

Erfolgreiche OpenSource Projekt-Maintenance

Michael Prokop <mika@grml.org>

4. März 2007



- Ich hab neue Software/Projekt! Und jetzt?
- Ich will Software/Projekt! Wie?
- Ich will mein Projekt erfolgreich machen. Wie?

- Ich hab neue Software/Projekt! Und jetzt?
- Ich will Software/Projekt! Wie?
- Ich will mein Projekt erfolgreich machen. Wie?

- Ich hab neue Software/Projekt! Und jetzt?
- Ich will Software/Projekt! Wie?
- Ich will mein Projekt erfolgreich machen. Wie?

- Projektanfang: zuerst Code, dann Öffentlichkeit
- Zielgruppe: für wen? Vermeidung von WFM!

- Projektanfang: zuerst Code, dann Öffentlichkeit
- Zielgruppe: für wen? Vermeidung von WFM!

- **Teamarbeit: niemanden überfordern**
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- Teamarbeit: niemanden überfordern
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- Teamarbeit: niemanden überfordern
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- Teamarbeit: niemanden überfordern
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- Teamarbeit: niemanden überfordern
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- Teamarbeit: niemanden überfordern
- Userfriendly: fehlertolerant
- Kompatibel: ab- und aufwärts
- Skalierungsfähig / Plattformübergreifend / Multilang.
- Single Point of Failure vermeiden (Truckfaktor)
- Langweilige Arbeit automatisieren

- **Bugreports: BTS, Webseite, Wiki, Mail**
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- Bugreports: BTS, Webseite, Wiki, Mail
- Produktvermarktung: Screenshots, Screencasts
- Produktverbreitung: Paketierung einfach machen, Announce-Feeds
- Werbung: Mundpropaganda
- Wichtige Sachen öffentlich dokumentieren: Changelog, Lizenz
- Contributions?
- Finanzierung: Sponsoring, Donations

- **Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .**
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- Versionkontrolle ist Pflicht! Eigene Infrastruktur, sourceforge, Google Code, . . .
- passendes Entwicklungsmodell dazu wählen: zentral (subversion) vs. verteilt (mercurial)
- Wiki (mit RSS-Feeds) und/oder BTS
- BTS: debugs, roundup, trac, bugzilla (Achtung vor Overhead!)
- Automatismen für Workflow einrichten (Bugreport, Bugfix, Changelog, Paketupload, Close Bug) einführen
- bei den Big Players[tm] Ideen anschauen! Debian, BSD, Solaris, Microsoft, . . .

- **KISS (Keep It Simple, Stupid)**
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten, . . .)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- KISS (Keep It Simple, Stupid)
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten, . . .)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- KISS (Keep It Simple, Stupid)
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten,...)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- KISS (Keep It Simple, Stupid)
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten,...)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- KISS (Keep It Simple, Stupid)
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten,...)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- KISS (Keep It Simple, Stupid)
- If it ain't broken don't fix it
- Schnittstellen (Plugins, Erweiterungsmöglichkeiten,...)
- Use the right tool, luke! „Wenn man nur einen Hammer als Werkzeug hat, sieht jedes Problem wie ein Nagel aus“
- Unit-Tests, Test-Suite
- Stresstests: read-only Partition, Festplatte voll, wenig RAM

- Hm - was will meine Zielgruppe eigentlich? Erwartungen?
- Leute befragen, ihre Probleme feststellen
- Umfrage durchführen - Webbased (z.B. phpsurvey) vs. Mail-Based
- Entscheidungen mit Zielpublikum koordinieren. Aber: Expertenmeinung einholen und mit anderen Entwicklern absprechen!

- Hm - was will meine Zielgruppe eigentlich? Erwartungen?
- Leute befragen, ihre Probleme feststellen
- Umfrage durchführen - Webbased (z.B. phpsurvey) vs. Mail-Based
- Entscheidungen mit Zielpublikum koordinieren. Aber: Expertenmeinung einholen und mit anderen Entwicklern absprechen!

- Hm - was will meine Zielgruppe eigentlich? Erwartungen?
- Leute befragen, ihre Probleme feststellen
- Umfrage durchführen - Webbased (z.B. phpsurvey) vs. Mail-Based
- Entscheidungen mit Zielpublikum koordinieren. Aber: Expertenmeinung einholen und mit anderen Entwicklern absprechen!

- Hm - was will meine Zielgruppe eigentlich? Erwartungen?
- Leute befragen, ihre Probleme feststellen
- Umfrage durchführen - Webbased (z.B. phpsurvey) vs. Mail-Based
- Entscheidungen mit Zielpublikum koordinieren. Aber: Expertenmeinung einholen und mit anderen Entwicklern absprechen!

- Lerne mit Kritik umzugehen
- Releasezyklen: Deadlines setzen, aber nicht stressen lassen (OS vs. Kommerz)

- Lerne mit Kritik umzugehen
- Releasezyklen: Deadlines setzen, aber nicht stressen lassen (OS vs. Kommerz)

- zsh
- Ekiga
- Ruby on Rails

- Antiverpeil-Howto
- 'Die Software Rebellen' von Glyn Moody (ISBN: 3478387302).
- 'The Cathedral and the Bazaar' Eric Steven Raymond
- 'Producing Open Source Software - How to Run a Successful Free Software Project' von Karl Fogel

Feedback

Danke für die Aufmerksamkeit!

Kontakt

Michael Prokop <mika@grml.org>

<http://michael-prokop.at/>

<http://grml.org/>