Logiciel libre, une introduction

Roberto Di Cosmo

Université Paris Diderot
UFR Informatique
Laboratoire Preuves, Programmes et Systèmes

roberto@dicosmo.org

3 Avril 2014

Part I

Génie Logiciel Libre

Génie Logiciel
- Linux
- Eric S. Raymond
- Apache
- Mythes et réalité

Distributions Linux
- Theoretical results

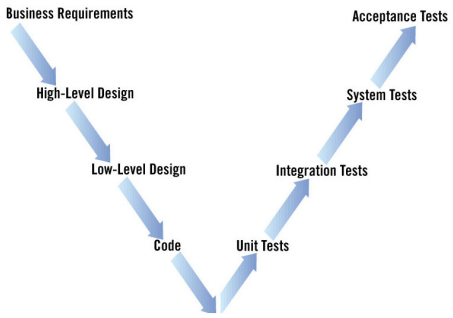
Securité
- Exemples du monde propriétaire
- Vote éléctronique

Génie Logiciel et Logiciel Libre

Des origines à nos jours

Un peu d'histoire du Génie logiciel

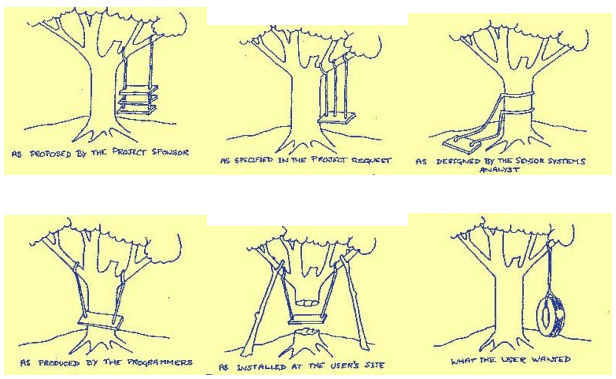
Le "modèle à V" incarne la vision du dévéloppement logiciel typique des années '80



Défauts du modèle

- il faut une organisation centralisée
- les spécifications initiales peuvent être fausses ou incomplètes
- la distance entre spécifications et test final peut être trop longue
- l'utilisateur final est exclu du processus jusqu'à la fin
- quand on découvre un erreur grave, il est souvent trop tard

Le problème en six images



Des solutions proposées ensuite. . .

- programmation orientée objet
- conception et modélisation via UML
- utilisation de méthodes formelles dans le dévéloppement
- diverses méthodologies plus récentes:
  - extreme programming
  - méthodes agiles
  - . . .

Voyons voir concrètement ce qui s'est passé avec certains projets en logiciel libre.

Analyses

Exemples de communautés significatives

- Linux
- Apache

Linux: une radiographie

- quelques dates:
  vers. 0.02 en 10/1991; 0.95 en 03/1992; 1.0 en 03/1994
- modèle du *dictateur bienveillant*:
  Linus Torvalds a le dernier mot sur tout
- élaboration des patchs sur la mailing list (cela permet de discuter les propositions)
- séparation entre fils de versions:
  stable (2.4.x, 2.6.x), et *experimental* (1.3.xx or 2.1.x)
- gestion de version avancée (plus de détails dans un cours à part)

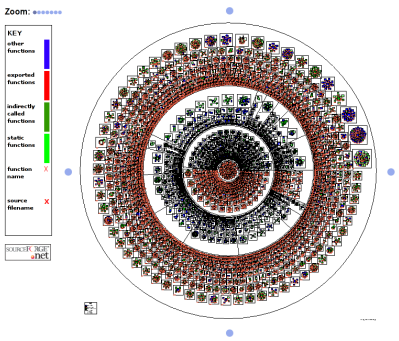
Exemple de patch dans la mailing list

```
Date Fri, 16 Mar 2007 12:36:15 +0100
From Jarek Poplawski <>
Subject [PATCH] Re: Fwd: oprofile lockdep warning on rc1

On 28-02-2007 01:46, Dave Jones wrote:
> This happened on a 2.6.21rc1 kernel.
...
> Richard Hughes <hughsient@gmail.com>
> Date:
> Tue, 27 Feb 2007 21:59:07 +0000
> To:
> Development discussions related to Fedora Core
> <fedora-devel-list@redhat.com>
...
> But also I get this (!!!):
>

Here is my patch proposal for testing.

Regards,
Jarek P.
```

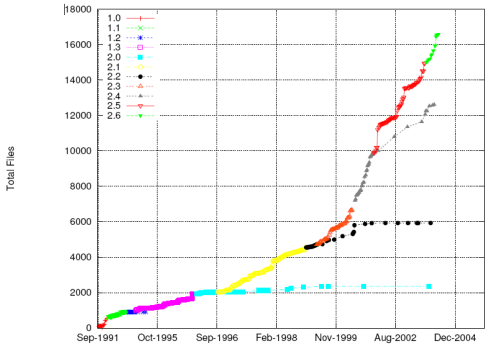

Exemple de patch dans la mailing list

```
lockdep found oprofilefs_lock is taken both in process
context (oprofilefs_ulong_from_user()) and from hardirq
(nmi_cpu_setup()), so the lockup is possible.

Reported-by: Richard Hughes <hughsient@gmail.com>
Signed-off-by: Jarek Poplawski <jarkao2@o2.pl>

---
diff -Nurp linux-2.6.21-rc1-/drivers/oprofile/oprofilefs.c linux-2.6.21-rc1/drivers/oprofile/oprofilefs.c
--- linux-2.6.21-rc1-/drivers/oprofile/oprofilefs.c 2007-02-27 10:47:38.000000000 +0100
+++ linux-2.6.21-rc1/drivers/oprofile/oprofilefs.c 2007-03-16 11:36:30.000000000 +0100
@@ -65,6 +65,7 @@ ssize_t oprofilefs_ulong_to_user(unsigned
 int oprofilefs_ulong_from_user(unsigned long * val, char const __user * buf, size_t count)

  char tmpbuf[TMPBUFSIZE];
+ unsigned long flags;

  if (!count)
  return 0;
@@ -77,9 +78,9 @@ int oprofilefs_ulong_from_user(unsigned
  if (copy_from_user(tmpbuf, buf, count))
  return -EFAULT;

- spin_lock(&oprofilefs_lock);
+ spin_lock_irqsave(&oprofilefs_lock, flags);
  *val = simple_strtoul(tmpbuf, NULL, 0);
- spin_unlock(&oprofilefs_lock);
+ spin_unlock_irqrestore(&oprofilefs_lock, flags);
  return 0;
```


Un aperçu: structuration



Un aperçu: évolution



Évolution du nombre des fichiers dans le noyau Linux (G. Robles)

Analyses

Un texte très connu tire des leçons du succès de Linux :

The cathedral and the Bazaar, Eric S. Raymond, 1997

cathedral : processus logiciel lourd et hierarchique

bazaar : dévéloppement par des équipes avec faible couplage

Règles de bonne conduite énoncées

1. Every good work of software starts by scratching a developer's personal itch.
2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
3. "Plan to throw one away; you will, anyhow." (Fred Brooks, The Mythical Man-Month, Chapter 11)
4. If you have the right attitude, interesting problems will find you.
5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
6. Treat your users as co-developers.
7. Release early. Release often.
8. Given enough eyeballs, all bugs are shallow.

Critiques

- analyse très orientée Linux
- separation entre Bazaar et Cathedral moins nette dans la réalité
- beaucoup d'approximations, par exemple, la dénégation de la loi de Brooks, qui oublie la différence entre core developers et utilis-acteurs à la marge du projet

Parenthèse: Free Software Lessons/Lean Software principles

Value

Every good work of software starts by scratching a developer's personal itch.

Customer first

Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.

Removing waste

Good programmers know what to write. Great ones know what to rewrite (and reuse).

Perfection, well... fix it fast

Release early, release often.
Given enough eyeballs, all bugs are shallow.

Lessons from E. S. Raymond's "The Cathedral and the Bazaar", 1997

Apache

- quelques dates:
  Création du projet en 1995 par Rob McCool, première version stable en 1996, version 1.3 en 1998, IBM choisit Apache pour WebSphere en juin 1998
- approx. 1500 committers sur les projets (une centaine sur httpd)
- organisation *oligarchique*

Committers get a shot at working on the code; good committers become members and thus get a piece of the ownership of the software and the direction. Commit access is a privilege, not a right, and is based on trust. The Apache Software Foundation is a meritocracy [. . .]
New candidates for membership are nominated by an existing member and then put to vote; a majority of the existing membership must approve a candidate in order to the candidate to be accepted.

Quelques mythes sur le Logiciel Libre

- dévéloppement anarchique sans leader
- connaissance qui emerge naturellement de la sagesse des masses

  The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations
  2004 James Surowiecki

- le logiciel libre est *toujours* de meilleure qualité que le logiciel propriétaire
- l'écosystème du logiciel libre est basé sur des valeurs d'altruisme communautaire...

Reverse Software Engineeering

On peut essayer d'analyser ces cas *systématiquement*:

- disponibilité des sources
- accès aux données historiques
- informations complètes sur les dévéloppeurs
- pas besoin d'accords formels

Un travail d'envergure a été entrepris par `http://libresoft.urjc.es/index`

Un point de vue moderne

Les analyses de Martin Michlmayr (ex Debian project leader) sont disponibles sur `http://opensource.mit.edu`

Cathedral phase	Transition phase	Bazaar phase
Original "idea" Project Author Core developers Unix philosophy	"Interest" ⇒ Prototype *Modular design*	⇒ Distributed development environment Community Parallel perfective and corrective maintenance Peer reviews

Exemples d'architectures

Focus sur la modularité

Un livre récent (2011) édité par Amy Brown et Greg Wilson, permet de se faire une idée de l'importance de la *modularité*.

The Architecture of Open Source Applications recueille une description de l'architecture de 25 projets Logiciel Libre, faite par quelques uns de leurs auteurs.

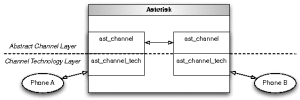
Voyons quelques exemples.

Asterisk (Russel Bryant)

Asterisk is...

an open source telephony applications platform distributed under the GPLv2.

Architecture basée sur l'abstraction des Channels et Channel Drivers:



Grâce à elle, un contributeur peut:

- fournir un nouveau pilote pour un nouveau type de téléphone sans se soucier du rester
- fournir une nouvelle fonctionnalite dans Asterisk sans se soucier des pilotes

Architecture extremement modulaire, qui a du succés depuis plus de 10 ans.

Eclipse (Kim Moir)

Eclipse is...

an open source platform for the creation of interoperable tools for application developers.

Eclipse's achievement

"Implementing software modularity is a notoriously difficult task. Interoperability with a large code base written by a diverse community is also difficult to manage. At Eclipse, we have managed to succeed on both counts."

Eclipse Architecture

Les composants essentiels sont:

three major elements, which corresponded to three major sub-projects: the Platform, the JDT (Java Development Tools) and the PDE (Plug-in Development Environment).

Observation

In order to encourage people to build upon the Eclipse platform, there needs to be a mechanism to make a contribution to the platform, and for the platform to accept this contribution. This is achieved through the use of extensions and extension points, another element of the Eclipse component model.

Eclipse Architecture

Modularité et extensibilité: les clefs du succès

This extensible architecture was one of the keys to the successful growth of the Eclipse ecosystem. Companies or individuals could develop new plugins, and either release them as open source or sell them commercially.
One of the most important concepts about Eclipse is that everything is a plugin. Whether the plugin is included in the Eclipse platform, or you write it yourself, plugins are all first class components of the assembled application.

Aujourd'hui: plus de 1000 composants sur le Eclipse Marketplace, et des milliers d'autres ailleurs!

Le cas des Distributions GNU/Linux

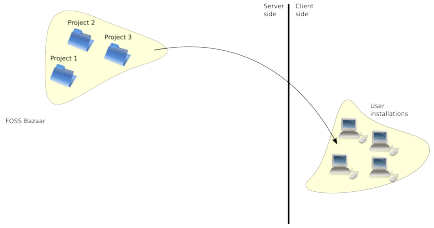
Systèmes logiciels complexes

On a vu comment l'architecture modulaire est à la base de bien de logiciels libres qui ont du succès.

Nous nous intéressons maintenant à comment on peut mettre en place une architecture modulaire pour des systèmes logiciels de grande evergure, comme les Distributions GNU/Linux.

Le cas des *distributions* GNU/Linux

Avant l'arrivée des *distributions*, le seul moyen d'installer du logiciels sur les postes clients était:

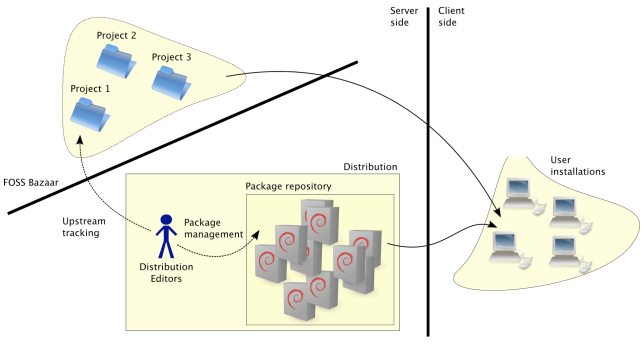


Mais:

- pas de *version standard* du poste client
- besoin de recompiler, . . . et reconfigure assez souvent
- trop compliqué!

Le cas des *distributions* Linux

Cela a fait naître les *distributions* comme intermédiaires entre les projets et les utilisateurs



Le role d'un éditeur de distribution:

upstream tracking : suivre l'évolution des sources
the developer is almost never the packager!

integration : offrir une collection coherente[1] de paquets
Pour cela, on doit traiter correctement des *dependances*

testing :

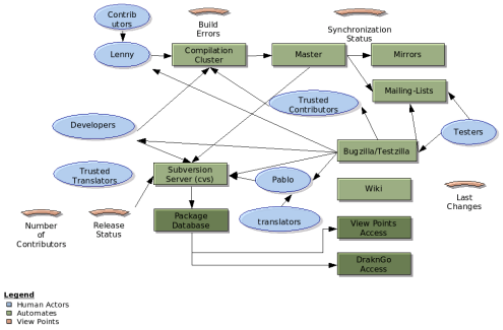
distribution : diffusion rapide sans casser l'éxistant

Ce n'est pas facile!:

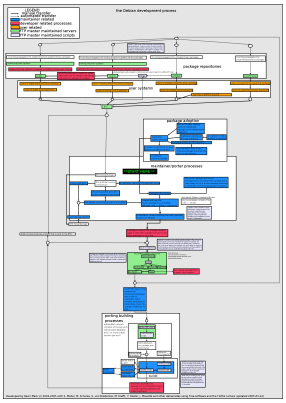
Le cycle de 6 mois de Mandriva nécéssite 30 années-homme.

[1] more formally?

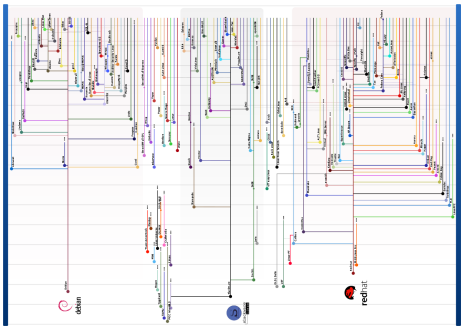
An overview of Mandriva's lifecycle



An overview of Debians's lifecycle

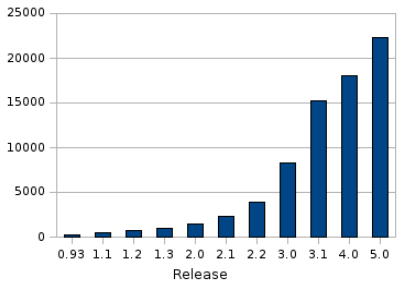


Distributions: a "somehow" successful idea . . .



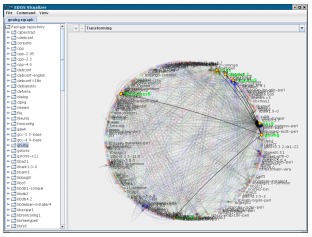
Notion centrale: pour abstraire sur la complexité de l'infrastructure existante, on utilise le package, avec des outils appropriés . . .

Un croissance superlinéaire



Nombre de paquets dans Debian

Des interdépendances complexes. . .



```
Package: gnubg
Version: 0.14.3+20060923-4
Depends: gnubg-data,
ttf-bitstream-vera, libartsc0
(>= 1.5.0-1), ..., libgl1-mesa-glx
| libgl1, ...
Conflicts: ...
```

Cela change *tous les jours*! Comment s'y retrouver?

Ces problèmes sont au coeur des projets EDOS, et MANCOOSI.

Packages, metadata, installation

Package = { some files, some scripts, metadata }

- Identification
- Inter-package rel.
  - Dependencies
  - Conflicts
- Feature declarations
- Other
  - Package maintainer
  - Textual descriptions

Example

```
Package: aterm
Version: 0.4.2-11
Section: x11
Installed-Size: 280
Maintainer: Göran Weinholt ...
Architecture: i386
Depends: libc6 (>= 2.3.2.ds1-4),
  libice6 | xlibs (> 4.1.0), ...
Conflicts: suidmanager (< 0.50)
Provides: x-terminal-emulator
...
```

- a package is the *elemental component* of modern distribution systems (not GNU/Linux specific)
- a working *system* is deployed by installing a package set (≈ 1000/2000 for GNU/Linux distro)

.

Installation process

Phase	Trace
User request	`# apt-get install aterm`

Constraint resolution	`Reading package lists... Done`
	`Building dependency tree... Done`
	`The following extra packages will be installed:`
	`  libafterimage0`
	`The following NEW packages will be installed`
	`  aterm libafterimage0`
	`0 upgraded, 2 newly installed, 0 to remove and 1786 not upgraded.`
	`Need to get 386kB of archives.`
	`After unpacking 807kB of additional disk space will be used.`
	`Do you want to continue [Y/n]? Y`

Package retrieval	`Get: 1 http://debian.ens-cachan.fr testing/main libafterimage0 2.2.8-2 [301kB]`
	`Get: 2 http://debian.ens-cachan.fr testing/main aterm 1.0.1-4 [84.4kB]`
	`Fetched 386kB in 0s (410kB/s)`

Pre-configuration	`{`

Unpacking	`Selecting previously deselected package libafterimage0.`
	`(Reading database ... 294774 files and directories currently installed.)`
	`Unpacking libafterimage0 (from .../libafterimage0_2.2.8-2_i386.deb) ...`
	`Selecting previously deselected package aterm.`
	`Unpacking aterm (from .../aterm_1.0.1-4_i386.deb) ...`

Configuration	`Setting up libafterimage0 (2.2.8-2) ...` `Setting up aterm (1.0.1-4) ...`

- *each* phase can fail (it actually happens quite often . . .)
- efforts should be made to identify errors as early as possible

Installability of a single package

In state of the art distributions:

- knowing if the installation process terminates without errors is undecidable (reason: *configuration scripts*)
- the largest number of issues affecting the users is related to the *dependency resolution* phase:
  - no solution is found
  - the solution found is puzzling

See
`http://www.dicosmo.org/MyOpinions/index.php/Mancoosi`
for some real life examples

Installation woes: no solution

```
apt-get -s install  baobab

Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
  gnome-settings-daemon: Breaks: gnome-screensaver (< 2.28.0)
                                 but 2.26.1-1 is to be installed
E: Broken packages
```


Installation woes: debatable solution

```
# sudo apt-get install  debhelper
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
 armagetron armagetron-common autoconf bonobo-activation codebreaker debconf
 debconf-i18n debconf-utils dialog esound-common fb-music-high fontconfig
 frozen-bubble-data gnupmail gv intltool-debian libaiksaurus-data
 libaiksaurus0c102 libatk1.0-0 libatk1.0-dev libbonobo-activation4 libbonobo2-0
 libbonobo2-common libdb3 libdbd-mysql-perl libdbi-perl libeel2-data libesd0
 ...
The following packages will be REMOVED:
  autoconf2.13 frozen-bubble frozen-bubble-lib gconf2 gnomemeeting itk3.1-dev
  libbonoboui2-0 libbonoboui2-common libdigest-md5-perl libforms0.89 libgconf2-4
  libgnome2-0 libgnome2-common libgnomeui-0 libgnomevfs2-0 libgnomevfs2-common
  libgtk1.2-dev libgtk2.0-0png3 libgtk2.0-dev libmime-base64-perl
  libpango1.0-dev libsdl-mixer1.2-dev libsdl-perl libsdl-ttf1.2-dev
  libsdl1.2-dev libsmpeg-dev libstorable-perl nautilus tk8.3-dev tktable-dev
  x-window-system x-window-system-core xaw3dg-dev xlib6g xlib6g-dev xlibmesa-dev
  xlibmesa3 xlibosmesa3 xlibs-dev xlibs-pic xpdf xpdf-reader
The following NEW packages will be installed:
  armagetron-common debconf-i18n fb-music-high fontconfig intltool-debian
  libaiksaurus-data libaiksaurus0c102 libeel2-data libfilehandle-unget-perl
  libfontconfig1 libforms1 libgdbm3 libgnutls7 libgsf-1 libice-dev libice6
  libid10 liblzo1 libmagick5.5.7 libmail-mbox-messageparser-perl
  libmysqlclient12 libncursesw5 libnet-daemon-perl libnewt0.51 libpaper1
  libplrpc-perl libsdl-console ...

75 packages upgraded, 80 newly installed, 42 to remove and 858  not upgraded.
Need to get 67.1MB of archives. After unpacking 26.9MB will be used.
Do you want to continue? [Y/n] Abort.
```


Formalising packages and repositories

Definition (Repository)

A *repository* (P, C, D) is a triple consisting of a set of packages P, an irreflexive and symmetrical *conflict relation* C ($C \subseteq P \times P$), and a *dependency function* $D : P \longrightarrow \wp(\wp(P))$.

Definition (Peaceful, abundant, and healthy subsets of a repository)

Given a *repository* $R = (P, D, C)$, a set of packages $I \subseteq P$ is *peaceful* if $I \times I \cap C = \emptyset$, and *abundant* if for all $p \in I$, for all $s \in D(p)$, $s \cap I \neq \emptyset$. If I is peaceful and abundant, then it is *healthy*.

Definition ((Co)Installability)

A set of packages S is (co)installable in a repository $R = (P, D, C)$ iff there exists a *healthy* $I \subseteq P$ with $S \subseteq I$.

How hard are the problems related to package installation?

Theorem

The following problems are NP-complete:

- *installability of a single package*
- *coinstallability of a set of packages*

Proof: bi-directional mapping between dependency resolution and boolean satisfiability (Di Cosmo et al, Ase 2006)
N.B.: recent SAT solvers are able to handle current instances (in Debian, single package installation leads to problems with a few thousands literals, and almost Horn formulae)

Package installation as a SAT problem

- All the version constraints in inter-package relationships are expanded to the disjunction of the packages in the repository that satisfy that constraint.
- For every package P version V in the repository a boolean variable P_v is introduced.
- For every dependency relation we introduce a logical implication of the form $P_v \rightarrow R_1 \wedge \cdots \wedge R_n$
- For every conflict relation we introduce a logical implication of the form $P_v \rightarrow \neg R_1 \wedge \cdots \wedge \neg R_2$
- The encoding of the repository is given by the conjunction of all the logical implication introduced by dependencies and conflicts.

Installation and SAT solving

Install `libc6` version `2.3.2.ds1-22` in

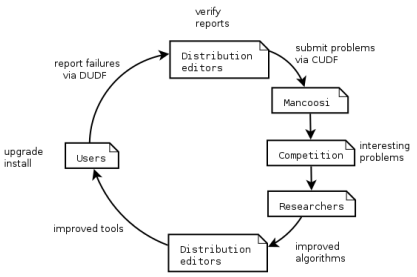
```
Package: libc6
Version: 2.2.5-11.8

Package: libc6
Version: 2.3.5-3

Package: libc6
Version: 2.3.2.ds1-22
Depends: libdb1-compat

Package: libdb1-compat
Version: 2.1.3-8
Depends: libc6 (>= 2.3.5-1)

Package: libdb1-compat
Version: 2.1.3-7
Depends: libc6 (>= 2.2.5-13)
```

becomes

$$
\begin{aligned}
& I^{2.3.2.ds1-22}_{libc6} \\
& \wedge \\
& \neg(I^{2.3.2.ds1-22}_{libc6} \wedge I^{2.2.5-11.8}_{libc6}) \\
& \wedge \\
& \neg(I^{2.3.2.ds1-22}_{libc6} \wedge I^{2.3.5-3}_{libc6}) \\
& \wedge \\
& \neg(I^{2.3.5-3}_{libc6} \wedge I^{2.2.5-11.8}_{libc6}) \\
& \wedge \\
& \neg(I^{2.1.3-7}_{libdb1-compat} \wedge I^{2.1.3-8}_{libdb1-compat}) \\
& \wedge \\
& I^{2.3.2.ds1-22}_{libc6} \rightarrow \\
& (I^{2.1.3-7}_{libdb1-compat} \vee I^{2.1.3-8}_{libdb1-compat}) \\
& \wedge \\
& I^{2.1.3-7}_{libdb1-compat} \rightarrow \\
& (I^{2.3.2.ds1-22}_{libc6} \vee I^{2.3.5-3}_{libc6}) \\
& \wedge \\
& I^{2.1.3-8}_{libdb1-compat} \rightarrow I^{2.3.5-3}_{libc6}
\end{aligned}
$$

Not that easy: pre-depends, optimizations, error explanation, ...

Mancoosi: algorithmes et optimisations



La sécurité

Sécurité et Logiciel libre

- les modèles de sécurité
- l'exemple du vote éléctronique

Modèles de sécurité

Fermé On construit un système dont les spécifications sont secrètes.
La sécurité repose en partie sur le secret des spécifications
(Ex: Enigma).

Avantages:

- On ajoute une difficulté supplémentaire pour les casseurs

Desavantages:

- La difficulté supplémentaire donne une fausse assurance
- Modèle traditionnellement sensible à la traison
- Difficile, voir impossible à mettre en oeuvre avec des composantes disponibles sur le marché

Modèles de sécurité

Difficile, voir impossible à mettre en oeuvre avec des composantes disponibles sur le marché,. . .

pourquoi?

- on ne maîtrise pas le code source propriétaire
- on ne sait pas faire d'audit de sécurité complet automatique sur 50M lignes de code
- chacune de ces lignes de code est suspecte
- même les outils de compilation[2] sont suspects!

[2] Ken Thompson: Reflections on trusting trust

Reflections on trusting trust

Ken Thompson, Turing Award, 1983.
Voir l'explication au tableau.

> *The moral is obvious. You can't trust code that you did not totally create yourself.(Especially code from companies that employ people like me.)No amount of source-level verification or scrutiny will protect you from using untrusted code.In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode.*

Communication of the ACM, Vol. 27, No. 8, August 1984, pp. 761-763.

Modèles de sécurité

Ouvert les spécifications du système sont connues de tous.
La sécurité repose aussi sur la vérification par les paires (Ex: RSA).

Avantages:

- Modèle étanche à la traison (inutile de voler un decodeur RSA).
- Modèle facile á mettre en place avec des composantes du marché, à condition qu'elles soient aussi "ouvertes" (i.e. livrées avec leur code, et ... vérifiés et reconstruits par vous!)

Desavantages:

- La qualité de la protection depend seulement de la qualité des algorithmes et de leur mise en oeuvre (crypto encore une art).
- La qualité de la protection depende aussi de la rapidité de correction des erreurs de conception ou mise en oeuvre (necessite encore du logiciel libre)

Microsoft Security Track Record

Toute information *peut* être *recueillie*, *tracée* et *transmise* **à votre insu**.
Microsoft l'a fait, le fait et continuera à le faire pour plusieurs raisons:

lutte aux copies illégales :

- Windows Registration Wizard (`http://www.lemonde.fr/nvtechno/gates/pirate.html`)
- Licence Windows Media Player
- fichiers Office avec GUID
- projet Tempest

WMP EULA (version 8, 2002)

> ** Digital Rights Management (Security). You agree that in order to protect the integrity of content and software protected by digital rights management ("Secure Content"), Microsoft may provide security related updates to the OS Components that will be automatically downloaded onto your computer. These security related updates may disable your ability to copy and/or play Secure Content and use other software on your computer. If we provide such a security update, we will use reasonable efforts to post notices on a web site explaining the update.*

Mais aujourd'hui, c'est bien pire...

- AppStore totalitaire
- filtrage du web, et collecte d'informations personnelles
- publicité qui vous suit
- flux d'information entre sites web
- deep-packet inspection...

Les politiques se reveillent (un peu tard):
`http://privacybydesign.ca/`

Un exemple: le vote éléctronique

Comme quoi le code source ne suffit pas...

Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

Le vote éléctronique: un exemple de l'existant

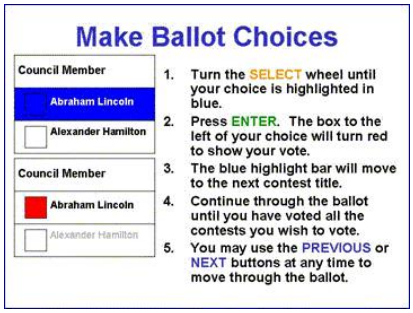
Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

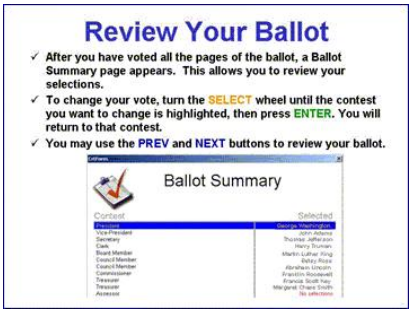
Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:



Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:



Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

Logiciel libre, une introduction
└Securité
└Vote éléctronique

Le vote éléctronique: un exemple de l'existant

Extrait de `http://www.alexandriavoter.org/eSlate/eSlate_slide_show.html`:

Est-ce que vous achétez ça?
Quel modèle de "sécurité" peut nous donner confiance dans des tels systèmes?

Logiciel libre, une introduction
└Securité
└Vote éléctronique

Logiciels Libres et vote éléctronique

Certains disent que:

> *Le code source des machines de vote éléctronique doit être entièrement du logiciel libre.*

Est-ce suffisant?Voir `http://avirubin.com/vote/response.html` pour une analyse du code source (non libre!) des machines Diebold.

Logiciel libre, une introduction
└Securité
└Vote éléctronique

Les propriétés d'un protocôle de vote éléctronique

pas de bourrage des urnes I seulement les élécteurs peuvent voter

pas de bourrage des urnes II personne peut voter plus d'une fois

pas de bourrage des urnes III personne peut voter à la place de quelqu'un d'autre

anonymat du vote personne connaît ce que *quelqu'un d'autre* a voté

contrôle cela peut prendre deux formes:

- chaque élécteur peut vérifier que sont vote est bien pris correctement en compte
- chacun peut vérifier que le vote de chaque élécteur est bien pris correctement en compte

pas de coercition personne ne peut "prouver" d'avoir voté dans un sens ou dans l'autre

Notez bien que ces deux dernières propriétés semblent incompatibles!

Logiciel libre, une introduction
└Securité
└Vote éléctronique

Une analyse du problème

- le vote à préférences multiples ordonnées
- les protocoles de vote éléctronique

Logiciel libre, une introduction
└Securité
└Vote éléctronique

Comparer avec l'existant

Discussion ouverte sur les modèles de dévéloppement du logiciel