A1 Scrum: Akzeptanzkriterien vs. Definition of Done

Erläutern Sie mit eigenen Worten den Unterschied zwischen den Akzeptanzkriterien einer User Story und der Definition of Done bei Scrum.

Lösung 1

Aus [1]:

> In Scrum gibt es zwei wichtige Konzepte, die oft verwechselt werden, aber unterschiedliche Zwecke erfüllen: Akzeptanzkriterien einer User Story und die Definition of Done. Hier ist eine klare Erläuterung der Unterschiede:
>
> 1. Akzeptanzkriterien einer User Story:
>   i. Was es ist: Akzeptanzkriterien sind spezifische Bedingungen, die eine User Story erfüllen muss, um als ërfülltöder äbgeschlossenängesehen zu werden. Sie sind in der Regel vom Product Owner festgelegt und geben an, was die User Story leisten soll.
>   ii. Zweck: Diese Kriterien helfen dem Entwicklungsteam zu verstehen, was der Product Owner von der User Story erwartet. Sie dienen als Leitfaden, um sicherzustellen, dass die entwickelte Funktionalität den Anforderungen des Benutzers entspricht.
>   iii. Fokus: Akzeptanzkriterien konzentrieren sich auf die spezifische Funktionalität und das Verhalten einer einzelnen User Story. Sie sind oft benutzerspezifisch und situationsspezifisch.
> 2. Definition of Done (DoD):
>   i. Was es ist: Die Definition of Done ist eine klare Liste von Anforderungen und Bedingungen, die erfüllt sein müssen, damit jede Arbeitseinheit (nicht nur User Stories, sondern alle Arbeitsergebnisse) im Scrum-Prozess als "fertig"betrachtet wird. Dies wird meist vom gesamten Scrum-Team festgelegt.
>   ii. Zweck: Die DoD stellt sicher, dass alle Arbeitsergebnisse eine konsistente Qualität aufweisen und alle notwendigen Schritte wie Code-Reviews, Testing, Dokumentation usw. durchlaufen haben. Sie ist ein Qualitätsstandard für die Auslieferung.
>   iii. Fokus: Die Definition of Done ist generisch und gilt für alle Arbeitspakete im Projekt. Sie stellt sicher, dass unabhängig von der spezifischen User Story alle Ergebnisse einen bestimmten Qualitätsstandard erfüllen.
>
> Zusammengefasst: Während die Akzeptanzkriterien einer User Story spezifizieren, was benötigt wird, damit eine spezifische Story die Bedürfnisse des Endbenutzers erfüllt, legt die Definition of Done allgemeine Qualitätsstandards und erforderliche Schritte fest, die für jede Art von Arbeit im Projekt

zu erfüllen sind. Beide Konzepte sind entscheidend für die erfolgreiche Umsetzung von Scrum-Projekten.

A2 Continuous Deployment

Nennen Sie mögliche Risiken einer Continuous-Deployment-Pipeline, die ein Softwaresystem über mehrere Schritte hinweg bis ins Produktivsystem einspielt.

Lösung 2

Aus [1]:

> Die Implementierung einer Continuous-Deployment-Pipeline für die Auslieferung von Software birgt verschiedene Risiken. Einige der möglichen Risiken sind:
>
> 1. Fehlerhafte Software-Updates: Automatisierte Deployments können fehlerhafte Updates schneller in die Produktionsumgebung bringen, insbesondere wenn die automatisierten Tests nicht ausreichend sind.
> 2. Sicherheitslücken: Schnelle Deployments können dazu führen, dass Sicherheitsprüfungen vernachlässigt werden, was Sicherheitslücken in der Software zur Folge haben kann.
> 3. Komplexität der Pipeline: Eine Continuous-Deployment-Pipeline kann komplex sein und erfordert ein tiefes Verständnis der Prozesse und Tools. Fehler in der Konfiguration können zu schwerwiegenden Problemen führen.
> 4. Abhängigkeiten zwischen Komponenten: In einem System mit vielen interdependenten Komponenten kann ein Update in einer Komponente unbeabsichtigte Auswirkungen auf andere haben.
> 5. Ressourcenüberlastung: Continuous Deployment kann zu einer ständigen Belastung der Ressourcen führen, insbesondere in Bezug auf Serverleistung und Speicherkapazität.
> 6. Mangelnde Rückfallmöglichkeiten: Fehlt ein effektiver Mechanismus zum Zurücksetzen von Änderungen (Rollback), kann es schwierig sein, auf einen stabilen Zustand zurückzukehren, wenn etwas schiefgeht.
> 7. Benutzererfahrung: Häufige Änderungen können für Endbenutzer verwirrend sein, insbesondere wenn neue Features oder Änderungen nicht klar kommuniziert werden.
> 8. Compliance und Regularien: In regulierten Branchen können schnelle Änderungen Compliance-Probleme verursachen, wenn nicht sichergestellt wird, dass jede Veröffentlichung den regulatorischen Anforderungen entspricht.

Um diese Risiken zu minimieren, ist es wichtig, robuste Teststrategien, effektive Überwachungs- und Logging-Tools sowie klare Kommunikationskanäle für das Team und die Endbenutzer zu implementieren.

A3 Klassisch oder agil?

Eine medizinische Gemeinschaftspraxis mit mehreren Ärzten:innen und Arzthelfern:innen möchte Sie als Softwaredienstleister beauftragen, ein Online-Reservierungssystem für Impfun- gen zu entwickeln. Es soll ein Festpreis für die Lieferung der Software vereinbart werden. Der Auftraggeber hat bisher kaum Erfahrung mit Softwareprojekten gemacht. Außerdem stellen Sie schon im ersten Gespräch fest, dass es verschiedene Auffassungen über den Funktionsumfang und die Details der fachlichen Anforderungen gibt.

Geben Sie an, welche Vorgehensweise Sie als Projektleitung für dieses Projekt empfehlen würden. Diskutieren Sie die Chancen und Risiken des Einsatzes von Scrum für dieses Projekt.

Lösung 3

Für ein solches Softwareprojekt mit einer medizinischen Gemeinschaftspraxis, die ein Online-Reservierungssystem für Impfungen entwickeln möchte, empfehle ich folgende Vorgehensweise:
Vorgehensweise:
1. **Anforderungsanalyse und Klärung:** - Führen Sie detaillierte Gespräche mit den Stakeholdern (Ärzten, Arzthelfern, ggf. Patientenvertretern) durch, um die Anforderungen genau zu verstehen und zu dokumentieren. - Erstellen Sie eine klare Spezifikation der Anforderungen, inklusive Funktionsumfang, Benutzeroberfläche, Datenschutzbestimmungen und anderen rechtlichen Anforderungen.
2. **Festpreisangebot mit Flexibilität:** - Da der Auftraggeber einen Festpreis wünscht, legen Sie diesen basierend auf der initialen Anforderungsanalyse fest. Beachten Sie jedoch, dass Änderungen im Projektverlauf wahrscheinlich sind. Planen Sie daher einen gewissen Spielraum für Anpassungen ein.
3. **Agile Methodik mit Scrum:** - Implementieren Sie eine agile Vorgehensweise, vorzugsweise Scrum, um Flexibilität im Entwicklungsprozess zu gewährleisten. Dies ermöglicht eine iterative Entwicklung und regelmäßige Abstimmungen mit dem Auftraggeber.
4. **Regelmäßige Kommunikation und Reviews:** - Planen Sie regelmäßige Meetings zur Überprüfung des Projektfortschritts und zur Diskussion von Anpassungen. Dies gewährleistet, dass das Endprodukt den Erwartungen des Auftraggebers entspricht.
5. **Prototyping und Feedback-Schleifen:** - Entwickeln Sie frühe Prototypen des Systems, um Feedback zu sammeln und sicherzustellen, dass die Entwicklung in die richtige Richtung geht.

6. **Dokumentation und Schulung:** - Dokumentieren Sie die Software umfassend und bieten Sie Schulungen für die Nutzer an, um eine effiziente Nutzung des Systems sicherzustellen.
Chancen und Risiken des Einsatzes von Scrum:
Chancen: 1. **Flexibilität:** Scrum erlaubt es, auf Änderungen in den Anforderungen schnell zu reagieren, was besonders wichtig ist, da das Verständnis des Auftraggebers für das Projekt zu Beginn begrenzt sein könnte. 2. **Transparenz:** Durch regelmäßige Sprints und Reviews bleibt der Auftraggeber stets über den Fortschritt informiert. 3. **Kundenorientierung:** Direktes Feedback des Auftraggebers wird in jeder Phase des Projekts berücksichtigt, was die Kundenzufriedenheit erhöht. 4. **Qualitätssicherung:** Kontinuierliches Testing und Anpassungen während der Entwicklung fördern eine hohe Qualität des Endprodukts.
Risiken: 1. **Unklare Anforderungen:** Anfangs unklare oder sich ändernde Anforderungen können zu häufigen Änderungen im Projektverlauf führen. 2. **Budgetüberschreitungen:** Bei einem Festpreisvertrag können häufige Anforderungsänderungen zu Budgetproblemen führen. 3. **Auftraggeber-Einbindung:** Der Erfolg von Scrum hängt stark von der aktiven Beteiligung und dem Engagement des Auftraggebers ab. 4. **Missverständnisse:** Ohne Erfahrung im Umgang mit agilen Methoden kann es beim Auftraggeber zu Missverständnissen bezüglich des Prozesses kommen.
Insgesamt wäre Scrum eine geeignete Methode für dieses Projekt, da es Flexibilität und ständige Anpassung an die Bedürfnisse des Auftraggebers ermöglicht. Es ist jedoch wichtig, von Anfang an klare Kommunikationskanäle zu etablieren und den Auftraggeber in den Prozess einzubinden.

A4 Online-Reservierungssystem für Impfungen

Sie erheben nun die Anforderungen für das Online-Reservierungssystem für Impfungen, siehe Aufgabe A3. Sie wollen zunächst auf fachlicher Ebene den Nachrichtenfluss zwischen Patienten und Web-Anwendung modellieren. Sie haben bereits die folgenden Komponenten identifiziert:

- Der Patient als menschlicher Akteur
- Die Web-Anwendung als GUI
- Ein Identitätsmanagement als Hintergrunddienst
- Ein Termin-Verwaltungssystem als Hintergrunddienst

Der reguläre Nachrichtenfluss zwischen diesen Komponenten gestaltet sich wie folgt:

- Der Patient besucht die Web-Anwendung und meldet sich mit seinem Benutzernamen und Passwort an (Sie können davon ausgehen, dass der Patient bereits registriert ist).
- Die Webseite prüft die Gültigkeit der Anmeldung im Identitätsmanagement.

- Bei ungültigen Anmeldedaten wird dem Patienten durch die Web-Anwendung angeboten, sein Passwort zurückzusetzen. Dazu wird eine entsprechende Operation im Identitätsmanagement aufgerufen. Der Nachrichtenfluss endet dann.
- Ansonsten ruft die Web-Anwendung die möglichen Wochentage für eine Impfung vom Termin-Verwaltungssystem ab und zeigt diese an.
- Der Patient wählt einen der möglichen Tage aus. Die Web-Anwendung ruft dann die möglichen Uhrzeiten vom Termin-Verwaltungssystem ab und zeigt diese an.
- Der Patient wählt eine Uhrzeit aus und bestätigt die Buchung. Dadurch wird der Termin im Termin-Verwaltungssystem reserviert und im persönlichen Kalender im Identitätsmanagement eingetragen.
- Falls bei der Buchung angegeben, erhält der Patient optional eine E-Mail mit dem gebuchten Termin. Diese wird vom Identitätsmanagement gesendet.

Modellieren Sie den Nachrichtenfluss zwischen Patienten und den beteiligten Systemen durch ein UML-Sequenzdiagramm.

Lösung 4

A5 Auto starten

Im Zip-Archiv »sequence.zip« finden Sie eine Java-Konsolenanwendung für den Start eines Autos, siehe ILIAS. Die Anwendung besteht aus den Klassen App, Battery, Car, Engine und OilSensor.

Modellieren Sie, ausgehend vom Aufruf der Methode main in der Klasse App, den Nachrichtenfluss der Anwendung als UML-Sequenzdiagramm. Berücksichtigen Sie dabei folgende Hinweise:

- Der Aufruf von main soll als »gefundene Nachricht« ohne Parameter modelliert werden.
- Die korrekten, verschachtelten Sequenznummern der Aufrufe sollen angegeben werden.
- Alle Datentypen, Parameter und Rückgaben sollen angegeben werden.
- Antwort-Nachrichten können weggelassen werden, soweit sie sich eindeutig aus den Aufrufen ergeben.
- Die Konsolenausgabe der App als Reaktion auf die Rückgabe von myCar.prepareStart() brauchen Sie nicht zu modellieren.

Lösung 5

Literatur

[1] OpenAI's ChatGPT, Version GPT-4, 2023-11-10 [Large language model], 2023. Adresse: https://chat.openai.com/share/9b92491c-b1fc-47a6-a54b-b20e6b36d7b8.