

Bonnes pratiques de développement

Claire Mouton - CREATIS, Lyon, France

claire.mouton@creatis.insa-lyon.fr

En collaboration avec

Vincent Miele - CNRS - Biométrie & Biologie Evolutive

Plan

I. Choisir un langage (Vincent)

II. Adopter une méthode de développement

III. Coder

IV. Partager, travailler en équipe

V. Profiling

VI. Débugger

II. Adopter une méthode de développement

I. Choisir un langage (Vincent)

II. Adopter une méthode de développement

III. Coder

IV. Partager, travailler en équipe

V. Profiling

VI. Débugger

II. Adopter une méthode de développement

J'ai choisi mon langage... Comment je procède maintenant?

J'aimerais pouvoir réutiliser et modifier mon code plus tard!

Et si je travaille en équipe?

II. Adopter une méthode de développement

Enthousiaste, je code dans un fichier...

qui finit par atteindre des milliers de ligne!

→ Comment en réutiliser une partie?

Ou faire une modification ?

Bonne pratique :

Séparer le code en plusieurs fichiers, chacun ayant une unité sémantique (fonction, classe),

un programme principal (main) y fera appel.

→ Concevoir l'architecture du logiciel en amont de l'écriture

II. Adopter une méthode de développement

Unified Modeling Language (UML)



Dessins qui résument les dépendances, la structure ou les fonctionnalités du programme

II. Adopter une méthode de développement

Unified Modeling Language (UML)

Langage de modélisation unifié

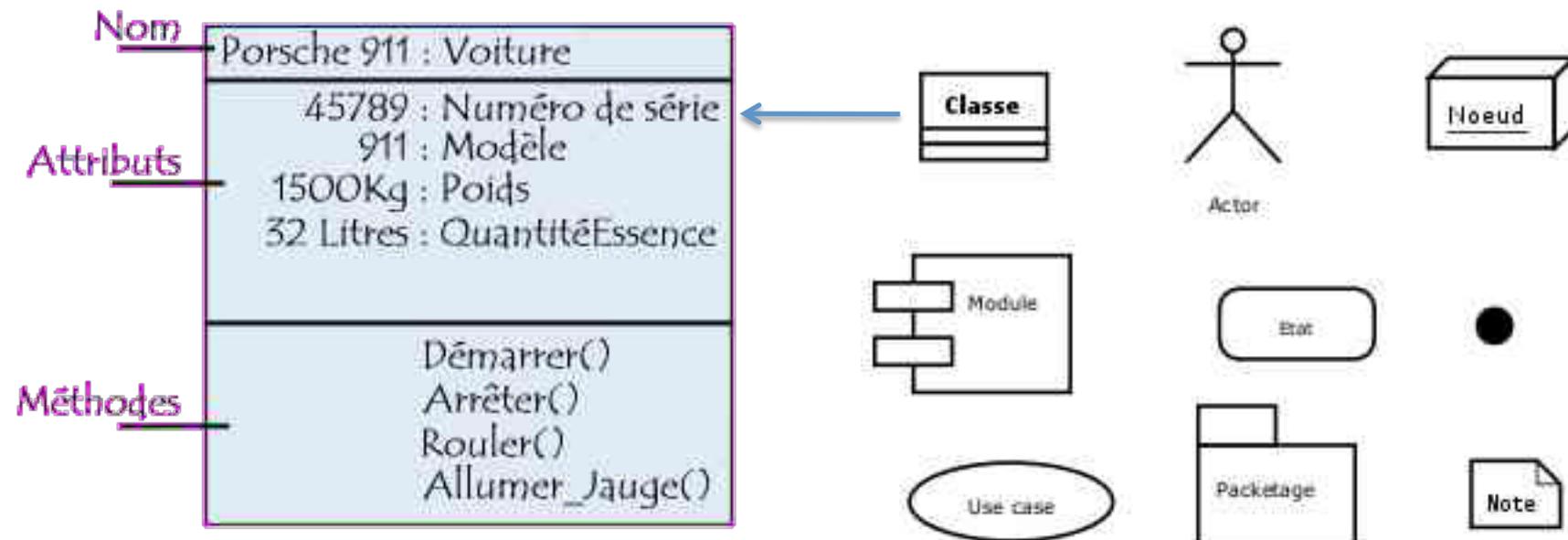
Notation permettant de modéliser un problème de façon standard

- Très lié au monde de la programmation objet
- Il existe un formalisme complet
- Le connaître et l'utiliser peut être utile:
 - pour mettre à plat le problème, réfléchir sur le programme avant de commencer à coder
 - pour être compris par les autres
 - pour comprendre les autres

II. Adopter une méthode de développement

Unified Modeling Language (UML)

Il existe un formalisme complet:



II. Adopter une méthode de développement

Unified Modeling Language (UML)

Éléments importants de l'UML : les « diagrammes »

13 diagrammes.

Les 2 diagrammes les plus utiles:

- Les diagrammes de classes
- Les diagramme de cas d'utilisation

II. Adopter une méthode de développement

Unified Modeling Language (UML)

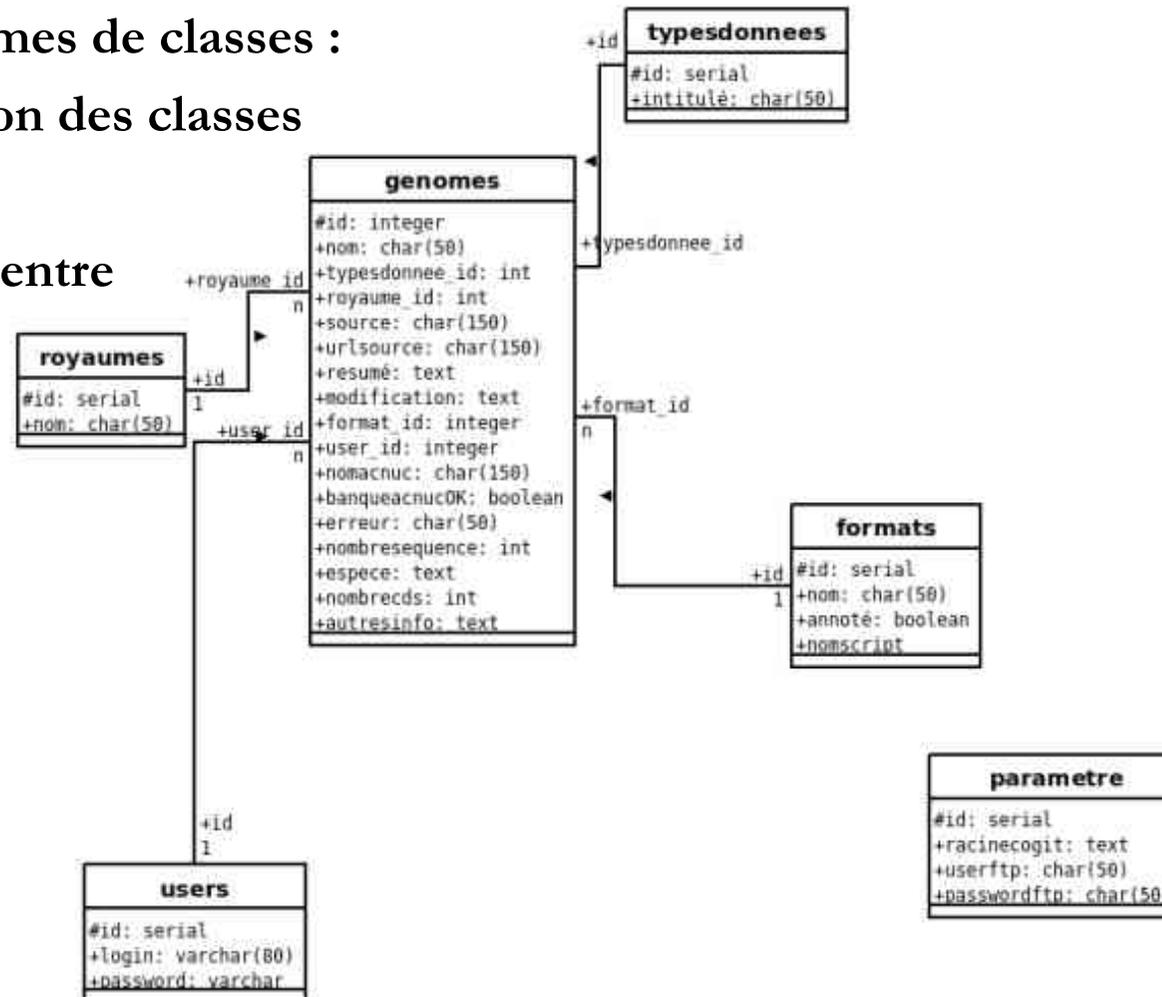
Les diagrammes de classes :

Représentation des classes

et

des relations entre

celles-ci

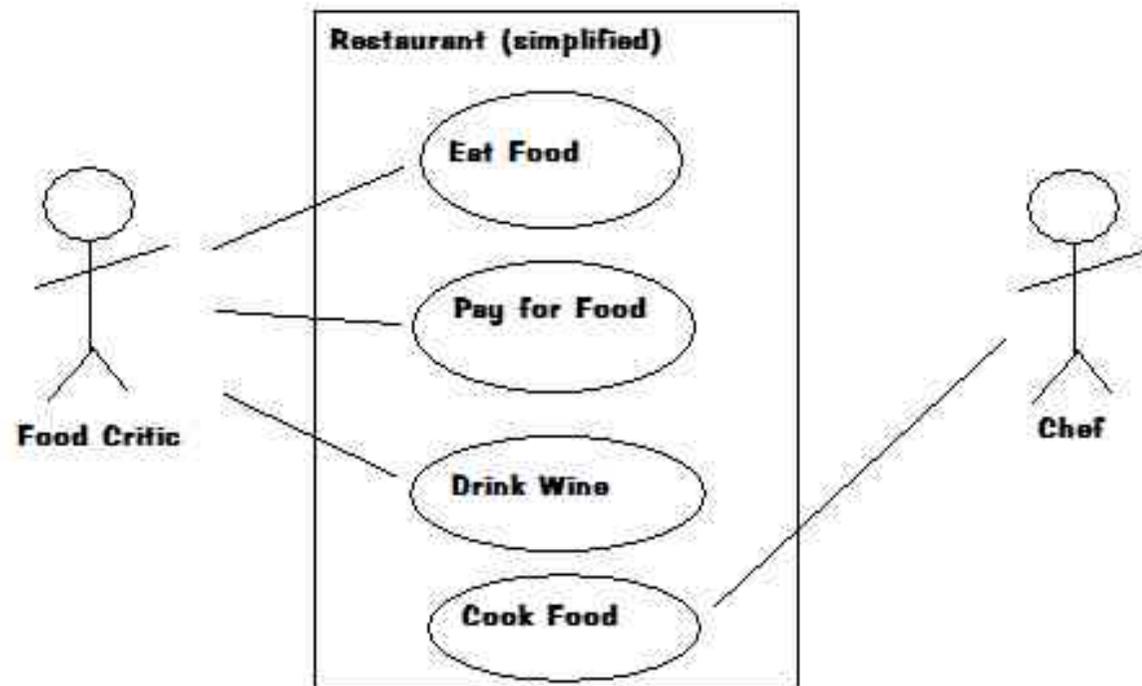


II. Adopter une méthode de développement

Unified Modeling Language (UML)

Les diagrammes de cas d'utilisation :

Vision globale du comportement fonctionnel d'un logiciel

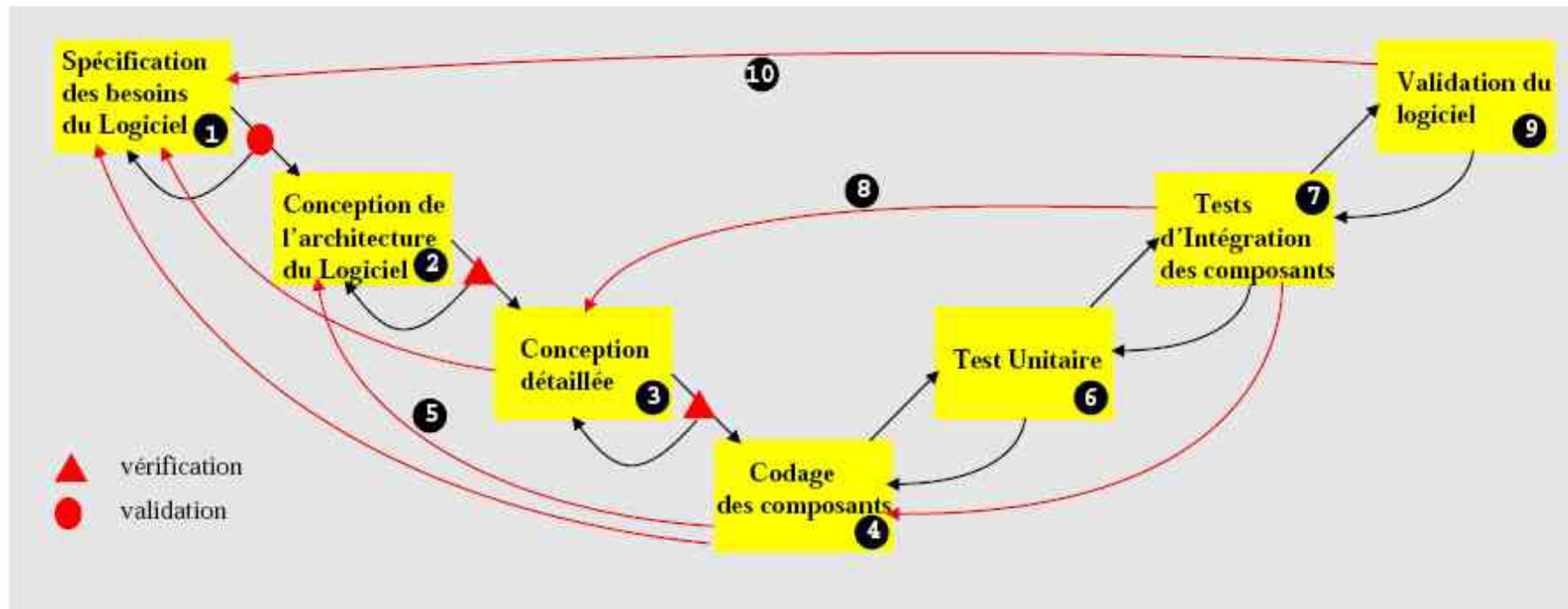


II. Adopter une méthode de développement

Cycle en V

Vision en un bloc

En réalité, le cycle de développement se déroule ainsi :



II. Adopter une méthode de développement

Extreme Programming

Agile Manifesto, signé par 17 personnalités du Génie Logiciel en 2001 :

- Un processus de développement
- Un état d'esprit
- Un ensemble de bonnes pratiques

→ Adapté aux collaborations ingénieur / chercheur pour lesquels les besoins sont fluctuants

II. Adopter une méthode de développement

Extreme Programming

Développement piloté par les tests

Règles de codage

Conception simple

Livraisons fréquentes

Travail en binôme

Metaphore

Intégration continue

Remaniement

Rythme durable

Client sur site

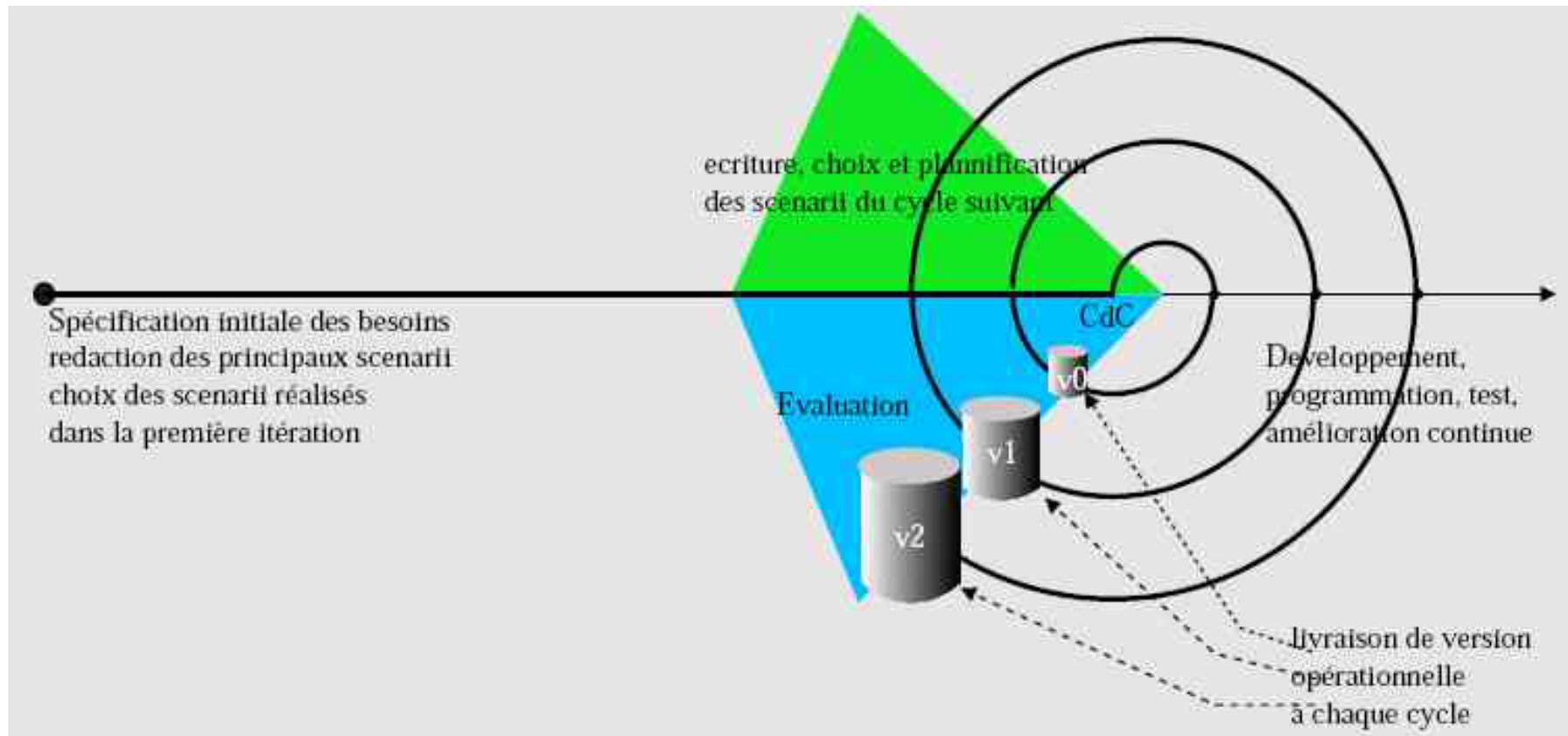
Responsabilité collective du code

Planification itérative



II. Adopter une méthode de développement

Extreme Programming



II. Adopter une méthode de développement

Extreme Programming : Intégration Continue

Vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

Prérequis:

- partage du code source via un gestionnaire de version
- intégration quotidienne des modifications par les développeurs
- développement de tests d'intégration de l'application

Avantages :

- le test immédiat des unités modifiées
- la prévention rapide en cas de code incompatible ou manquant
- les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de dernière minute
- une version est toujours disponible pour un test, une démonstration ou une distribution

II. Adopter une méthode de développement

Extreme Programming : Intégration Continue

Exemple : Jenkins
(ex Hudson),
serveur d'intégration
continue pour Java

The screenshot displays the Hudson web interface. On the left, there are navigation links: [New Job](#), [Configure](#), and [Reload Config](#). Below these are two tables:

Build Queue	
hudson	⊘
jaxb-ri	⊘

Build Executor Status	
No.	Status
1	Idle
2	Idle
3	Building javanet-maven-repository-daemon #826 ⊘
4	Building jaxb-ri #3181 ⊘
5	Building glassfish #105 ⊘
6	Idle

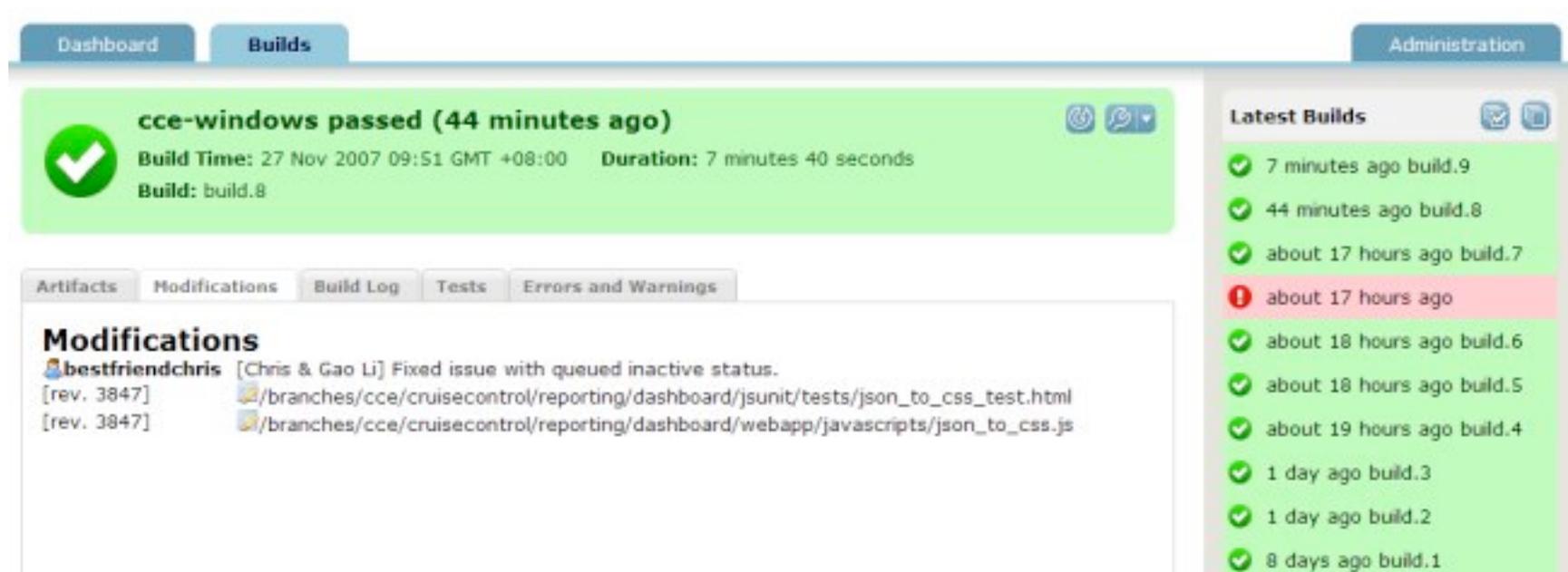
The main part of the interface shows a table of jobs:

Job	Last Success	Last Failure	Last Duration
Common annotations	4 days (#16)	9 months (#3)	39 seconds
bsh	6 months (#11)	10 months (#2)	59 seconds
dtd-parser	6 months (#8)	N/A	1 minute
fi	28 days (#586)	1 month (#567)	7 minutes
fi (weekly)	6 days (#53)	13 days (#52)	5 minutes
glassfish	4 hours (#104)	1 day (#88)	1 hour
hudson	4 minutes (#201)	N/A	1 minute
istack-commons	12 days (#19)	16 days (#5)	14 seconds
iapex	3 days (#55)	9 hours (#64)	1 minute
java-ws-xml community discussion updater	4 minutes (#16146)	10 hours (#16125)	1 minute
java.net acl processor	18 hours (#162)	N/A	0 seconds

II. Adopter une méthode de développement

Extreme Programming : Intégration Continue

Exemple : CruiseControl, logiciel d'intégration continue multilingue



The screenshot displays the CruiseControl web interface. At the top, there are navigation tabs for 'Dashboard', 'Builds', and 'Administration'. The main content area features a large green notification box with a checkmark icon, stating 'cce-windows passed (44 minutes ago)'. Below this, it provides details: 'Build Time: 27 Nov 2007 09:51 GMT +08:00', 'Duration: 7 minutes 40 seconds', and 'Build: build.8'. To the right, a 'Latest Builds' sidebar lists recent builds with their status (green checkmark for success, red exclamation mark for failure) and timestamps. Below the notification, there are tabs for 'Artifacts', 'Modifications', 'Build Log', 'Tests', and 'Errors and Warnings'. The 'Modifications' tab is active, showing a commit by 'bestfriendchris' with the message '[Chris & Gao Li] Fixed issue with queued inactive status.' and two file paths: '/branches/cce/cruisecontrol/reporting/dashboard/jsunit/tests/json_to_css_test.html' and '/branches/cce/cruisecontrol/reporting/dashboard/webapp/javascripts/json_to_css.js'.

II. Adopter une méthode de développement

Extreme Programming : Intégration Continue

Exemple : CDash, open-source
serveur de test
avec interface web

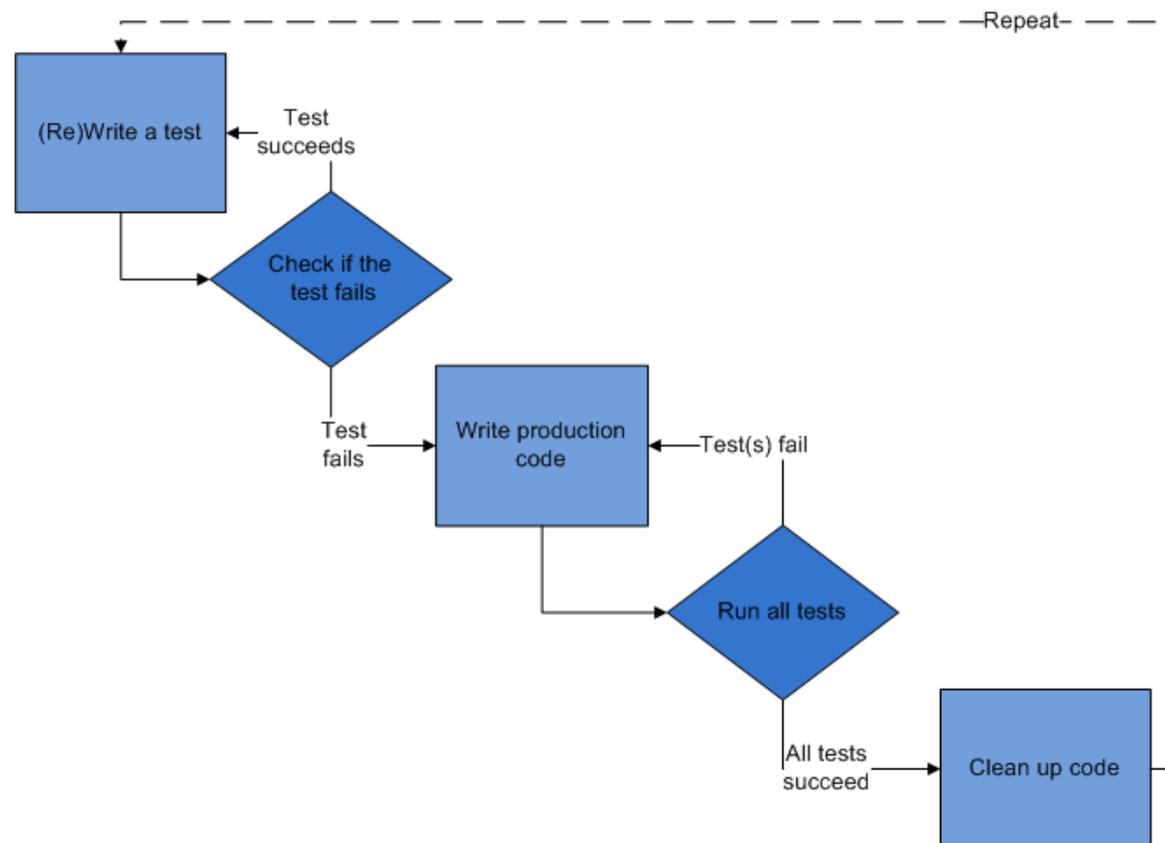


The screenshot shows the CMAKE Dashboard web interface. The top navigation bar includes links for **DASHBOARD**, **CALENDAR**, **PREVIOUS**, **CURRENT**, **NEXT**, and **PROJECT**. Below the navigation bar, there is a table of build and test results. The table is organized into two main sections: **Style** and **Nightly Expected**. Each section contains a table with columns for **Site**, **Build Name**, **Update**, **Qty**, **Build** (with sub-columns for **Error**, **Warn**, **Min**), **Test** (with sub-columns for **NotRun**, **Fail**, **Pass**, **Min**), and **Build Time**. The **Build** and **Test** columns use color coding: green for success, red for failure, and orange for warnings.

Site	Build Name	Update	Qty	Build			Test			Build Time
				Error	Warn	Min	NotRun	Fail	Pass	
cmake.com	cmake-2.8.10	28	1	0	0	0	0	0	0	2008-06-18 02:11:00 EDT
Nightly Expected										
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 01:05:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 00:01:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 10:04:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-17 22:42:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-17 21:08:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 00:51:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 00:29:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 01:15:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 01:12:00 EDT
cmake.com	cmake-2.8.10	28	0	0	0	0	0	0	0	2008-06-18 00:58:00 EDT

II. Adopter une méthode de développement

Extreme Programming : Développement piloté par les tests



III. Coder

I. Choisir un langage (Vincent)

II. Adopter une méthode de développement

III. Coder

IV. Partager, travailler en équipe

V. Profiling

VI. Débugger

III. Coder

J'ai choisi mon langage, adopté une méthode de développement... Ça y est, je peux écrire du code?

Sur une feuille? Dans un bloc-notes?

Il n'y aurait pas des outils pour coder plus efficacement?

Comment savoir ce qui existe et comment l'utiliser?

Et si je veux documenter pour les autres?

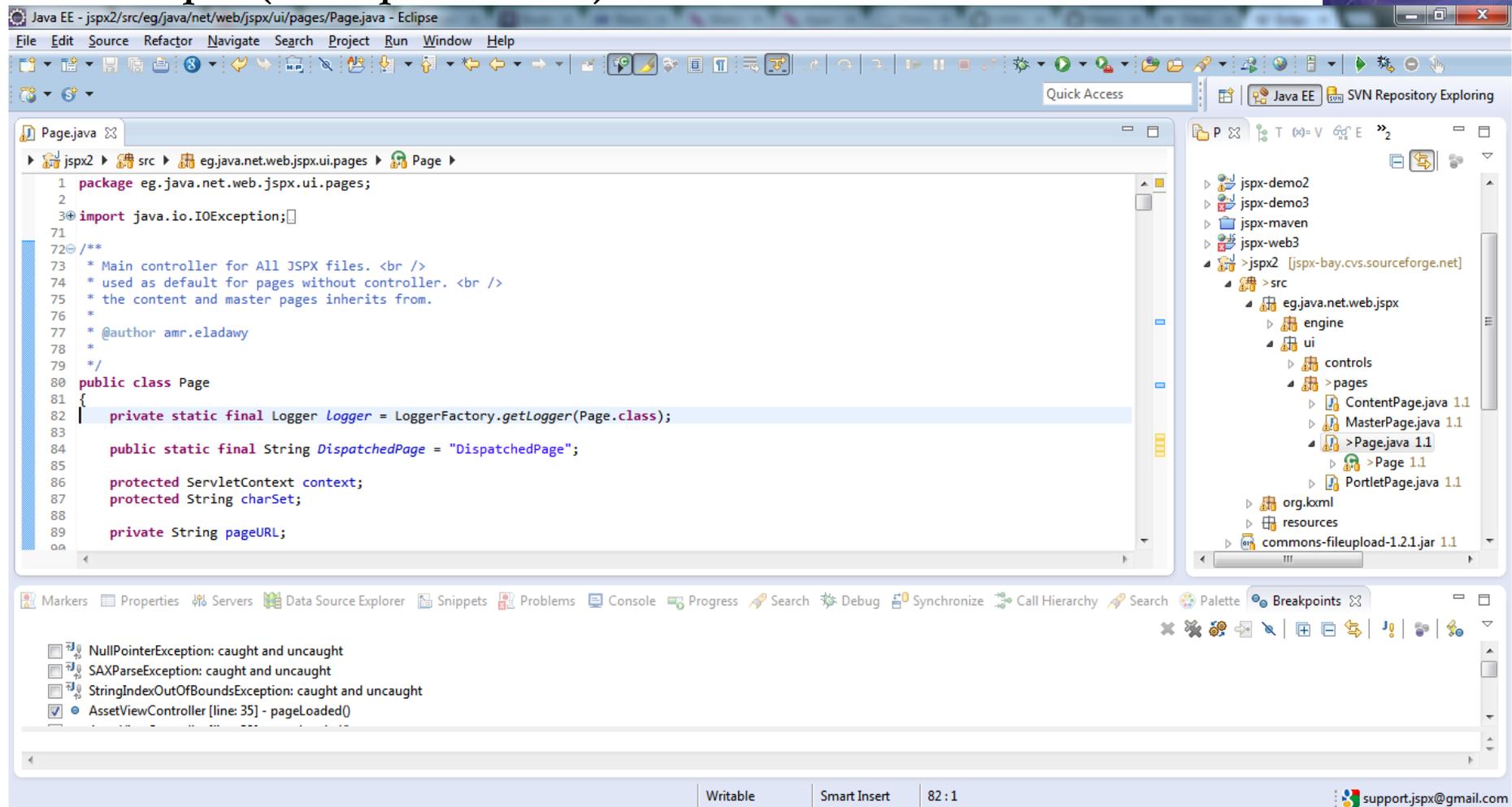
Les environnements de développement intégré (IDE)

Application fournissant

- Éditeur de code (complétion du code, surlignage)
- Compilateur et/ou interpréteur
- Debugger
- Gestion de version
- Outils de création d'IHM
- Class browser
- Indépendance par rapport à l'OS
- Pour plein de langages :
Java, C, C++, PHP, Perl, Ruby, HTML, Python, Latex... Fortran

III. Coder

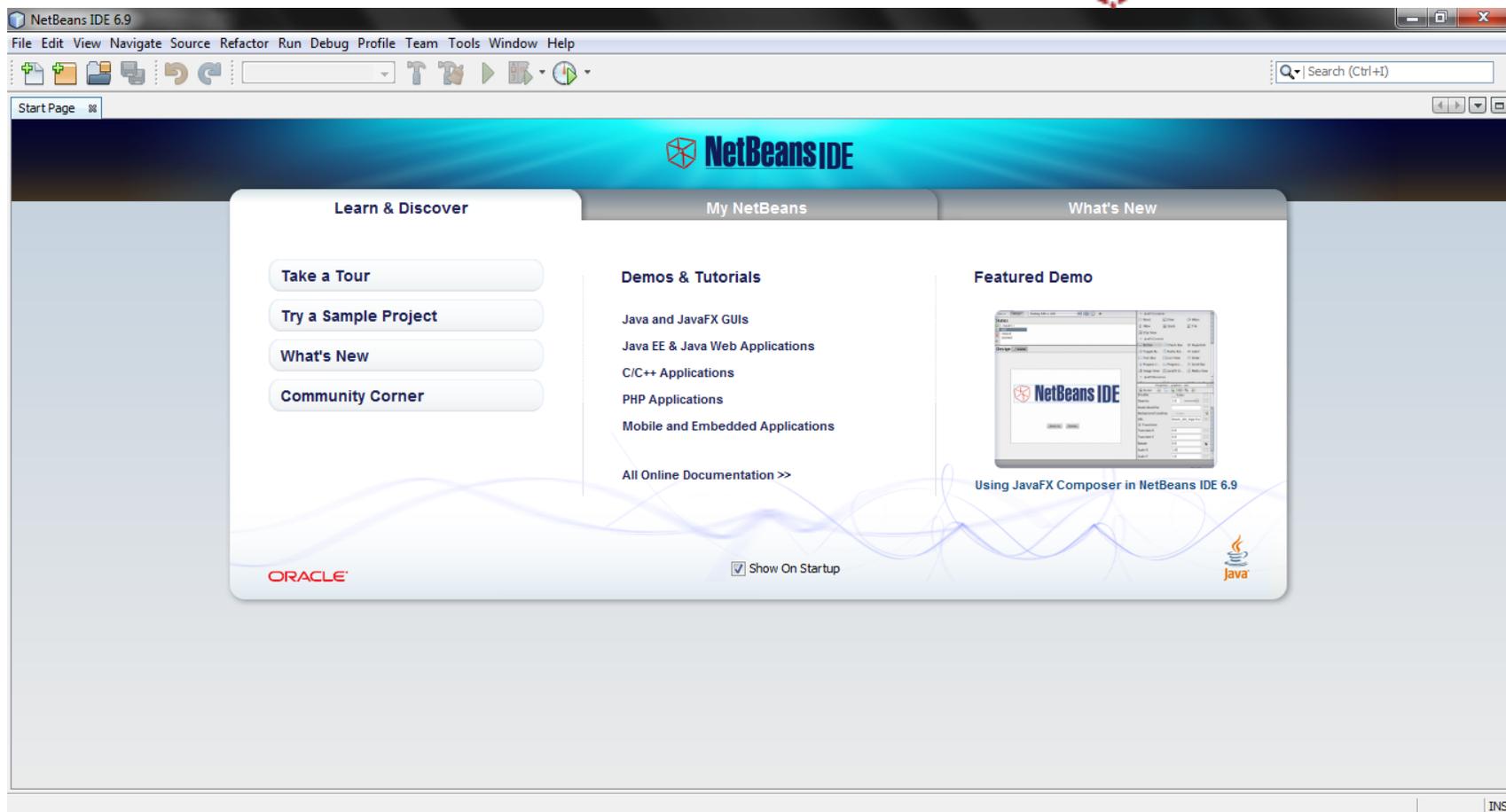
Les environnements de développement Eclipse (multi plate-forme)



III. Coder

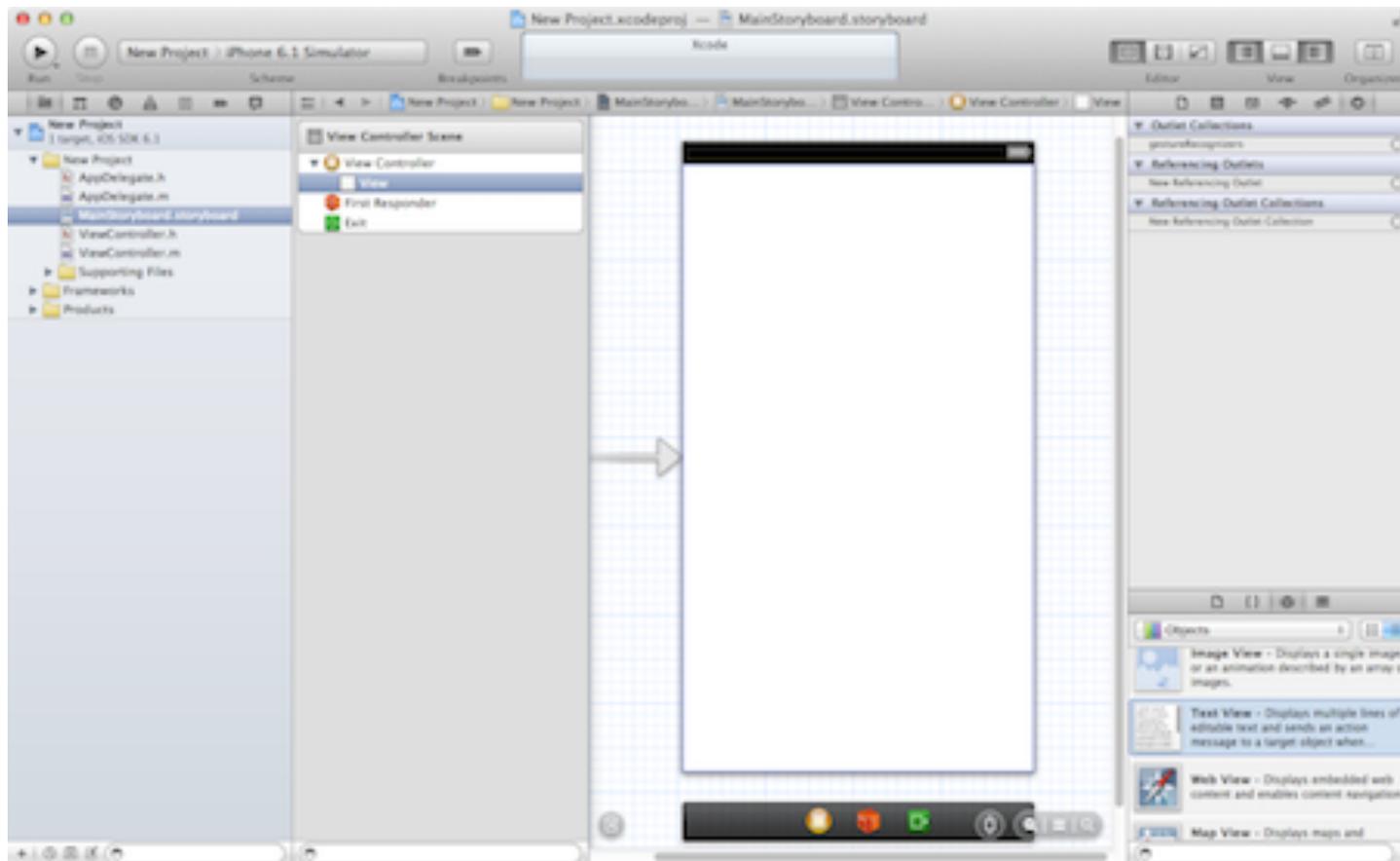
Les environnements de développement

Netbeans (Windows, Linux)



III. Coder

Les environnements de développement Xcode (Mac OS X)



III. Coder

Les environnements de développement

Visual Studio (Windows)

A screenshot of the Microsoft Visual Studio code editor. The window title is 'Solution - Microsoft Visual Studio'. The menu bar includes FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEST, SQL, TOOLS, TEST, ANALYZE, WINDOW, and HELP. The toolbar shows various icons for file operations and development. The main editor area displays C# code for a class named 'Form1'. The code includes a 'Main' method with a 'while' loop and several 'MessageBox.Show' calls. The code is as follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                MessageBox.Show("Have you tried turning it off and on again?", "Restart?", MessageBoxButtons.YesNo);

                if (MessageBox.Show("Have you booted it?", "Booted it?", MessageBoxButtons.YesNo) == DialogResult.No)
                {
                    MessageBox.Show("Single it again!");
                }
                else if (MessageBox.Show("Is it fixed?", "Fixed?", MessageBoxButtons.YesNo) == DialogResult.No)
                {
                    Console.WriteLine("I = True");
                    MessageBox.Show("Congratulations, you are now a certified computer technician!");
                }
                else
                {
                    MessageBox.Show("Single it!");
                }
            }
        }
    }
}
```

La documentation

Diverses vocations :

- Pour les développeurs comme moi (on peut générer une documentation de référence automatiquement!)
- Pour l'utilisateur final (site web avec tutoriels)
- Pour les chercheurs (articles au contenu scientifique)

III. Coder

La documentation pour l'utilisateur final

Site Web

Guide de l'utilisateur

Contenu scientifique

Verdandi, data assimilation library

<http://verdandi.sourceforge.net/index.php>

Verdandi user's guide

<http://verdandi.sourceforge.net/doc-1.5/index.php>

http://verdandi.sourceforge.net/doc-1.5/reduced_order_unscen...

Verdandi generic library for data assimila

INTRODUCTION CONTENTS DOCUMENTATION DOWNL

Verdandi is a generic C++ library for data assimilation.

Verdandi is currently developed at **INRIA**. It aims at providing methods and tools for data assimilation large class of problems involving high-dimensional numerical models.

To guarantee the highest performance, the library is implemented in C++. In addition, Verdandi provides **Swig**.

Models implemented in Fortran, C, C++, Python, ... can be plugged to Verdandi using either a C++ or I

Verdandi is provided under the **GNU Lesser General Public License (LGPL)**.

Scientific Context

Data assimilation is the process of combining different sources of information in order to better estimate extension, some parameters can also be estimated. These methods were originally introduced to deal with pertaining mostly to geophysics, but it is now widely recognized that they have a tremendous potential in e.g. heart example below).

Whether the system be biological, environmental, mechanical, etc., the main sources of information are observations and error statistics. Data assimilation methods can be written independently of the system method can be applied to a wide class of systems. Therefore methods are generic and can be put together

What is Verdandi for?

What is Verdandi designed for?

- to provide data assimilation methods to non-specialists;
- to facilitate the application of methods to a great number of problems;
- to provide a framework for perennial development;
- to improve the diffusion and impact of data assimilation algorithms.

Who can be a Verdandi user?

- non-specialists, engineers or researchers, who could directly use the available data assimilation i
- a specialist taking advantage of a modular framework, which should ease development, transfers

The users provide the numerical model and the observations with the appropriate interface.

Acknowledgment



The development of Verdandi is financially supported by the European research and innovation program Horizon 2020 at developing, sharing and integrating patient-specific multi-physics and multi-scale models in normal and pathological conditions to address clinical challenges. Data assimilation is thus intended to allow the personalization of the biophysical

1 sur 2

29/08/13 23:21

Verdandi generic library for data assimila

INTRODUCTION CONTENTS DOCUMENTATION DOWNL
PREV. VERSIONS VERSION 1.3 VERSION 1.4 VERSIO

This documentation is also available for [download in PDF format](#).

USER'S GUIDE

Introduction

Getting Started

Dependencies

Assimilation Methods

Example Models

Observations

Tools

Plugging in Verdandi

Debugging

Python Interface

API REFERENCE

Classes

Class List

Class Hierarchy

Class Members

Functions

Search for

Support

Verdandi User's G

Verdandi is a generic C++ library for data assimilation.

Verdandi is currently developed at **INRIA**. It aims at providing methods and tools for data assimilation. It is designed to be relevant to a large class of problem numerical models.

To guarantee the highest performance, the library is implemented in C++. In addition, Verdandi provides **Swig**.

Models implemented in Fortran, C, C++, Python, ... can be plugged to Verdandi using either a C++ or I Python interface.

Verdandi is provided under the **GNU Lesser General Public License (LGPL)**.

Scientific context

Data assimilation is the process of combining different sources of information in order to better estimate the state of a system. By extension, some parameters can also be estimated. These methods were originally introduced to deal with uncertainties present in most geophysics, but it is now widely recognized that they have a tremendous potential in e.g. heart example below).

Whether the system be biological, environmental, mechanical, etc., the main sources of information are observations and error statistics. Data assimilation methods can be written independently of the system to which they are applied, and can be put together to deal with a wide class of systems. Therefore methods are generic and can be put together

What is Verdandi for

What is Verdandi designed for?

- to provide data assimilation methods to non-specialists;
- to facilitate the application of methods to a great number of problems;
- to provide a framework for perennial development;
- to improve the diffusion and impact of data assimilation algorithms.

Who can be a Verdandi user?

- non-specialists, engineers or researchers, who could directly use the available data assimilation i
- a specialist taking advantage of a modular framework, which should ease development, transfers and interactions.

Verdandi user's guide

Functions

Search for

Support

3. Optionally initialize a step with **InitializeStep()**. This

4. Perform a step forward and propagate the state error variance

5. Compute the analysis with **Analyze()**, whenever observations

6. Compute the data assimilation until the model has finished: if the simulation is done, false otherwise.

Reduced Order Unscented Kalman

Assuming that P is of reduced rank p – typically much smaller than the dimension of the state – a basic idea in reduced-order filtering is, in essence, to be able to manipulate a factorized form

$$P = LU^{-1}L^T,$$

where U – in the group of invertible matrices \mathcal{GL}_p – is of much smaller rank than P and represents the main uncertainties in the system. What is crucial here is that computations on L and U without needing to compute P .

Simplex case

In this section, we focus on the simplex distribution. Consider some $(V^{(i)})_{1 \leq i \leq r} \in \mathbb{R}^p$ associated with some coefficients $(\alpha) = (\alpha_i)_{1 \leq i \leq r}$ matrix of these sigma-points denoted by $[V^*] \in \mathcal{M}_{r,p}$ and the mean $D_\alpha = \text{diag}(\alpha_1 \dots \alpha_r) \in \mathcal{M}_r$.

1. Sampling:

$$C_h = \sqrt{U_{h+1}^{-1}},$$

$$x_h^{(i)\alpha} = x_h^\alpha + L_h C_h I^{(i)}, \quad 1 \leq i \leq p+1$$

2. Prediction:

$$x_{h+1}^\alpha = E_\alpha(\mathcal{M}_h(x_h^\alpha))$$

$$x_{h+1}^{(i)f} = \begin{cases} x_{h+1}^\alpha + [\mathcal{M}_h(x_h^\alpha)] D_\alpha [V^*]^T ([V^*] I^{(i)} \\ \text{or} \\ \mathcal{M}_h(x_h^{(i)\alpha}) \end{cases}$$

$$L_{h+1} = [x_{h+1}^{(i)f}] D_\alpha [V^*]^T \in \mathcal{M}_{n,p}$$

$$P_{h+1}^\alpha = L_{h+1} (P_h^\alpha)^{-1} L_{h+1}^T$$

3. Update:

$$y_{h+1}^{(i)} = \mathcal{H}_{h+1}(x_{h+1}^{(i)f})$$

$$\{HL\}_{h+1} = [y_{h+1}^{(i)}] D_\alpha [V^*]^T$$

$$U_{h+1} = P_{h+1}^\alpha + \{HL\}_{h+1} R_{h+1}^{-1} \{HL\}_{h+1}^T \in \mathcal{M}_n$$

$$x_{h+1}^\alpha = x_{h+1}^\alpha + L_{h+1} U_{h+1}^{-1} \{HL\}_{h+1}^T R_{h+1}^{-1} (y_{h+1} - \mathcal{H}_{h+1}(x_{h+1}^\alpha))$$

$$P_{h+1}^\alpha = L_{h+1} U_{h+1}^{-1} L_{h+1}^T$$

With:

x_h^α forecast state vector;

x_h^α analysis state vector;

29/08/13 23:28

<http://verdandi.sourceforge.net>

La documentation par et pour les développeurs

Les commentaires – concis et pertinents pour se repérer dans le code

Les diagrammes UML

La documentation de référence automatique : Doxygen, Javadoc

La documentation de référence automatique

Pourquoi générer automatiquement la documentation ?

- La seule source d'information absolument juste est le code
- La rédaction de la documentation technique est laborieuse et complexe
- Faciliter la maintenance, le développement d'un code écrit seul et surtout à plusieurs
- La documentation est écrite dans le code, et il est donc relativement facile de la tenir à jour

La documentation de référence automatique

Comment ça marche ?

- Extraction de l'information à partir du code source et d'autres données laissées à la responsabilité du développeur
- Tient compte de la syntaxe et de la structure du langage du programme ainsi que des commentaires associés

La documentation de référence automatique



The screenshot shows the Doxygen website homepage. At the top, the Doxygen logo is on the left, and the text "latest release v1.5.7.1 - last page update 19 Dec 2008" is on the right. The main heading is "Source code documentation generator tool". Below this, a paragraph describes Doxygen as a documentation system for various languages. A list of three points explains how it can help. A "Doxygen license" section is at the bottom left. On the right side, there are several navigation menus: "Home" (Manual, Mailing Lists, Mail archive, FAQ, ChangeLog, Todo / Wish List, Report bugs, Doxygen users, Articles, Other Doc Tools), "SVN" (SVN tarballs, Download, Download manual, Helper tools, RPM Packages, Debian Packages), "Help doxygen!" (Donate), "Contributors", "Preisvergleich", "Affallen", and "Javadoc Tool". At the bottom right of the page, there are sections for "Downloads", "Reference", "Community", and "Learning".

Source code documentation generator tool

Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#, and to some extent D.

It can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in $\text{E}\text{T}\text{T}\text{X}$) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can **configure** doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. You can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can even 'abuse' doxygen for creating normal documentation (as I did for this manual).

Doxygen is developed under [Linux](#) and Mac OS X, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

Doxygen license

Copyright © 1997–2008 by [Dimitri van Heesch](#).

Javadoc is a tool for generating API documentation in HTML format from doc comments in source code. It can be **downloaded** only as part of the Java 2 SDK. To see documentation generated by the Javadoc tool, go to [J2SE 1.5.0 API Documentation](#).

- **Javadoc FAQ** - This FAQ covers where to download the Javadoc tool, how to find a list of known bugs and feature requests, workarounds for known bugs, how to increase memory for Javadoc, and more.
- **Javadoc Documentation** - Enhancements, Standard Doclet, Doclet overview, Doclet and Taglet APIs - See [Javadoc 1.5](#), [Javadoc 1.4](#), [Javadoc 1.3](#), or [Javadoc 1.2](#).
- **Javadoc Reference Pages** - See [Javadoc 1.5](#), [Javadoc 1.4](#), [Javadoc 1.3](#), or [Javadoc 1.2](#) for options and examples for calling the Javadoc tool.
- **How to Write Doc Comments for Javadoc** - Sun conventions for writing documentation comments.
- **Requirements for Writing API Specifications** - Standard requirements used when writing the Java 2 Platform Specification. Covers requirements for packages, classes, interfaces, fields and methods to satisfy testable assertions.

Doclets

The standard doclet generates HTML and is built into the Javadoc tool. Other doclets that Java Software has developed are listed [here](#).

- **Doclet API** is an API provided by the Javadoc tool for use by doclets. See [Doclet Overview](#) for a basic description and simple examples. (These documents are for version 1.3 of Java 2 SDK, Standard Edition.)
- **Taglet API** is an interface provided for custom formatting the text of Javadoc tags. [Taglet Overview](#) for a basic description and simple examples. (These documents are for version 1.5 of Java 2 SDK, Standard Edition.)
- **MIF Doclet** - Want beautiful PDF? This doclet can automate the generation of API documentation in PDF by way of MIF. It also enables you to print directly to a printer. MIF is Adobe FrameMaker's interchange format.
- **DocCheck Doclet** checks doc comments in source files and generates a report listing the errors and irregularities it finds. It is part of the Sun Doc Check Utilities.
- **Exclude Doclet** is a simple wrapper program that enables you to exclude from the generated documentation any public or protected classes (or packages) that you specify. It takes a list of classes in a file and removes them from the RootDoc before delegating execution to the standard doclet.
- **Doclet Toolkit** is an API and implementation for doclets emulating the standard doclet. We plan to release this toolkit hopefully during 2004. More information will be available here and through the Javadoc announce email listed below.

III. Coder

Génération via le code

```

//| A test class.
//|
//| A more elaborate class description.
//|
class Test
{
public:

//| An enum.
//| More detailed enum description. */
enum TEnum {
    TVal1, /*|< Enum value TVal1. */
    TVal2, /*|< Enum value TVal2. */
    TVal3 /*|< Enum value TVal3..*/
}

//| Enum pointer.
//| Details. */
*enumPtr,
//| Enum variable.
//| Details. */
enumVar;

//| A constructor.
//|
//| A more elaborate description of the constructor.
//|
Test();

//| A destructor.
//|
//| A more elaborate description of the destructor.
//|
~Test();

//| A normal member taking two arguments and returning an integer value.
//|
//| \param a an integer argument.
//| \param s a constant character pointer.
//| \return The test results
//| \sa Test(), ~Test(), testMeToo() and publicVar()
//|
int testMe(int a,const char *s);

//| A pure virtual member.
//|
//| \sa testMe()
//| \param c1 the first argument.
//| \param c2 the second argument.
//|
virtual void testMeToo(char c1,char c2) = 0;

//| A public variable.
//|
//| Details.
//|
int publicVar;

//| A function variable.
//|
//| Details.
//|
int (*handler)(int a,int b);
};
    
```

Utilisation



Test Class Reference

A test class. More...

List of all members.

Public Types

enum **TEnum** { **TVal1**, **TVal2**, **TVal3** }
An enum. More...

Public Member Functions

Test ()
A constructor.

~Test ()
A destructor.

int **testMe** (int a, const char *s)
A normal member taking two arguments and returning an integer value.

virtual void **testMeToo** (char c1, char c2)=0
A pure virtual member.

Public Attributes

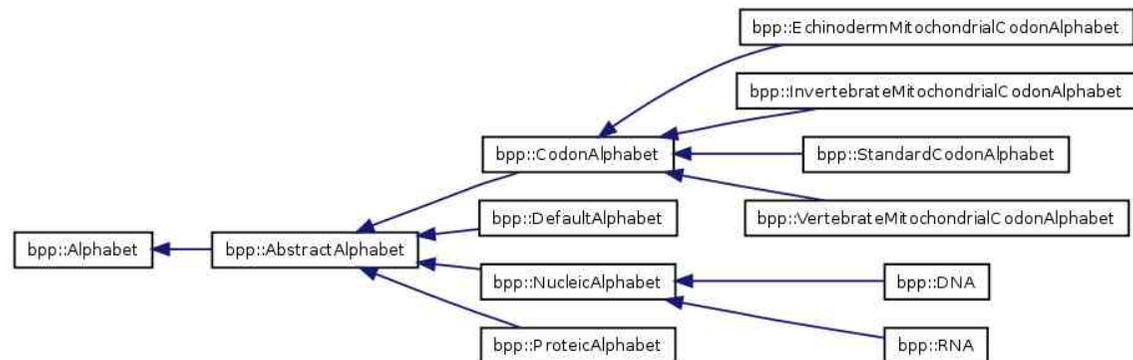
Bio++ Class documentation

Search for

SeqLib Graphical Class Hierarchy

[Go to the textual class hierarchy](#)

hpp::AbstractAlphabet::sletter



A more elaborate description of the constructor.

IV. Partager, travailler en équipe

I. Choisir un langage (Vincent)

II. Adopter une méthode de développement

III. Coder

IV. Partager, travailler en équipe

V. Profiling

VI. Débugger

IV. Partager, travailler en équipe

Ça compile, ça tourne, et le résultat est correct!

Des collègues veulent utiliser mon programme, certains veulent même contribuer en développant...

Comment leur donner la main?

Et du point de vue légal?

IV. Partager, travailler en équipe

Licences

Deux aspects :

- Que puis-je faire avec un programme selon sa licence ?
- Dois-je utiliser une licence pour mes programmes ?
Si oui, pourquoi?
Et laquelle ?

IV. Partager, travailler en équipe

Licences

- Le logiciel est couvert par le droit d'auteur
- Le logiciel ne se limite pas au code source, mais comprend aussi documents d'analyse fonctionnelle, de conception technique, maquette, prototype, aide en ligne...
- Quelle est la licence d'un logiciel? Elle doit être spécifiée dans les sources, et dans un fichier (par exemple LICENSE.TXT)
- Pourquoi utiliser une licence?
 - Sans licence, personne ne peut utiliser un programme.
 - Contrôler l'utilisation et la diffusion du programme.

IV. Partager, travailler en équipe

Licences

Qui détient le copyright du logiciel ?

- **Sur commande** : Réglé dans le contrat entre le commanditaire et le prestataire.
- **Salarié** : Droits d'exploitation transmis de plein droit à l'employeur
Le laboratoire n'a pas les droits. Seul le CNRS ou l'université ont les droits.
- **Temps libre** : Droits d'exploitation et droits moraux reviennent à l'auteur.
- **Stagiaires** : Voir la convention de stage...

IV. Partager, travailler en équipe

Licences de logiciel libre

Qu'est-ce qu'une licence de logiciels libres ?

- La liberté d'exécuter le logiciel
- La liberté d'étudier le fonctionnement du logiciel
- La liberté de redistribuer des copies du logiciel
- La liberté d'améliorer le logiciel et de publier ses améliorations

IV. Partager, travailler en équipe

Licences de logiciel libre : Typologie

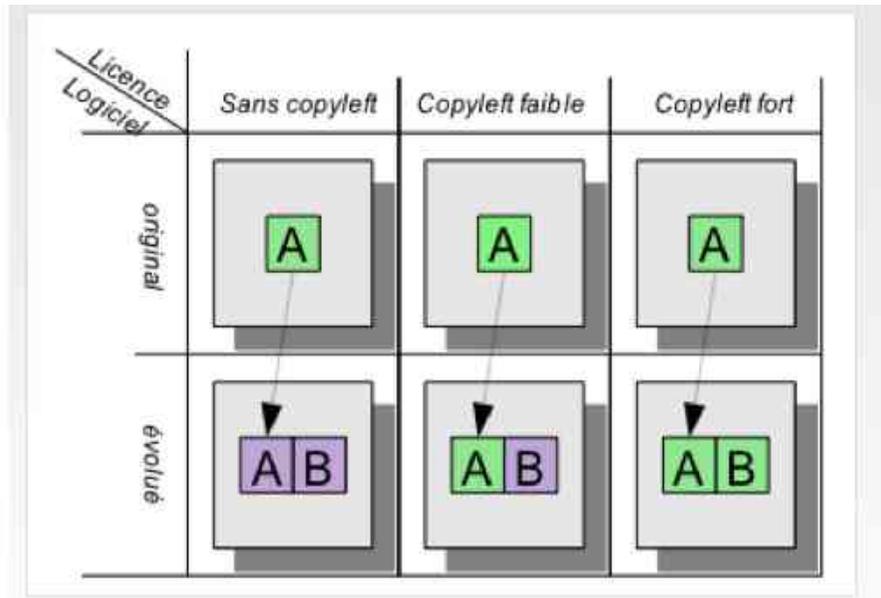
Déterminer le caractère copyleft ou non de la licence

Copyleft :

- Exigence de réciprocité : le code source (le logiciel) sera redistribué sous la même licence qu'il a été reçu
- Héritaire

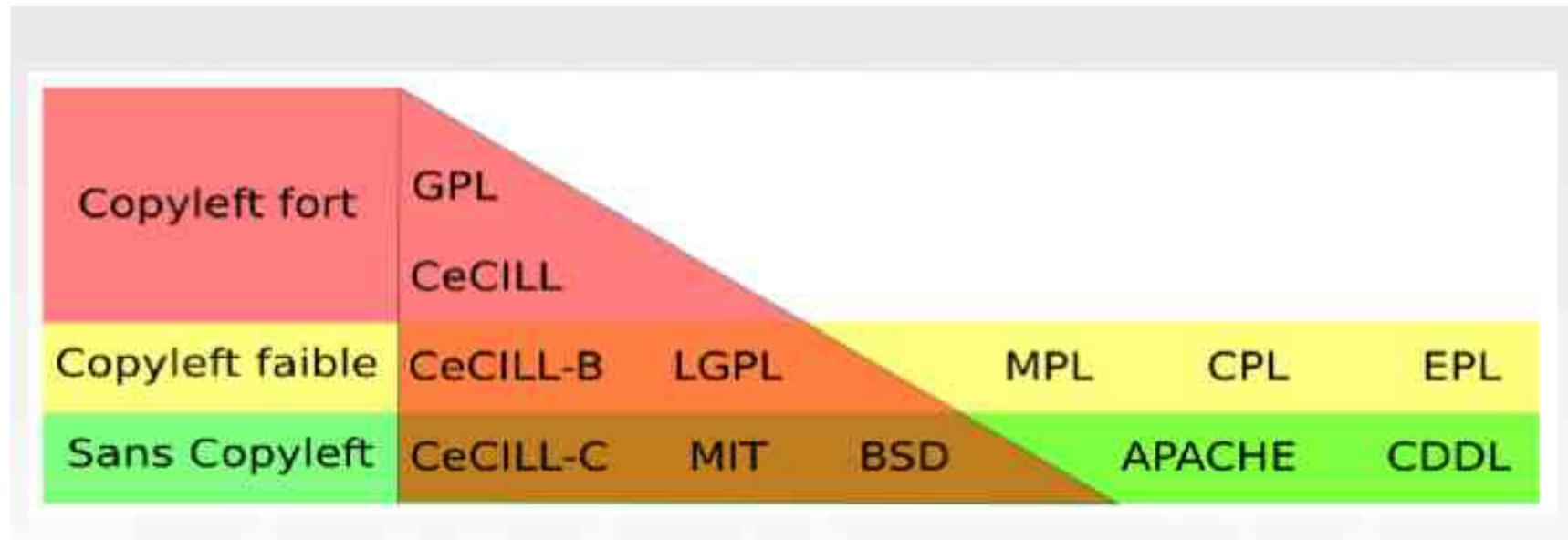
Portée du copyleft

Copyleft faible ou fort



IV. Partager, travailler en équipe

Licences de logiciel libre : Compatibilité



IV. Partager, travailler en équipe

Licences de logiciel libre : Pourquoi?

- Outils idéaux pour la préservation du patrimoine intellectuel d'un laboratoire de recherche
- Aucune renonciation à un usage ultérieur du logiciel
- Coût nul : l'ajout des mentions de licences dans chaque fichier source suffit pour bénéficier des termes de la licence
- Le dépôt des organismes de type APP (pour un coût un coût dérisoire) apporte une preuve d'antériorité
- Mutualisation de l'effort de développement
- Maximisation des possibilités d'irriguer la société

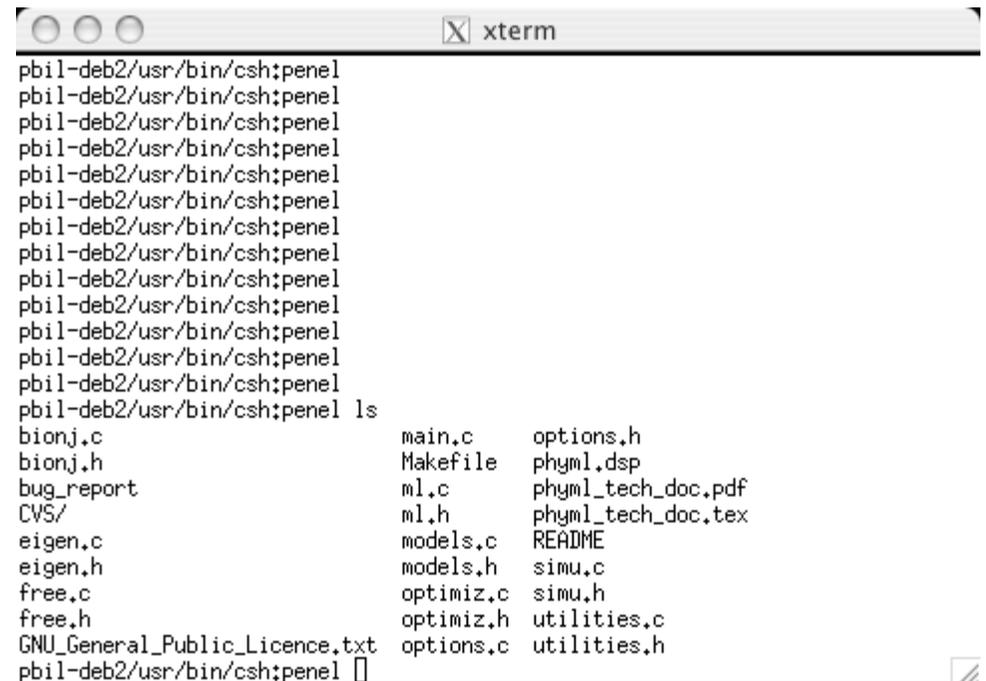
IV. Partager, travailler en équipe

Licences de logiciel libre : Comment?

Mention des auteurs et de la licence dans l'en-tête des fichiers

Licence présente dans l'arborescence

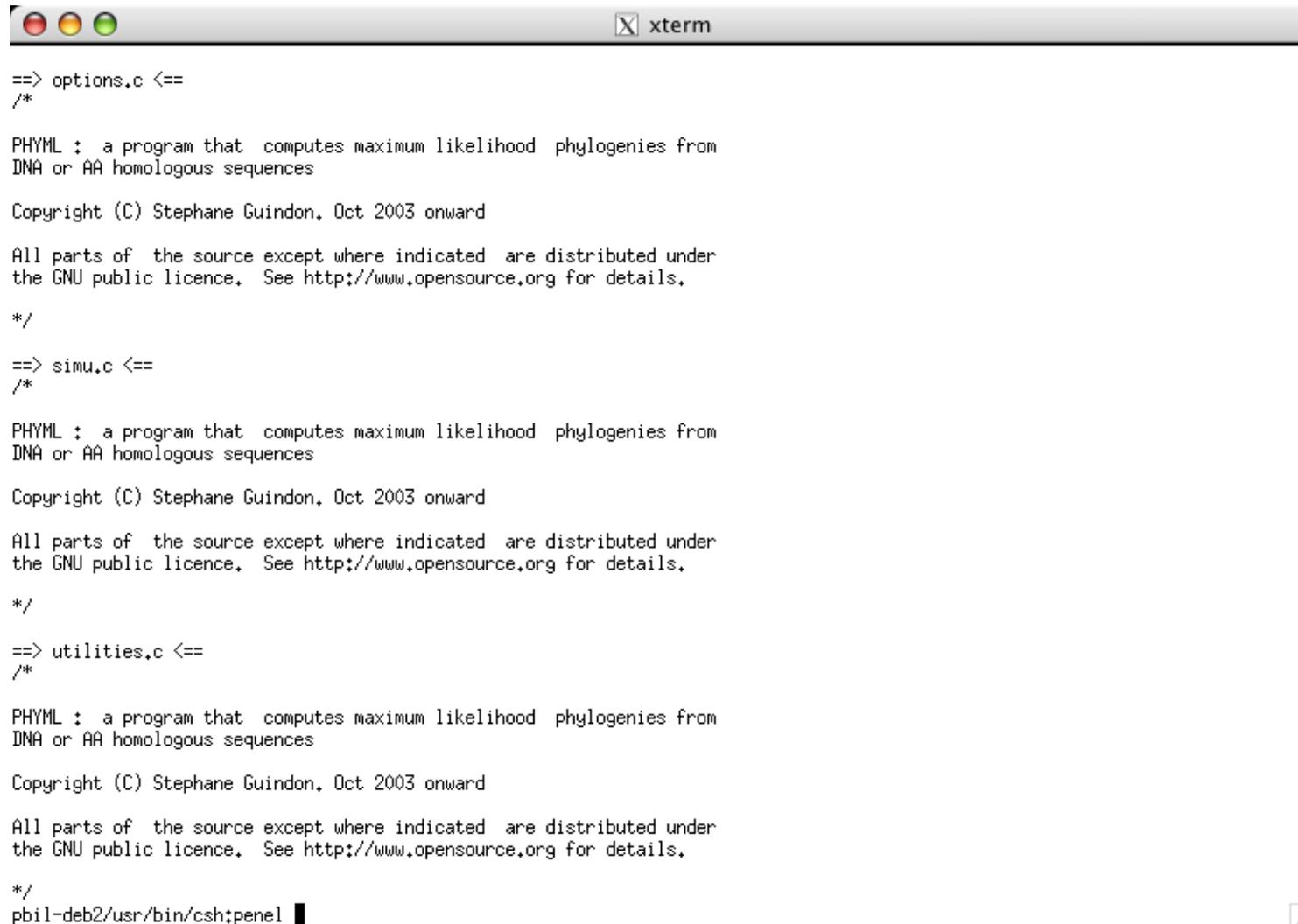
Exemple : le programme phym1



```
pbil-deb2/usr/bin/csh:penel
pbil-deb2/usr/bin/csh:penel ls
bionj.c          main.c          options.h
bionj.h          Makefile       phym1.dsp
bug_report      ml.c           phym1_tech_doc.pdf
CVS/            ml.h           phym1_tech_doc.tex
eigen.c         models.c       README
eigen.h         models.h       simu.c
free.c          optimiz.c      simu.h
free.h          optimiz.h      utilities.c
GNU_General_Public_Licence.txt options.c      utilities.h
pbil-deb2/usr/bin/csh:penel
```

IV. Partager, travailler en équipe

Licences de logiciel libre : Comment?



```
xterm

==> options.c <==
/*

PHYML : a program that computes maximum likelihood phylogenies from
DNA or AA homologous sequences

Copyright (C) Stephane Guindon, Oct 2003 onward

All parts of the source except where indicated are distributed under
the GNU public licence. See http://www.opensource.org for details.

*/

==> simu.c <==
/*

PHYML : a program that computes maximum likelihood phylogenies from
DNA or AA homologous sequences

Copyright (C) Stephane Guindon, Oct 2003 onward

All parts of the source except where indicated are distributed under
the GNU public licence. See http://www.opensource.org for details.

*/

==> utilities.c <==
/*

PHYML : a program that computes maximum likelihood phylogenies from
DNA or AA homologous sequences

Copyright (C) Stephane Guindon, Oct 2003 onward

All parts of the source except where indicated are distributed under
the GNU public licence. See http://www.opensource.org for details.

*/
pbil-deb2/usr/bin/csh:penel █
```

IV. Partager, travailler en équipe

Partager : Archive des sources

Donner accès au répertoire compressé des sources
(format tar.gz, tar.bz2, zip)

→ Permet à l'utilisateur de :

- Mieux comprendre qui se cache derrière les appels aux fonctionnalités
- Mieux les utiliser
- Eventuellement contribuer en tant que développeur
- Marche pour toutes les plateformes

IV. Partager, travailler en équipe

Partager : Packaging

Problématique :

- Installation facile : éviter à chaque utilisateur l'étape de compilation, longue (à cause des éventuelles dépendances) et source potentielle d'erreurs
- Portabilité : défi du multi-plateforme (Windows, Mac OS, Linux)

Exemples de packages :

- RPM sous Fedora
- make dist
- CPack multi-plateforme
- Distutils (Python)
- R packaging

IV. Partager, travailler en équipe

CMake

Système de construction logicielle
multi-plateforme

Génération des Makefile,
à la base du mécanisme de compilation
(gestion de plusieurs fichiers source,
édition de lien avec éventuelles dépendances,
via une commande simple)



```
cmake_minimum_required(VERSION 2.6)

#Déclaration du projet
project(MyProject)

#Génération de la liste des fichiers sources
file(
    GLOB_RECURSE
    source_files
    src/*
)

#Déclaration de l'exécutable
add_executable(
    my_exe
    ${source_files}
)
```

IV. Partager, travailler en équipe

Travailler en équipe : Gestion de version

Problématique

- Ça marche plus... je voudrais annuler ma modification!
- On est plusieurs à modifier le même document... quelle version est la dernière?

IV. Partager, travailler en équipe

Travailler en équipe : Gestion de version

Qu'est-ce que c'est ?

Activité permettant de gérer les modifications d'un ensemble de données.

Typiquement : code source d'un logiciel

Également applicable à d'autres catégories de données :

- Documentation (par exemple Latex)
- Site web
- Fichiers de configuration d'un système

IV. Partager, travailler en équipe

Travailler en équipe : Logiciels de gestion de version

Permet de :

- Travailler à plusieurs
- Garder une trace des différents stades de développement, éventuellement de revenir en arrière
- Suivre l'évolution du programme
- Développer des versions différentes du programme en parallèle

IV. Partager, travailler en équipe

Travailler en équipe : Logiciels de gestion de version

Fonctions de base :

- Un dépôt (projet + historique), des copies « locales » du projet
- Conserve un historique des modifications
- Permet de travailler à plusieurs:
fusion, gestion des conflits
- Permet les modifications en parallèle

IV. Partager, travailler en équipe

Travailler en équipe : Gestion de version

Modèle client-serveur :

Programmes :

- CVS 
- Subversion



IV. Partager, travailler en équipe

Travailler en équipe : Gestion de version

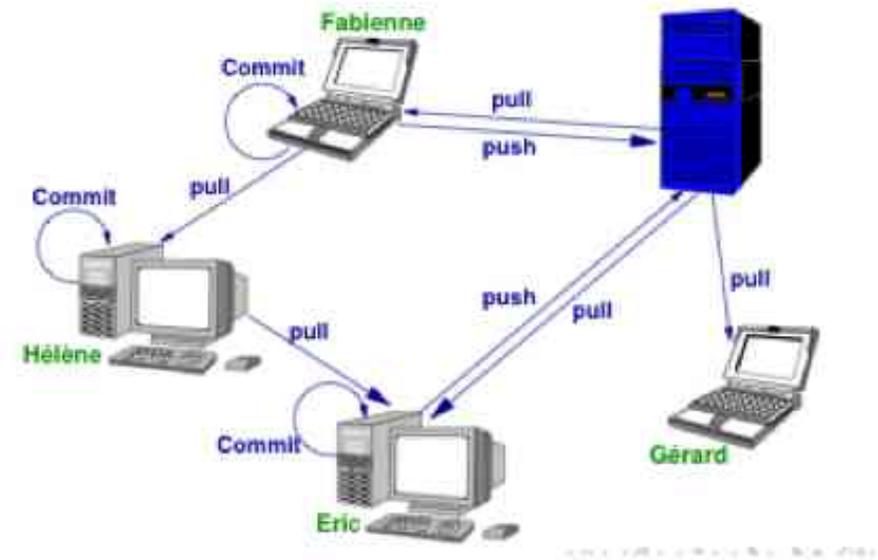
Système distribué :

Programmes :

- Git



- Bitkeeper
- Mercurial



Pas de dépôt centralisé

- Chaque développeur a sa copie avec ses branches privées
- Plus de liberté
- Opération de synchronisation des dépôts des développeurs

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Définition :

Une Forge a plusieurs facettes :

- Un portail communautaire
- Un outil de gestion de projets
- Un environnement de développement collaboratif
- Un site pour une communauté

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Contenu d'une Forge : services aux projets

Une Forge offre un ensemble d'outils permettant la gestion des projets logiciels

- Gestion des sources : CVS / SVN
- Trackers : feature requests, bug tracker, tâches
- Livraisons (fichiers, packages)
- Gestion des documents, Wiki
- Autres services aux projets
 - Forums
 - Mailing lists
 - Sondages, news
 - Administration : gestion des membres, des services associés

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemples :

<http://sourceforge.net/>

<https://github.com/>

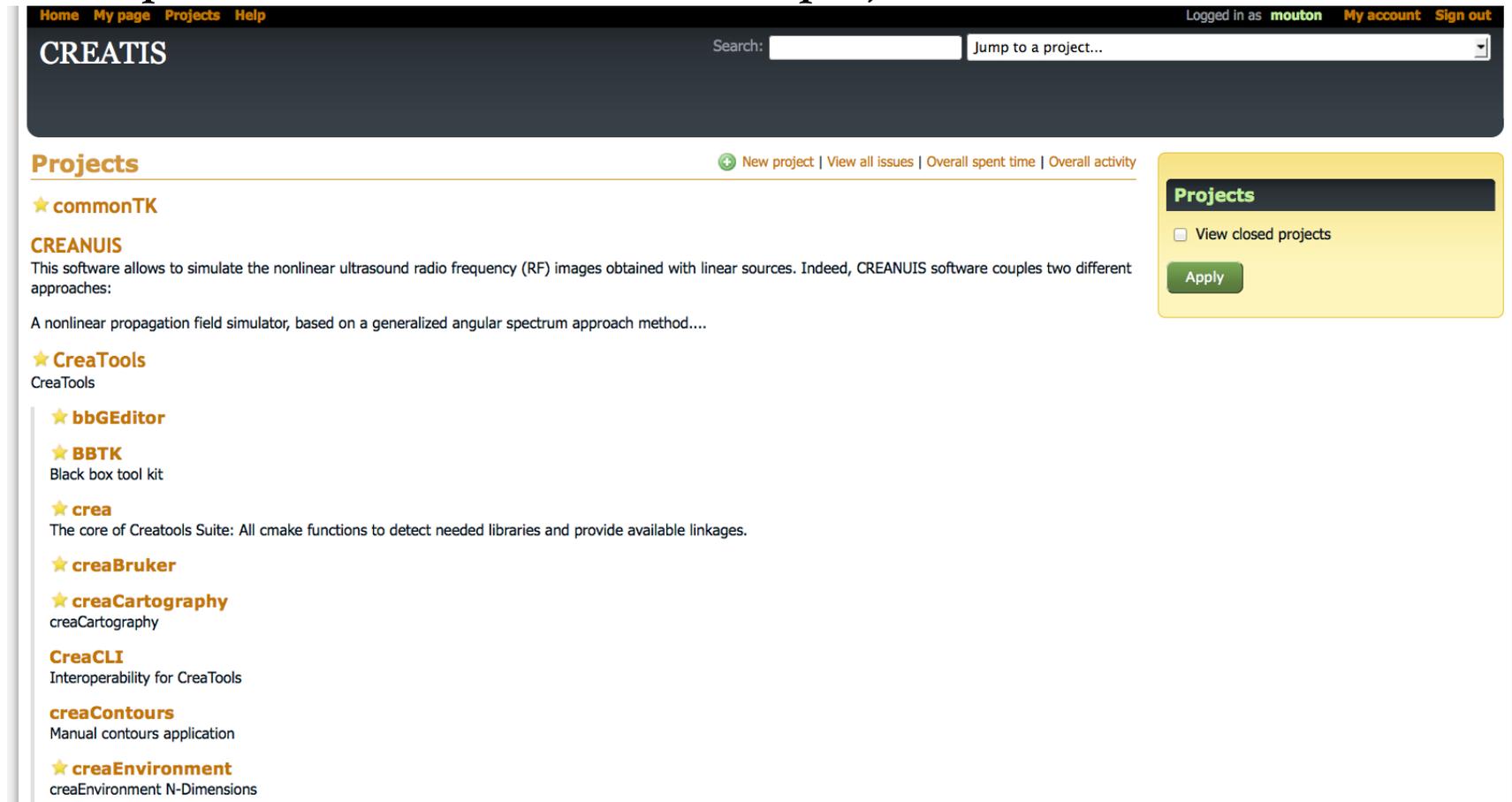
<http://gitorious.org/>

<http://vip.creatis.insa-lyon.fr:9002/projects>

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemple : Redmine à CREATIS – Les projets



The screenshot shows the CREATIS Redmine interface. At the top, there is a navigation bar with links for Home, My page, Projects, and Help. The user is logged in as 'mouton' and can access My account or Sign out. A search bar and a 'Jump to a project...' dropdown are also present. The main content area is titled 'Projects' and includes links for 'New project', 'View all issues', 'Overall spent time', and 'Overall activity'. A list of projects is displayed, each with a star icon and a brief description. The sidebar on the right contains a 'Projects' section with a checkbox for 'View closed projects' and an 'Apply' button.

Home My page Projects Help

Logged in as **mouton** My account Sign out

CREATIS Search: Jump to a project...

Projects [New project](#) | [View all issues](#) | [Overall spent time](#) | [Overall activity](#)

★ **commonTK**

★ **CREANUIS**
This software allows to simulate the nonlinear ultrasound radio frequency (RF) images obtained with linear sources. Indeed, CREANUIS software couples two different approaches:
A nonlinear propagation field simulator, based on a generalized angular spectrum approach method....

★ **CreaTools**
CreaTools

- ★ **bbGEditor**
- ★ **BBTK**
Black box tool kit
- ★ **crea**
The core of Creatools Suite: All cmake functions to detect needed libraries and provide available linkages.
- ★ **creaBruker**
- ★ **creaCartography**
creaCartography
- CreaCLI**
Interoperability for CreaTools
- creaContours**
Manual contours application
- ★ **creaEnvironment**
creaEnvironment N-Dimensions

Projects

View closed projects

Apply

<http://vip.creatis.insa-lyon.fr:9002/projects>

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemple : Redmine à CREATIS – Le projet OsiriX Plug-in >> inTag

The screenshot shows the Redmine interface for the project 'OsiriX Plug-in » inTag'. The top navigation bar includes 'Home', 'My page', 'Projects', and 'Help'. The user is logged in as 'mouton'. The search bar contains 'inTag'. The main navigation tabs are 'Overview', 'Activity', 'Roadmap', 'Issues', 'New issue', 'Gantt', 'Calendar', 'News', 'Documents', 'Wiki', 'Files', 'Repository', and 'Settings'. The 'Overview' tab is active, showing a description of the project, a list of members, an issue tracking summary, and a 'Spent time' section. The 'Members' section lists the Manager (Claire Mouton), Developer (Claire Mouton, DAVILA Eduardo, Patrick Clarysse, pierre croisille, William A. Romero R.), Reporter (Claire Mouton, DAVILA Eduardo, Patrick Clarysse, pierre croisille, William A. Romero R.), and Viewer (Claire Mouton, coralie vandroux, DAVILA Eduardo, Frédéric Cervenansky, Maciej Orkisz, Patrick Clarysse, pierre croisille, William A. Romero R.). The 'Issue tracking' section shows 2 open bugs, 4 open features, and 0 open support, research, integration, deployment, or test issues. The 'Spent time' section shows 0.00 hours. The 'Manager' section displays profile cards for Claire Mouton, DAVILA Eduardo, Maciej Orkisz, Patrick Clarysse, pierre croisille, and William A. Romero R. (Research Engineer at CNRS). A 'Download' button for 'inTag inTag v1.2-1B' is visible at the bottom right. The footer indicates the page is powered by Redmine © 2006-2013 Jean-Phillippe Lang.

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemple : Redmine à CREATIS – Le projet OsiriX Plug-in >> inTag/Issues

Home My page Projects Help Logged in as **mouton** My account Sign out

OsiriX Plug-in » inTag Search: » inTag

Overview Activity Roadmap **Issues** New issue Gantt Calendar News Documents Wiki Files Repository Settings

Issues

Filters: Status open Add filter

Options: Apply Clear Save

#	#	Project	Tracker	Status	Priority	Subject	Updated	
<input type="checkbox"/>	2012	2012	inTag	Bug	New	Normal	Ambiguity with parameter identifier in xls files	05/28/2013 04:56 pm
<input type="checkbox"/>	1962	1962	inTag	Feature	New	Normal	Alternatives Motion estimators	03/29/2013 09:55 am
<input type="checkbox"/>	1937	1937	inTag	Bug	New	Normal	Limited access to sample data	03/18/2013 05:02 pm
<input type="checkbox"/>	1936	1936	inTag	Feature	In Progress	Normal	Contour interactive correction	08/26/2013 07:11 pm
<input type="checkbox"/>	1935	1935	inTag	Test	New	Normal	RefIntag Test	03/18/2013 05:03 pm
<input type="checkbox"/>	1934	1934	inTag	Test	New	Normal	inTag Evaluation	03/18/2013 05:04 pm
<input type="checkbox"/>	1933	1933	inTag	Feature	New	Normal	Point value picking	03/18/2013 05:04 pm
<input type="checkbox"/>	1931	1931	inTag	Feature	In Progress	Normal	Strain rates, peak strain rates computation	03/18/2013 05:17 pm

(1-8/8)

Also available in: Atom | CSV | PDF

Issues
View all issues
Summary

Graphs
Open aging issues
Total issues over time
Total bugs over time
Calendar
Gantt

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemple : Redmine à CREATIS – Le projet OsiriX Plug-in >> inTag/Wiki

The screenshot shows a Redmine Wiki page for 'inTag'. The page title is '[OsiriX Plug-in] inTag WIKI'. The main content area is titled 'Content' and lists four items: 1. Software development road map, 2. Developer Guide, 3. Meetings log, and 4. Recommended links and resources. A sidebar on the left also lists these items under the heading 'Content'. A blue arrow points from the sidebar to the main content area. Below the content list is a UML diagram showing a 'User' actor and an 'inTag' use case. The 'User' actor is connected to the 'Start inTag' use case, which is connected to the 'inTag' use case. The diagram is a sequence diagram with a start node leading to a 'Start inTag' use case, which then leads to the 'inTag' use case.

IV. Partager, travailler en équipe

Travailler en équipe : Les Forges

Exemple : Redmine à CREATIS – Le projet OsiriX Plug-in >> inTag/Dépôt

The screenshot shows a Redmine interface for a repository named 'OsiriX Plug-in » inTag'. The user is logged in as 'mouton'. The interface includes a search bar, navigation tabs (Overview, Activity, Roadmap, Issues, New issue, Gantt, Calendar, News, Documents, Wiki, Files, Repository, Settings), and a file browser showing the repository structure. Below the file browser is a table of 'Latest revisions' with columns for revision number, date, author, and comment.

root @ master Statistics | Branch: | Tag: | Revision:

Name	Size
OsiriXPlugin	
wxInTag.skel	
.gitignore	162 Bytes
AxinoeLogo.xpm	23 KB
CMakeLists.txt	2.98 KB
CreatisLogo.png	4.99 KB
CreatisLogo.xpm	22.9 KB
README	10 KB
builtwithwx.png	2.7 KB

Latest revisions

#	Date	Author	Comment
a8cb985e	05/24/2012 02:18 am	Jean-Charles BERTIN	Added parameters to debug save panel.
ee1a20c5	05/24/2012 02:17 am	Jean-Charles BERTIN	Added inTag to saved parameters.
3f71b14d	05/24/2012 01:45 am	Jean-Charles BERTIN	Make wxInTag target optional.
c5f77ff4	05/24/2012 01:44 am	Jean-Charles BERTIN	Fixed compilation warning.
8214dea9	05/23/2012 08:44 pm	Jean-Charles BERTIN	Use Alternate key to access debug save panel.
153fa0ba	05/23/2012 08:30 pm	Jean-Charles BERTIN	Minor cleanup.
30343549	05/23/2012 08:27 pm	Jean-Charles BERTIN	Added Wait.h file.
37e03b5f	05/23/2012 08:25 pm	Jean-Charles BERTIN	Added slices versioning.
2d74de58	05/23/2012 08:25 pm	Jean-Charles BERTIN	First import.
83305de5	05/23/2012 08:25 pm	Jean-Charles BERTIN	Added recalculation of initial results with warping algorithm.

[View differences](#)

[View all revisions](#) | [View revisions](#)

Also available in: [Atom](#)

Powered by [Redmine](#) © 2006-2013 Jean-Philippe Lang

Fin du cours...

À vous la main!

Des questions avant de plonger dans le code?