MGL7460 — Automne 2016

Gestion de la configuration et contrôle du code source

Guy Tremblay
Professeur

Département d'informatique
UQAM
http://www.labunix.uqam.ca/~tremblay

8 septembre 2016

Parmi les premières choses à faire, quand on développe du code de façon professionnelle…

« *You need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

- *?*
- *?*
- *?*

? *needs to come before anything else. It's the first bit of infrastructure we set up on any project.* »

«Practices of an Agile Developer—Working in the Real World», Subramaniam & Hunt, 2006.

Parmi les premières choses à faire, quand on développe du code de façon professionnelle…

« *You need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

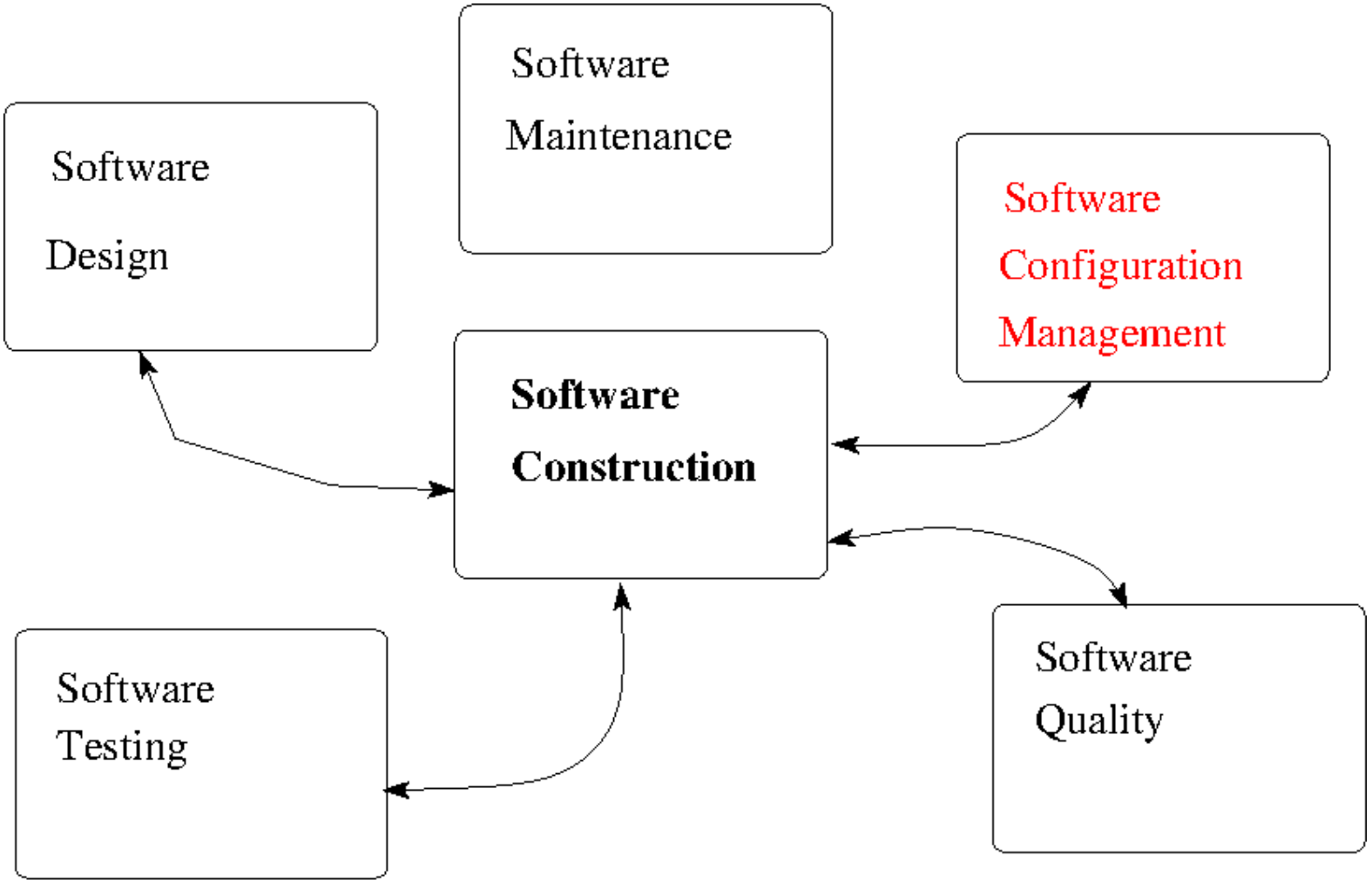
- *Version control*
- *Unit Testing*
- *Build automation*

Version control needs to come before anything else. It's the first bit of infrastructure we set up on any project. »

«Practices of an Agile Developer—Working in the Real World», Subramaniam & Hunt, 2006.

Gestion de la configuration

Les KAs reliés à *Software Construction*



Gestion de la configuration logicielle selon SWEBOK

Configuration

The configuration of a system is the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product ;

it can also be thought of as a collection of specific **versions** *of hardware, firmware, or software items combined according to specific build procedures to serve a particular purpose.*

Gestion de la configuration logicielle selon SWEBOK

Configuration Management

Configuration management (CM) [. . .] is the discipline of identifying the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration and maintaining the integrity and traceability of the configuration throughout the system life cycle.

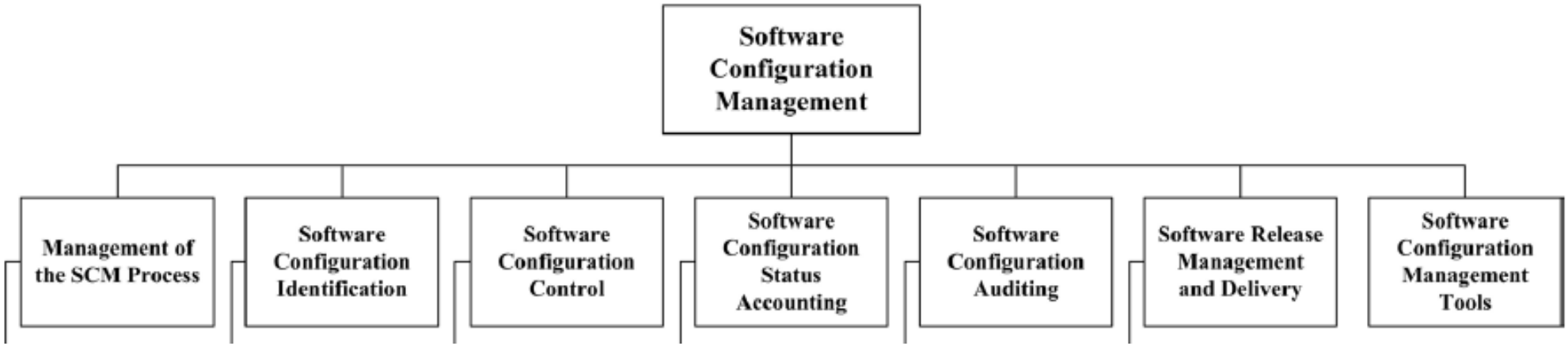
Gestion de la configuration logicielle selon ISO

Configuration Management

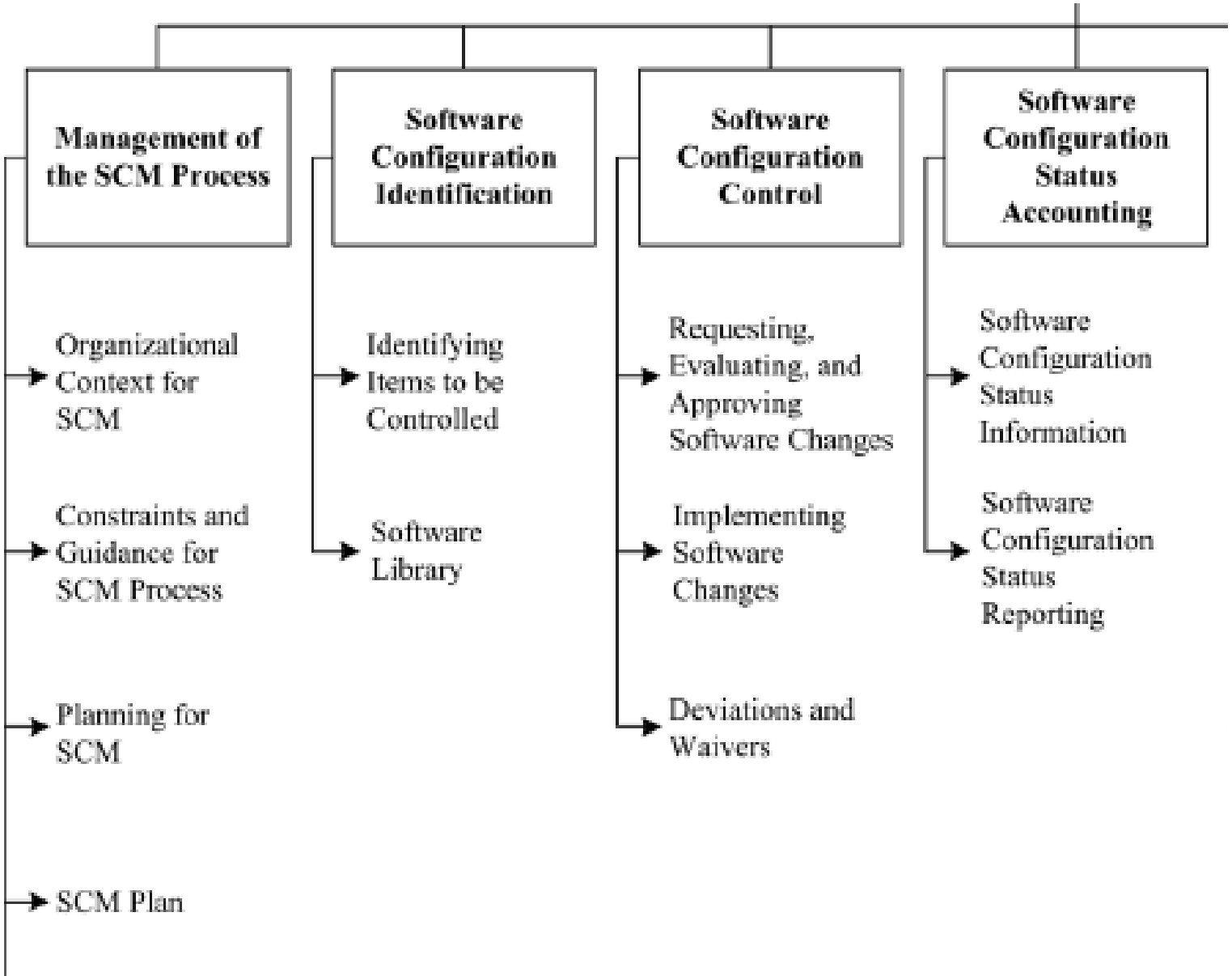
A discipline applying technical and administrative direction and surveillance to : identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

Source : ISO/IEC/IEEE 24765 :2010 Systems and Software Engineering-Vocabulary, ISO/ IEC/IEEE, 2010.

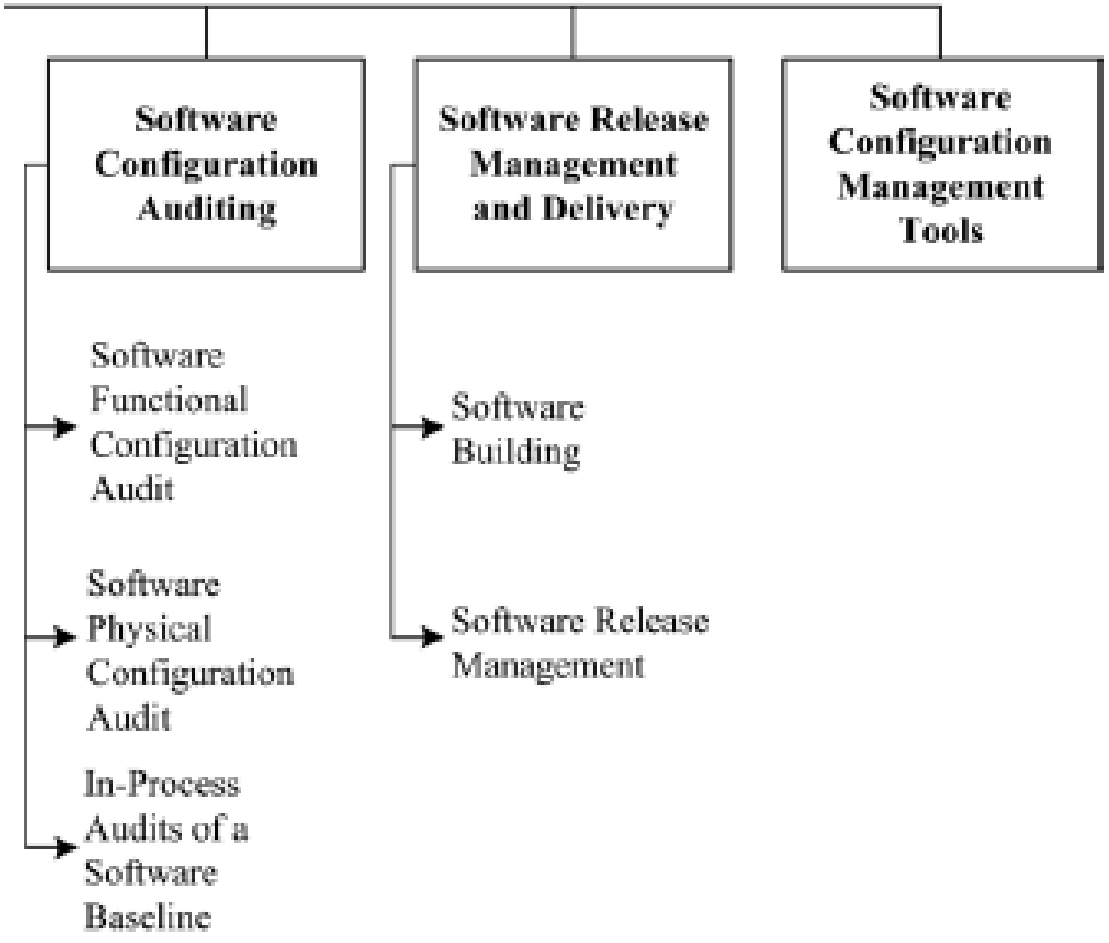
Aperçu du *Software Configuration Management Knowledge Area* du Guide to the SWEBOK



Aperçu du *Software Configuration Management Knowledge Area* du Guide to the SWEBOK



Aperçu du *Software Configuration Management Knowledge Area* du Guide to the SWEBOK



Software Configuration Identification

Software Configuration Identification

Software configuration identification identifies items to be controlled, establishes identification schemes for the items and their versions, and establishes the tools and techniques to be used in acquiring and managing controlled items. These activities provide the basis for the other SCM activities.

Semantic Versioning : Un système maintenant courant de numérotation des versions

Question : Que nous indiquent les numéros de versions suivants quant à l'évolution de la bibliothèque `foo` :

- `foo-1.0.0`
- `foo-1.0.1`
- `foo-1.0.2`
- `foo-1.1.0`
- `foo-1.2.0`
- `foo-2.0.0-pre`
- `foo-2.0.0`

Semantic Versioning 2.0.0

Given a version number MAJOR.MINOR.PATCH, increment the :

1. *MAJOR version when you make incompatible API changes,*
2. *MINOR version when you add functionality in a backwards-compatible manner, and*
3. *PATCH version when you make backwards-compatible bug fixes.*

Source*:* `http://semver.org/`

Semantic Versioning 2.0.0 (suite)

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

- *A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately [. . .]*
  Examples : 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.
- *Build metadata MAY be denoted by appending a plus sign and a series of dot separated identifiers [. . .]*
  Examples : 1.0.0-alpha+001, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85.

Source: `http://semver.org/`

Software Configuration Management Tools

Les principaux outils associés à SCM selon SWEBOK

Version control tools

track, document, and store individual configuration items such as source code and external documentation

Build handling tools

compile and link an executable version of the software. More advanced building tools extract the latest version from the version control software, perform quality checks, run regression tests, and produce various forms of reports, among other tasks

Change control tools

support the control of change requests and events notification

Les principaux outils associés à SCM selon SWEBOK

Version control tools

track, document, and store individual configuration items such as source code and external documentation

Build handling tools

compile and link an executable version of the software. More advanced building tools extract the latest version from the version control software, perform quality checks, run regression tests, and produce various forms of reports, among other tasks

Change control tools

support the control of change requests and events notification

Cours de ce soir : Contrôle du code source !

Un prochain cours : Assemblage de logiciels (*build*)

Contrôle du code source

Qu’est-ce qu’un système de contrôle du code source ?

?

Qu’est-ce qu’un système de contrôle du code source ?

a source code control system [is like] a giant UNDO key—a project-wide time machine that can return you to those alcyon days of last week, when the code actually compiled and ran.

«The Pragmatic Programmer», Hunt & Thomas, 2000.

Qu'est-ce qu'un système de contrôle du code source ?

a source code control system [is like] a giant UNDO key—a project-wide time machine that can return you to those alcyon days of last week, when the code actually compiled and ran.

«The Pragmatic Programmer», Hunt & Thomas, 2000.

Alcyon

1. *Calm and peaceful ; tranquil.*
2. *Prosperous ; golden : halcyon years.*

YOU KNOW YOU'RE IN A SOFTWARE PROJECT

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

- Identifier quels fichiers ont été modifiés
- Déterminer qui a modifié un bout de code
- Comparer des versions
- Fusionner des versions développées de façon concurrente

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

- Identifier quels fichiers ont été modifiés
- Déterminer qui a modifié un bout de code
- Comparer des versions
- Fusionner des versions développées de façon concurrente

- Identifier et gérer les *releases*, les versions, les branches de développement

Les différents SCCS

Qui a utilisé ou utilise. . .

CVS	Git
Perforce	Mercurial
Subversion (SVN)	Bazaar

?

Les différents SCCS

Qu'est-ce qui différencie...

CVS	Git
Perforce	Mercurial
Subversion (SVN)	Bazaar

?

Les différents SCCS

Qu'est-ce qui différencie...

Centralisé	Distribué
CVS	Git
Perforce	Mercurial
Subversion (SVN)	Bazaar

!

SCCS Centralisé vs. SCCS Distribué

Centralisé

Tout l'historique des changements est conservé sur un serveur central, duquel n'importe qui peut obtenir la version la plus récente ou envoyer les changements les plus récents.

Distribué

Chaque usager a une copie locale de tout l'historique des changements. Il n'est donc pas nécessaire d'être connecté à un réseau/serveur pour sauvegarder des changements. En outre, n'importe quel usager peut se synchroniser avec n'importe quel autre.

SCCS Centralisé vs. SCCS Distribué

Centralisé

Centralized VCS systems are designed with the intent that there is One True Source that is Blessed, and therefore Good. All developers work (checkout) from that source, and then add (commit) their changes, which then become similarly Blessed.

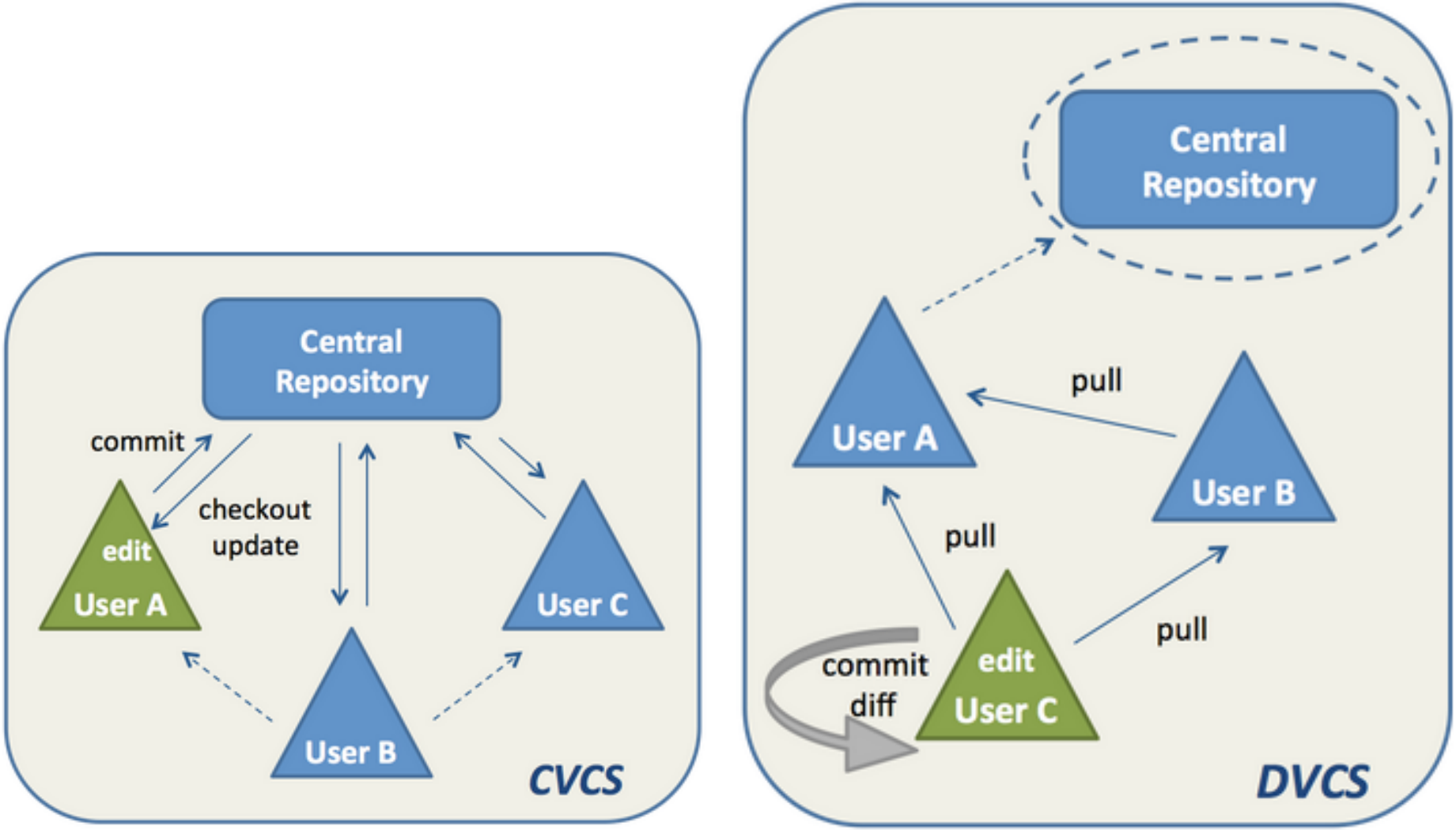
Distribué

Distributed VCS systems are designed with the intent that one repository is as good as any other, and that merges from one repository to another are just another form of communication.

Source: `http://stackoverflow.com/questions/111031/comparison-between-centralized-and-distributed-version-control-systems`

SCCS Centralisé vs. SCCS Distribué

Source : https://www.appfusions.com/display/StashSCMImporter/CVCS+vs.+DVCS+In+a+Nutshell



Avantages/désavantages des SCCS distribués

Avantages

- Chaque développeur a son espace privé — son *sandbox*
- On peut travailler sans être en ligne
- Plusieurs opérations sont très rapides — exécution locale
- La création et fusion de branches est efficace

Désavantages

- Préférable d'avoir un dépôt (central) de sauvegarde
- Pas nécessairement de «version la plus récente»

Source:

`https://www.appfusions.com/display/StashSCMImporter/CVCS+vs.+DVCS+In+a+Nutshell`

Utilisation de `git` dans le cours

Dans le cadre des projets d'équipe. . .

Vous devrez utiliser `git` :

- Pour collaborer avec vos collègues
- Pour remettre votre code source — je ferai un `git clone`

Dans le cadre des projets d'équipe. . .

Vous devrez utiliser `git` :

- Pour collaborer avec vos collègues
- Pour remettre votre code source — je ferai un `git clone`

Vous pourrez utiliser

- GitHub
- BitBucket

Si vous ne connaissez pas `git`...

- Notes de cours sur `git` :
  `http://www.labunix.uqam.ca/~tremblay/MGL7460/Materiel/git.pdf`

- Divers liens sur `git` :
  `http://www.labunix.uqam.ca/~tremblay/MGL7460/Liens/`

Quelques recommandations d’utilisation

«*The Pragmatic Programmer*» : Tip 23

Tip 23

Always Use Source Code Control

Always. Even if you are a single-person team on a one-week project. Even if it's a "throw-away" prototype. Even if the stuff you're working on isn't source code. Make sure that everything is under source code control !

«*Ship It !*» : Tip 3

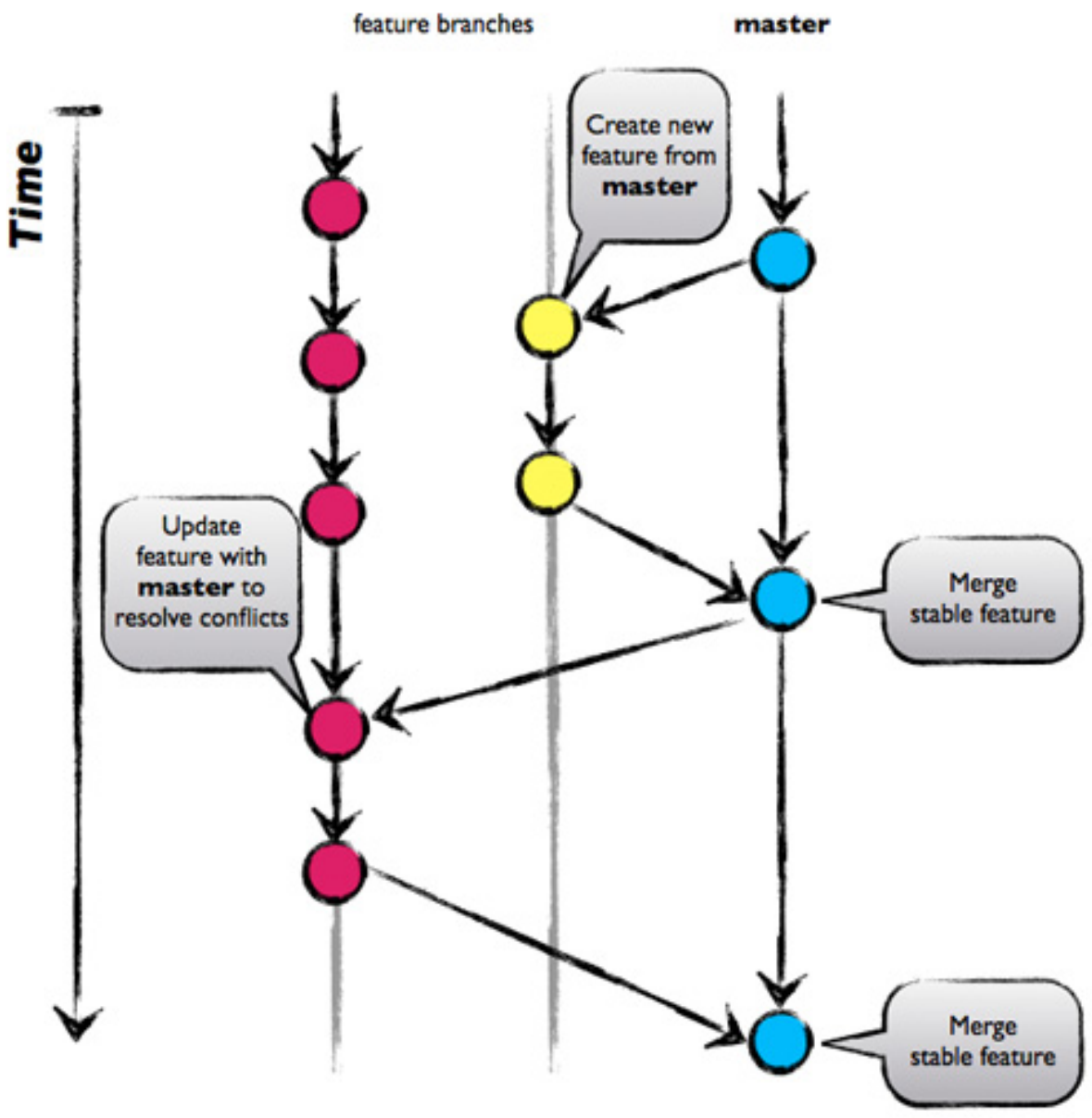
Tip 3

If you need it, check it in

If your development shop goes up in flames, you should be able to recover with just a backup of your repository. You should have everything you need to build the entire project ; if you don't, then perhaps you aren't using the tool properly.
[. . .]
The only exception to the «Keep everything you need to build your product in the SCM» top are files that you can generate.

Le modèle GitHub flow (simplifié) illustré

Source : http://www.nicoespeon.com/fr/2013/08/quel-git-workflow-pour-mon-projet/



Le modèle GitHub flow (simplifié)

Source : `http://www.nicoespeon.com/fr/2013/08/quel-git-workflow-pour-mon-projet/`

1. Tout ce qui est dans `master` peut être déployé en production
2. Créer des branches explicites depuis `master` (*features*)
3. Pousser sur `origin` régulièrement
4. Ouvrir une *pull-request* à tout moment
5. Fusionner seulement après une *pull-request review*
6. Déployer immédiatement après `merge` dans `master`

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even days —between code check-ins,
  you aren't using the system actively.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins,
  you aren't using the system actively.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.

- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.

- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than a day or two , consider changing the way you work.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.

- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.

- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.

- *[H]ow long would it take you to get a new machine up and running for development ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.
- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.
- *[H]ow long would it take you to get a new machine up and running for development ?*
- *Can you perform SCM operations quickly ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.
- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.
- *[H]ow long would it take you to get a new machine up and running for development ?*
- *Can you perform SCM operations quickly ?*
- *Are you backing up the SCM's repository ?*

Références

P. Bourque and R.E. Fairley, editors.

Guide to the Software Engineering Body of Knowledge (Version 3.0).

IEEE Computer Society, 2014.

`http://www.computer.org/web/swebok/v3`.

A. Hunt and D. Thomas.

The Pragmatic Programmer—From Journeyman to Master.

Addison-Wesley, 2000.

J. Richardson and W.A. Gwaltney.

Ship it ! : A Practical Guide to Successful Software Projects.

The Pragmatic Bookshelf, 2005.

V. Subramaniam and A. Hunt.

Practices of an Agile Developer—Working in the Real World.

The Pragmatic Bookshelf, 2006.