat Vor- und Nachteile und bestimmte Szenarien, wo sie gut geeignet ist. Die eine, richtige Teamzusammenstellung, die immer und überall passt, gibt es nicht.

Produktteams

Jedes Team entwickelt genau ein Produkt und dafür alle Features. Produkte sind dabei nach fachlichen Gesichtspunkten definiert. Dies setzt voraus, dass ein Produkt klein genug ist, um von einem Team betreut zu werden. Beispiel: 1 ERP-Team, 1 Backoffice-Team, 1 CRM-Team etc.

Vorteile von Produktteams:

- ☑ Team identifiziert sich mit seinem Produkt
- ☑ Hohes Verantwortungsgefühl
- ☑ Jedes Team baut in seinem Produkt Fach-Know-How auf
- ☑ Motivation durch Experten-Status in der Fachdomäne

Nachteile von Produktteams:

- Engpässe, wenn bei einem Produkt viel zu tun ist
- Unterauslastung, wenn für ein Produkt wenig zu tun ist
- Produktübergreifende Basisfunktionen sind schwer umzusetzen und erfordern mehrere Teams und viel Koordination. Oft ist unklar, wer diese Koordination übernimmt.
- „Silodenken“. Jeder denkt nur an sein Produkt
- Entwickeln und Sicherstellen einer einheitlichen, produktübergreifenden Architektur im Unternehmen ist schwierig
- Synergien zwischen den Produkten schwer zu heben
- Dinge werden eventuell unabsichtlich mehrfach gebaut

Produktteams sind gut geeignet wenn

- Kleine Produkte, die ein Team von A bis Z entwickelt
- Viel produktspezielles Fach-Know-How nötig
- Wenige unterschiedliche Technologien
- Wiederverwendung von Komponenten nicht erforderlich

Komponententeams

Jedes Team verantwortet eine oder mehrere technische Komponenten. Die Komponenten können dabei entweder nach fachlichen oder nach technischen Gesichtspunkten definiert werden. Beispiel: 1 Kernel-Team, 1 Import-Modul-Team, 1 Export-Modul-Team, 1 Faktura-Modul-Team etc.

Vorteile von Komponententeams

- ☑ Entwickler können sich auf eine Technologie konzentrieren und dort Experten werden
- ☑ Team identifiziert sich mit seiner Komponente
- ☑ Hohes Verantwortungsgefühl
- ☑ Hohe Effizienz in der Entwicklung der Komponenten
- ☑ Fördert tendenziell die Wiederverwendung von Komponenten in mehreren Produkten

Nachteile von Komponententeams

- Fokus eher auf Technologie als auf Kundennutzen. Niemand fühlt sich für das Gesamtprodukt verantwortlich.
- Engpässe und Wartezeiten bei abhängigen Teams, wenn in einer Komponente viel zu tun ist
- Unterauslastung, wenn wenig zu tun ist
- Fachliche Features benötigen oft mehrere technische Komponenten für die Umsetzung. Solche komponentenübergreifenden Features erfordern mehrere Teams und dadurch viel Koordination. Oft unklar, wer diese Koordination übernimmt.
- „Silodenken“. Jeder denkt nur an seine Komponente
- Entwickeln und Sicherstellen einer einheitlichen, komponentenübergreifenden Architektur ist schwierig
- Dinge werden eventuell unabsichtlich mehrfach gebaut

Komponententeams sind gut geeignet wenn

- Wenige Produkte, im Idealfall nur ein einziges
- Hohe technische Komplexität der einzelnen Komponenten
- Kleine Features, die meist nur eine Komponente betreffen
- Unterschiedliche Technologien, die Spezialisten erfordern

Schichtenteams

Jedes Team verantwortet eine oder mehrere Schichten in der Architektur. Beispiel: 1 UI-Team, 1 Business-Logik-Team, 1 API-Team, 1 Datenbank-Team etc.

Vorteile von Schichtenteams

- ☑ Entwickler können sich auf eine Technologie konzentrieren und dort Experten werden.
- ☑ Team identifiziert sich mit seiner Schicht
- ☑ Hohe Effizienz in der Entwicklung der Schichten

Vollständiger Knowledge Letter Zugang

Wir freuen uns, dass Sie an diesem Thema Interesse haben und den Knowledge Letter von Software Quality Lab bis hierher gelesen haben.

Dieser Knowledge Letter ist eine Vorschau (gekürzte Version des gesamten Artikels).

Wenn Sie den ungekürzten Knowledge Letter lesen möchten, registrieren Sie sich bitte unter
http://www.software-quality-lab.com/download/knowledge-letter/anfrage-knowledge-letter/

Sie erhalten nach der Registrierung vollen Zugang zu allen bisherigen Knowledge Lettern von Software Quality Lab und erhalten automatisch künftige Knowledge Letter per E-Mail.