DevOps

Les numéros de versions

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

2020

Introduction

Les numéros de version permettent d'attribuer des identifiants uniques à des états particuliers d'un logiciel.

Comment construire ces identifiants?

Agenda

Cycle de vie logiciel

Les numéros de version

Les étapes de développement

Les 5 étapes principales:

Les étapes de développement

Les 5 étapes principales:

- Prototype
- Alpha
- Beta
- Release candidate
- Version stable

Prototype

- Premier jet de l'application
- Ne dispose que de peu de fonctionnalités
- Sert de démonstrateur

Version alpha

- Version interne (non accessible au public)
  - Dans les projets open source, les versions alpha peuvent être accessibles au public
- Début de la phase de test du logiciel
  - Test du fonctionnement interne du logiciel (white-box testing)
- Toutes les fonctionnalités de la version finale ne sont pas encore mises en œuvre
- Contient des bugs importants
  - Risques de crashs et/ou d'altération des données
- La phase *alpha* se termine par un *feature freeze*
  - Arrêt de l'ajout de nouvelles fonctionnalités

Version beta

- Objectif: corriger les bugs et les problèmes de performance
- Phase de tests intensifs des fonctionnalités (black-box testing)
- Utilisation de *Beta-testeurs*:
  - Des employés de la société, des bénévoles, un sous-ensemble des clients, des clients potentiels ...
  - Rapportent les problèmes rencontrés et apportent des suggestions
- La version beta peut servir de démonstrateur pour de futurs utilisateurs

Release candidate (RC)

Version admissible

- Version pouvant potentiellement être un produit final
  - Ne contient plus de bugs majeurs
- Objectif: détecter et corriger les derniers bugs

Version stable

- Version prête à être distribuée (Release to manufacturing – RTM)
  - Version ne comportant plus de bugs (ou presque)
  - "Going gold"
- General Availability (GA)
  - Toutes les étapes nécessaires à la commercialisation sont passées.
- Release To Web (RTW)
  - Equivalent de RTM
  - Meilleure description de la manière de distribuer un logiciel aujourd'hui

Agenda

Cycle de vie logiciel

Les numéros de version

Les numéros de version

Identification basée sur des numéros de séquence:

- Utilisation de séquences de lettres ou de nombres

Idée générale

- Plusieurs niveaux de séquence: 1.4.3
- Le choix du niveau à incrémenter est fait en fonction de l'importance des changements depuis la dernière version

Exemples de schémas de numérotation

- [major].[minor].[revision] (1.4.3)
- [major].[minor][revisiontype] (1.0b3, 1.0-rc2)

Changer de version

L'incrémentation des numéros de version est à l'appréciation de l'équipe de développement.

Pratique courante

- Incrémentation du *major*
  - Changements importants dans les fonctionnalités, risques d'incompatibilité avec les versions précédentes
  - Contre exemple: noyau Linux 2.6.39 $\rightarrow$ 3.0
- Incrémentation du *minor*
  - Changements mineurs des fonctionnalités ou correction de bugs majeurs
- Incrémentation de la *revision*
  - Correction de bugs mineurs

Ordre des version

Ordonner les versions

- $1.1.1 < 1.1.2 < 1.2.1$
- $1.1\text{-a1} < 1.1\text{-a2} < 1.1\text{-b1} < 1.1\text{-rc1} < 1.1\text{-rc2} < 1.1$

La version 1.0

- Première version stable du logiciel
  - Les versions alpha et beta sont numérotées en 0.X
- Dans le monde du libre, première version complète du logiciel

Notations

- a: alpha
- b: beta
- rc: release candidate
- rtm: release to market
- ga: general availability

Semantic versioning

http://semver.org/

Problème dans la gestion de dépendances avec plusieurs packages:

- Version lock: Si les dépendances sont trop contraignantes, impossibilité de mettre à jour un package sans publier une nouvelle version d'autres packages.
- Version promiscuity: si les dépendances ne sont pas assez contraignantes, risque de considérer comme compatibles des versions qui ne le sont pas.

Quel schéma de numéros de version adopter pour simplifier le problème?

Semantic versioning

http://semver.org/

Avant tout

- Définir l'API publique de mon logiciel

Règles de numérotation

Format: MAJOR.MINOR.PATCH

- Incrémenter MAJOR en cas de changement de l'API avec incompatibilités.
- Incrémenter MINOR en cas d'ajout de fonctionnalités avec retro-compatibilité
- Incrémenter PATCH en cas de correction de bugs avec retro-compatibilité

Long-term Support

Cycle de vie des logiciels

En particulier pour les codes open-source:

- *release early, release often*
- Une nouvelle version du logiciel inclus des corrections de sécurité et des nouvelles fonctionnalités
  - Les anciennes versions du logiciel ne sont très vite plus maintenues.
  - Parfois, on aimerait n'avoir que les correctifs pour les failles de sécurité

Long-term Support

- Version du logiciel pour laquelle seuls des correctifs de sécurité seront publiés
- Version du logiciel maintenue pendant une longue durée
  - Windows XP maintenu pendant 12 ans

Références

- Notes de D. Donsez
- `https://fr.wikipedia.org/wiki/Version_d%27un_logiciel`