Version Control

Een stoomcursus Subversion

Sven van Haastregt

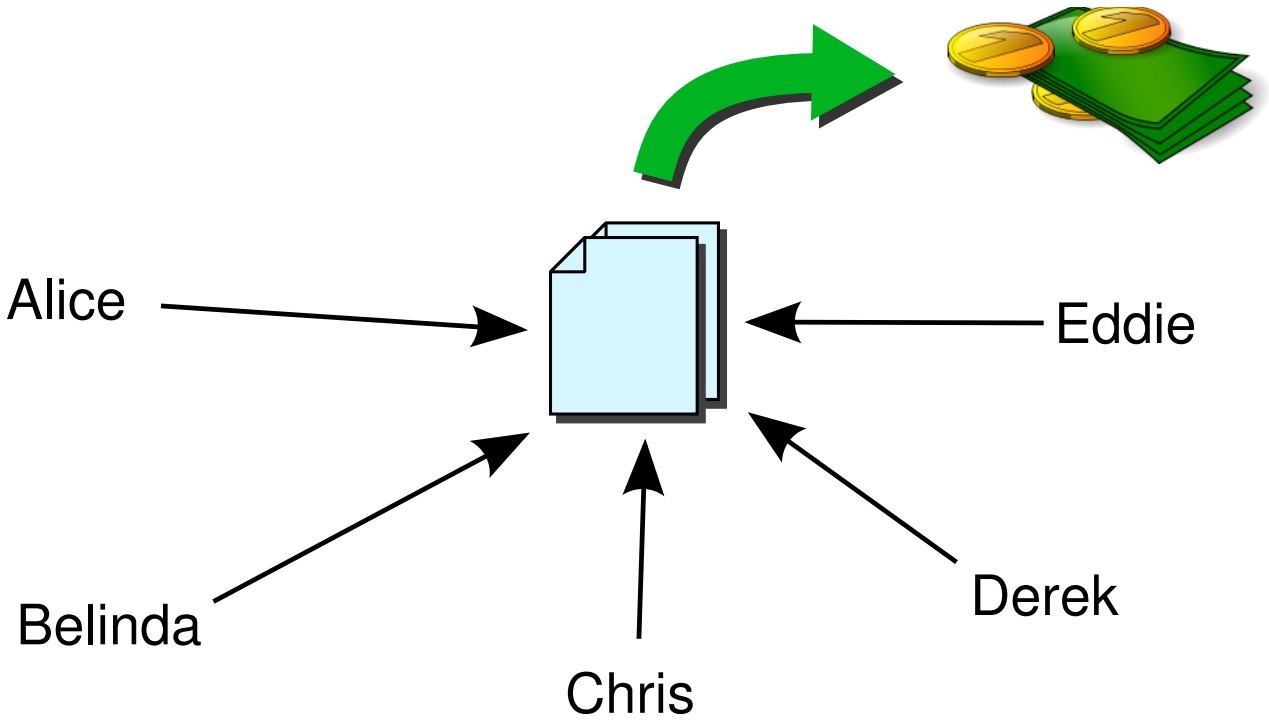
Challenges in Computer Science Seminar
LIACS, Universiteit Leiden

12 Februari 2013

Inhoud

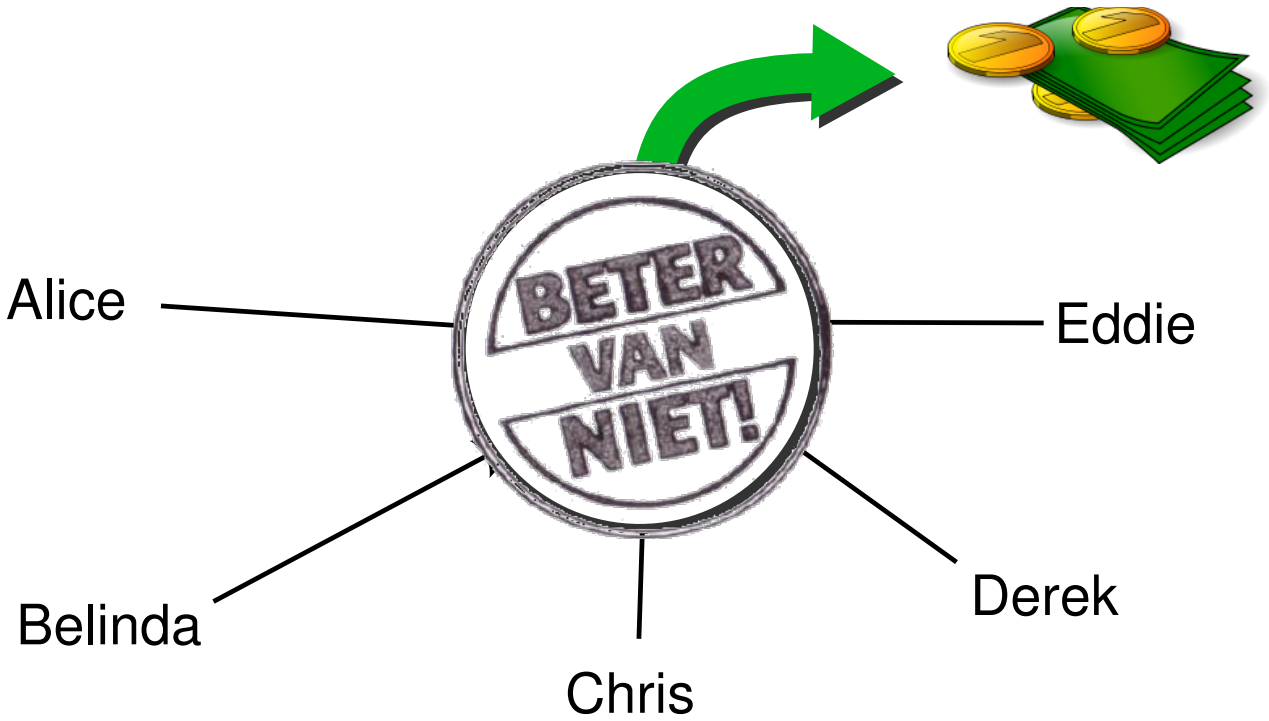
Samenwerken aan een project

1 directory waarin iedereen werkt?

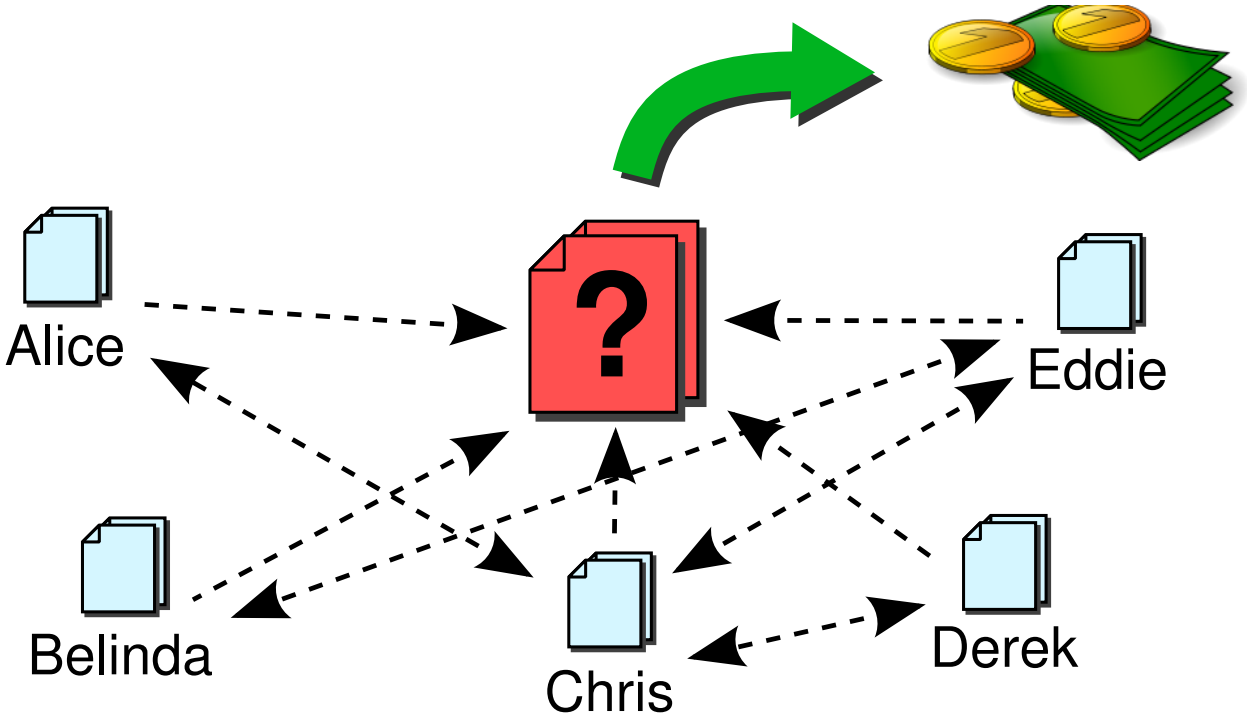


Samenwerken aan een project

1 directory waarin iedereen werkt?



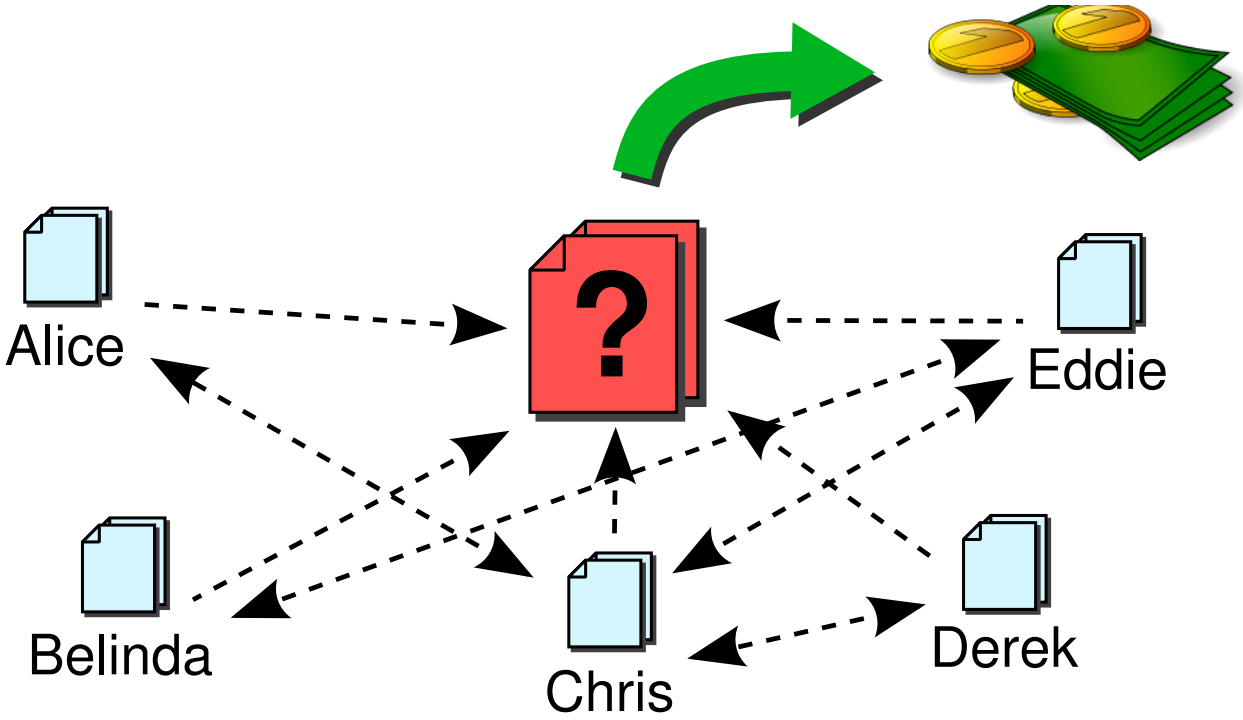
Samenwerken aan een project



Maar hoe verspreid je nieuwe code?

- Iedereen stuurt nieuwe versie naar iedereen.

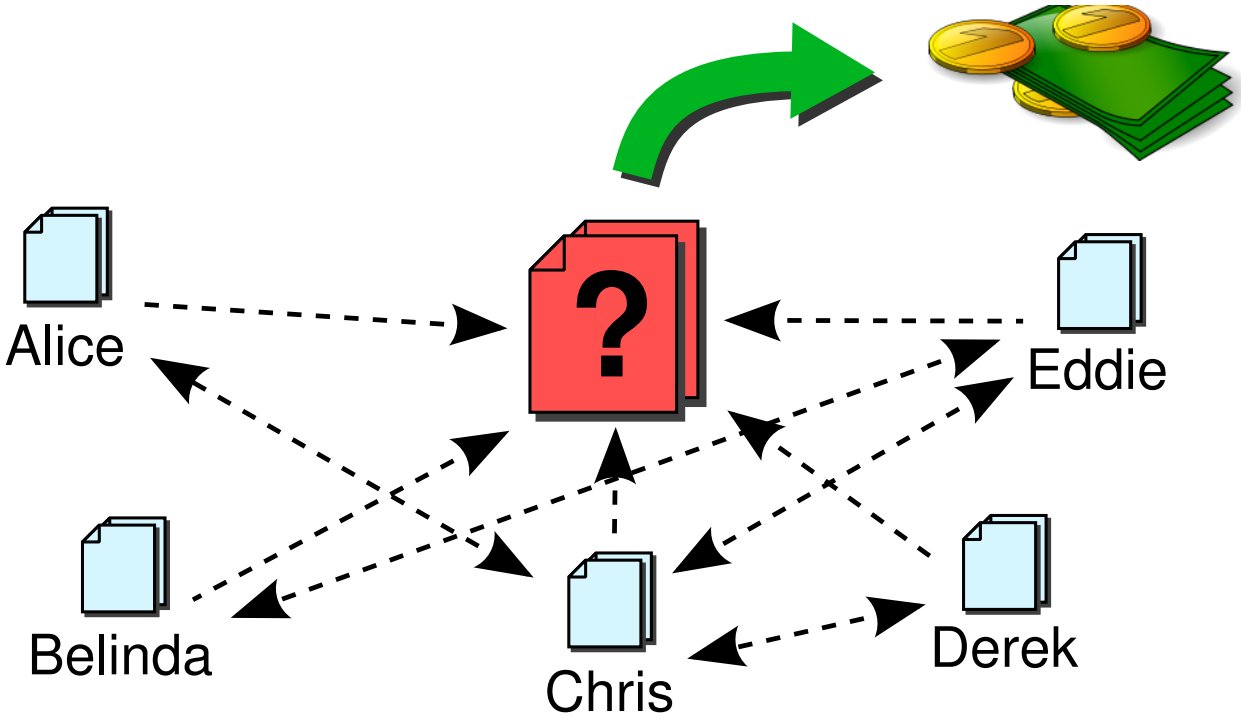
Samenwerken aan een project



Maar hoe verspreid je nieuwe code?

- Iedereen stuurt nieuwe versie naar iedereen.
- Iedereen stuurt nieuwe versie naar “integrator”.

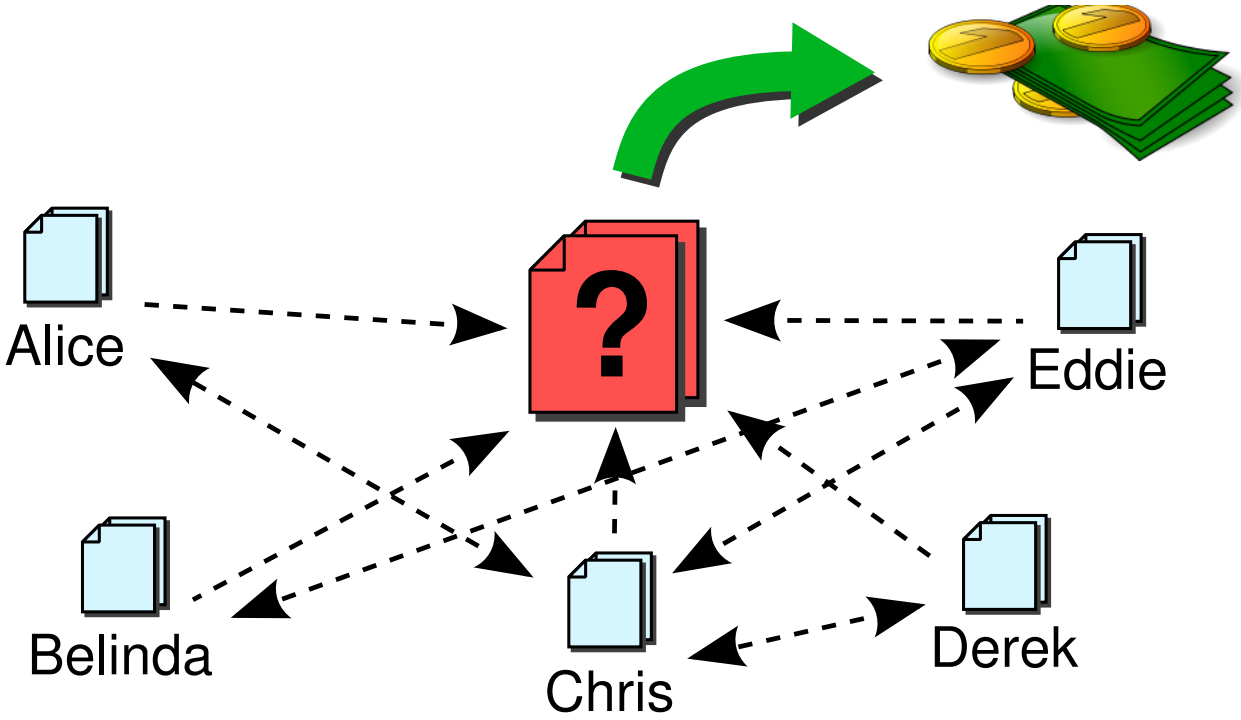
Samenwerken aan een project



Maar hoe verspreid je nieuwe code?

- Iedereen stuurt nieuwe versie naar iedereen.
- Iedereen stuurt nieuwe versie naar “integrator”.
- Iedereen stuurt alleen wijzigingen naar “integrator”.

Samenwerken aan een project



Maar hoe verspreid je nieuwe code?

- Iedereen stuurt nieuwe versie naar iedereen. 

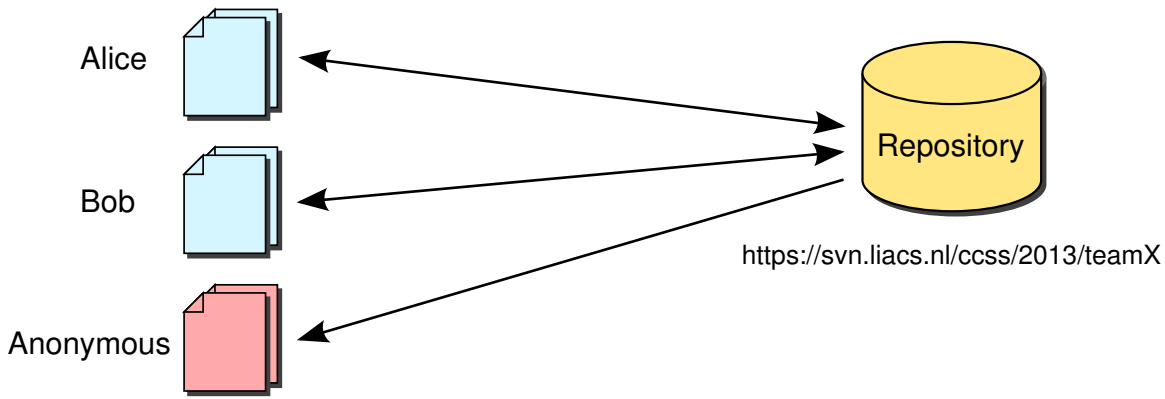
- Iedereen stuurt nieuwe versie naar “integrator”. 

- Iedereen stuurt alleen wijzigingen naar “integrator”. mwahhh...
- Iedereen gebruikt version control software. ✔

Version control (/ revision control / source control)

Gecentraliseerd model:

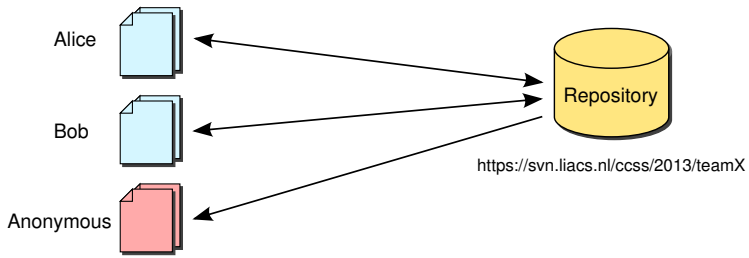
- Ergens wordt een "master kopie" bewaard (op een server).
- Iedere programmeur heeft een eigen lokale kopie.
- Wijzigen van de master kopie gebeurt via version control software.
- Werk van anderen wordt binnengehaald via version control software.
- Alle wijzigingen worden bewaard. Dus oude versies van de code zijn opvraagbaar.



Probleem: delen van files

Hoe een file delen zonder elkaar in de weg te zitten?

- Lock-Modify-Unlock:
  Een file kan alleen gewijzigd worden als je een exclusieve *lock* hebt.
  - Hindert parallel werken in dezelfde file.
  - Vergeten te unlocken.

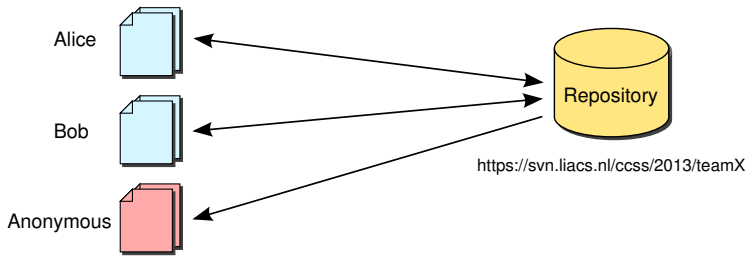


Probleem: delen van files

Hoe een file delen zonder elkaar in de weg te zitten?

- Lock-Modify-Unlock:
  Een file kan alleen gewijzigd worden als je een exclusieve *lock* hebt.
  - Hindert parallel werken in dezelfde file.
  - Vergeten te unlocken.

- Copy-Modify-Merge:
  Iedereen werkt ongehinderd in een eigen kopie. Version control systeem integreert verschillende versies.
  - Vereist een "slimmer" version control systeem.
  - Alleen bij overlappende wijzigingen (conflict) nog mensenwerk nodig.

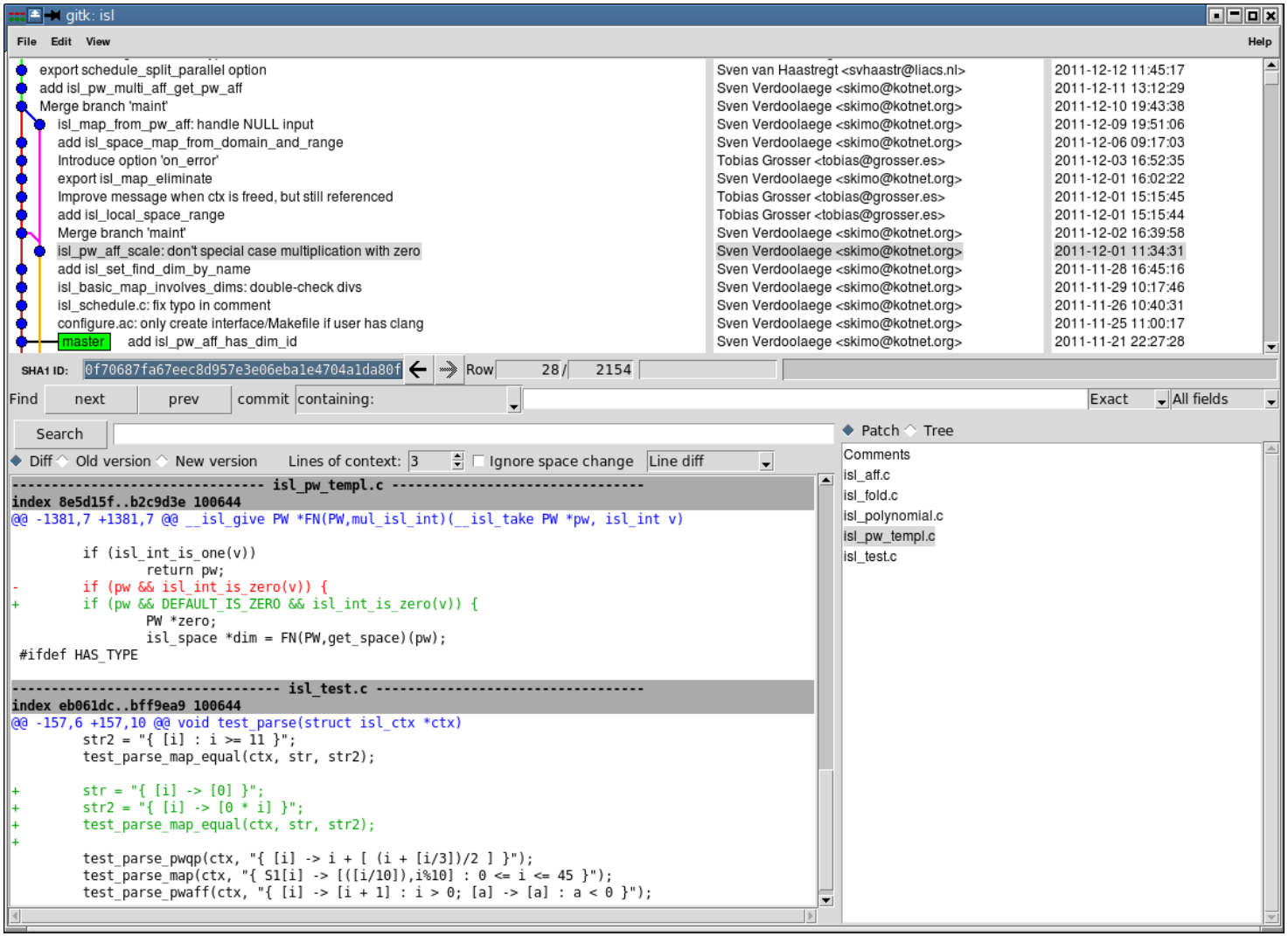


Version Control Systemen

- **CVS**: Concurrent Versioning System (1990).
  Voor vele projecten gebruikt (o.a. Mozilla, Python & Gnome).
- **SVN**: Subversion (2000).
  "CVS done right."
- **Git** (2005).
  "There is no way to do CVS right." – Linus Torvalds
  `http://www.youtube.com/watch?v=4XpnKHJAok8`

En dan is er ook nog: SCCS, Monotone, Mercurial, Bazaar, ...

Voorbeeld van een repository



Diff

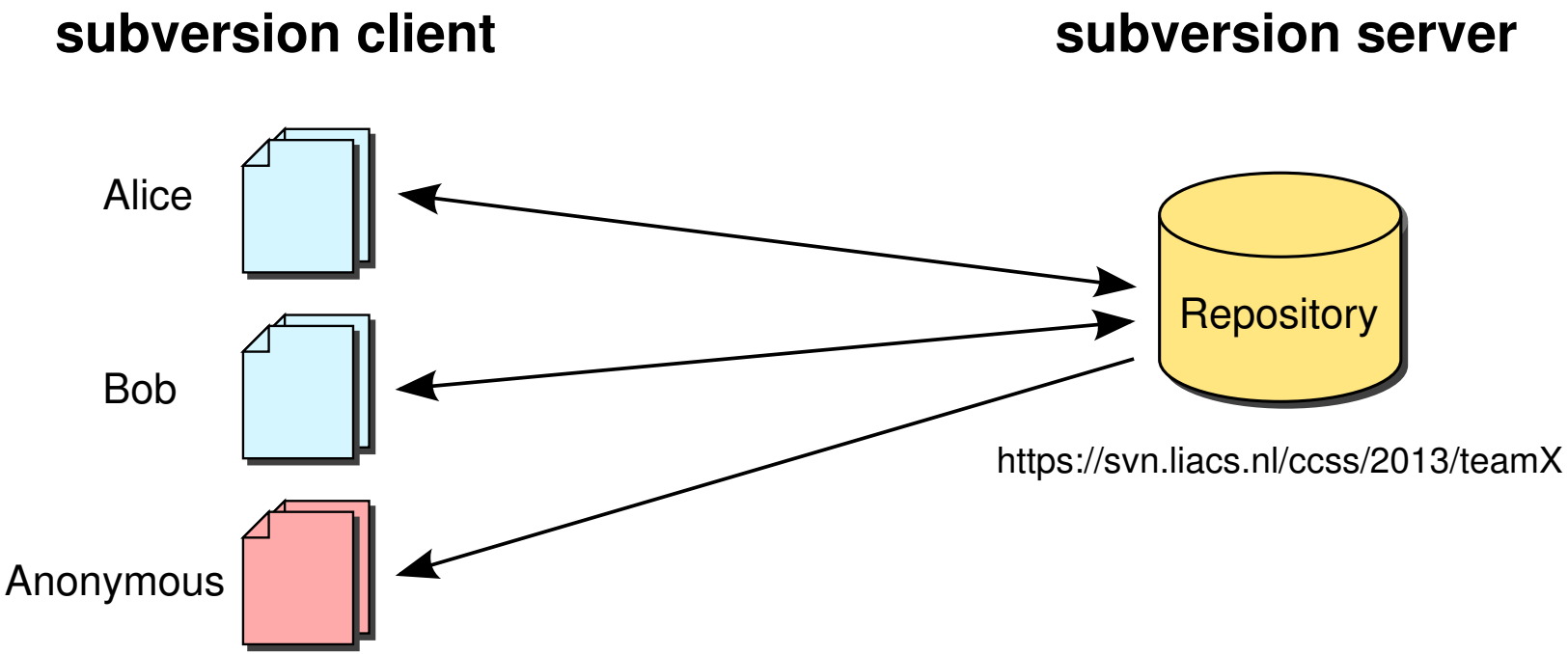
```
@@ -94,6 +114,6 @@ int main(int argc, char *argv[])
     cout << endl;
   }

+  pdgInfo.analyzeSCCs(sccs);
   delete sccs;

-  cout << "Nodes: " << nodes.size() << endl;
-  cout << "Edges: " << edges.size() << endl;
+  pdgInfo.printSummary();

   pdg->free();
   delete pdg;
```

- Handig bij uitwisselen van "platte tekst" (C++, Java, LaTeX, ...).
- Zie ook Unix commando's `diff` en `patch`.

Inhoud

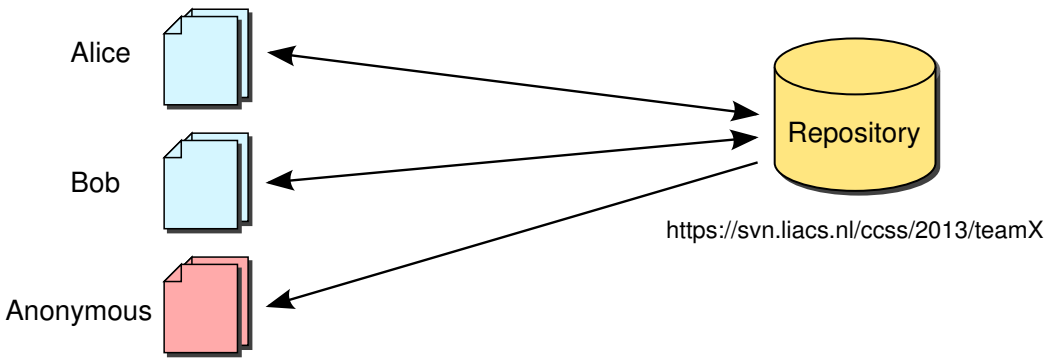
Subversion

- Wordt doorgaans gezien als opvolger van CVS.
- Onder andere gebruikt voor GCC, LLVM, Apache en Mono.
- Gecentraliseerd model, dus er is één server waar iedereen naar uploadt en van downloadt.



Terminologie

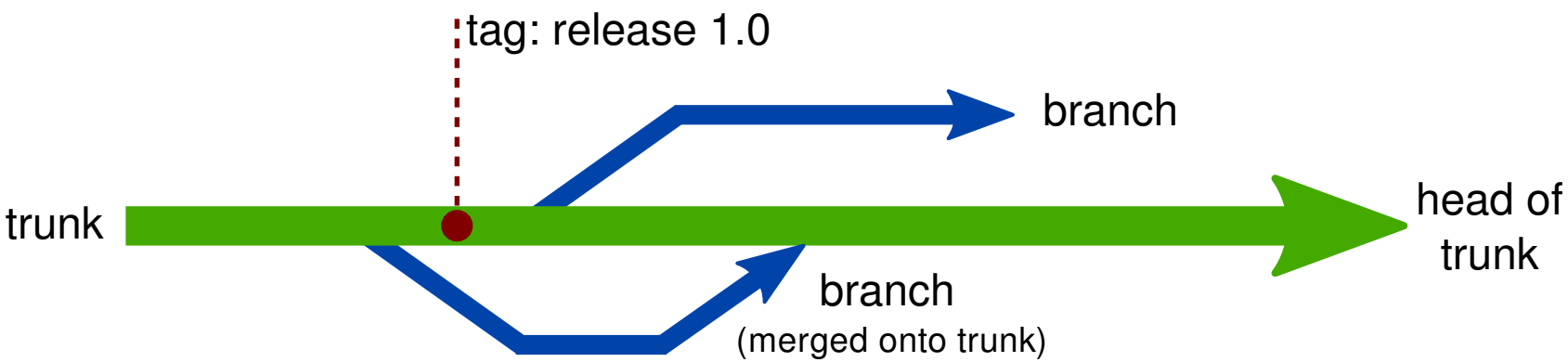
- **repository/repo**: centrale opslagplaats op de server waar files+geschiedenis bewaard worden.
- **working copy**: een lokale kopie van files uit de repo.
- **checkout**: ophalen van een working copy uit repo.
- **commit**: wijzigingen naar repo uploaden.
- **update/merge**: haal (laatste) versie op uit repo en voeg deze samen met de working copy.
- **head**: de meest recente versie in de repo.



Structuur van een SVN repository

Hoewel niet strikt vereist, is een SVN repository meestal als volgt opgezet:

- **trunk**: versie waarop iedereen normaalgesproken werkt.
- **branches**: vertakkingen van de trunk, met bijvoorbeeld experimentele features. Deze kunnen later weer gemerged worden met de trunk.
- **tags**: snapshots van belangrijke versies (bijv. releases).



Subversion gebruiken (client)

- Verschillende grafische tools (RapidSVN).
- Integratie in Windows/KDE (TortoiseSVN/Kdesvn).
- Integratie in ontwikkelomgevingen (Eclipse).
- Commandline: `svn`

Subversion gebruiken (client)

- Verschillende grafische tools (RapidSVN).
- Integratie in Windows/KDE (TortoiseSVN/Kdesvn).
- Integratie in ontwikkelomgevingen (Eclipse).
- Commandline: `svn`

  Algemene syntax van een aanroep:

  ```
  svn commando [opties] [files]
  ```

  bijvoorbeeld:

  - `svn checkout https://...`
  - `svn update`
  - `svn mkdir opgave1`
  - `svn commit -q hello.c`
  - `...`

checkout: Een lokale kopie ophalen van de server

- Om een lokale kopie te verkrijgen gebruik je `svn checkout`.
- Gebruik `--username` om een andere username te gebruiken.

```
[sven]$ svn checkout https://svn.liacs.nl/ccss/2013/teamX
Authentication realm: <https://svn.liacs ...
Password for 'sven':
Checked out revision 0.
```

- Als de checkout slaagt, is er nu een directory `teamX` waarin de laatste versie van de repository staat.
- In het bovenstaande voorbeeld is de repository nog leeg.

add+commit: Een file toevoegen en uploaden

- Om nieuwe files toe te voegen gebruik je `svn add`.

```
$ cd teamX/trunk
$ edit hello.c
$ svn add hello.c
A         hello.c
```

add+commit: Een file toevoegen en uploaden

- Om nieuwe files toe te voegen gebruik je `svn add`.
- Om **A**dded files te uploaden naar de server gebruik je `svn commit`. Tijdens het committen wordt een commit message gevraagd.

```
$ cd teamX/trunk
$ edit hello.c
$ svn add hello.c
A         hello.c
$ svn commit
Adding         trunk/hello.c
Transmitting file data .
Committed revision 1.
```

- Vi(m): **i** = invoeren van tekst; daarna **ESC :wq**

commit: Een file wijzigen

- Om wijzigingen aan "files onder SVN" te uploaden gebruik je weer `svn commit`.

```
$ edit hello.c
$ svn commit
Sending        trunk/hello.c
Transmitting file data .
Committed revision 2.
```

- Vi(m): **i** = invoeren van tekst; daarna **ESC :wq**

update: De laatste versie binnenhalen

- Om je lokale working copy te synchroniseren met de laatste versie op de repository, gebruik je `svn update`.

```
$ svn update
U    hello.c
Updated to revision 4.
```

- SVN zal indien nodig proberen om jouw versie te mergen met de versie van de repository.

Merge Conflicten

- Automatisch mergen is niet altijd mogelijk, bijvoorbeeld:
  1. Alice en Bob doen beiden een checkout.
  2. Alice en Bob wijzigen dezelfde regel van `hello.c`.

Merge Conflicten: versies van de working copies

Versie van Alice:

```
1  int main() {
2    printf("Hello world by Alice!\n");
3    return 0;
4  }
```

Versie van Bob:

```
1  int main() {
2    printf("Hello world by Bob!\n");
3    return 0;
4  }
```

Merge Conflicten

- Automatisch mergen is niet altijd mogelijk, bijvoorbeeld:
  1. Alice en Bob doen beiden een checkout.
  2. Alice en Bob wijzigen dezelfde regel van `hello.c`.
  3. Alice commit haar wijzigingen.

Merge Conflicten

- Automatisch mergen is niet altijd mogelijk, bijvoorbeeld:
  1. Alice en Bob doen beiden een checkout.
  2. Alice en Bob wijzigen dezelfde regel van `hello.c`.
  3. Alice commit haar wijzigingen.
  4. Bob probeert ook te committen, maar dan:

```
$ svn commit
Sending        trunk/hello.c
Transmitting file data .
svn: Commit failed (details follow):
svn: File '/trunk/hello.c' is out of date
svn: Your commit message was left in a temporary file:
svn:    'svn-commit.tmp'
```

- Bob's `hello.c` is "out of date", dus Bob moet een update doen.

Merge Conflicten: een conflicterende update

Bob doet een update . . . :

```
$ svn update
svn update
Conflict discovered in 'hello.c'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (h) help for more options: p
```

- SVN weet niet (en kan niet weten) wat er nu met Bob's wijziging moet gebeuren. Daarom krijgt `hello.c` de conflict status.
- Bob moet nu eerst het conflict in `hello.c` oplossen voordat hij kan committen.

Merge Conflicten: conflict markers

```
1  int main() {
2  <<<<<<< .mine
3    printf("Hello world by Bob!\n");
4  =======
5    printf("Hello world by Alice!\n");
6  >>>>>>> .r6
7    return 0;
8  }
```

- In dit geval wil Bob handmatig de juiste oplossing van het conflict invoeren.
- Hiervoor dienen de overbodige code en alle conflict markers verwijderd te worden.
- Controleer altijd of er niet nog meer conflicten in de file staan!

Merge Conflicten: resolve

```
1  int main() {
2    printf("Hello world by Alice and Bob!\n");
3    return 0;
4  }
```

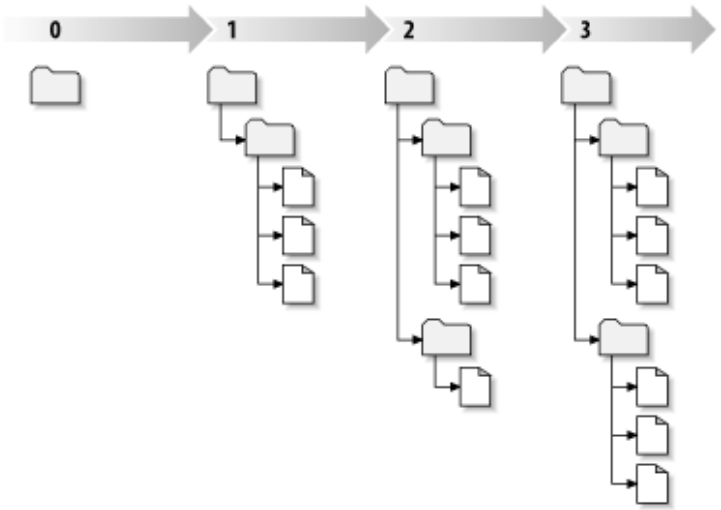
- Nu kan Bob met `svn resolve` het conflict oplossen:

```
$ svn resolve --accept working hello.c
Resolved conflicted state of 'hello.c'
$ svn commit
Committed revision 7.
```

- Als alternatief voor `working` kun je ook `base`, `mine-full` of `theirs-full` gebruiken.

Revision Numbers

- Een revision number verwijst naar een volledige tree. (dus niet alleen naar de files die toen gewijzigd zijn)
- "Revision 7 van hello.c" betekent
  - ✘ *niet* de 7e versie van hello.c
  - ✔ *wel* hello.c zoals die in revision 7 opgeslagen is.



log: Commit geschiedenis

Met `svn log` is de geschiedenis te bekijken:

```
$ svn log
r2 | sven | 2012-01-12 15:47:44 | 1 line
Add hello.c
------------------------------------------------------------------------
r1 | sven | 2012-01-12 15:45:20 | 1 line
Create initial repository structure
```

log: Commit geschiedenis

Met `svn log` is de geschiedenis te bekijken:

```
$ svn log
r2 | sven | 2012-01-12 15:47:44 | 1 line
Add hello.c
------------------------------------------------------------------------
r1 | sven | 2012-01-12 15:45:20 | 1 line
Create initial repository structure

$ svn log Type.h
r42936 | fjahanian | 2007-10-13 00:10:42 | 2 lines
Check and diagnose that objective-c objects may not  ....
------------------------------------------------------------------------
r42858 | lattner | 2007-10-11 05:36:41 | 1 line
slightly simplify interface
------------------------------------------------------------------------
r42856 | fjahanian | 2007-10-11 02:55:41 | 1 line
Patch to create protocol conforming class types.
```

update: Oude versies ophalen

Kies een versie met de `-r` optie van `svn update`.

```
$ svn update -r 3 hello.c
U    hello.c
Updated to revision 3.
```

update: Oude versies ophalen

Kies een versie met de `-r` optie van `svn update`.

```
$ svn update -r 3 hello.c
U    hello.c
Updated to revision 3.

$ svn update -r 1 hello.c
D    hello.c
Updated to revision 1.
```

update: Oude versies ophalen

Kies een versie met de `-r` optie van `svn update`.

```
$ svn update -r 3 hello.c
U    hello.c
Updated to revision 3.

$ svn update -r 1 hello.c
D    hello.c
Updated to revision 1.

$ svn update -r "{‘date -Is -d "1 day ago"‘}" hello.c
A    hello.c
Updated to revision 3.
```

update: Oude versies ophalen

Kies een versie met de `-r` optie van `svn update`.

```
$ svn update -r 3 hello.c
U    hello.c
Updated to revision 3.

$ svn update -r 1 hello.c
D    hello.c
Updated to revision 1.

$ svn update -r "{‘date -Is -d "1 day ago"‘}" hello.c
A    hello.c
Updated to revision 3.

$ svn update
U    hello.c
Updated to revision 7.
```

cat: Oude versies bekijken

Kies een versie met de `-r` optie of met `@`-notatie:

```
$ svn cat -r 2 hello.c
// Hier komt hello world
```

cat: Oude versies bekijken

Kies een versie met de `-r` optie of met `@`-notatie:

```
$ svn cat -r 2 hello.c
// Hier komt hello world

$ svn cat hello.c@3
#include <stdio.h>

int main() {
  printf("Hello world!\n");

  return 0;
}
```

status: Status van files

```
$ svn status -u
?                       untracked.c
A                  0    added.c
M                  7    README
?                       hello
       *           7    hello.c
Status against revision:      8
```

- `untracked.c` en `hello` zijn niet in beheer bij SVN (**?**).
- `added.c` is **A**dded, maar nog niet gecommit.
- `README` is **M**odified, maar nog niet gecommit.
- Van `hello.c` is een nieuwe versie beschikbaar (*****).

Andere commando's

- `svn diff`: geef verschillen tussen twee versies.
- `svn revert`: gooi lokale wijzigingen weg.
- `svn export`: exporteer een "schone" versie van de repo (zonder alle `.svn/` directories).
- `svn delete`: verwijder files uit de repo.
- en meer ...

blame: Wie, wat, wanneer?

`svn blame` laat zien in welke revision en door wie iedere regel gecommit is.

```
$ svn blame hello.c
     3 hendrikjan #include <stdio.h>
     3 hendrikjan
     3 hendrikjan int main() {
     7        sven   printf("Hello world!\n");
     8        sven   printf("Bye!\n");
     3 hendrikjan
     3 hendrikjan   return 0;
     3 hendrikjan }
```

Tags

De huidige trunk taggen als `v1.0`:

```
$ svn copy https://svn.liacs.nl/ccss/2013/teamX/trunk \
           https://svn.liacs.nl/ccss/2013/teamX/tags/v1.0 \
           -m "Tag version 1.0"
Committed revision 10.
```

Na een `svn update` is deze versie te vinden in `.../teamX/tags/`.
De getaggede versie is ook opvraagbaar met:

```
$ svn checkout \
  https://svn.liacs.nl/ccss/2013/teamX/tags/v1.0
A    v1.0/hello.c
Checked out revision 10.
```

Meer informatie

- Gebruik
  `svn help`
  voor een lijst van beschikbare SVN commando's.
- Gebruik
  `svn help commando`
  voor uitleg over een commando, bijvoorbeeld:
  `svn help commit`.
- Voor uitgebreide uitleg:
  `http://svnbook.red-bean.com/`

Inhoud

Commit: Do's en Don'ts

- Maak kleine logische commits; spaar niet alles op tot eind van de dag.
- Doe eerst een update voordat je commit, los eventuele merge conflicten op en test het resultaat.
- Geef duidelijke commit messages:
  - ✘ "bugfix"
  - ✘ " "
  - ✔ "Avoid division by zero in score calculation"
- Commit geen executables, object of class files.
- Commit geen files met nog niet opgeloste conflicts.
- Commit geen onnodige whitespace/code style wijzigingen.
- Zorg dat de repo altijd compileerbaar is en correct werkt.

Valkuilen

- Vergeet niet om nieuwe files toe te voegen voordat je commit.
  Tip: doe altijd `svn status` voordat je commit.
- Wees voorzichtig met het gebruik van `add *`. Bijna altijd voegt dit meer files toe dan je zou willen.
- Controleer altijd of een file echt conflictvrij is voordat je een conflict markeert als "opgelost" met `svn resolve`.
  Tip: zoek even op de string <<<< .
- Een file verwijderen uit de repo is niet wat het lijkt.
  Tip: commit geen wachtwoorden/pincodes/....

“Ik heb een eenmanszaakje . . .”

“. . . dus voor mij is dat hele version control nutteloos.”

“Ik heb een eenmanszaakje . . .”

“. . . dus voor mij is dat hele version control nutteloos.”
– Niet dus!

Version control is al nuttig vanaf meer dan 0 personen:

- Eenvoudig en veilig experimenteren op de code.
- Repository is een (remote) backup.
- Volledige geschiedenis beschikbaar:
  - Whatever you delete today, you desperately need tomorrow.
  - Welke code hoort bij de versie die de klant gebruikt?
  - Hoeveel tijd heb ik gestoken in feature X?

Een eigen repository opzetten

Do it yourself:

- `svnadmin create /path/to/my_repo`
  `svnserve -d -r /path/to/my_repo`
- `git init`

Hosted:

- `http://github.com/`
- `http://repo.or.cz/`
- `http://code.google.com/projecthosting/`
- `http://sourceforge.net/`

Voor de hackers . . .

- `svn propset svn:ignore "*.class" .`:
  Alle `.class` files in `.` negeren.

- `git svn`:
  "Bidirectional operation between a Subversion repository and git."
  - `git svn clone` <*svn URL*>
  - `git svn fetch`
  - `git svn rebase`
  - `git svn dcommit`
  - `...`

Inhoud

Repository voor dit vak

Ieder team heeft een eigen repository, te vinden op:
`https://svn.liacs.nl/ccss/2013/teamX`

Houd de volgende repository-structuur aan:

```
teamX/
+-- branches/
+-- tags/
+-- trunk/
    +-- opgave1/
    |   +-- Pacman.java
    |   +-- Doom.java
    |   +-- README
    +-- opgave2/
    +-- verslag/
```

Opdracht

1. Zet de java-opgave in de SVN repository van je team.
2. Laat ieder teamlid tenminste één commit maken onder zijn/haar eigen SVN account.
3. Forceer een merge conflict, door twee teamleden dezelfde regel te laten wijzigen, en los dit conflict vervolgens weer op. Zie ▸ Conflicten.
4. Lever de opgave in door de gewenste versie de tag "OPGAVE1" te geven. Zie ▸ Tags.
   (test door zelf een nieuwe checkout te doen van tags/OPGAVE1)