1. GRAD

ROTER GRAD

Prinzipien

- Don’t Repeat Yourself (DRY)
- Keep it simple, stupid (KISS)
- Vorsicht vor Optimierungen!
- Favour Composition over Inheritance (FCoI)
- Integration Operation Segregation Principle (IOSP)

Praktiken

- Die Pfadfinderregel beachten
- Root Cause Analysis
- Ein Versionskontrollsystem einsetzen
- Einfache Refaktorisierungs-muster anwenden
- Täglich reflektieren

2.GRAD

ORANGER GRAD

Prinzipien

- Single Level of Abstraction (SLA)
- Single Responsibility Principle (SRP)
- Separation of Concerns (SoC)
- Source Code Konventionen

Praktiken

- Issue Tracking
- Automatisierte Integrationstests
- Lesen, Lesen, Lesen
- Reviews

5. GRAD

BLAUER GRAD

Prinzipien

- Entwurf und Implementation überlappen nicht
- Implementation spiegelt Entwurf
- YAGNI - You Ain´t Gonna Need It

Praktiken

- Continuous Delivery
- Iterative Entwicklung
- Komponentenorientierung
- Test first

generic//de
the clean code company

4.GRAD

GRÜNER GRAD

Prinzipien

- Open Closed Principle (OCP)
- Tell, don´t ask
- Law of Demeter

Praktiken

- Continuous Integration
- Statische Codeanalyse (Metriken)
- Inversion of Control Container
- Erfahrung weitergeben
- Messen von Fehlern

3.GRAD

GELBER GRAD

Prinzipien

- Interface Segregation Principle (ISP)
- Dependency Inversion Principle
- Liskov Substitution Principle
- Principle of Least Astonishment
- Information Hiding Principle

Praktiken

- Automatisierte Unit Tests
- Mockups (Testattrappen)
- Code Coverage Analyse
- Teilnahme an Fachveranstaltungen
- Komplexe Refaktorisierungen

DIE CLEAN CODE DEVELOPER GRADE

Clean Code Developer ist man nicht einfach, sondern wird es. Es geht nämlich nicht darum, ein paar Regeln auswendig zu lernen, sondern das CCD-Wertesystem wirklich zu verinnerlichen. Das braucht Zeit und Übung. Deshalb ist das CCD-Wertesystem in Stufen unterteilt, die man als Entwickler eine nach der anderen erklimmt. Hierbei wird der gesamte Prozess als Kreis gesehen: wer alle Grade bearbeitet hat, beginnt wieder von vorne.

Jeder Entwicklungsstufe ist eine Farbe zugeordnet. Und wer an sich als Clean Code Developer (CCD) auf einer Stufe arbeitet, der trägt ein CCD-Armband als Zeichen seines Willens, sie zu meistern. Anders als im Judo-Sport entspricht die Farbe also nicht einem erreichten Grad, sondern dem in Arbeit befindlichen.

Roter Grad

Der wirkliche Weg zum Clean Code Developer beginnt mit dem roten Grad. Mit dem roten Grad setzt die Übungspraxis ein. Er enthält deshalb nur Elemente des CCD-Wertesystems, die absolut unverzichtbar sind. Der Einstieg soll so leicht wie möglich sein. Auf dieser Stufe geht es deshalb noch nicht so sehr um Softwareentwicklungsprinzipien, als vielmehr um den Aufbau einer fundamentalen Haltung zur Softwareentwicklung und zum Clean Code Developer.

Oranger Grad

Nachdem im roten Grad die Grundlagen für den Prozess der kontinuierlichen Verbesserung geschaffen wurden, geht es im orangen Grad darum, einige fundamentale Prinzipien auf den Code anzuwenden und erste Erfahrungen mit dem Mittel Nr. 1 zur Produktivitätssteigerung zu gewinnen: der Automatisierung von Abläufen. Da nur korrekter Code guter Code ist, dient die Automatisierung der Korrektheitsprüfung. Es geht also nicht um eine nice-to-have Eigenschaft von Code, sondern um seine Essenz.

Gelber Grad

Der gelbe Grad steht ganz im Zeichen automatisierter Tests. Beim orangen Grad ging es noch um die von außen ansetzbaren Integrationstests. Für sie war nicht unbedingt ein Eingriff in den Code nötig. Ab dem gelben Grad allerdings geht es nicht mehr ohne Tests unter der Oberfläche. Und nicht nur das: Getestet werden sollen die kleinstmöglichen Einheiten, nicht nur funktionale Durchstiche. Das bedeutet eine Änderung der Codierungspraxis, denn sonst lassen sich einzelne Klassen nicht isoliert, d.h. unabhängig von genutzten Diensten prüfen. Deshalb gehören zum gelben Grad auch objektorientierte Prinzipien, denn nur mit ihnen ist eine Ablösung von zu testendem Code von seinem „Untergrund“ möglich.

Grüner Grad

Im grünen Grad geht es weiter mit der Automatisierung. Die ist einfach der Schlüssel zur Produktivität und Reaktionsfähigkeit. Nur wenn maximal viele Tätigkeiten in der Softwareentwicklung automatisiert sind, kann sich der CCD auf das Wesentliche konzentrieren: die Implementation von Kundenanforderungen. Ohne Automatisierung hängt die Entwicklung sonst oft an Kleinigkeiten - das kostet Zeit.

Korrektheitsprüfungen und Releases sind dann mehr Strafe denn Mittel zum Erfolg. Nach der Automatisierung der Tests steht jetzt allerdings die Produktion auf dem Plan. Code am Entwicklerarbeitsplatz zu testen, ist eine Sache. Ihn auf einem unabhängigen Rechner erfolgreich zu übersetzen und zu testen, eine andere. Nur so lassen sich mehr oder weniger subtile Abhängigkeiten von den einzelnen Entwicklerarbeitsplätzen finden. Garniert wird diese Praxis dann noch mit weiteren Prinzipien zur Codestrukturierung und einem Werkzeug für bessere Architekturen.

Blauer Grad

Mit dem blauen Grad geht es in die Zielgerade des CCD-Wertesystems. Die Automatisierung ist noch einen Schritt weiter zu treiben. Nach Übersetzung und Test steht jetzt das Deployment auf dem Programm. Vor allem geht es im blauen Grad aber nun um Aspekte der Softwareentwicklung jenseits von Code und Tools: CCD kümmern sich nicht nur um gute Strukturen im Kleinen, sondern planen sie von vornherein im Großen. Es geht also um Architektur. Da wir uns jedoch bewusst sind, dass keine Planung eine perfekte Lösung definieren kann, gehört nicht nur zur Architektur, sondern zur Softwareentwicklung insgesamt auch ein passendes Vorgehensmodell. Das ist iterativ und soll während der Arbeit am blauen Grad nun auch eingeübt werden.

Evolvierbarkeit | **Korrektheit** | **Team** | **Produktions-effizienz** | **Kontinuierliche Verbesserung** | **Single Developer**