

Créer une application:
le guide incomplet...
mais pratique quand même

Luis Menina

- 2002 : Ingénieur (filière apprentissage)
- 2003 : Windows → GNU/Linux
- 2014 : Développeur et intégrateur pour Peaks

Présentation de la société Peaks



Peaks est une société de conseil et d'ingénierie spécialisée dans les technologies de l'information.

Peaks adresse ses services à une soixantaine de clients allant de la Startup aux Grands Comptes.



Chiffres clés

105 consultants talentueux répartis sur

4 sites : Paris, Lyon, Reims et Nantes

8,6M€ de chiffre d'affaires

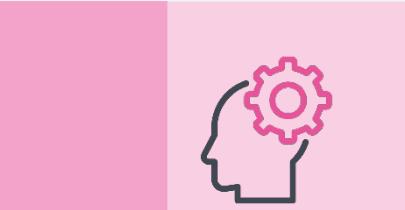
10 années d'expérience en pilotage d'activités de service

L'offre de Peaks



Vous développez un produit ?

Il vous manque des compétences ?



Business Managers
Qualification du besoin



Peaks Lab

Peaks Digital Factory



Peaks Talents



Livrables

Site Web

Application mobile

Application Desktop

Web Apps

AMOA

Chef de Projet

Designer UX, intégrateur

Développeur

Architecte Technique

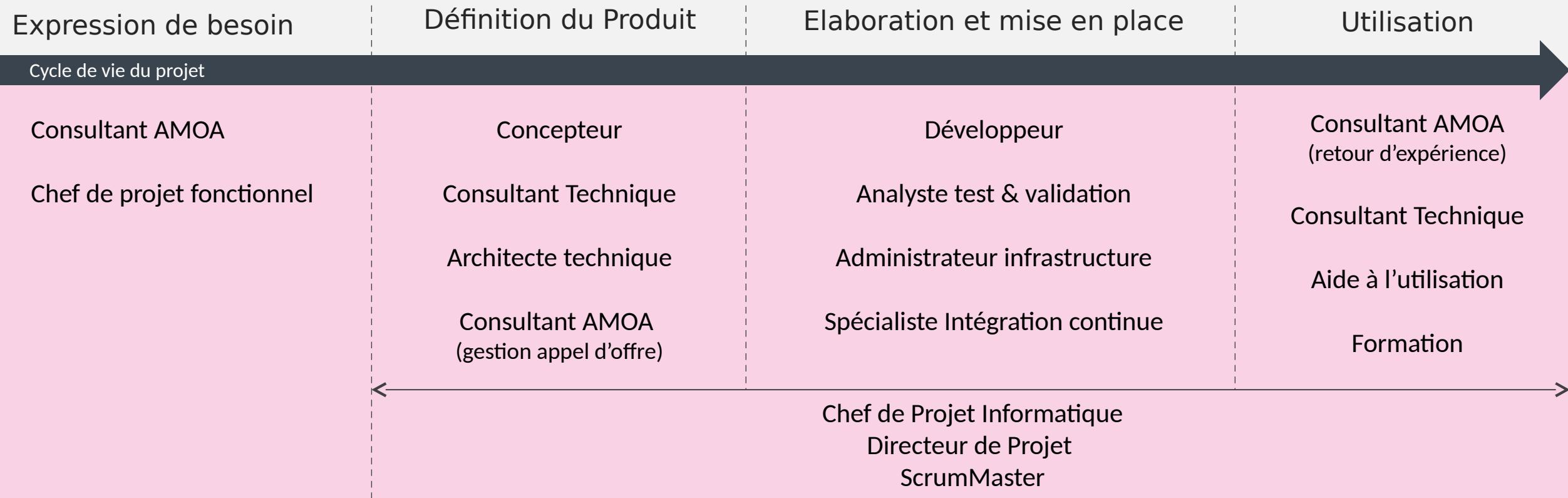
Intégrateur Front

L'offre Peaks

Peaks dans le cycle de vie d'un projet



Peaks est en mesure d'intervenir à tous les niveaux du cycle de vie d'un projet digital

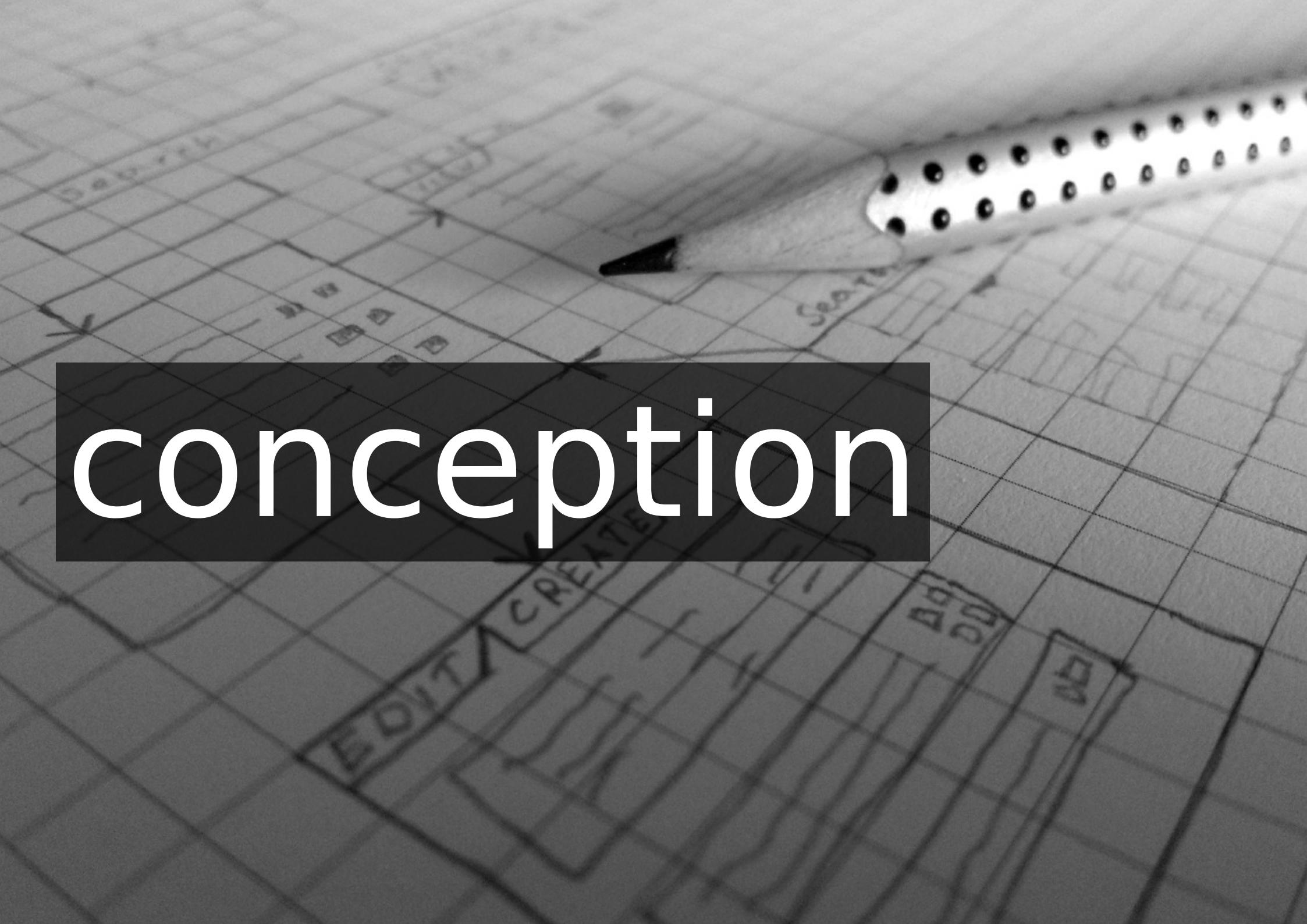


À la base de chaque application :

idée



conception



```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;
```

implémentation

```
while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3)) {
            continue;
        }
    }
}
```

```
#include <stdio.h>

int main (int argc, char *argv[ ] )
{
    printf ("Hello, World !");
    return 0;
}
```

```
./hello-world  
Hello, World !
```

profit!
#penelope



Simple, et pourtant...

... impossible sans
outils

Même pour une application triviale,
vous aurez utilisé...



- éditeur de texte
- langage de programmation
- compilateur/interpréteur
- système d'exploitation

« Il est tentant, quand on n'a pour seul outil un marteau, de tout traiter comme un clou »

— Abraham Maslow
The Psychology of Science:
A Reconnaissance
(1966)



Trouvez vos outils



30 langages = 30 marteaux



L'éditeur de texte

Indépendant

Vim

Emacs

SublimeText

Notepad++

etc.

Intégré dans un IDE

Eclipse
Visual Studio
Code::Blocks
GNOME Builder
etc.

Ce qu'on attend d'un éditeur

coloration syntaxique

Ce qu'on attend d'un éditeur
marque-pages

Ce qu'on attend d'un éditeur

indentation automatique

Ce qu'on attend d'un éditeur
macros

Ce qu'on attend d'un éditeur

remplacement de texte

Ce qu'on attend d'un éditeur
multi-documents

Ce qu'on attend d'un éditeur

sélection rectangulaire

Ce qu'on attend d'un éditeur
extensible

Bonnes pratiques d'édition :

Respectez les conventions
du document que vous éditez...

...ou adoptez les conventions du langage

Python PEP8: <https://www.python.org/dev/peps/pep-0008/>

Respectez l'indentation

→ configurez votre éditeur !

- facilite la lecture des diffs
- évite l'attribution
- sémantiquement important



+1

SANGIORGI & WALKER



Guzdial
Rose



Squeak

Open Personal Computing
and Multimedia

CAMBRIDGE

DYBVIG

THE SCHEME PROGRAMMING LANGUAGE ANSI SCHEME

SECOND EDITION

Nelson

Systems Programming with Modula-3

SECOND EDITION



Learning Python

Lutz & Ascher

THIRD EDITION

Programming Perl

Wall,
Christiansen
& Orwant



The Craft of
Functional
Programming

Thompson

ML97 EDITION

ULLMAN

ELEMENTS OF COMPUTER PROGRAMMING

ML97 EDITION

The Little MLer

Felleisen and Friedman



The Java™ Programming Language

Arnold
Gosling



JAVA



The Dylan
Reference Manual

Shalit

Addison Wesley

STROUSTRUP



THE C++ PROGRAMMING LANGUAGE

THIRD
EDITION

KERNIGHAN • RITCHIE

THE C PROGRAMMING LANGUAGE

SECOND EDITION

Software Construction

Addison Wesley

PTR

Choix du langage

le plus adapté... parmi ceux que maîtrisez

Contraintes

Quelle plate-forme(s) logicielle(s) ?

Quelle version(s) de plate-forme ?

Quelle plate-forme(s) matérielle(s) ?

Quel usage ?

portabilité
performance
maintenance
écosystème

Quel langage... compilé ?

cible différente de l'hôte ?
→ compilation croisée

Quel langage... interprété ?

Quelle version de standard ?

C++: C++98, C+03, C++11

C: C89, C99, C11

Python: 2.7, 3.x

etc.

Occasion d'apprendre
un nouveau langage ?



librsvg

<https://git.gnome.org/browse/librsvg/>

A close-up photograph of a young man with dark hair and brown eyes. He is blowing a large, iridescent bubble from a pink bubble wand held near his mouth. The bubble is filled with colorful, rainbow-like patterns. He is wearing a light-colored, short-sleeved shirt. In the background, there is a chain-link fence and some greenery, suggesting an outdoor setting like a park or playground. The entire image is overlaid with large, bold red text.

c
2001
Federico
XML → bitmap

Migration C → Rust

oxydation

<https://people.gnome.org/~federico/news-2016-10.html#25>

C

langage difficile
gestion mémoire
sécurité (buffer overflow)

C : risques

fichiers mal formés :

→ crash

→ faille de sécurité

Critique en production

Rust

soutenu par Mozilla et Samsung
utilisé dans Firefox (Servo, Quantum)

<https://www.rust-lang.org/fr/>

Objectifs de Rust

parallélisme
gestion mémoire
sécurité

```
fn main() {  
    println!("Hello, Rust!");  
}
```

Migration → Rust

intégrable dans du code C
migration par morceaux

« Vous n'avez pas besoin de tout réécrire en une fois.
Je ne connais pas d'autre langage qui permette cela. »

— *Federico Mena Quintero, 2016*

IoT ?

Bonnes pratiques de codage :

Respectez la casse

snake_case
CamelCase
mixedCase

Respectez le codage (charset)

```
printf ("©o®ação = ♥");
```

UTF-8 vs ISO 8859-1

Écrivez du code portable

(essayez)

chemins du système de fichiers

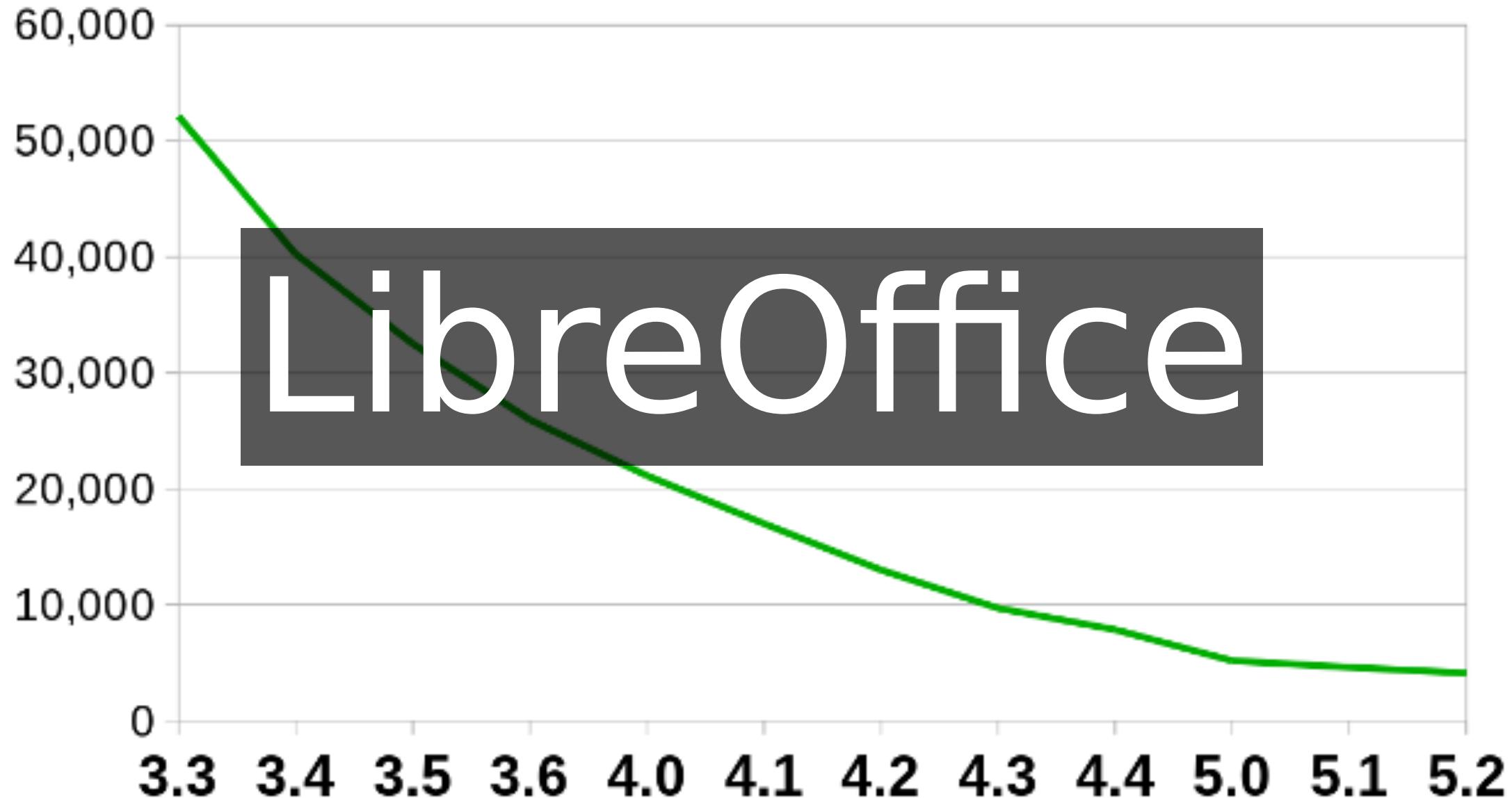
Respectez la langue

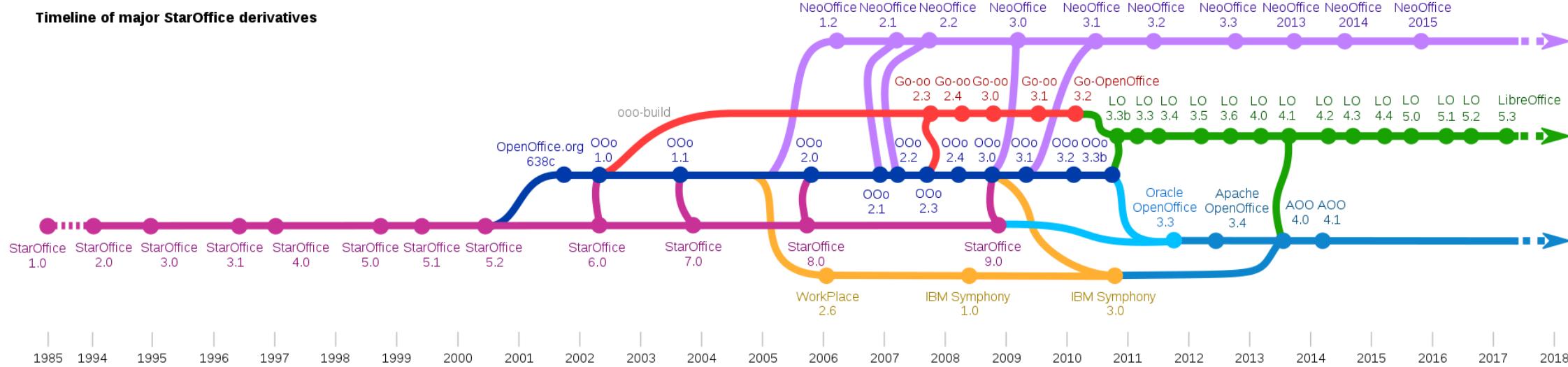
fonctions
variables
commentaires

en anglais !

- plus court
- plus facile à diffuser (S.O ☺)
- maintenable

Detected lines of German comment





32 ans !



Anglais US

The illustration depicts a stylized, voluminous hairdo in shades of yellow, orange, and brown, resembling Donald Trump's signature hairstyle. The hair is rendered with sharp, angular lines and soft shading to create a textured, layered effect against a solid black background.

Do we get the leaders we deserve?

Mantras du développeur

**Don't
Repeat
Yourself**

Keep
It
Simple
Stupid

**Not
Invented
Here**

Zen of Python

<https://www.python.org/dev/peps/pep-0020/>



+1

Feature
for future
release

Feature
for
next release

fixed for
production:
hotfix 0.2

Incorporate
bugfix in
develop

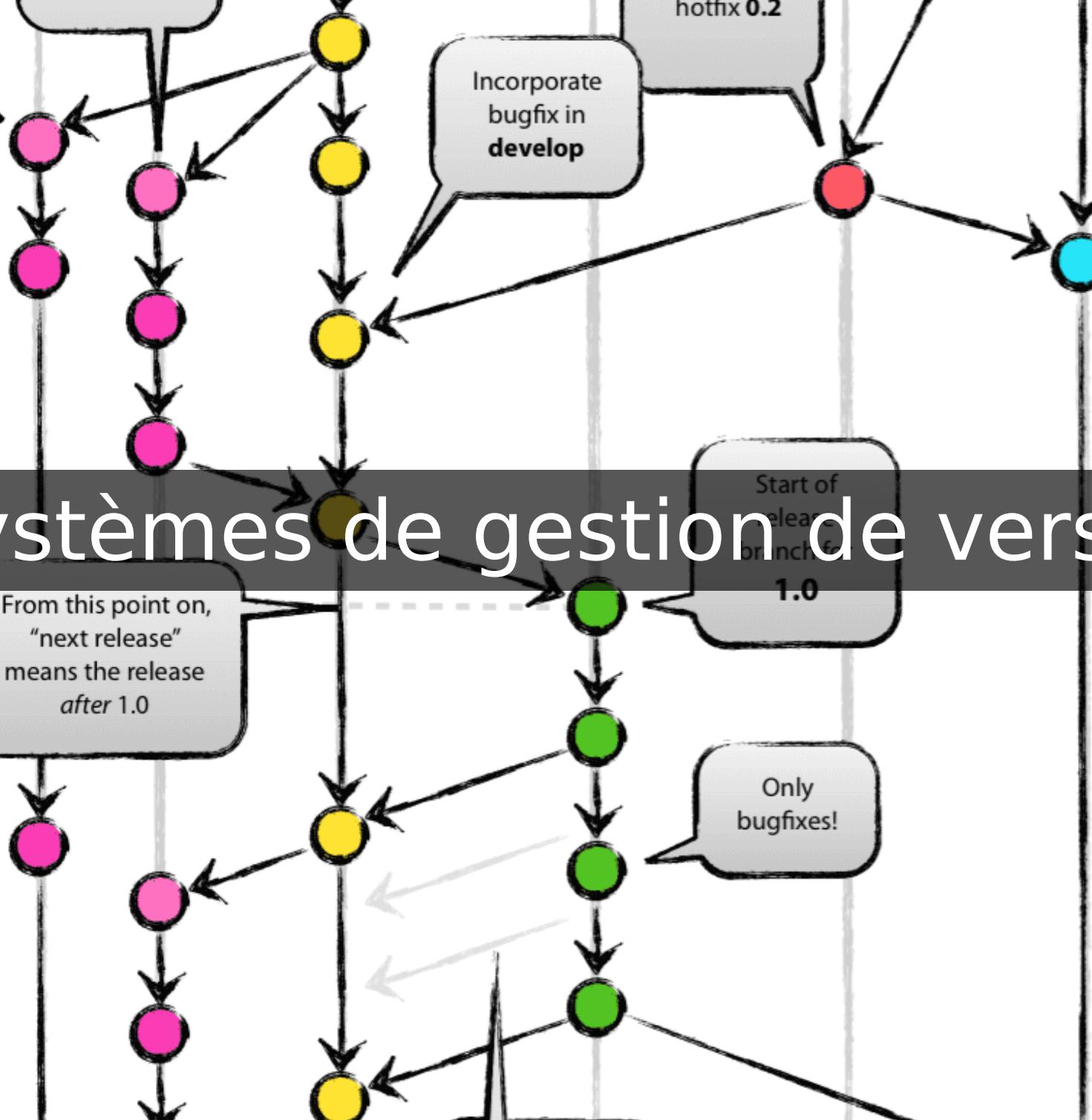
Tag
0.2

Systèmes de gestion de version

From this point on,
“next release”
means the release
after 1.0

Start of
release
branch for
1.0

Only
bugfixes!





Sondage !

- ✓ CVS
- ✓ Subversion (SVN)
- ✓ Git
- ✓ Perforce
- ✓ Mercurial
- ✓ PVCS
- ✓ Microsoft SourceSafe
- ✓ Rationale Synergy
- ✓ cpold



Joel Spolsky

<http://french.joelonsoftware.com/Articles/TheJoelTest.html>

Le test de Joël

- 1. Utilisez-vous un système de gestion de code source ?**
2. Pouvez-vous faire un build en une seule étape ?
3. Faites-vous des builds quotidiens ?
4. Avez-vous une base de données de bugs ?
5. Corrigez-vous vos bugs avant d'écrire du nouveau code ?
6. Avez-vous un planning à jour ?
7. Avez-vous une spec ?
8. Les programmeurs bénéficient-ils d'un environnement de travail calme ?
9. Utilisez-vous les meilleurs outils que vous puissiez vous payer ?
10. Avez-vous des testeurs ?
11. Les candidats écrivent-ils du code pendant leur entretien d'embauche ?
12. Faites-vous des tests d'utilisabilité de couloir ?

Définitions

checkout/clone
commit
branch
merge

Systèmes centralisés (SVN)

serveur et réseau critiques
distant = lent (latence, débit)

$1 \rightarrow n$

Systèmes décentralisés (Git)



git

<https://git-scm.com/>

Linus
1991: Linux
2005: git



Git : avantages

local = rapide
agilité

branches multiples

merge facilité

1 → n ou n → n

Git : bonus

interopérabilité SVN (git-svn)
chaque clone est une sauvegarde
workflow de petits commits



Git

plus grande complexité
développeur → intégrateur
gros binaires

Git: créer un dépôt

git init

git add .

git commit

Bonnes pratiques gestion de version :

diff avant commit !



+1

Systèmes de construction



Le test de Joël

1. Utilisez-vous un système de gestion de code source ?
- 2. Pouvez-vous faire un build en une seule étape ?**
3. Faites-vous des builds quotidiens ?
4. Avez-vous une base de données de bugs ?
5. Corrigez-vous vos bugs avant d'écrire du nouveau code ?
6. Avez-vous un planning à jour ?
7. Avez-vous une spec ?
8. Les programmeurs bénéficient-ils d'un environnement de travail calme ?
9. Utilisez-vous les meilleurs outils que vous puissiez vous payer ?
10. Avez-vous des testeurs ?
11. Les candidats écrivent-ils du code pendant leur entretien d'embauche ?
12. Faites-vous des tests d'utilisabilité de couloir ?

Systèmes de construction

Makefile

Projet Eclipse, Visual Studio

autotools

CMake

SCons

etc.

Systèmes de construction

aide à la configuration
debug/release
construit les artefacts

Meson

<http://mesonbuild.com>

Meson

dépendances : Ninja, python 3
= facilement déployable

Ninja

<https://ninja-build.org/>

Ninja : buts

rapidité
faire une chose, la faire bien → KISS

Ninja : particularités

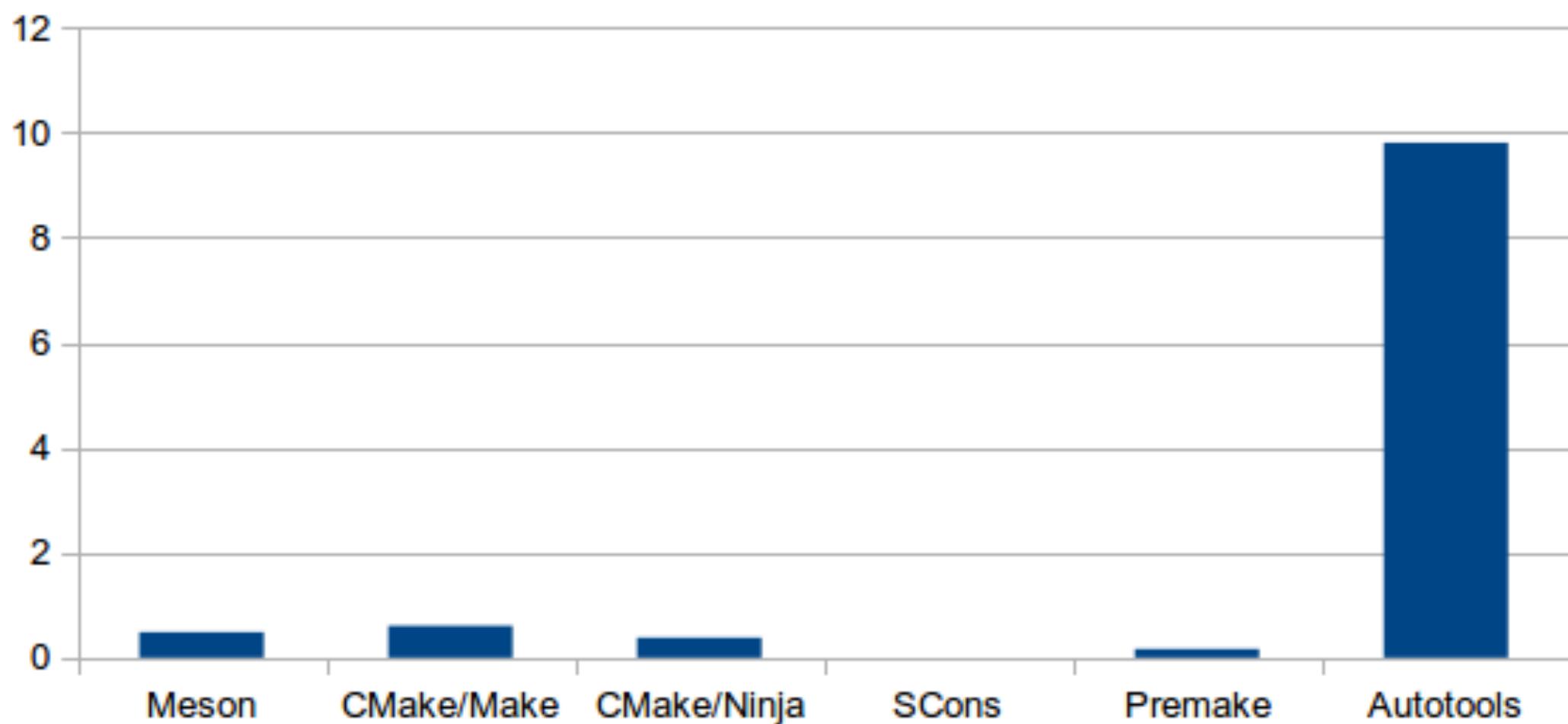
langage de description, pas de programmation
fichiers verbeux
fichiers générés
remplace make et les Makefiles
backend pour CMake

Ninja : utilisateurs

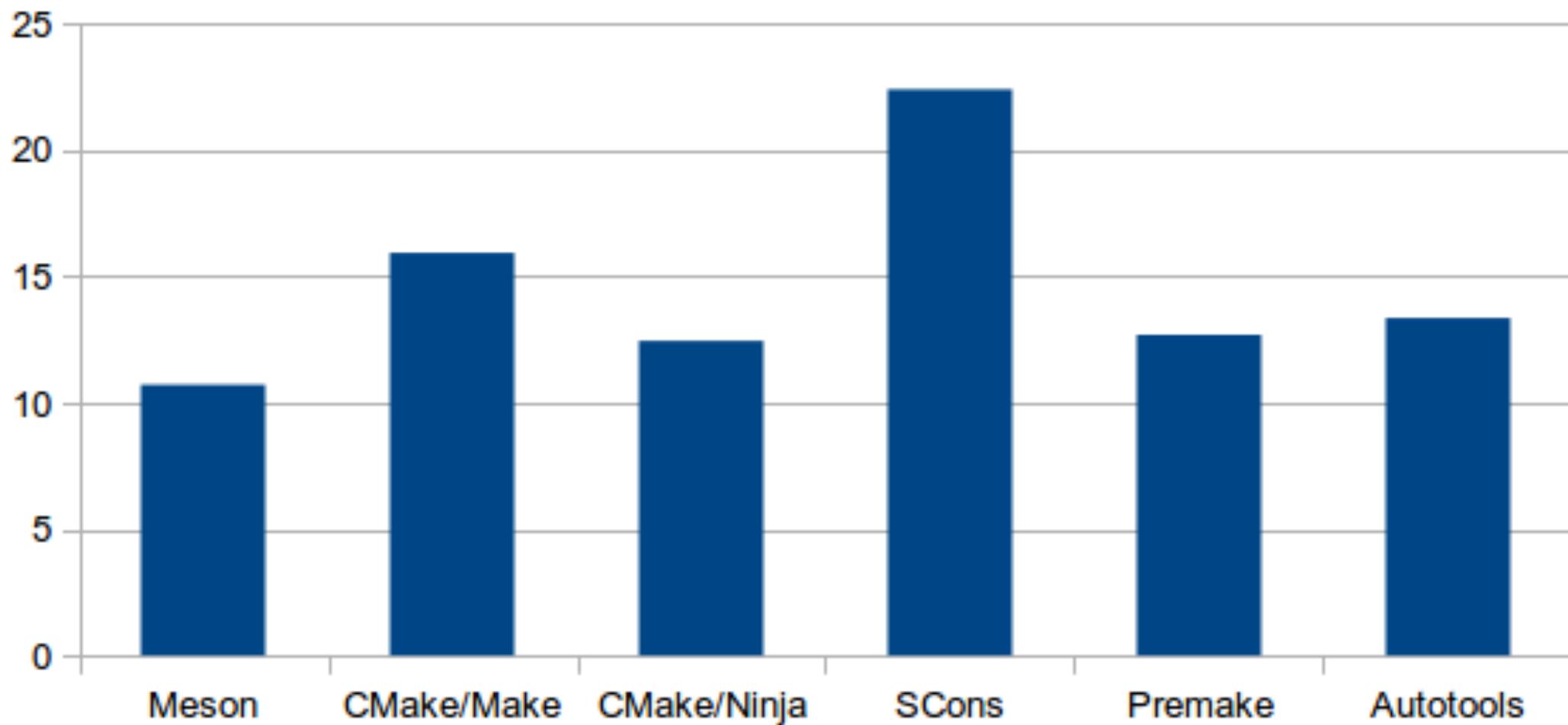
Google Chrome
GStreamer

```
mkdir build && cd build && meson ..  
ninja
```

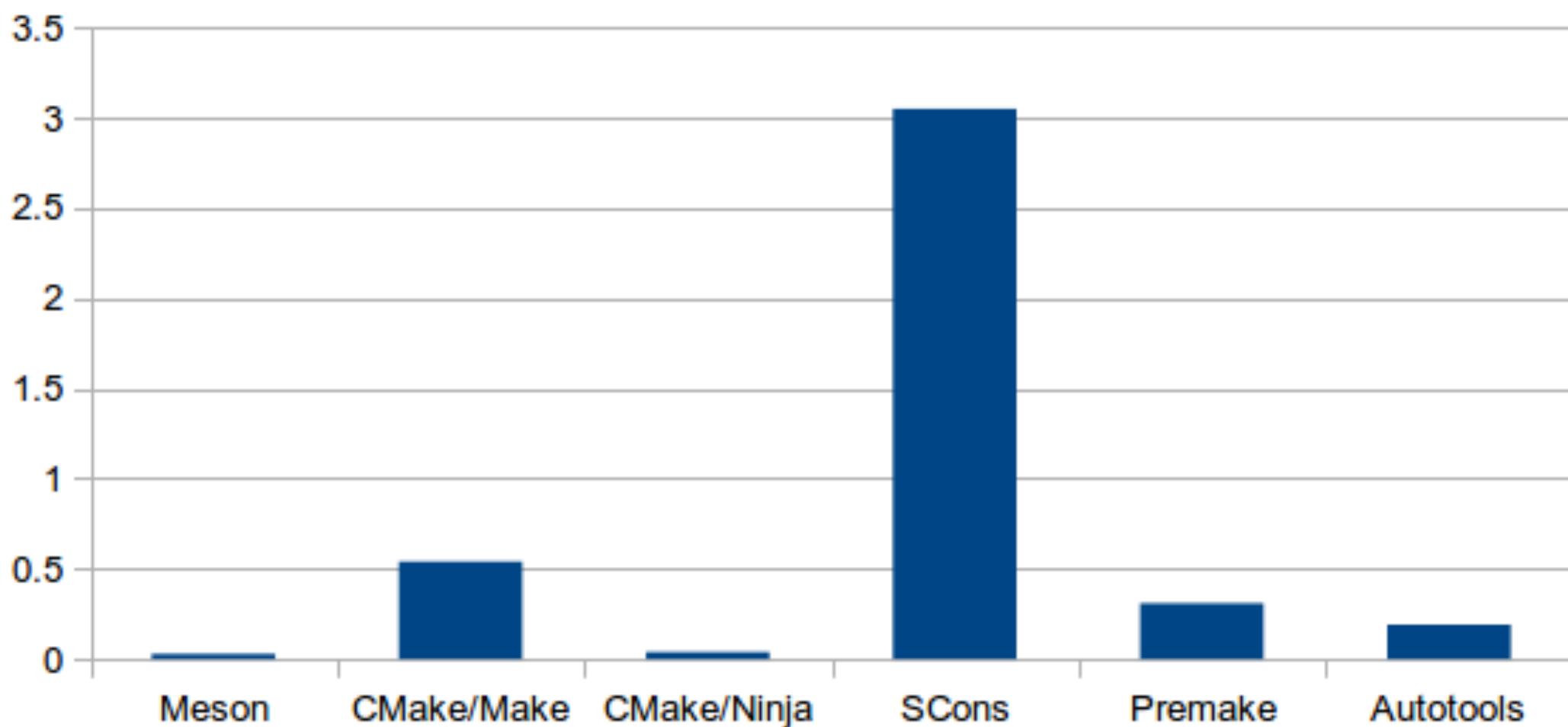
Configuration time



Build time



Empty build time



Source: <https://github.com/mesonbuild/meson/wiki/Simple%20comparison>

Meson : résultats

« About 2.5x faster on Linux
and 10x faster on Windows
for building the core GStreamer repository »

(comparé aux autotools)

<http://blog.nirbheek.in/2016/05/gstreamer-and-meson-new-hope.html>

Bonne pratique :

Ne construire que le nécessaire



Trouvez vos outils



Trouvez votre workflow

FIN

Questions ?

**ASK MORE
QUESTIONS**

GET MORE ANSWERS
Anthony Doerr

Analyseurs statiques

*lint, deblint, pylint, coala, shellcheck
indispensable pour les langages interprétés
enseigne les bonnes pratiques

Coala

vim + git + pinpoint

vim + git + lilypond + GNU Make

vim + git + dot

❤ Git

git rebase --interactive

 Git

git commit - - amend

♥ Git

git add --patch



Git

git stash

 Git

git bissect

♥ Git

git format-patch