Implementation och Testning

Implementation

En bra implementation

- Completeness
- Maintainability
- Traceability
- Correctness
- Readability
- Performance

Guidelines

- Storlek på funktioner/metoder
- Namn med flera ord
- Indentering och mellanslag
- Speciella egenskaper i ett språk
- Namngivning
- Namn på filer

Kommentarer

- Referens till externa dokument
- Förklaring av koden
- Vad det är tänkt att koden ska göra
- Skriva om koden som kommentar
- Markeringar
- En summering av koden

Debugging

- Verifiera
- Hitta
- Fixa
- Reproducera

Assertions

Optimering

- Profiler
- Algoritm
- Verktyg för optimering
- Implementation

Refactoring

Code smell

- "Avundsjuka"
- För intimt
- Swicth-statements (OOP)
- Stora klasser
- Långa metoder
- Duplicerad kod

Testning

Hitta fel

Testing

- Skapa testfall
- Köra testen
- Verifiera resultatet

- Inspections and reviews
- Formal methods
- Static analysis

Kvalitet

- Verifiering
- Validering

Fel

- Fault
- Failure
- Error

Testing

Techniques

Who?

- Programmers
- Testers
- Users

What?

- Unit testing
- Functional testing
- Integration and system testing

Why?

- Acceptance testing
- Conformance testing
- Configuration testing
- Performance testing
- Stress testing
- User interface testing

What test cases?

- Intuition
- Specification testing
  - Black-box testing
- Code
  - White-box testing
- Existing test case
  - Regression testing
- Faults

- Equivalence Class Partitioning
- Boundary Value Analysis
- Path Analysis
- Combinations of conditions
- TDD

När ska man sluta testa?

Inspection and reviews

Steg

- Planning
- Overview
- Preparation
- Examination
- Rework
- Follow-up

Formal methods

Static Analysis

- Intermediate documents
- Structured format
- Source code
- Executable files

/Implementation och Testning/Testning

Eftersom att det är ganska svårt att bevisa att ett program är korrekt så finns det egentligen bara ett sätt att ta reda på om ett program fungerar som tänkt eller inte: testa det. Nackdelen med testning är givetvis att det inte är någon garanti för att allting är korrekt, det är bara en indikation på att de fall som testades fungerar under de omständigheter som testen skedde i.

/Implementation och Testning/Testning/Hitta fel/Inspections and reviews

En systematisk genomgång av systemet av två eller flera personer.

/Implementation och Testning/Testning/Hitta fel/Formal methods

Bevisa att ett program är korrekt genom att analysera programmet och genom någon form av bevissystem visa på att det är korrekt.

/Implementation och Testning/Testning/Hitta fel/Static analysis

Använda verktyg för att kolla på kod och/eller “binärer” för att se om det finns några fel.

/Implementation och Testning/Testning/Testing/Techniques/Equivalence Class Partitioning

Dela upp det möjliga indatat i sådana grupper att innehållet i varje grupp kan anses vara lika när det gäller testning - dvs om ett värde i en grupp ger korrekt resultat borde alla andra värden i samma grupp också ge korrekt resultat.

/Implementation och Testning/Testning/Testing/Techniques/Boundary Value Analysis

Att kolla olika gränsvärden är ofta en bra strategi, med gränsvärden menar jag då de värden som finns mellan olika grupper av värden. T.ex. när jag sätter upp ett kalkylblad för att beräkna betyg för en kurs så testar jag alltid gränserna mellan U och 3, 3 och 4, 4 och 5 men sällan alla värden som t.ex. ger en 3:a. Jag använder mig alltså av equivalence class partitioning och boundary value analysis tillsammans.

/Implementation och Testning/Testning/Testing/Techniques/Path Analysis

Försök att använda sådana indata att alla exekveringsvägar testas.

/Implementation och Testning/Testning/Testing/Techniques/TDD

Jonas …

/Implementation och Testning/Testning/Testing/När ska man sluta testa?

Jonas …

/Implementation och Testning/Testning/Testing/Formal methods

Bevisföring att ett program är korrekt

/Implementation och Testning/Implementation

Målet med implementation är givetvis att komma fram till något som fungerar och som kunden kan använda på ett bra sätt.

/Implementation och Testning/Implementation/En bra implementation

Alla vill väl göra en bra implementation men frågan är hur man gör och kanske framför vad “bra” betyder. Vi har tidigare nämnt:

/Implementation och Testning/Implementation/En bra implementation/Completeness

Förhoppningsvis så ska systemet vara komplett också, dvs innehålla de delar du och kunden har kommit överens om.

/Implementation och Testning/Implementation/En bra implementation/Maintainability

I de allra flesta fall så konstrueras inte ett program för att sedan användas tills det ersätts med något annat program. Det normala är att programmet konstrueras, tas i drift,fel upptäcks, fel fixas, det upptäcks att programmet behöver ytterligare funktionalitet, implementation av denna funktionalitet, osv.
Det är alltså nödvändigt att det är någorlunda lätt att underhålla ett system.

/Implementation och Testning/Implementation/En bra implementation/Traceability

Det ska vara möjligt att se varför olika koddelar existerar. Det ska alltså gå att spåra varför koddel X är implementerad till t.ex. en user story.

/Implementation och Testning/Implementation/En bra implementation/Correctness

Programmet ska ju inte bara gå att köra, det ska ju ge rätt resultat också!! Och med rätt resultat menar jag det resultat som kunden förväntar sig - inte det du tycker att det ska vara.

/Implementation och Testning/Implementation/En bra implementation/Readability

Att skriva kod är ganska lätt, att skriva kod som är lätt att läsa och förstå är inte lika lätt. Här gäller det att få till både algoritm, dokumentation och implementation på ett sätt som gör att de som senare tittar på koden har lätt att förstå och modifiera.

/Implementation och Testning/Implementation/En bra implementation/Performance

Prestandan på ett program är ofta av mindre betydelse än en del tror. Naturligtvis är det så att bättre prestanda är bra om alla andra relevanta problem är lösta. Men det är kanske inte det viktigaste och det man ska ägna allra mest tid åt (beror naturligtvis på vad det är man implementerar).
Det kan också vara dumt att ägna mycket tid åt optimering om det är oklart vad som behöver optimeras, bättre att göra systemet klart för att sedan titta på vart flaskhalsarna finns.

/Implementation och Testning/Implementation/Guidelines

De flesta större organisationer har någon sorts standard över hur kod ska skrivas. Denna standard kan innefatta “triviala” saker som namngivning och indentering men ibland så finns det även riktlinjer för vilka språkkonstruktioner som inte får användas eller hur något ska implementeras.
Oftast är det inte t.ex. antalet mellanslag som är det viktiga, det viktiga är att alla gör likadant. På det viset blir det mycket lättare att sätta sig in i existerande kod.
De flesta av er har ju redan provat att följa en kodstandard på AP-Javan. Vanliga saker som man tar upp är:

/Implementation och Testning/Implementation/Guidelines/Indentering och mellanslag

Detta borde inte vara någon nyhet för er vid det här laget men indenteringen ska vara sådan att det avspeglar kodstrukturen på ett lämpligt sätt.

/Implementation och Testning/Implementation/Guidelines/Speciella egenskaper i ett spr...

Detta är kanske inte det första du tänker på men en del organisationer förbjuder användningen av vissa programkonstruktioner eller språkegenskaper. Jag har själv stött på tre olika saker när jag har hälsat på hos olika företag: rekursion (svårt, tar mycket minne, komplicerat), templates i C++ (code explosion), trådar (“generellt så klarar inte våra programmerare av detta så det är bara två stycken som får skriva den koden och de andra får använda färdiga klasser”)

/Implementation och Testning/Implementation/Guidelines/Namngivning

Har tycker jag att det är viktigt att man kallar en sak för samma sak hela tiden. Samt att man som Jonas sa använder termer som ligger i problemdomänen.

/Implementation och Testning/Implementation/Kommentarer/Förklaring av koden

Dvs koden är så komplex att man behöver förklara vad som händer, fundera på om koden kan förenklas.

/Implementation och Testning/Implementation/Kommentarer/Vad det är tänkt att koden ska...

Dvs det är denna kommentar som koden ska uppfylla, om inte koden gör detta så är koden fel.

/Implementation och Testning/Implementation/Kommentarer/Skriva om koden som kommentar

```
i = i + 1 // Öka i med 1
```

/Implementation och Testning/Implementation/Kommentarer/Markeringar

```
// FIXME
// TODO - add blablabla to blipp
```

/Implementation och Testning/Implementation/Kommentarer/En summering av koden

Vad koden gör - om detta är en del av en funktion/metod borde det kanska vara en separat funktion/metod/klass.

/Implementation och Testning/Implementation/Assertions

Dvs “denna kod antar att följande villkor är sanna”. Post-condition - “denna kod garanterar att följande villkor gäller efter exekvering”

/Implementation och Testning/Implementation/Debugging/Hitta

Vart i koden finns felet? Det bästa som finns här är en källkodsdebugger och något sorts logförfarande - beroende på typ av fel.

/Implementation och Testning/Implementation/Debugging/Reproducera

Dvs reproducera felet så att det är möjligt att genomför felsökning

/Implementation och Testning/Implementation/Optimering

Som sagt: inte för tidigt

/Implementation och Testning/Implementation/Refactoring

Skriva om kod så att den blir bättre uppdelad

/Implementation och Testning/Implementation/Refactoring/Code smell

Intressant men kan vara svårt att ge en klar definition av.

/Implementation och Testning/Implementation/Refactoring/Code smell/“Avundsjuka”

En klass använder ett objekt från en annan klass i stor utsträckning

/Implementation och Testning/Implementation/Refactoring/Code smell/För intimt

En klass använder interna delar av andra klasser.