Gestió de versions d'un projecte

Programació de Baix Nivell

Sebastià Vila-Marta

Enginyeria de Sistemes TIC
Universitat Politècnica de Catalunya
http://epsem.upc.edu

1 d'abril de 2020

1 Gestió de projectes i versionat

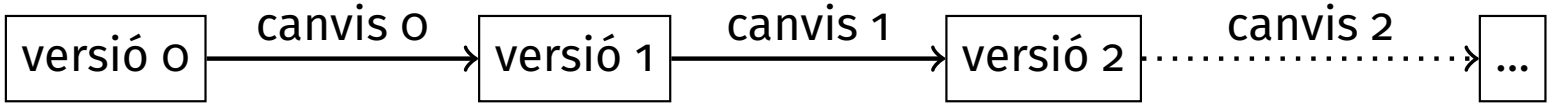
2 Subversion

Cicle de vida d'un projecte

- El projectes de desenvolupament de programari (i altres projectes):
  - Són entitats vives que canvien conforme passa el temps.
  - Comporten molts elements que transcendeixen el programari: documentació, comptabilitat, gestió dels equips, etc.
- La solvència programant no garanteix l'èxit d'un projecte: cal ser solvent també en la resta d'aspectes que inclou un projecte.
- L'objectiu de l'«enginyeria del programari» és avançar en aquests aspectes. El programari com a indústria.
- Una de les dimensions cabdals en la gestió dels projectes és la gestió del versionat.

Les versions

- El programari canvia durant el temps, tant mentre es desenvolupa com quan està en explotació.
- No existeix «el programa» sinó que existeixen «fotografies instantànies» del projecte en diferents moments del temps. Genèricament d'aquestes «fotografies» en diem versions.



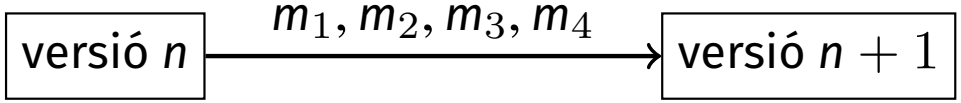
- En el cas dels projectes de programari, una versió es concreta en els directoris i fitxers que contenen els fonts, documents, dades i altres informacions que constitueixen el projecte.

Els canvis

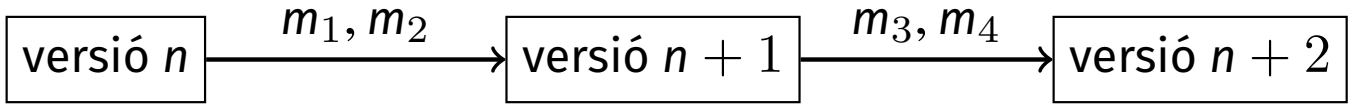
- Anomenem canvi al conjunt de modificacions que transformen una versió en una altra.
- Els canvis poden afectar, entre d'altres:
  - Al contingut dels fitxers que formen el projecte.
  - A l'existència o no dels propis fitxers.
  - A l'estructura de directoris del projecte.
- Per exemple, un canvi pot estar constituït per les següents modificacions:
  1. Afegir la línia `return l[3]` al fitxer `video.py`.
  2. Esborrar els ':' de la línia 327 del fitxer `stack.py`.
  3. Afegir el fitxer `queue.py`.
  4. Canviar el nom del directori `fotos` per `pictures`.
- Els canvis poden tenir molts orígens. Per exemple:
  - Avenços en el desenvolupament.
  - Millores quan el programa ja s'està usant.
  - Correcció d'errors detectats pels clients.
  - Adaptacions específiques per a un client concret.

Els canvis tenen significat I

- Quan una sèrie de modificacions constitueix un canvi?
- Considerem les següents modificacions:
  1. Afegir la línia `return l[3]` al fitxer `video.py`.
  2. Esborrar els ':' de la línia 327 del fitxer `stack.py`.
  3. Afegir el fitxer `queue.py`.
  4. Canviar el nom del directori `fotos` per `pictures`.
- Podriem considerar que totes juntes són un canvi:



- O també que constitueixen dos canvis:

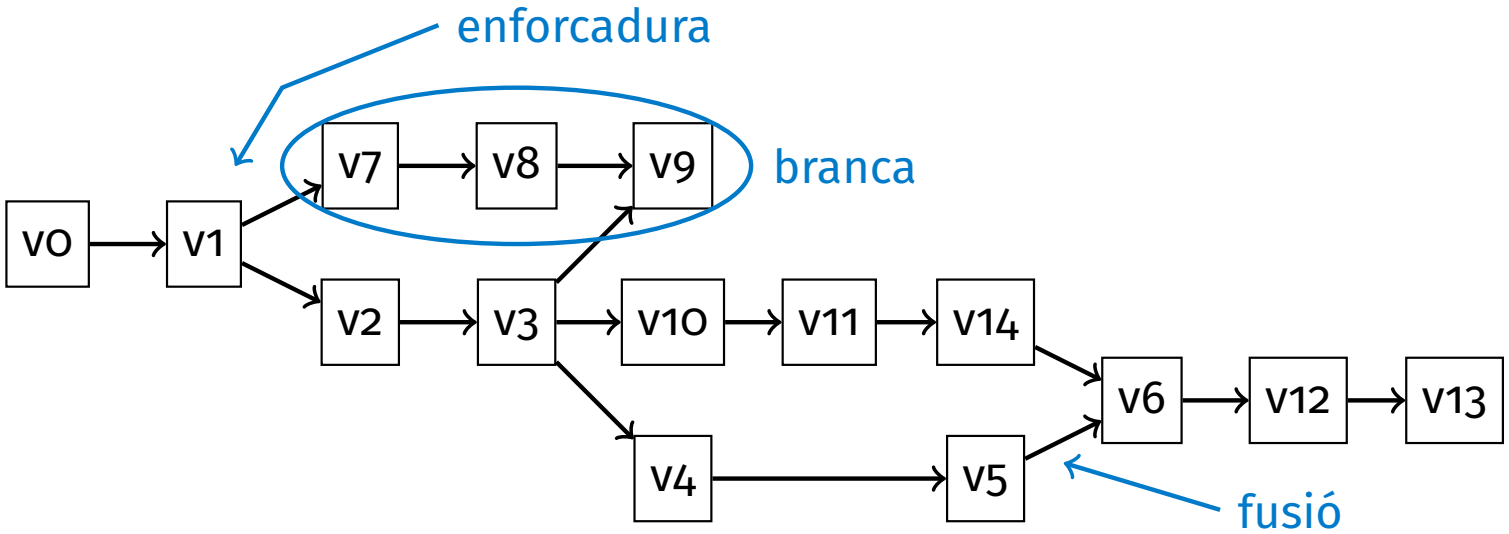


Els canvis tenen significat II

- S'agrupen les modificacions de manera que una versió tingui un significat coherent en el marc del projecte. Per exemple:
  - Correcció un error concret.
  - Una millora concreta.
  - Una reorganització particular del projecte.
- La decisió de quines modificacions constitueixen un canvi no pot automatitzar-se: és una decisió estratègica que prenen les persones de manera explícita.

L'històric de versions

- A mida que la vida d'un projecte avança, l'històric de versions es pot fer molt complex:



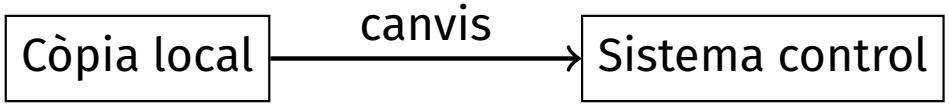
- Apareixen branques que segueixen variacions del projecte original i que evolucionen autònomament:
  - Adaptacions a clients concrets o maquinaris específics.
  - Proves de concepte.
  - Modificacions que desenvolupa una persona o equip.
- Algunes branques s'integren amb altres (fusió).
- Algunes branques es retroalimenten de canvis fets en altres.

La gestió de versions és inevitable i complexa

- No gestionar les versions d'un projecte condueix a situacions de fallida greu del projecte. Per exemple:
  - No poder reproduir un problema declarat per un client per que es desconeix el programari precís que està usant o no se'n té cap còpia exacta.
  - Haver d'integrar manualment les millores al projecte que han desenvolupat al departament de Queensland.
- Per gestionar aquest marasme s'usen eines de control de versions.
- La disciplina i les eines són complexes i cal una aproximació tranquila i paulatina per anar adquirint les habilitats i coneixements necessaris.

Eines de control de versions

- Les eines de control de versions segueixen el següent esquema de treball:
  - Un desenvolupador té una còpia local del projecte.
  - Alguns fitxers es declaren com a part del projecte —i són controlats per l'eina— i altres no.
  - El desenvolupador modifica la còpia local i va informant al sistema de control.
  - Quan el desenvolupador ho creu escaient, crea una versió amb totes les modificacions que ha fet i n'informa al sistema de control.
  - El sistema de control en pren nota.
- L'esquema és:



- Hi ha dos models de sistemes de control de versions:
  1. Centralitzats: una sola BD de versions (subversion).
  2. Distribuïts: diverses BD de versions (mercurial, git).

Sessió de subversion I

1. Creem una còpia local del projecte (buit) que hi ha en el SCV (Sistema de Control de Versions).

```
$ svn checkout http://server.svn.cat/project  mycopy
```

   - `mycopy` és el directori que es crearà per encabir la còpia del projecte.
   - Inicialment el projecte és buit i per tant `mycopy` també.
   - `http://server.svn.cat/project` és la URL del nostre projecte en el servidor de SCV.

2. Treballem en la còpia local creant l'estructura pel nostre projecte (la pràctica 4):

```
$ cd mycopy
$ mkdir p4
$ touch p4/README
```

Sessió de subversion II

3. Informem a l'SCV que hi ha dos nous fitxers que ha de tenir sota control.

```
$ svn add p4
```

   - Recursivament afegeix als fitxers que cal controlar `p4` i `README`.

4. Editem `README` i hi afegim contingut.

```
$ emacs p4/README
```

5. Considerem que les modificacions fetes constitueixen una canvi coherent i són constitutives d'una nova versió. A tal efecte informem a l'SCV.

```
$ svn commit -m 'Creació de l'esquelet de la pràctica'
```

6. Continuem treballant. Afegim un nou mòdul python. Ampliem el `README`.

```
$ emacs p4/recepta.py
$ emacs p4/README
```

Sessió de subversion III

7. Informem a l'SCV que també ha de controlar al fitxer `recepta.py` i informem que els canvis fets constitueixen una nova versió.

```
$ svn add p4/recepta.py
$ svn commit -m 'Afegit el mòdul recepta`
```