Confluence+Jira nahtlos kombiniert

Inhalt

1. Einleitung

Die Methode cards+ ist ein agiler Ansatz mit dem Ziel, Produktwissen strukturiert und mit hoher Qualität zu erfassen. Für das Projektmanagement gibt es unterschiedliche Vorgehensmodelle. Agile Ansätze setzen vor allem auf Flexibilität und Anpassung. Statt ausführlicher und umfangreicher Planung zu Beginn eines Projekts werden das adaptive Planen in Iterationen, schnelles Erkennen von Problemen (engl. fail fast) sowie die effiziente Kommunikation im Team unterstützt. Hier kommen `Confluence` und `Jira` ins Spiel.

`Jira` wird vom Hersteller als Werkzeug zur Vorgangs- und Projektverfolgung für agile Teams bezeichnet. `Confluence` ist ein ausgereiftes Produkt mit vielen Funktionen, die man für ein Wiki braucht. Die beiden Produkte der australischen Firma Atlassian bilden ein sehr leistungsfähiges Paar. Hier gilt der (verkürzte) Satz von Aristoteles: Das Ganze ist mehr als die Summe seiner Teile.

> Copy. Paste. Done. So schnell kann man die URL eines Vorganges oder eines Filters in Jira in einer Confluence-Seite einfügen. Es ist so einfach.
>
> — Terrence Caldwell

Jira ist ein sehr leistungsstarkes Werkzeug. Wir können mit Jira sehr gut die Methode Scrum unterstützen, mit Vorgängen für das Backlog vom Typ Epic, Story und Sprint. Das Backlog-Management profitiert von den Möglichkeiten zur Abbildung von Abhängigkeiten von Stories untereinander, von Story zu Ersteller (der Produktverantwortliche) bzw. Bearbeiter (jemand aus dem Team) und zum Produktwissen in Confluence. Die Einschätzung der Komplexität des Teams aus Refinement- und Planning-Meeting sichert der Produktverantwortliche in der Story, als Grundlage für seine Planung. Der aktuelle Fortschritt wird mit einem Burn-Down-Diagramm dargestellt. Jira hat noch viele weitere Funktionen für die "Prozesssicht", für die Sicht auf den Weg des Scrum-Teams zum richtigen Produkt. Jira beantwortet die Frage, wie wir unser Ziel erreichen.

Confluence hingegen ist ein Werkzeug für die "Produktsicht",

die Sicht auf den Zustand des Produktes. Die Methode cards+ versetzt uns in die Lage, eine Produktdokumentation inkrementell zu erstellen. Ein Case wird in der Anforderungsanalyse angelegt, mit der abstrakten Beschreibung des Problems und konkreten Beispielen. Erst in Zusammenarbeit mit dem Team wird die Lösung skizziert. Mit der Übernahme des Case als Story in den Sprint beginnt die Umsetzung. Am Ende eines Sprints gibt es immer eine neue Version des Produktes (das Produktinkrement), bestehend aus Code, Test und Dokumentation. Der aktuelle Leistungsumfang des Produktinkrementes wird durch die Bausteine der Systembeschreibung sichtbar. Art, Umfang und Struktur der Software wird durch weitere Bausteine angemessen beschrieben.

2. Baustein Epic

Der Baustein Epic ist das Artefakt in unserer Systembeschreibung, mit dem wir ein Ergebnis der Analyse einer Anforderung sichern. Er gibt einer Anforderung einen eindeutigen Namen und enthält die grobe Ausarbeitung der fachlichen Problemstellung. Er aggregiert Vorgänge, Anwendungsfälle, Sonderfälle und Fehlersituation zu einem sinnvollen Ganzen.

Ein Epic in `Jira` lässt sich ganz leicht mit seinem "Zwilling" in `Confluence` verknüpfen. Ganz pragmatisch erreichen wir dadurch eine hohe Wiedererkennung der Anforderungen in `Jira` und `Confluence`. Mit einem Klick kann der Nutzer zwischen den beiden Werkzeugen wechseln.

3. Baustein Case

Der Baustein Case ist das Artefakt in unserer Systembeschreibung, mit dem wir ein konkretes fachliches Problem aus der Analyse einer Anforderung sichern. Er gibt einem Vorgang, Anwendungsfall, einer besonderen Situation oder einem bekannten und akzeptierten Fehler einen aussagekräftigen Titel. Er kann aber auch eine Situation bezeichnen, für die es keine Lösung gibt (z.B. wegen technischer Hürden oder schlicht zu teuer).

Für die Realisierung eines Case, d.h. die Transformation in Code und Test, benötigt das Team eine Story in `Jira`. Der Titel für Case und Story ist gleich. Wie beim Epic ergibt sich auch hier automatisch eine hohe Wiedererkennung. In der Beschreibung der Story ergänzt der Produktverantwortlicher kurzfristig benötigtes detailreiches Wissen (Beispiele, Szenarien, Testfälle). Spätestens in einem Refinement-Meeting entwickelt das Team zusammen mit dem Produktverantwortlicher die Lösungsidee und dokumentiert sie im Case als Folge von

Essenzschritten, in der die Rolle von Services und Events beschrieben wird. Akzeptanzkriterien schließlich sorgen dafür, dass Produktverantwortlicher, Entwickler und Tester das gleiche Verständnis von Umfang und Ziel der Story haben.

Eine Story ist im Normalfall mit der Confluence-Seite des umzusetzenden Cases verbunden. Mit einem Klick auf den Link bekommt der Entwickler alle Informationen aus der Anforderungsanalyse, die für die Story relevant sind. Er wechselt von Jira zu Confluence. Er wechselt einfach die Perspektive. Auch die in der Lösungsidee erwähnten Confluence-Seiten für Services und Events werden mit der Story verknüpft. Alle für Entwickler wissenswerten Besonderheiten einer Komponente oder notwendigen Spezifikationen sind "nur einen Klick" entfernt. Diese Zusammenstellung der wichtigsten Confluence-Seiten in der Story hat den enormen Vorteil, dass jeder Entwickler sicher und leicht an alle notwendigen Informationen kommt. Der Wechsel von Jira in Confluence und wieder zurück ist völlig transparent.

Gerade bei neuen Produkten gibt es immer wieder die Situation, in der erst die Voraussetzungen innerhalb der Software-Architektur geschaffen werden müssen, bevor eine Case umgesetzt werden kann. Dazu erstellt der Produktverantwortlicher häufig eine sogenannte Enabler-Story im Backlog. In manchen Fällen wird für die Umsetzung eines Qualitätsmerkmales (z.B. Logging, Monitoring, Sicherheit) oder eine Optimierung im Zusammenhang mit einem Case eine separate Story angelegt. Die Gründe dafür sind vielfältig: Der Sprint ist voll, die Umsetzung erfordert Spezialkenntnisse im Team oder das Qualitätsmerkmal ist viel niedriger priorisiert als die fachliche Funktion. Jede dieser technisch motivierten Stories hat wenig bis keinen wirksamen fachlichen Mehrwert. Trotzdem wird jede Story wird mit dem Case verknüpft, für den sie die Voraussetzungen oder Verbesserungen schafft. Damit bekommen wir in `Confluence` eine wunderbare Übersicht über alle Stories, die für einen Case relevant sind. Mit dieser zusätzlichen zum Backlog in `Jira` verfügbaren Sicht auf Stories fällt es dem Produktverantwortlicher leichter, Konflikte zwischen Stories zu erkennen. Mit einem Blick sieht er den aktuellen Status und Bearbeiter

der für einen Case geplanten Stories.

4. Baustein Service

Der Baustein Service ist das Artefakt der Systemstruktur, mit dem wir einen Dienst unserer Anwendung so beschreiben, wie wir ihn realisiert haben. Im Steckbrief des Bausteins stehen grundlegende Informationen über die gewählten Technologien.

In der Regel schreiben wir Stories, um Fähigkeiten des Produktes zu ergänzen, zu ändern oder eventuell sogar zu entfernen. Es gibt aber auch Qualitätsmerkmale, die keinen direkten Bezug zu einem Case haben. Die Aufarbeitung von technischen Schulden zählt dazu. Gleiches gilt für den Fall, dass es eine wesentliche Änderung in der Software-Architektur gibt. Das kann eine geplante Aktualisierung einer Bibliothek oder Middleware sein, der Einsatz neuer Features der Programmiersprache oder der Austausch eines Infrastrukturproduktes (z.B. Datenbank-Server, Messaging-Broker, Streaming-Cluster). Nicht unerwähnt bleibt die grundlegende Optimierung eines Dienstes.

Ein daraus resultierendes Refactoring im Code mit einer ausreichenden Komplexität und dem damit verbundenen Risiko nimmt ein Produktverantwortlicher gerne als Story in das Backlog auf. Die Story wird mit den Confluence-Seiten der Services verknüpft, die von der Story betroffen sind. Nach Abschluss des Refactorings kann das Team lückenlos die notwendigen Änderungen an der Dokumentation vornehmen. Der Produktverantwortlicher kann in der Zwischenzeit erkennen, welche Services gerade technisch "renoviert" werden. Er wird einen Case verschieben, wenn der ebenfalls Änderungen an einem der betroffenen Services erfordert. Hier schafft Confluence ohne große Mühe eine Sicht, die der Produktverantwortlicher im Backlog in Jira nur sehr schwer nachbilden kann.

5. Fazit

Ein Produktverantwortlicher zieht große Vorteile für sein Backlog-Management aus der Kombination von Jira und Confluence.

- Er sieht, ob ein Case schon im Backlog berücksichtigt wurde.
- Er sieht, welcher Service in welchem Case relevant ist.
- Er sieht, in welchen Cases ein Service verwendet wird.
- Er kann die Vollständigkeit des Backlog einschätzen.
- Er bekommt mit einem Klick den aktuellen Status in Jira.
- Er sieht den aktuellen Bearbeiter im Team.
- Er kann Konflikte bei Stories vor der Umsetzung lösen.
- Er erkennt Duplikate im Backlog.

Mit Jira hat er den agile Entwicklungsprozess im Griff. Jira zeigt ihm den Fortschritt im Projekt. In Confluence sammeln er

inkrementell das dafür nötige Produktwissen. Dadurch behält er die Übersicht über den Zustand des Produkte.

Confluence und Jira sind web-basiert. Sie können ganz leicht in jeder Arbeitsumgebung integriert werden. Auch unterwegs können sie genutzt werden. Damit sind sie Werkzeuge für das ganze Team zu jeder Zeit.

Confluence und Jira passen sehr gut zusammen. Durch die sehr gute Integration der beiden Produkte behält das Team gleichzeitig Überblick über den Entwicklungsprozess und die Produktinkremente. Es gibt ausreichend viele gute Gründe, Jira und Confluence zusammen mit den agilen Methoden Scrum und cards+ einzusetzen.