Logiciels Libres dans la recherche et l'enseignement: enjeux et perspectives.

Roberto Di Cosmo

Professeur d'Informatique
Université Paris 7 Denis Diderot

8 Juillet 2008

Logiciel Libre: quelques définitions

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

- Liberté d'utiliser le logiciel

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

- Liberté d'utiliser le logiciel
- Liberté d'étudier les sources du logiciel et de l'adapter à ses besoins

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

- Liberté d'utiliser le logiciel
- Liberté d'étudier les sources du logiciel et de l'adapter à ses besoins
- Liberté de distribuer des copies

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

- Liberté d'utiliser le logiciel
- Liberté d'étudier les sources du logiciel et de l'adapter à ses besoins
- Liberté de distribuer des copies
- Liberté de distribuer les sources (même modifiées)

Logiciel Libre: quelques définitions

Gratuit (anglais: free):
logiciel non payant (aujourd'hui)

Libre (anglais: free):
logiciel avec 4 droits

- Liberté d'utiliser le logiciel
- Liberté d'étudier les sources du logiciel et de l'adapter à ses besoins
- Liberté de distribuer des copies
- Liberté de distribuer les sources (même modifiées)

Il y a des obligations aussi, qui varient selon la licence: GPL/BSD/Mozilla/X, etc.

La qualité de la formation

das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor

La qualité de la formation

das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor
Logiciel libre = accès au code, liberté de modifier et distribuer:

La qualité de la formation

das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor
Logiciel libre = accès au code, liberté de modifier et distribuer:

- égalité des chances: tous les étudiants ont accès à l'ensemble des outils, sans restrictions, sans besoin d'accords spécifiques

La qualité de la formation

das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor
Logiciel libre = accès au code, liberté de modifier et distribuer:

- égalité des chances: tous les étudiants ont accès à l'ensemble des outils, sans restrictions, sans besoin d'accords spécifiques
- accès à une meilleure formation (à l'informatique):
  pas de barrière à la connaissance
  "comme si un élève ingénieur pouvait participer à la construction du pont d'Aquitaine"

La qualité de la formation

das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor
Logiciel libre = accès au code, liberté de modifier et distribuer:

- égalité des chances: tous les étudiants ont accès à l'ensemble des outils, sans restrictions, sans besoin d'accords spécifiques
- accès à une meilleure formation (à l'informatique):
  pas de barrière à la connaissance
  "comme si un élève ingénieur pouvait participer à la construction du pont d'Aquitaine"
- formations spécifiques: nombreux Master en Logiciel Libre en Europe

La qualité de la formation

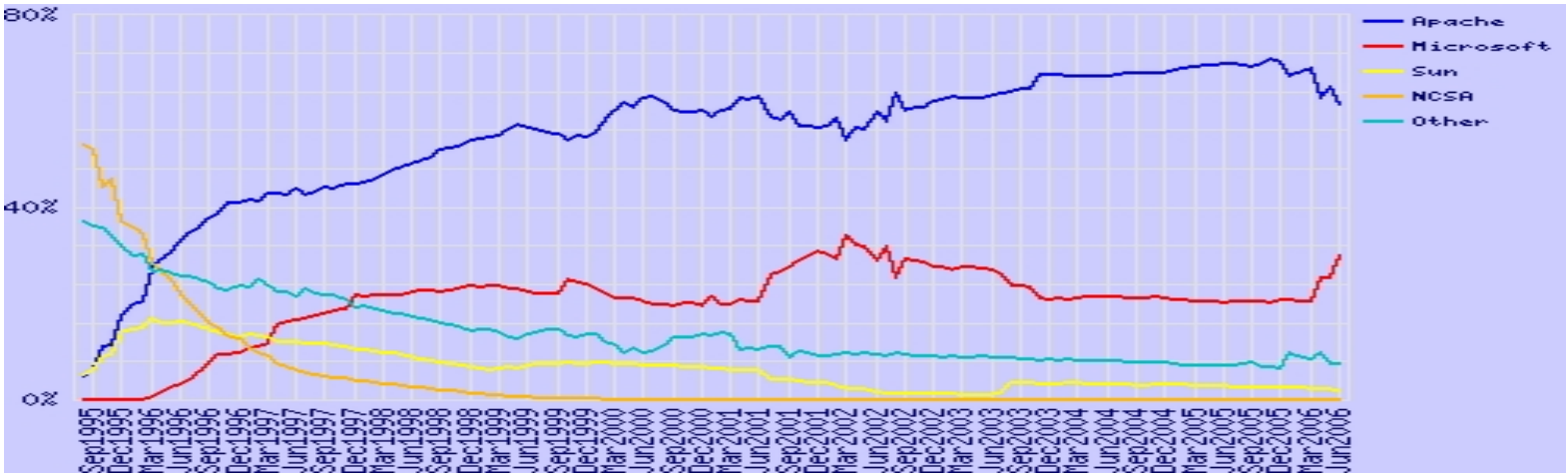
das Wesen der Mathematik liegt gerade in ihrer Freiheit
Georg Cantor
Logiciel libre = accès au code, liberté de modifier et distribuer:

- égalité des chances: tous les étudiants ont accès à l'ensemble des outils, sans restrictions, sans besoin d'accords spécifiques
- accès à une meilleure formation (à l'informatique):
  pas de barrière à la connaissance
  "comme si un élève ingénieur pouvait participer à la construction du pont d'Aquitaine"
- formations spécifiques: nombreux Master en Logiciel Libre en Europe

Le Logiciel Libre est incontournable pour l'enseignement

Le contexte économique

- Apache (free software) domine le marché des serveurs web



(Source: NetCraft, Juin 2006)

- `http://impots.gouv.fr`: 5,5 x 10^6 déclarations en 2006
- `OpenOffice.org`: Minefi (80.000), Gendarmerie (70.000), Intérieur (50.000), Equipement (55.000), Douanes (16.000)
- ...

Le Logiciel Libre est une réalité économique

Marché France (Pierre Audoin Consultants)

Year	Income (M euros)	Market share	Growth
2003	100	0.4%	-
2004	140	0.5%	40%
2005	250	0.9%	79%
2006	430	1.4%	72%
2007	730	2.2%	63%

CA des grands groupes (PAC)

SSII	2004	2005	Growth (%)
Capgemini	7 Me	15 Me	+ 114,3%
Thales DSV	10 Me	14 Me	+ 40,0%
Atos Origin	9 Me	13,5 Me	+ 50,0%
Bull	8,5 Me	13 Me	+ 52,9%
SQLI	6,3 Me	12 Me	+ 90,5%
Unilog-LogicaCMG	5,5 Me	9,2 Me	+ 67,3%
IBM Global Services	4 Me	9,2 Me	+ 130,0%
EDS	6 Me	7 Me	+ 16,7%
Devoteam	4,6 Me	6,3 Me	+ 37,0%
Sopra Group	3,4 Me	6 Me	+ 76,5%
TOTAL	64,3 Me	105,2 Me	+ 63,6%

C'est une *revolution*!

L'administration publique comme moteur

L'État *n'est pas* une "entreprise" comme les autres!
Quelques exigences *spécifiques* de l'administration

- archivage à très long terme, intégrité des données (état civil, imp ôts. . .)
- sécurité (respect de la vie privée, confidentialité, protection des in formations sensibles, défense, ...)
- coût modéré
- identification sûre du citoyen
- *devoir* de transparence

  tout cela doit se faire dans le cadre d'un

- oecuménisme technologique

Voir aussi la position du parlement européen du 5 juillet 2006

Interopérabilité, standards et qualité

Des grandes administrations demandent des solutions interopérables, et donc les veulent

- conformes à des standards normalisés (ISO, RFC, etc.)
- garanties contre la non conformité sur tout le spectre
- avec livraison du code source...

Il est très difficile de repondre à ces demandes avec des offres propriétaires...

- se re-assurer sur le fournisseur peut suffir à court terme,
- à long terme, on ouvre un boulevard à des nouveaux concurrents qui connaissent bien les Logiciels Libres

Logiciel libre au service de la recherche

evolution des plateformes : pas d'obligations d'évolution basées sur des agendas externes aux notres

dissemination : on diffuse et construit nos logiciels selon la même approche que les mathématiques

coût et simplicité : pas de coûts recurrents, pas de gestion de licences (y compris personnel qui suit les conventions Select et similia)

besoins techniques pointus : les logiciels propriétaires empêchent les corrections et adaptations, le logiciel libre le permet

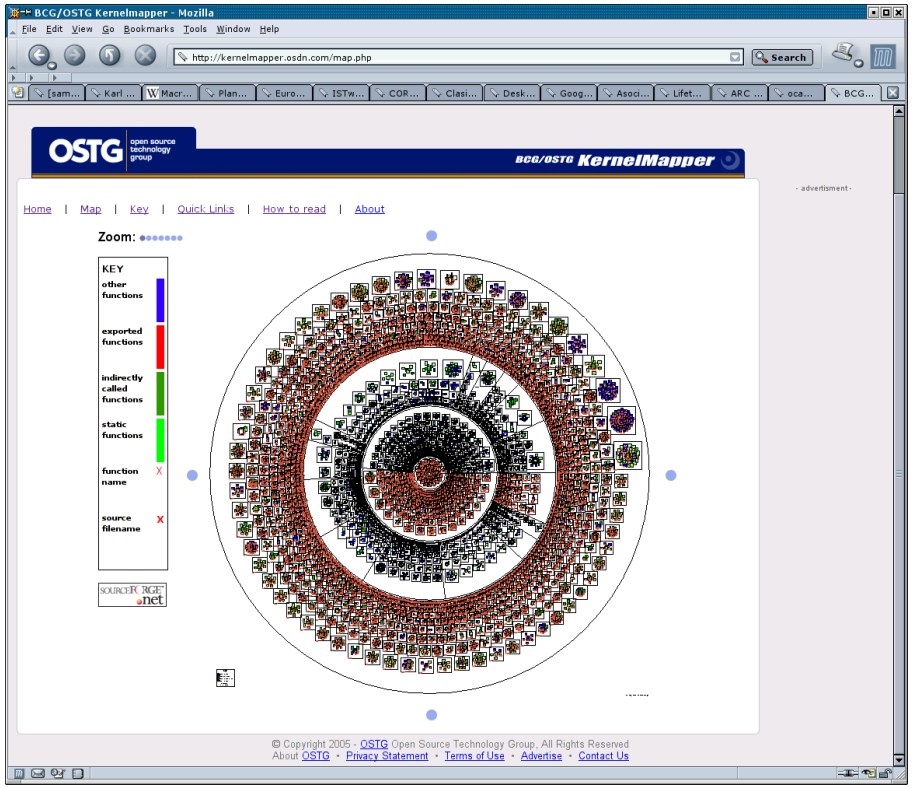
Le défi scientifique des Logiciels Libres

Un logiciel libre *n'est pas* un logiciel comme les autres:

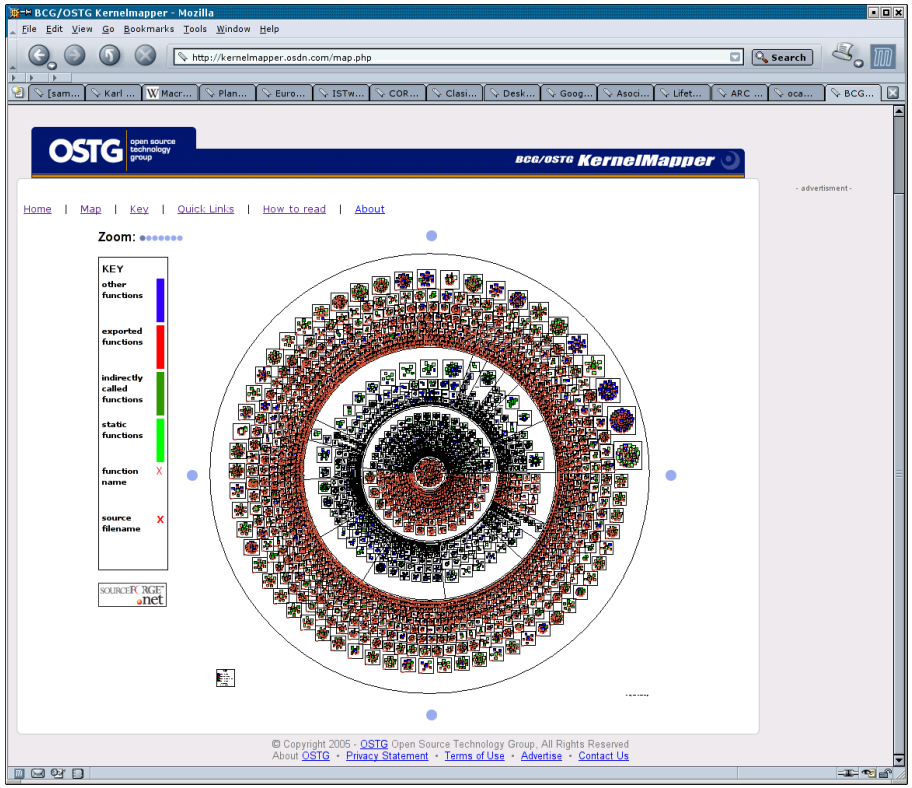
- pas d'architecte unique
- développement distribué
- cycle de développement très rapide
- interdépendances fortes
- disponibilité des sources pour des grandes masses de logiciels

  SourceForge.net: 123,736 projets, 1,342,153 utilisateurs

Des logiciels complexes. . .

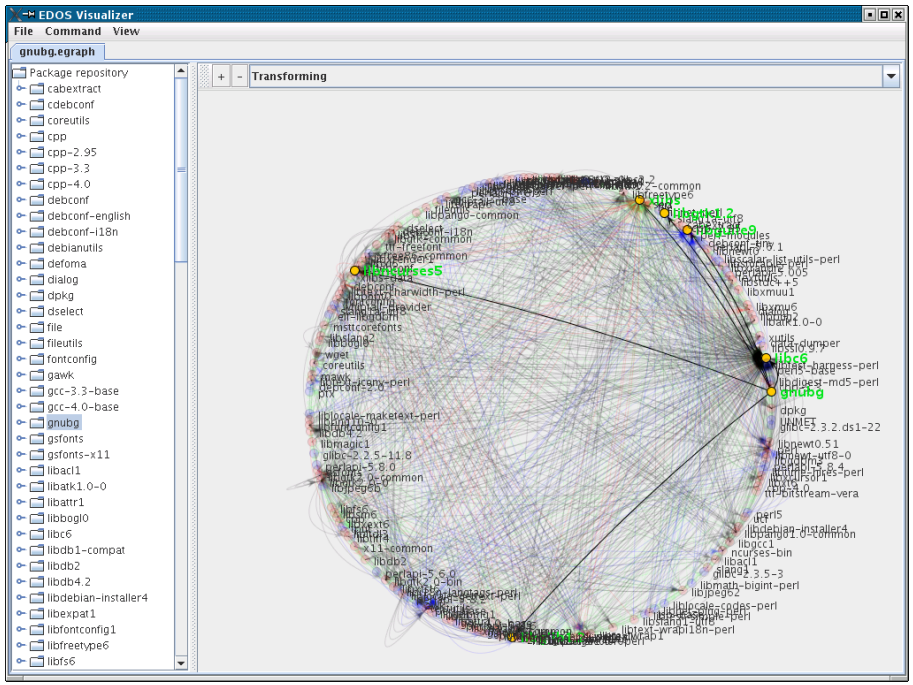


Des logiciels complexes. . .

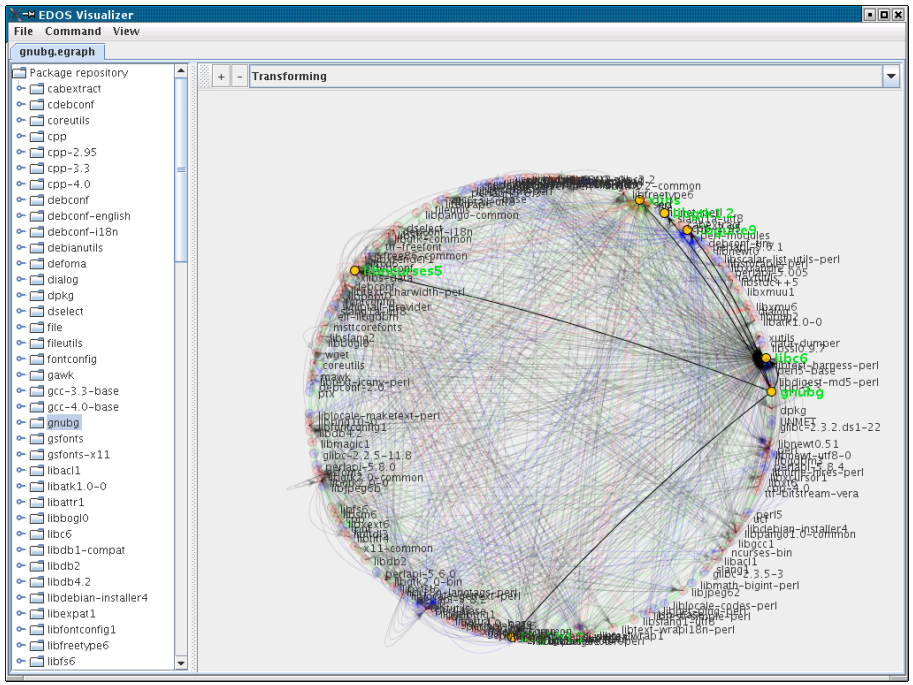



```
linux-2.6.16.20 > sloccount .
...
Total Physical Source Lines of Code (SLOC) = 4,827,227
Data generated using David A. Wheeler's 'SLOCCount'.
```

Des interdépendances complexes. . .

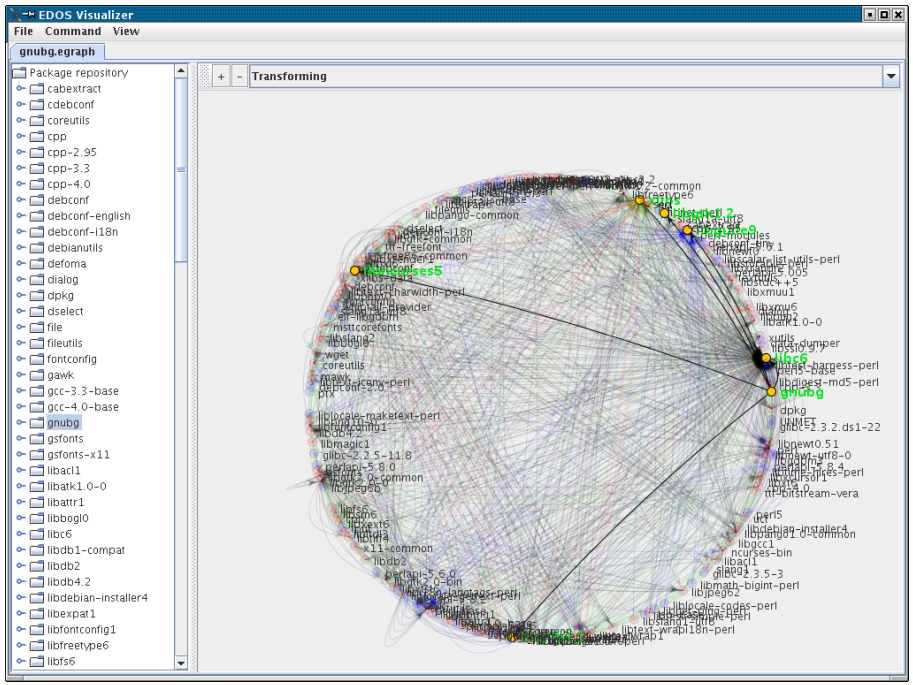


Des interdépendances complexes. . .




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

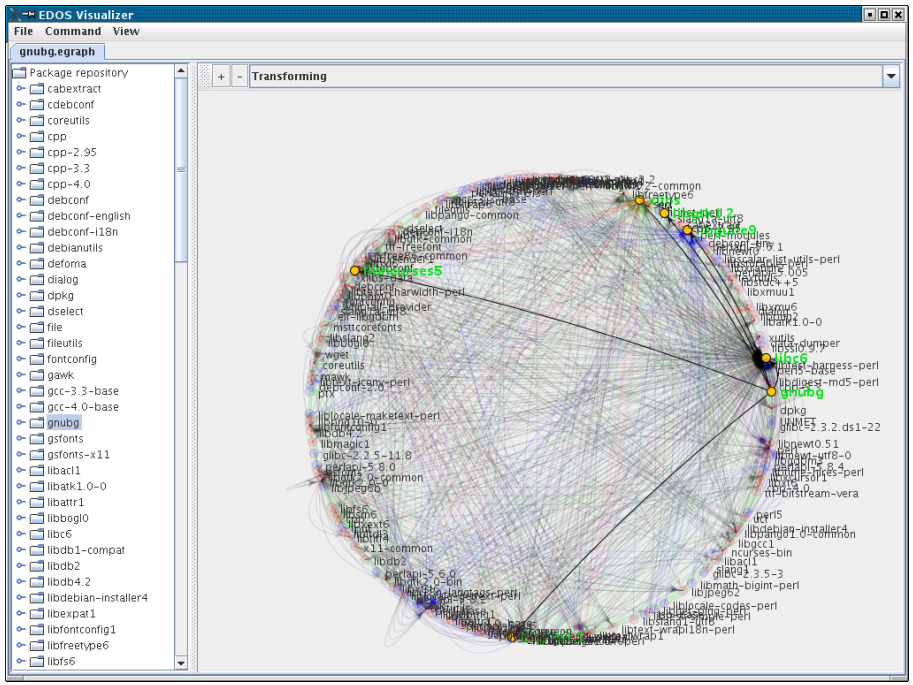
Des interdépendances complexes. . .




```
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*!

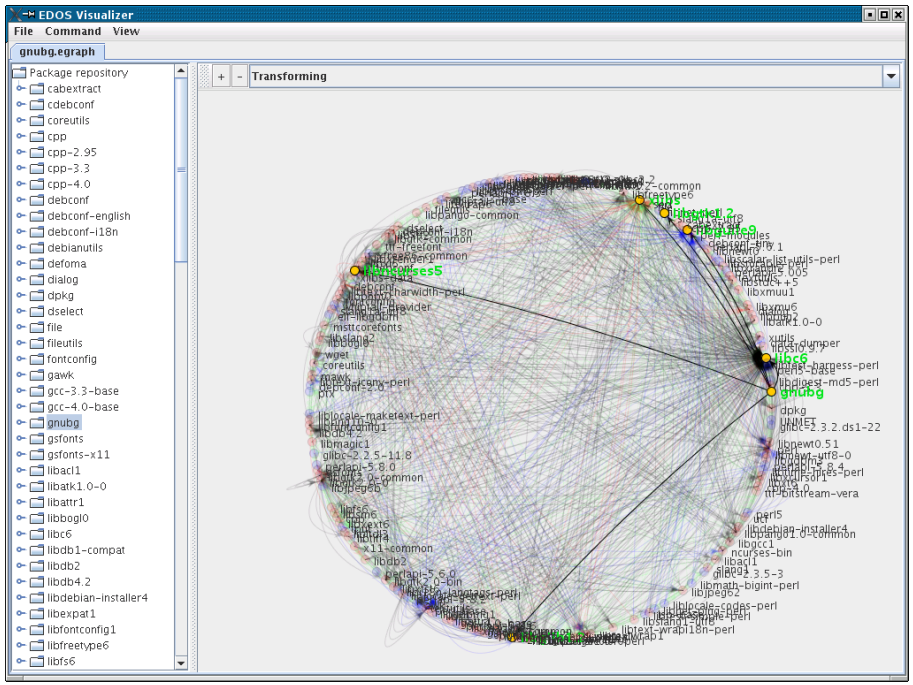
Des interdépendances complexes. . .




```
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?

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.

Installer `libc6` dans:

```
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)
```

Installer `libc6` dans:

```
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)
```

peut devenir

Installer `libc6` dans:

```
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)
```

peut devenir

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

Installer `libc6` dans:

```
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)
```

peut devenir

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

Mais il faut aller beaucoup plus loin:

- comment trouver les *meilleures* solutions?
- comment garantir la réversibilité des mises à jours?

Installer `libc6` dans:

```
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)
```

peut devenir

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

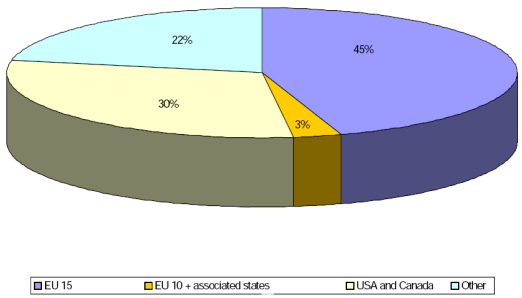
Mais il faut aller beaucoup plus loin:

- comment trouver les *meilleures* solutions?
- comment garantir la réversibilité des mises à jours?

Ces problèmes intéressants, et bien d'autres, ont un lien fort avec la réalité économique...

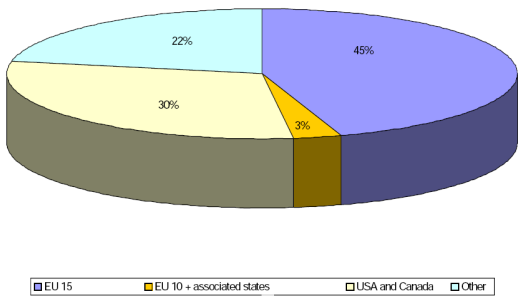
Leadership européenne et impact économique

Figure 21: Geographic distribution of leadership in development



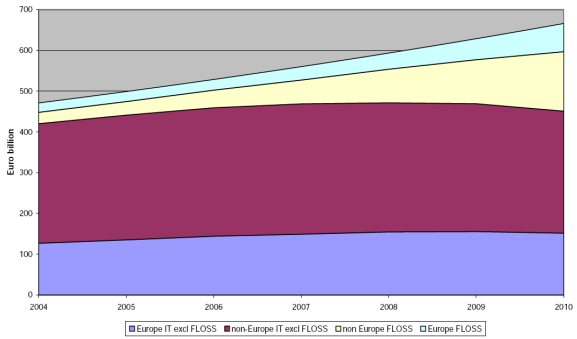
Leadership européenne et impact économique

Figure 21: Geographic distribution of leadership in development



Copyright © 2006 MERIT. Source: db.debian.org

Figure 47: FLOSS-related and IT services revenue, Europe and world



Copyright © 2006 MERIT. MERIT estimates and projections based on sources including Gartner (IT services market size); IDC (Linux server and PC sales); GGDC (software investment ratios). Shares add up to the total (Euro 667 billion in 2010).

Nos défis:

scientifique : *résoudre* les problèmes nouveaux

Nos défis:

scientifique : *résoudre* les problèmes nouveaux

pédagogique : *former* les ingénieurs de demain

Nos défis:

scientifique : *résoudre* les problèmes nouveaux

pédagogique : *former* les ingénieurs de demain

économique : *retenir* les succès créateurs de valeur

A good engineer has a demanding life

- design real-world systems that will go into production
- understand complex software,
  at least as much as necessary to modify and adapt it
- build complex systems by reusing existing components
- interact with other, often strongly opinioned, developers

Yet, we still teach computer science like 20 years ago!

- one algorithm at a time
- one monolithic program (big or small) for each project
- one student at a time

this needs to change, and free software is the key

An example: teaching algorithms in a modern way

Let's take one of the favorite intruductions to dynamic programming

Longest Common Subsequence (LCS)

> *given two sequences* $X = (x_1, x_2, \ldots x_n)$ *and* $Y = (y_1, y_2, \ldots y_m)$, *we whish to find a maximum length common subsequence of X and Y.*

For example, for X = BDCABA and Y = ABCBDAB, the sequence BCBA is such a common subsequence (BDAB is another one).
How do we find one?
Sould we enumerate all subsequences of X and Y, then find the common ones and pick a longest one?
Hey, that would require exponential time!

The algorithmic insight, 1

We remark that the LCS problem has an *optimal substructure* property:
for $X = (x_1, x_2, \ldots x_n)$ and $Y = (y_1, y_2, \ldots y_m)$, and $Z = z_1, \ldots, z_k$ an LCS

- if $x_n = y_m$ then $z_k = x_n = y_m$ and Z_{k-1} is an LCS of X_{n-1} and Y_{m-1}
- if $x_n \neq y_m$ then $z_k \neq x_n$ implies Z is an LCS of X_{n-1} and Y
- if $x_n \neq y_m$ then $z_k \neq y_m$ implies Z is an LCS of X and Y_{m-1}

The algorithmic insight, 2

So we can fill an n by m table $c[i,j]$ containing the length of the LCS of X_i and Y_j

$$c[i,j] = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ c[i-1,j-1]+1 & x_i > 0, y_j > 0, x_i = y_j \\ max(c[i,j-1], c[i-1,j]) & x_i > 0, y_j > 0, x_i \neq y_j \end{cases}$$

The algorithmic insight, 3

This can be done bottom up with the simple code that follows

```
for i = 1 to n do c[i,0] = 0
for j = 1 to m do c[0,j] = 0
for i = 1 to n do
for j = 1 to m do
if x[i]=y[j] then c[i,j] = c[i-1,j-1] +1
else c[i,j] = max(c[i,j-1], c[i-1,j])
```

Notice that:

- we can actually recover an LCS from the matrix c
- the algorithm runs in $O(mn)$ time
- the algorithm requires $O(mn)$ space

The algorithmic insight, 4

Many lecturers conclude “this is how the diff program works!”

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?
- Is diff *really* building an n by m array of *text lines*?

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?
- Is diff *really* building an n by m array of *text lines*?
- Is diff *really* comparing *text lines*?

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?
- Is diff *really* building an n by m array of *text lines*?
- Is diff *really* comparing *text lines*?

Is *the student* asking himself these fundamental questions?

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?
- Is diff *really* building an n by m array of *text lines*?
- Is diff *really* comparing *text lines*?

Is *the student* asking himself these fundamental questions?
With proprietary software, you would never know. . . . .

The algorithmic insight, 4

Many lecturers conclude "this is how the diff program works!"

really?

- Is $O(nm)$ an acceptable space and time complexity, *in practice*?
- Is diff *really* building an n by m array of *text lines*?
- Is diff *really* comparing *text lines*?

Is *the student* asking himself these fundamental questions?
With proprietary software, you would never know. . . . .
With free software, things change radically!

A look at diff internals

```
apt-get source diffutils
cd diffutils-2.8.1/src
less analyze.c
...
/* The basic algorithm is described in:
"An O(ND) Difference Algorithm and its Variations", Eugene
Myers,
Algorithmica Vol. 1 No. 2, 1986, pp. 251-266;
see especially section 4.2, which describes the variation
used below.
Unless the --minimal option is specified, this code uses
the TOO_EXPENSIVE
heuristic, by Paul Eggert, to limit the cost to O(N**1.5
log N)
at the price of producing suboptimal output for large input
with
```

A look at diff internals, 2

```
less io.c
...
/* Lines are put into equivalence classes of lines that
match in lines_differ.
Each equivalence class is represented by one of these struc
but only while the classes are being computed.
Afterward, each class is represented by a number. */
struct equivclass
{
lin next; /* Next item in this bucket. */
hash_value hash; /* Hash of lines in this class. */
char const *line; /* A line that fits this class. */
size_t length; /* That line's length, not counting its
newline. */
};
```

A look at diff internals, 3

```
less analyze.c
...
/* Discard lines from one file that have no matches in
the other file.

A line which is discarded will not be considered by the
actual
comparison algorithm; it will be as if that line were not
in the file.
The file's 'realindexes' table maps virtual line numbers
(which don't count the discarded lines) into real line
numbers;
this is how the actual comparison algorithm produces result
that are comprehensible when the discarded lines are counte

When we discard a line, we also mark it as a deletion or
```

Free software makes a difference

By looking at the *free source code* of a real-world, industry-strength implementation of the diff algorithm, our students have learned :

- a real-world program is much more than just *one* algorithm
  - optimize the common case (the $O(DN)$)
  - use hashing where appropriate (line equivalence classes)
  - reduce the size of the problem (remove lines that are not common)
- follow references to *freely accessible* research papers
- documentation, and comments, are essential to understand the code

Free software and sustainable development

New profiles for our students

Leveraging free software, we can identify three different levels:

FLOSS technician : knows how to use some successful FLOSS projects, off the shelf (Apache, Linux, etc.) This can be done with e-learning, at a 2 or 3 year graduate level, see `http://www.eof.eu.org/`.

FLOSS engineer : she also knows how to inspect, modify and fix some complex code coming from the FLOSS community

FLOSS core developer : she is able to become part of a FLOSS community, and have her changes accepted back in it

See also Paterson's letter in Communication of the ACM (03/06): "Computer Science Education in the 21st Century".
This is key to nonvolatile, highly qualified jobs.