UNIVERSITÄT BASEL

MEDIZINISCHE FAKULTÄT

Über Bugs und Crashes

Adrian Schneider

Open-Source Seminar
University of Basel

Inhalt

- Bug-Klassifizierung
- Bug-Erkennung
  - Test-Driven-Development
  - Build-Systeme
  - Code-Analyse
  - Laufzeitanalyse
- Bug-Tracking-Systeme
  - Bug-Report
  - Patches

Klassifizierung

Nach Schweregrad:

- Schweregrad 1: Software-Absturz
- Schweregrad 2: Hauptfunktionalität beschädigt
- Schweregrad 3+: Zunehmend vernachlässigbar

Nach Auftreten:

- Immer
- System abhängig
- Manchmal
- Der Heisenbug ☺

Bug-Erkennung

> *If debugging is the process of removing software bugs, then programming must be the process of putting them in. (Edsger Dijkstra, Turing Award 1972)*

Das Erkennen von Bugs sollte

- **Vor dem Release geschehen**
- Günstig sein (automatisch)

Test-Driven-Development (TDD)

Die Implementierung eines neuen Feature beginnt mit seinem eigenen Test-Code.

Bekannte Testbibliotheken:

- CppUnit für C++
- jUnit für Java

Build-Systeme

Zweck:

- Kompilieren des Codes prüfen
- Erzeugen von Binaries
- Funktionalität prüfen
- Entwickler benachrichtigen

Schritte:

- Neusten Code aus Repository nehmen
- Build
- Unit Tests
- Falls fehlgeschlagen, Entwickler benachrichtigen

Code-Analyse

Für die statische Source-Code-Analyse können beispielsweise folgende Tools verwendet werden:

- Compiler (Warnungen angehen!)
- Cppcheck (open-source)
- ...

Laufzeitanalyse

Eine Laufzeit-Software-Analyse kann mit diesen Tools durchgeführt werden:

- Debugger wie *gdb*. Benötigt DEBUG Build.
- *strace*, *valgrind* und *ldd* kann für DEBUG und RELEASE Builds gebraucht werden.

Bug-Tracking-Systeme

Die Hauptzwecke von Bug-Tracking-Systemen sind

- Bugs zu erfassen und verifizieren
- Bugs unter den Entwicklern zu verteilen
- Formale interne Abläufe zu befolgen

Bug-Tracking-Systeme sind oft Teil eines **Issue-Tracking-System**.

Bekannte Tools sind

- Mantis
- Bugzilla
- Track+

Bug-Report

Wenn man einen Bug gefunden hat, sollte zuerst abgeklärt werden ob

- Man die neuste Version installiert hat.
- Schon ein Bug-Report dazu geschrieben wurde.

Ein guter Bug-Report sollte

- Eine sehr kurze und klare Zusammenfassung des Bugs vorweisen
- **Detaillierte Schritte** zur Reproduzierbarkeit des Bugs beinhalten
- Das beobachtete und erwartete Software-Verhalten beschreiben

Bug-Report

Weitere hilfreiche Informationen wären

- Der Stack-Trace (Error Call Stack)
- Bei welcher Version der Bug reinkam (Regression?)

Unterstütztes Melden:

- Automatisches senden von Bugs an eine zentrale Stelle (Ubuntu, Windows).
- Spezifische Diagnose-Tools des Software-Hersteller (Firefox, QT).

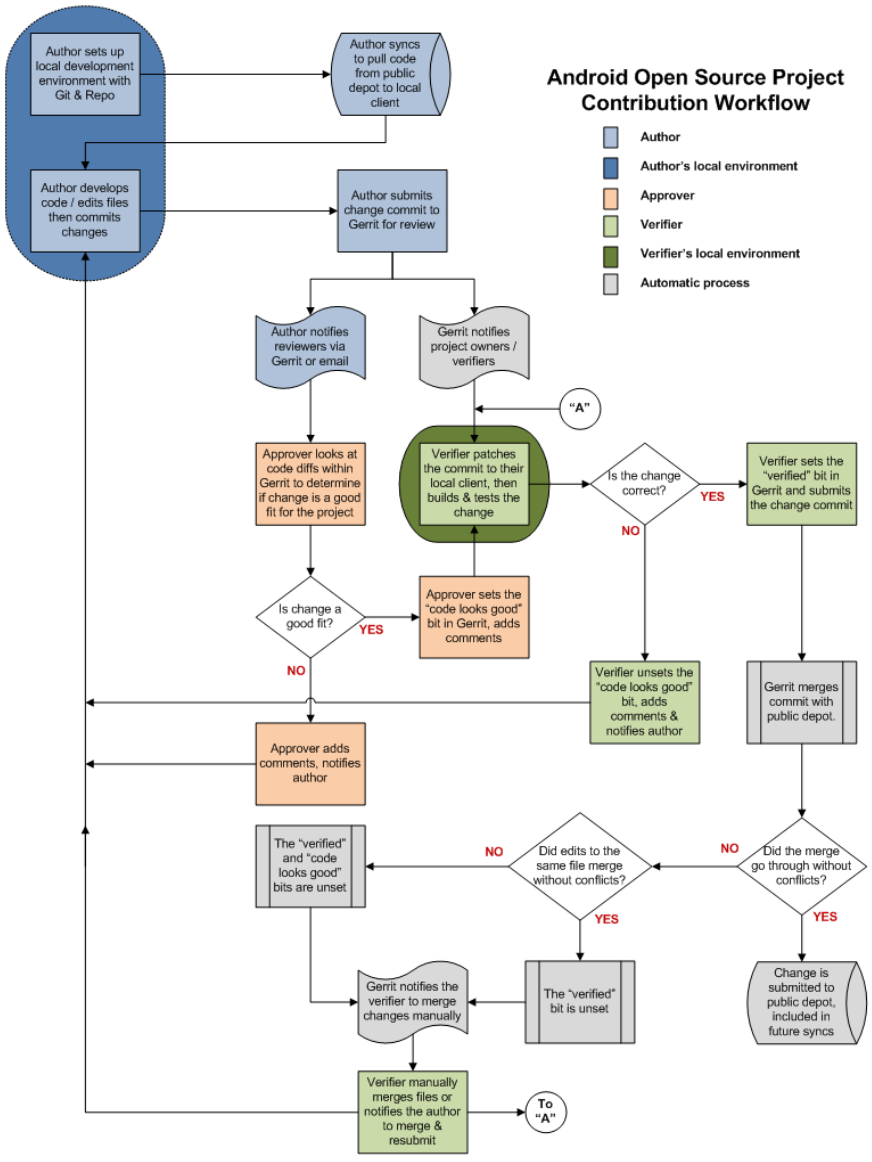
Patch

Der Patch sollte nur minimale Code Änderungen beinhalten.

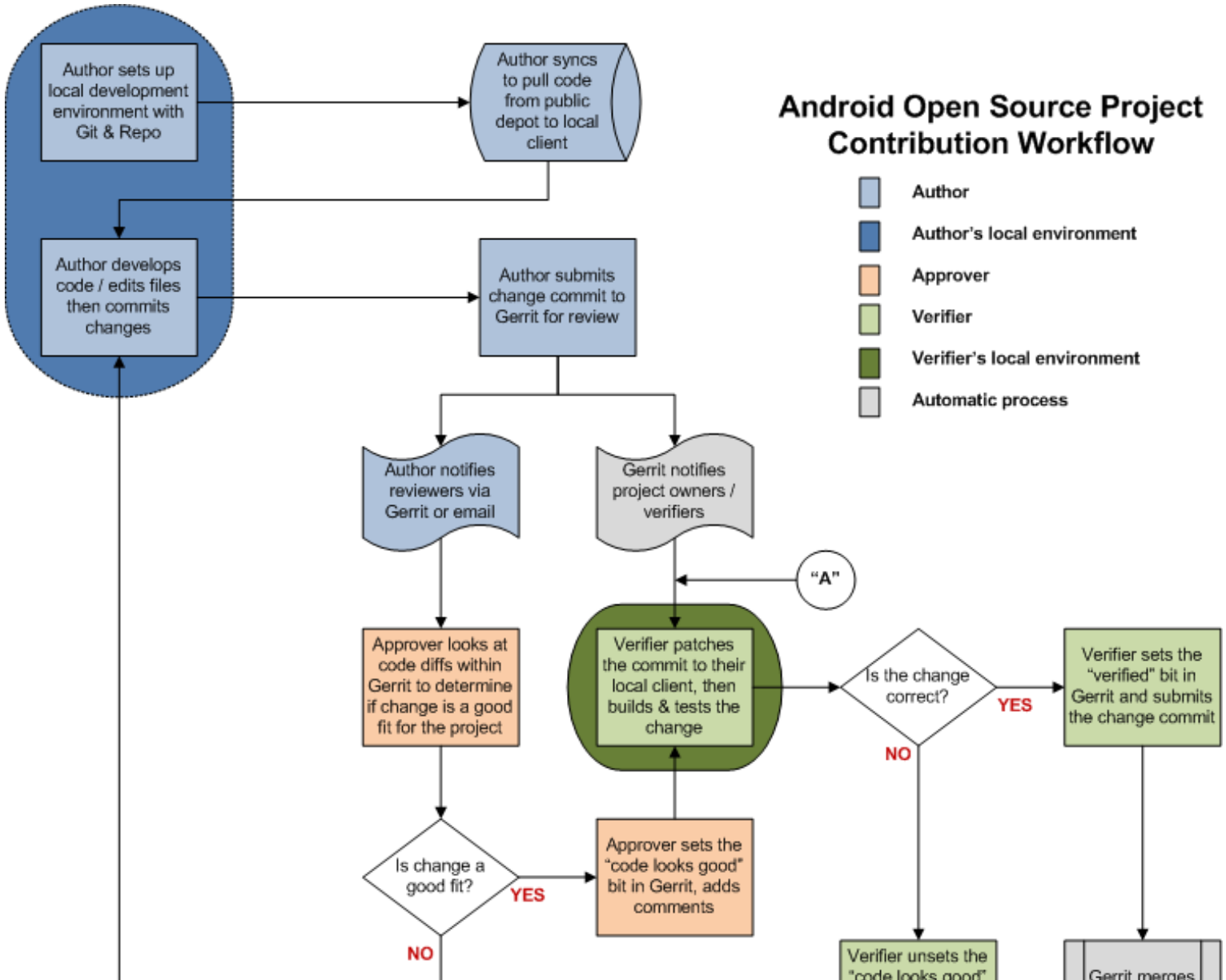
Das Patch-Handling variiert stark von Projekt zu Projekt.
Generell können vier Schritte beobachtet werden:

- Patch schreiben (Bug Fixing)
- Patch commit (Source Repository)
- Patch Code wird kontrolliert (Akzeptiert oder Abgelehnt?)
- Patch wird getestet (Bug gelöst?)

Life of a Patch (AOSP)



Life of a Patch (AOSP)



Life of a Patch (AOSP)