Contrasting Development & Release Stabilization on Linux Kernel

Md Tajmilur Rahman
Peter C Rigby

Concordia University
Montreal, Quebec, Canada

mdt_rahm@encs.concordia.ca
peter.rigby@concordia.ca
Development vs Stabilization

Stabilization Process

RC1  RC2  RC3  RC4  RC5  Release
Research Questions

- **RQ1:** What *release process* does Linux use?
- **RQ2:** How much *effort* is expended during development vs stabilization?
- **RQ3:** Do *developers work* in the same set of files during development and stabilization?
- **RQ4:** How long does it take (*lag/transit time*) for development vs stabilization changes to be released?
- **RQ5:** Is there a *rush-to-release* period?
Results - RQ1: Release Process

- Rolling Release Model
- Has longer release cycle in comparison to Firefox and Google projects
- Has more branches to get merged in merge window
Results – RQ2: Effort Spent

- Data: From 2005 to 2013, 331 releases, **39** are stable
- Commits: 381k (Development: **77%**, Stabilization: **23%**)
- Churns: Development: **91%**, Stabilization: **9%**
- LOC/Commit: Development: **105**, Stabilization: **41**
Results – RQ2: Core Developers

- 80% changes in development by: 55 devs
- 80% changes in stabilization by: 23 devs
Results – RQ3: Developers' work

480 Developers' work get re-worked by others

171 Developers' work do not require any re-work

161 Developers re-work
Results – RQ4: Transit Time

- Development Commit Date to Release Date
- Development Commit Date to rc1
- Bug Fix Commit Date to rc*
- Bug Fix Commit Date to Release
Results – RQ4: Transit time

- Development commits: **35** days, Stabilization Commits: **8** days to be integrated to mainline
Results – RQ5: Rush to Release ??

Development seems to be slow and steady, although long distance between merge windows.
Results – RQ5: Rush to Release

- Kolmogrov-Smirnov test
- p-value: 0.18 in indicates there is no significant difference between churn in merge window and before merge window
Question?