Herzlich willkommen bei der OSC

Verstehen, was Entwickler tun

Grundlagen für PO & SM

3. November 2025, 17:00 - 18:30 Uhr

Online Scrum Community

Marvin Marion

Strategie-Meeting

Strategie-Meeting im November

OSC-Komitee

Euer Input ist gefragt!

Vorstellung

Marion Engert & Marvin Kranz

Sergej Werfel

Ablauf

- Begrüßung
- Motivation & was euch erwartet
- Vorstellung technischer Themen & Interview-Fragen an einen Dev
- Was könnt ihr als SM/PO noch tun?
- Weitere Themenzusammlung
- Abschluss & Feedback
- 18 - 18:30 Uhr: offener Austausch

Motivation & Erwartungshaltung

Unsere Inspiration für diesen Thema

Was hilft es, die technischen Verständnis als PO und SM zu haben?

- **SM: Kann die Entwickler besser dabei unterstützen, ihre Dynamisierung wahrzunehmen**
- **PO: Kann besser beurteilen, wie es um die Qualität seines Produktes steht und ob geplant dafür gutes wird.**
- **Kann Erwartetes Kommunikation mit den Entwicklern, mit deren Anforderungen und Bedürfnissen besser verstanden werden.**

Wir haben hierfür 18 Themen für heute vorbereitet. Davon werden wir dezidiert näher betrachten.

Das Board gibt als Rawdoct und bildet für sich betrachtet, anlässe für eine Exit bei Bedarf im Nachgang durchchauen könnt.

Themen

Build Pipeline
Pair Programming
APIs
Unit Tests
Clean Code
Testpyramide
Stages
Security
Monitoring
Automatisierte vs. manuelle Tests
Mikrotrilog
Frontend-Backend-Datenbank
Softwarearchitektur
Softwaremetriken
Technische Schulden
Test Driven Development (TDD)
Merge-Request / Pull Request
Interviewfragen
Weitere interessante Themen
CI / CD

Wir starten mit "Innere & äußere Qualität" und "Testpyramide" und danach entscheidet ihr, mit welchen Themen es weitergeht. Verständnisfragen gerne jederzeit, Diskussion und Austausch gerne im Nachgang.

Innere und äußere Qualität

Innere und äußere Qualität

Begriff: Innere und äußere Qualität

Erklärung
- **Innere Qualität** dient vor allem Programmierern unterstützen.
- **Äußere Qualität** dient vor allem Anwendern (z. B. Bedienung, Stabilität, Geschwindigkeit, Benutzerfreundlichkeit, Barrierefreiheit, Sicherheit, Architektur und Clean Code-Prinzipien). Der ist für Entwickler immer wichtig, weil sie die Auswirkungen ihrer Entscheidungen sehen.
- **Äußere Qualität** kann auch Auswirkungen auf die Entwicklung haben, z. B. wenn die Nutzer immer komplexere oder ineffizientere Anforderungen, Funktionalität und Stabilität. Sie ist ebenso wichtig wie die innere Qualität.

Alltagsnutzen
- **Innere Qualität:** Technischer Zustand des Codes (z.B. Lesbarkeit, Wartbarkeit, Testbarkeit, Modularität, Einhaltung von Clean Code-Prinzipien). Der ist für Entwickler immer wichtig, weil sie die Auswirkungen ihrer Entscheidungen sehen.
- **Äußere Qualität** kann auch Auswirkungen auf die Entwicklung haben, z. B. wenn die Nutzer immer komplexere oder ineffizientere Anforderungen, Funktionalität und Stabilität. Sie ist ebenso wichtig wie die innere Qualität.

Quelle: ist KI erstellt

Impact
- Innere Qualität ermöglicht langfristige Weiterentwicklung, geringere Wartungskosten, schnellere Fehlersuche.
- Äußere Qualität sichert Akzeptanz, z. B. zufriedenere Nutzer/innen, besserer Akzeptanz und Weiterkauf, durch Produktfehler oder Ausfälle, weniger Frust und Supportaufwand.

Erklärung des PO & SM:
- **Product Owner:** Innere und äußere auf äußere Qualität Anforderungen sollte ohne die Entwicklerqualität zu berücksichtigen. Dies führt zu Unzufriedenheit bei Entwicklung einer Produkts, Bedienung, technische Schulden erkauert.
- **Scrum Master:** Fördert eine ausgewogene Qualitätsverständnis zu fördern, technische Schulden abgebaut Prinzipien in Sprint und Backlog zu unterstützen Entwicklerteams Qualität zu Verbesserungen zu helfen. Dafür sorgen, dass Dokumente ist Ziele gut im Team gelebt werden.

Weiterführend
- Robert C. Martin: Clean Code
- Martin Fowler: Refactoring

Clean Code

Clean Code

Begriff: Clean Code

Erklärung
- Clean Code ist systematisch Qualitätsziele, den einfach lesbar, verständlich und wartbar. Ist Ziel ist nicht nur, dass der Code funktioniert, sondern dass er auch für andere auch für einen erfahrenen Entwickler nachvollziehbar bleibt.
- Clean Code ist kein Stil, sondern eine Haltung.

Quelle: https://commons.wikimedia.org/wiki/File:Clean_Code.jpg

Alltagsnutzen
- **Lesbarkeit:** Code ist für Menschen, das man gegenüber ist, mit anderen Menschgenossen, durch Kollegen, Entwickler, Entwickler und gleich Wichtige, dieses kann so verstehen.
- **Einfacher Code:** Kann einfacher Kleinigkeiten Fehler, sollte Abkürzungen und Verkürzen vermeiden, weil sie den Code schwer verständlich machen.

Quelle: ist KI erstellt

Impact
- Kürzere Lesezeit und Wartbarkeit
- Weniger Fehler und Bugs
- Schnellere Einarbeitung neuer Entwickler/innen
- Geringere Kosten für Refactoring und Fehlersuche

Erklärung des PO & SM:
- **Product Owner:** Kann sich nicht um Clean Code kümmern, weil ihr Verständnis fehlt.
- **Scrum Master:** Fördert den Austausch zwischen Entwicklern, Code Reviews und Refactoring-Sessions, damit Clean Code Prinzipien gelebt werden.

Weiterführend
- Robert C. Martin: Clean Code
- Martin Fowler: Refactoring
- Kent Beck: Test-Driven Development: By Example

Technische Schulden

Technische Schulden

Begriff: Technische Schulden

Erklärung
- Technische Schulden entstehen, wenn LÖ Sie kürzfristige Reihenze technische Maßnahmen gezwungen werden, die spätere Arbeit Aufwand für Wartung, Anpassung oder Erweiterung erhöhen. Technische Schulden sind vergleichbar mit finanziellen Schulden: Sie sparen kurzfristig Zeit, müssen später aber zurückgezahlt werden.

Quelle: https://commons.wikimedia.org/wiki/File:Technical_debt_quadrant.jpg

Alltagsnutzen
- Technische Schulden sind etwas wie ein Kredit: Man nimmt schnell eine Zeit ein ohne Sorgen, kostet, bis dem irgendwann ihre Rechnung, das man dann zahlen muss. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Langsamere Entwicklung
- Mehr Fehler und Bugs
- Schwierigere Einarbeitung neuer Entwickler/innen
- Geringere Motivation
- Steigende Kosten für Refactoring und Fehlersuche

Erklärung des PO & SM:
- **Product Owner:** Kann, als „schnelle" technische Schulden verzichten und Refactoring Zeiten einplanen. Benötigt einen Technical Debt Backlog und langfristiger Qualität im architektonischen Sinne.
- **Scrum Master:** Fördert Bewusstsein für Technical Debt und Refactoring-Sessions, damit Technische Schulden im Backlog und unterstützt bei der Priorisierung mit Product Owner. Hilft Entwicklern, Technical Debt im Backlog sichtbar zu machen.

Weiterführend
- Martin Fowler: Technical Debt Quadrant (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)

APIs

APIs

Begriff: APIs (Application Programming Interface)

Erklärung
- APIs (Application Programming Interface, deutsch: Programmschnittstelle) ist eine standardisierte Schnittstelle zwischen Begriff Protocolen und Funktionen, über die Anwendungen oder Systeme miteinander kommunizieren können. Sie legen fest, welche Daten und Befehlen sie nutzen können, ohne zu wissen, wie die jeweils andere intern funktioniert. Über verschiedene Protocolle und Datenformaten können sie Informationen austauschen.

Quelle: https://commons.wikimedia.org/wiki/File:Application_Programming_Interface.svg

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Schnellere Entwicklung
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

Unit Tests

Unit Tests

Begriff: Unit Tests

Erklärung
- Unit Tests sind automatisierte, isolierte Einheiten-Ebene Codes (z. B. Funktionen, Methoden, Klassen). Sie prüfen, ob kleinste Bestandteile eines Programms, die sicherstellen, dass Software zuverlässig funktioniert, Fehler frühzeitig erkennen und Qualitätsziele erreichen.

Quelle: ist KI erstellt

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Schnellere Entwicklung
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

Test Driven Development (TDD)

Test Driven Development (TDD)

Begriff: Test Driven Development (TDD)

Erklärung
- Test Driven Development (TDD) ist eine Methode, bei der Tests vor dem eigentlichen Code geschrieben werden. Der Ablauf folgt einer Schleife: Red (Fehlschlagender Test) → Green (Code macht Test grün) → Refactor (Code verbessern, Tests bleiben grün).

Quelle: ist KI erstellt

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

CI / CD

CI / CD

Begriff: Continuous Integration (CI) / Continuous Deployment (CD)

Erklärung
- CI/CD ist eine automatisierte Pipeline, die sicherstellt, dass Software kontinuierlich integriert, getestet und bereitgestellt wird.
- Ziel: Änderungen schnell, zuverlässig und mit hoher Qualität in Produktion bringen.

Quelle: ist KI erstellt

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Schnellere Entwicklung
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

Build Pipeline

Build Pipeline

Begriff: Build Pipeline

Erklärung
- Eine Build Pipeline ist eine automatisierte Abfolge von Schritten, die aus geschriebenen Programmcode eine lauffähige Anwendung erzeugen. Dabei werden z. B. kompiliert, Tests ausgeführt, Pakete erstellt und gut Deployment.

Quelle: ist KI erstellt

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Schnellere Entwicklung
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

Monitoring

Monitoring

Begriff: Monitoring

Erklärung
- Monitoring beschreibt die systematische Beobachtung von Systemen, Anwendungen und Prozessen, um Leistung, Stabilität und Sicherheit sicherzustellen.
- Ziel ist es, Abweichen Fehlerfälle zu erkennen und gezielt zu reagieren, bevor Nutzer/innen Probleme bemerken. Es umfasst Messen, Alarmieren und Analysieren.

Quelle: ist KI erstellt

Alltagsnutzen
- Wenn du mal Urlaub machst und eine andere Mann kommt schnell ein Zim-Z ohne Sorgen Sorgen, hat dein Kollegin ihre Berechtigung, dein Mann hat sie mit dem Kollegen. Oder sollte schon jetzt, weil sich die Zinsen hoch.

Impact
- Schnellere Entwicklung
- Weniger Fehlerquellen durch klare Schnittstellen + unabhängige Entwicklung
- Einfachere Wartbarkeit durch getrennte Verantwortungen, geringere Abhängigkeiten
- Bessere Skalierbarkeit und Integrationsfähigkeit durch standardisierte Schnittstellen
- **Nachteile:** Mehrkosten

Erklärung des PO & SM:
- **Product Owner:** Kann APIs als Produktbestandteil, gute APIs sorgen
- **Scrum Master:** Fördert das Verständnis von Schnittstellen und unterstützt bei der Aufteilung der Systemarchitektur.

Weiterführend
- Martin Fowler: Internal Client Libraries (Blogpost)
- Professionalsoft und technische Schulden (Online Scrum Community)
- Kent Beck: Implementation Patterns

Was noch tun als SM / PO

Was könnt ihr als SM/PO noch tun, um technisches Verständnis zu erlangen?

Fragt die Devs :)
Schreibt mit Entwicklern Tickets / Stories / Aufgaben
Nehmt Entwickler mit in Meetings / Planungen / Reviews
Bildet eure Entwickler weiter (Kurse, Bücher, Konferenzen, Workshops)
Schreibt eure Tickets / Stories und lasst eure Entwickler sie ergänzen
Macht eure Entwickler zu Experten für eure Domäne
Macht eure Entwickler zu Experten für eure Domäne
ein sadisch Ticket / Story schreiben
Kauft ein Buch
Macht eure Entwickler zu Experten für eure Domäne
Macht eure Entwickler zu Experten für eure Domäne
Macht eure Entwickler zu Experten für eure Domäne