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

Unified Modeling Language (UML)

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

Unified Modeling Language (UML)

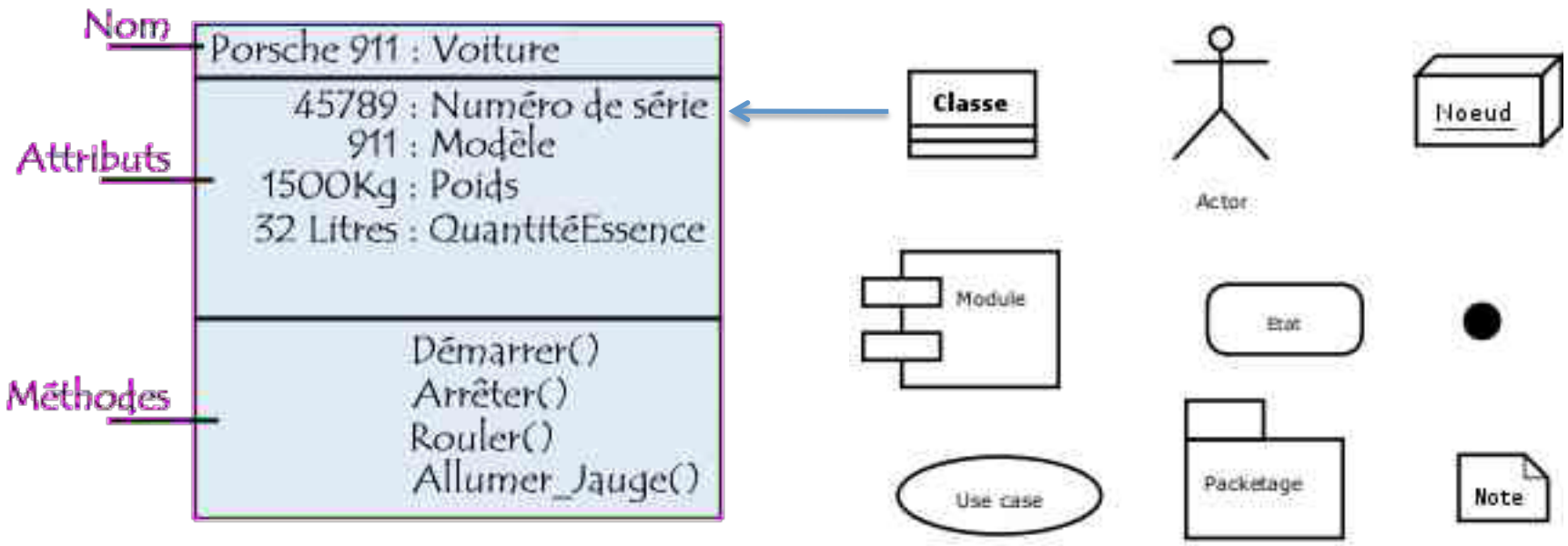
Langage de modélisation unifié

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

Unified Modeling Language (UML)

Il existe un formalisme complet:



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

Unified Modeling Language (UML)

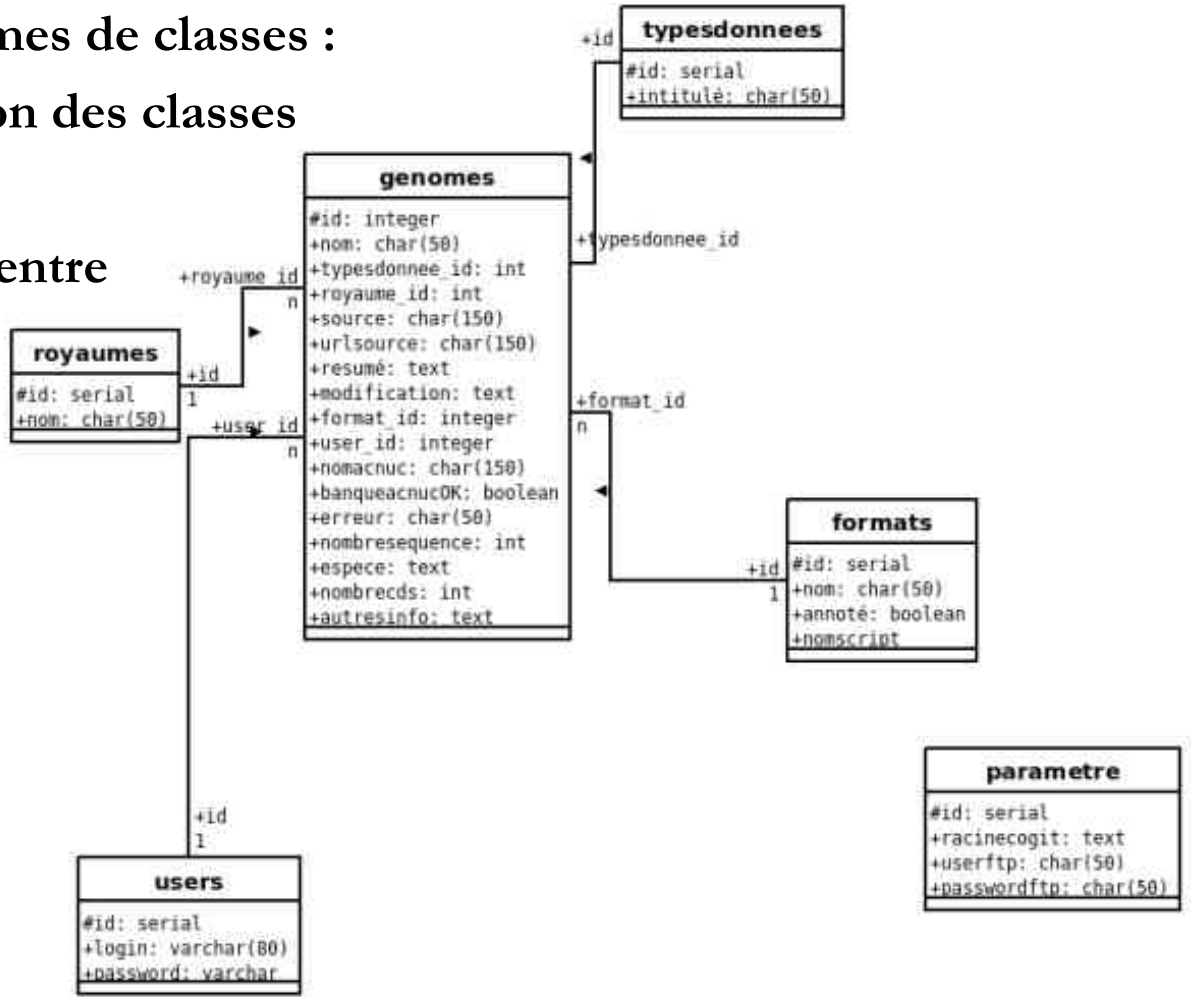
Les diagrammes de classes :

Représentation des classes

et

des relations entre

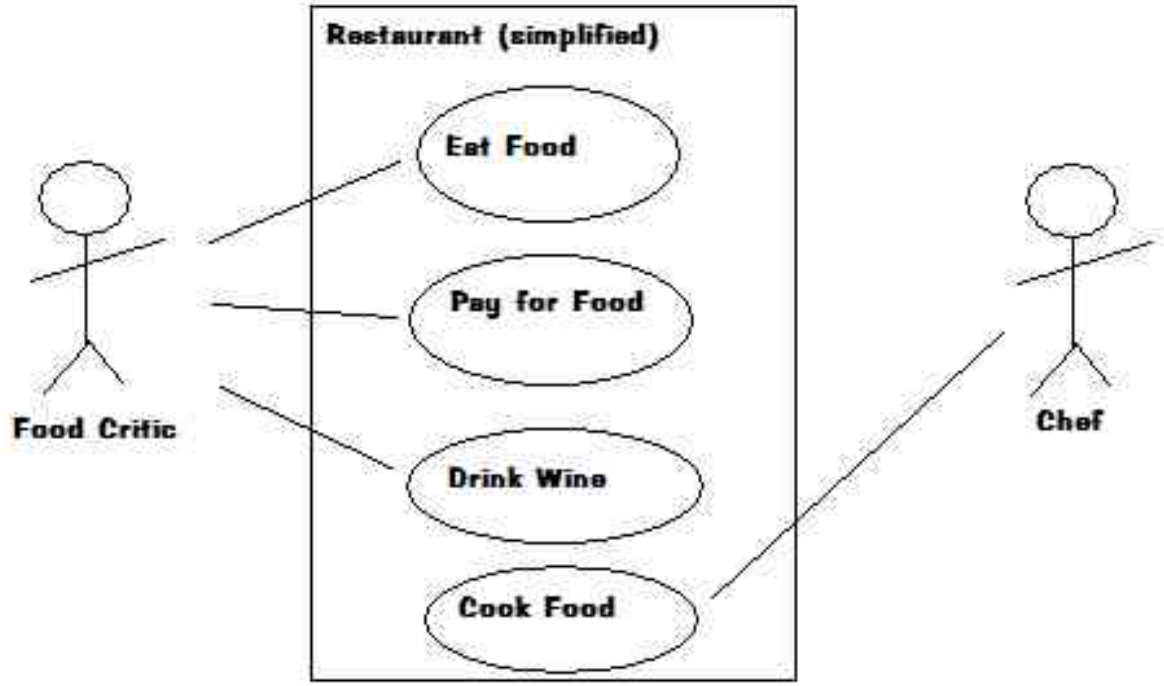
celles-ci



Unified Modeling Language (UML)

Les diagrammes de cas d'utilisation :

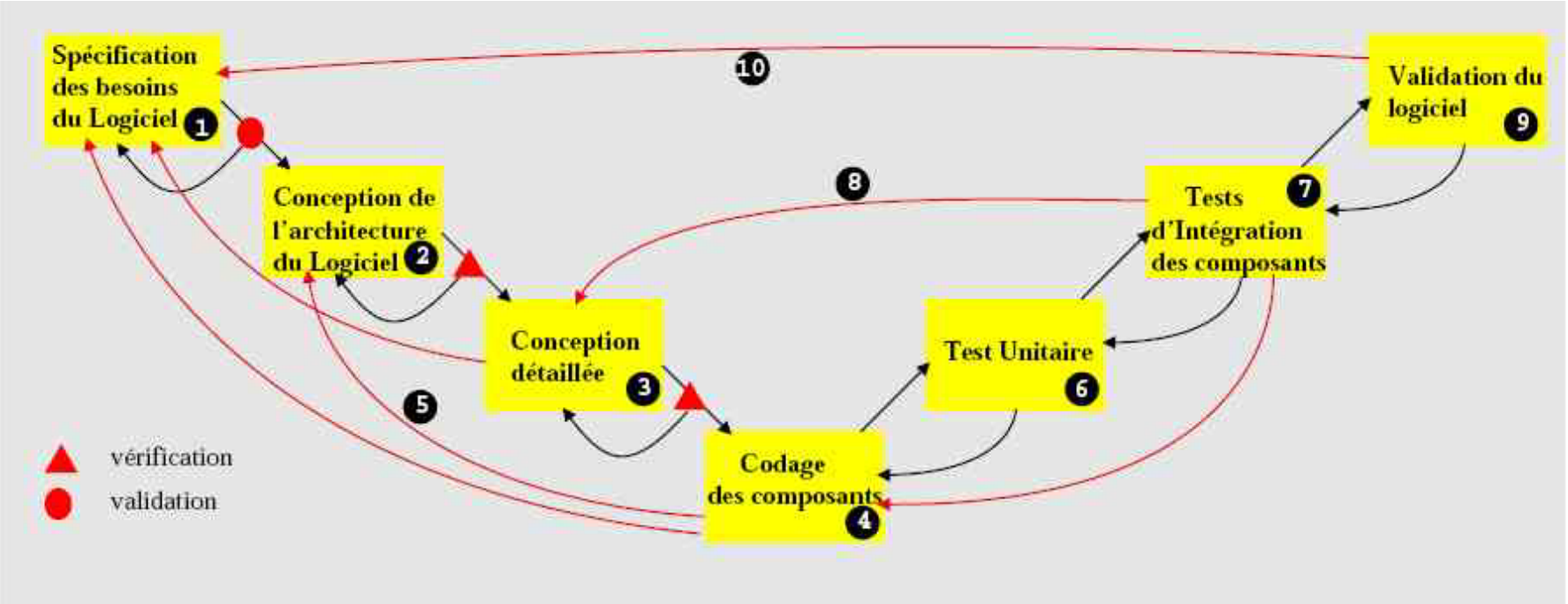
Vision globale du comportement fonctionnel d'un logiciel



Cycle en V

Vision en un bloc

En réalité, le cycle de développement se déroule ainsi :



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

Extreme Programming

Développement piloté par les tests

Règles de codage

Conception simple

Livraisons frequentes

Travail en binome

Metaphore

Integration continue

Remaniement

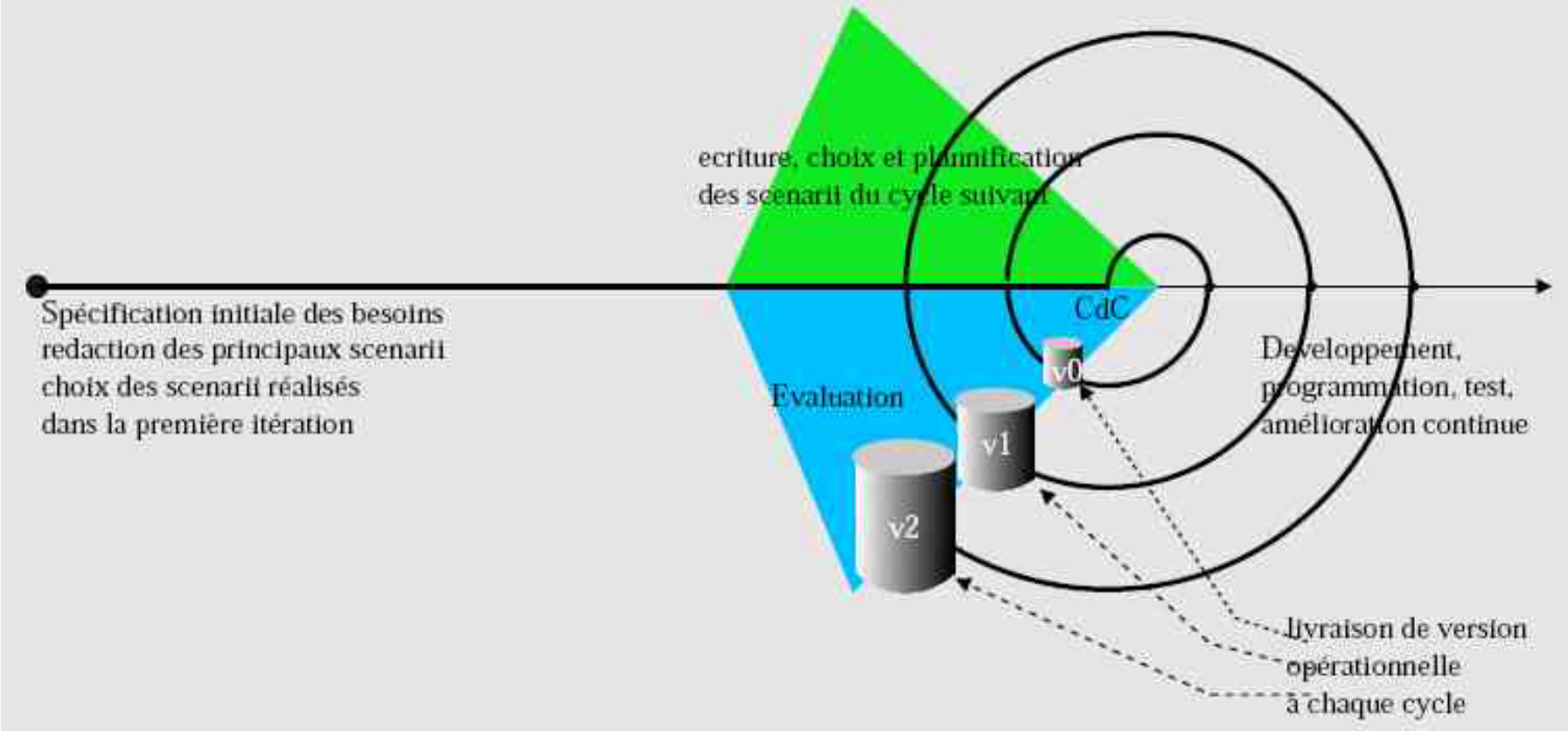
Rythme durable

Client sur site

Responsabilite collective du code

Planification iterative

Extreme Programming



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:

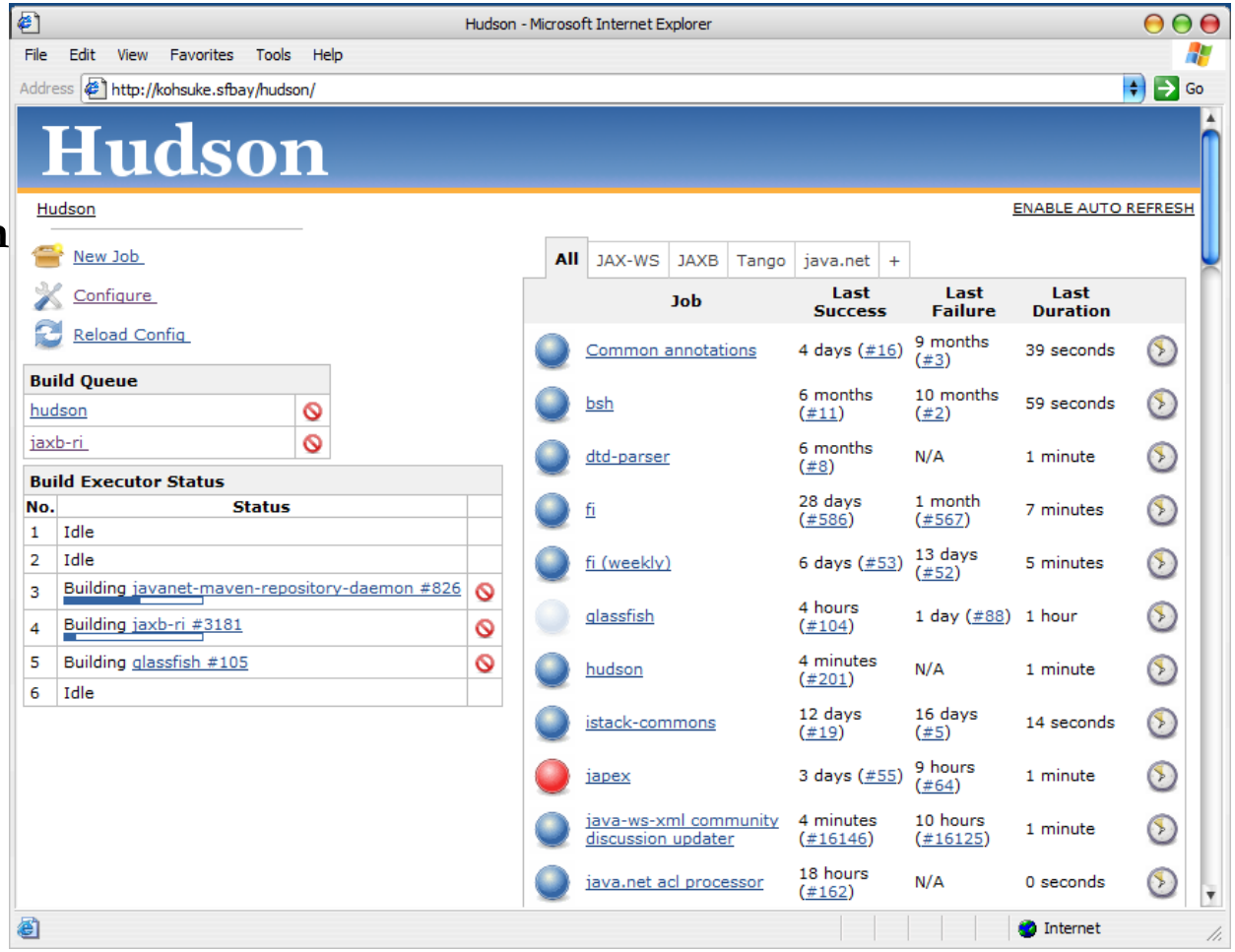
- partage du code source via un gestionnaire de version
- intégration quotidienne des modifications par les développeurs
- développement de tests d'intégration de l'application

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

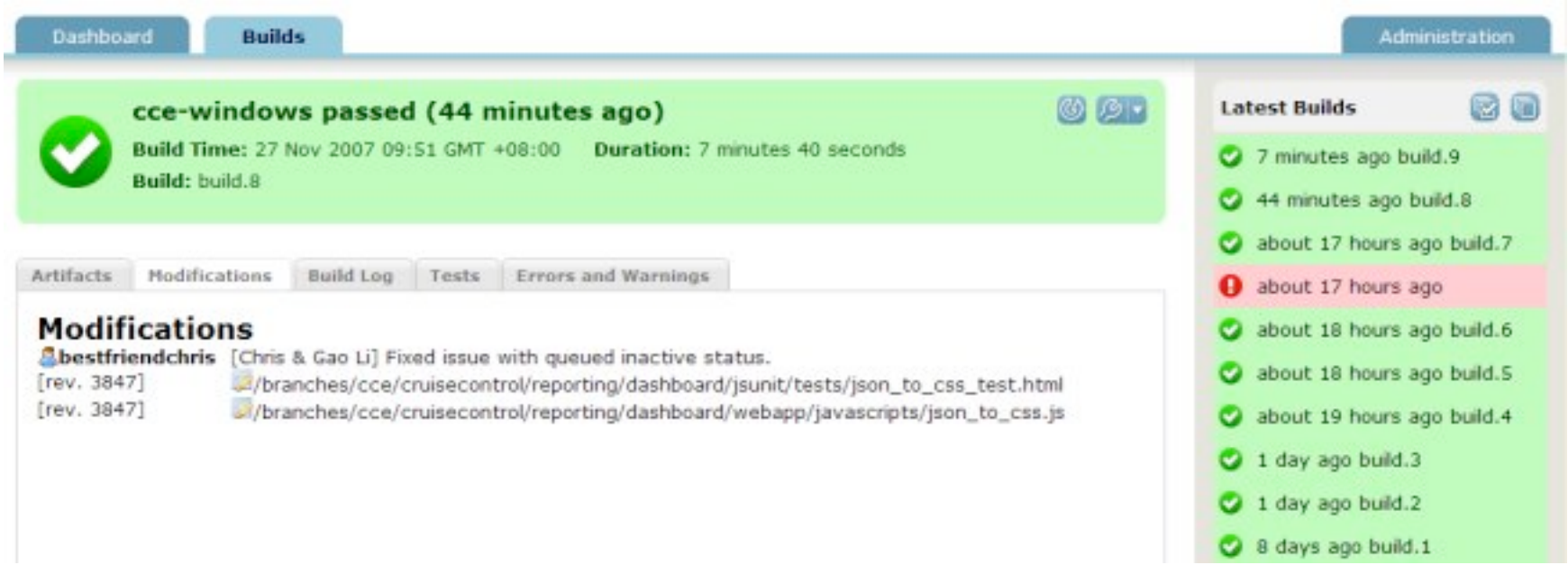
Extreme Programming : Intégration Continue

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



Extreme Programming : Intégration Continue

Exemple : CruiseControl, logiciel d'intégration continue multilingue

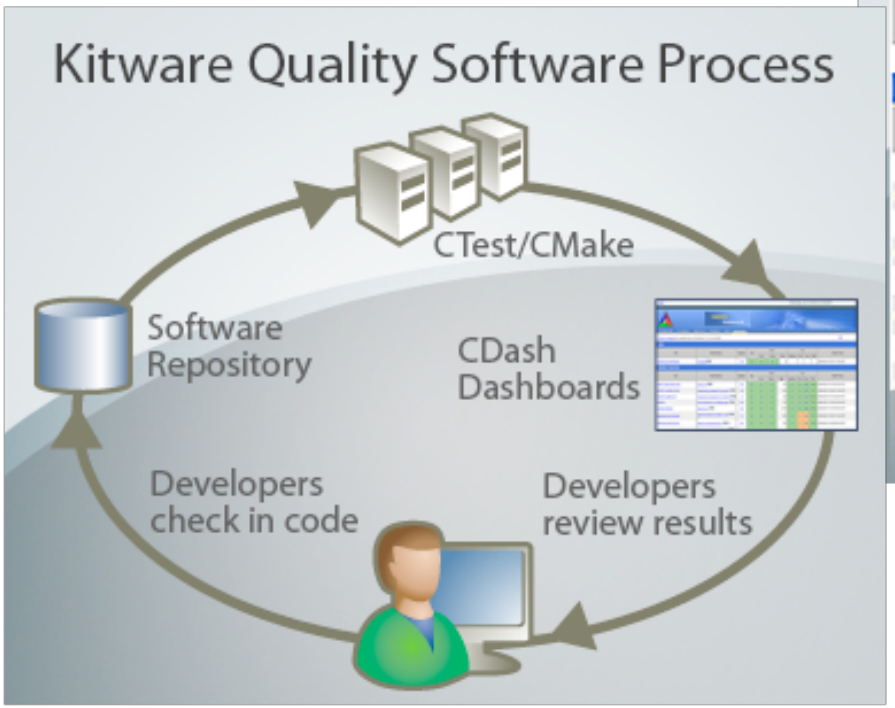


Extreme Programming : Intégration Continue

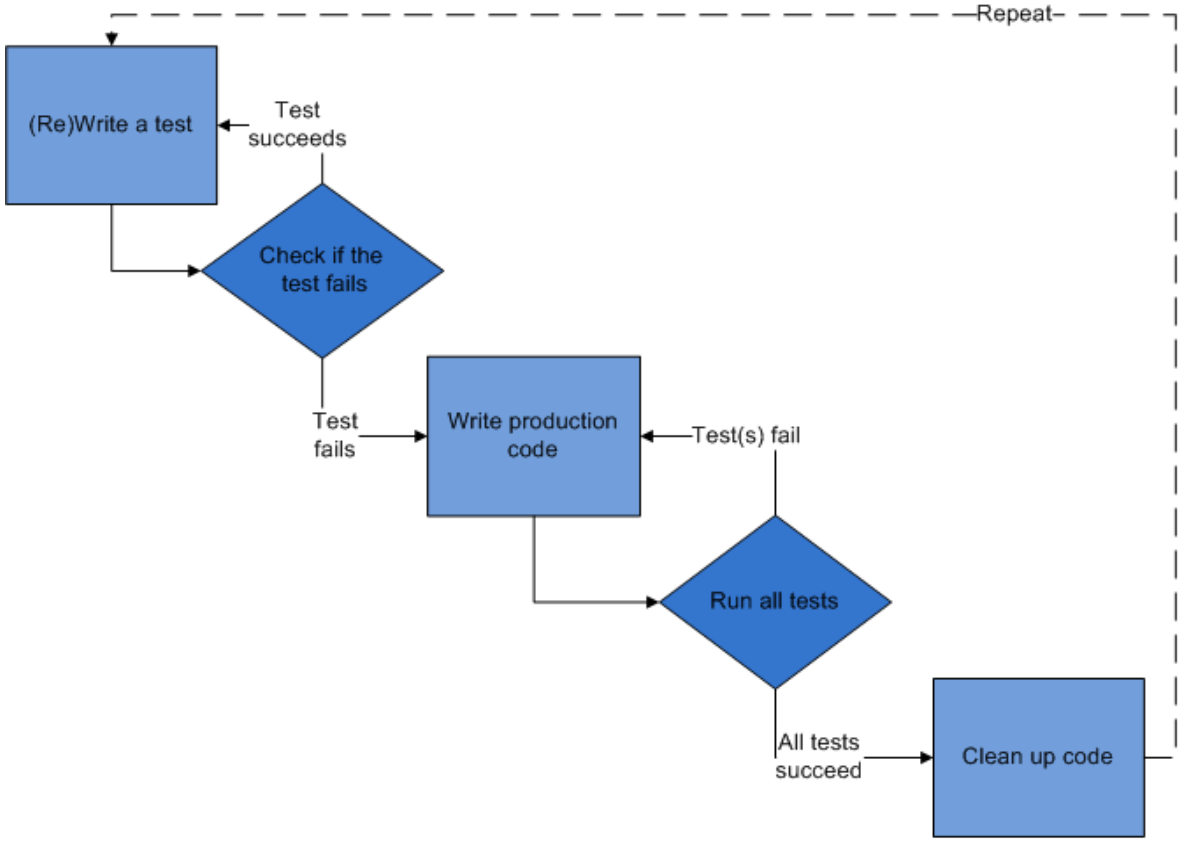
Exemple : CDash, open-source

serveur de test

avec interface web



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?

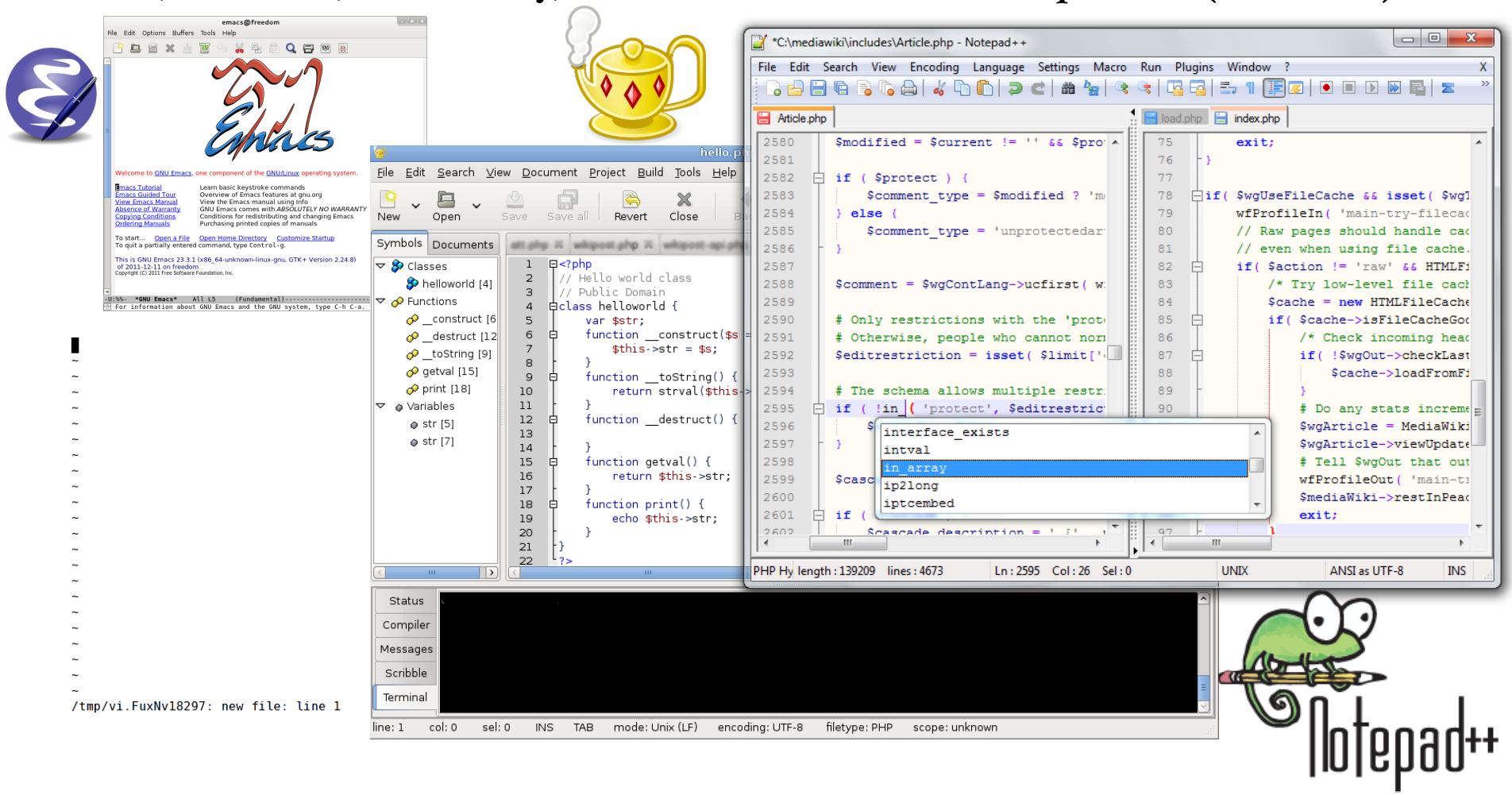
Il n'y aurait pas des outils pour coder plus efficacement?

Comment savoir ce qui existe et comment l'utiliser?

Et si je veux documenter pour les autres?

Les éditeurs de texte

Vi, Emacs, Geany, Notepad ++ (Windows)



Les environnements de développement intégré (IDE)

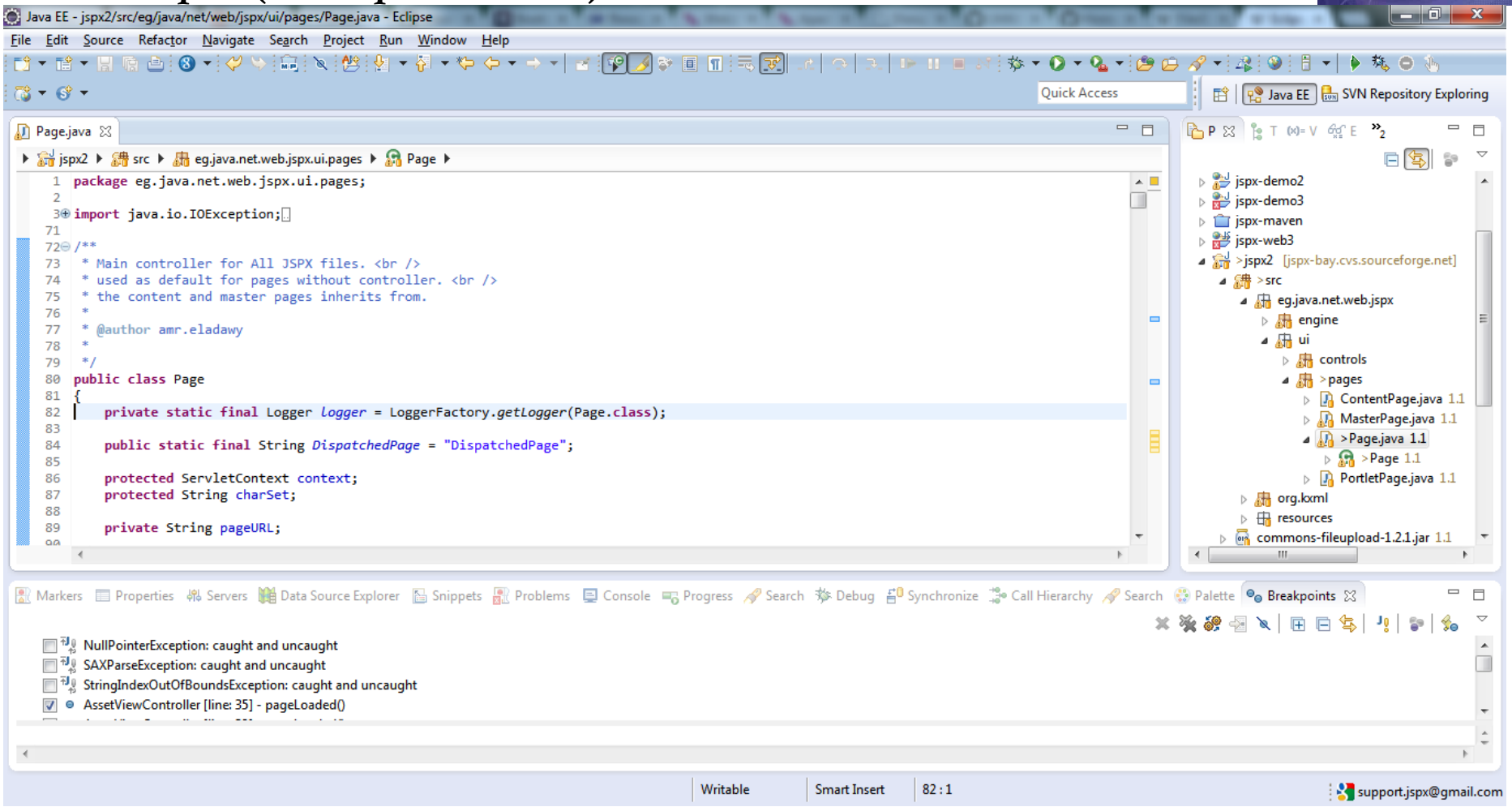
Application fournissant

- 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

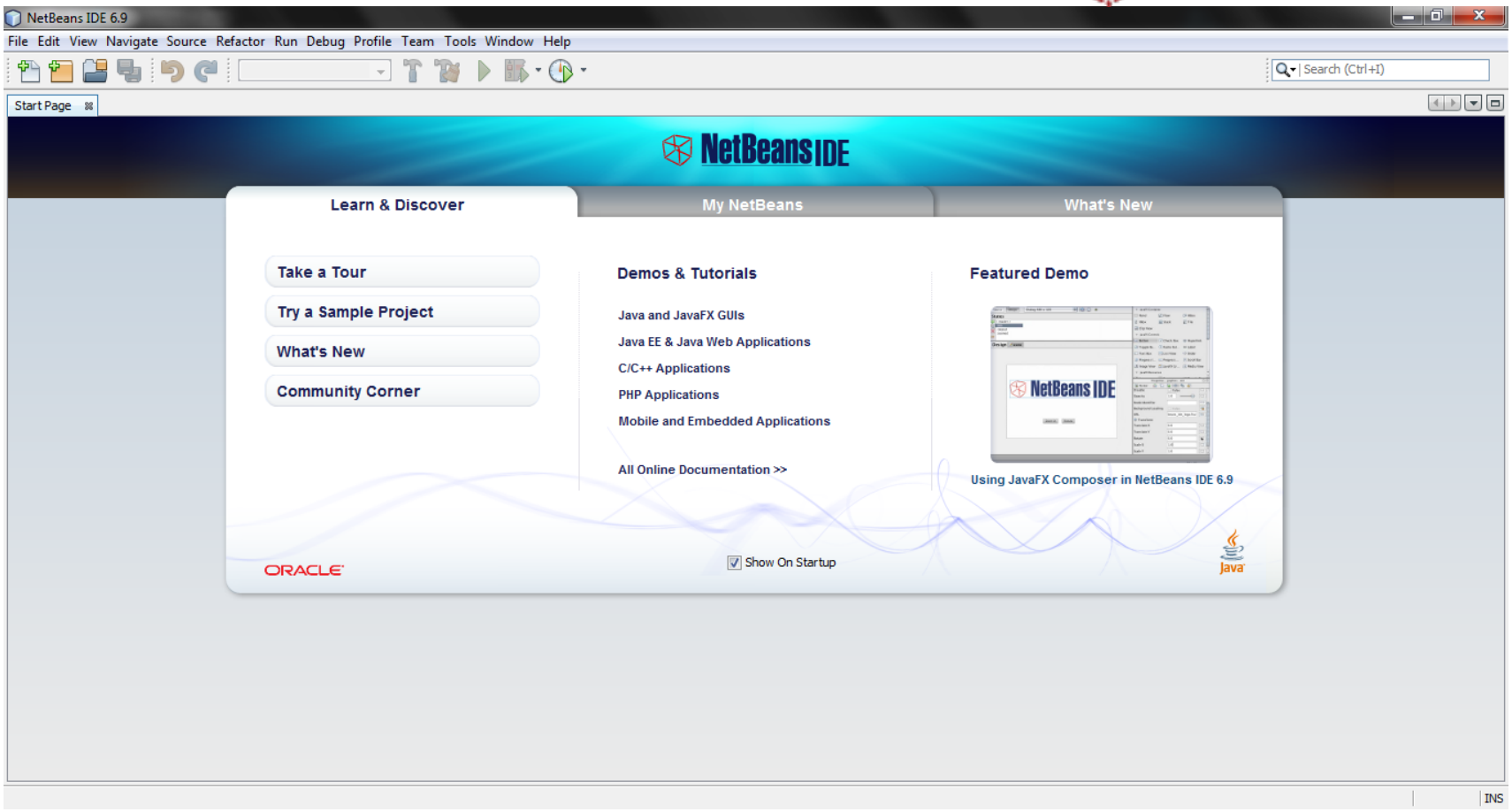
Les environnements de développement

Eclipse (multi plate-forme)



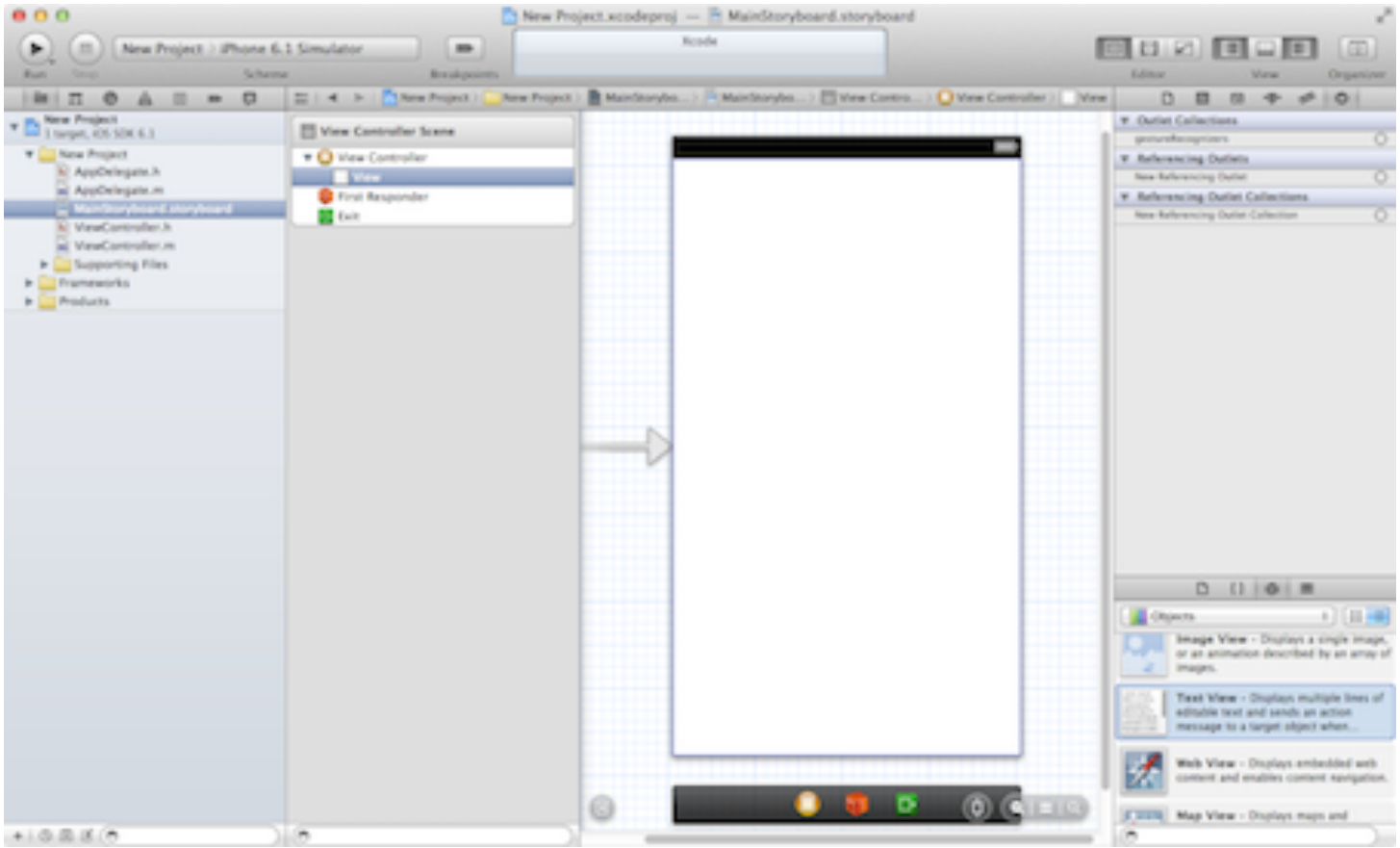
Les environnements de développement

Netbeans (Windows, Linux)



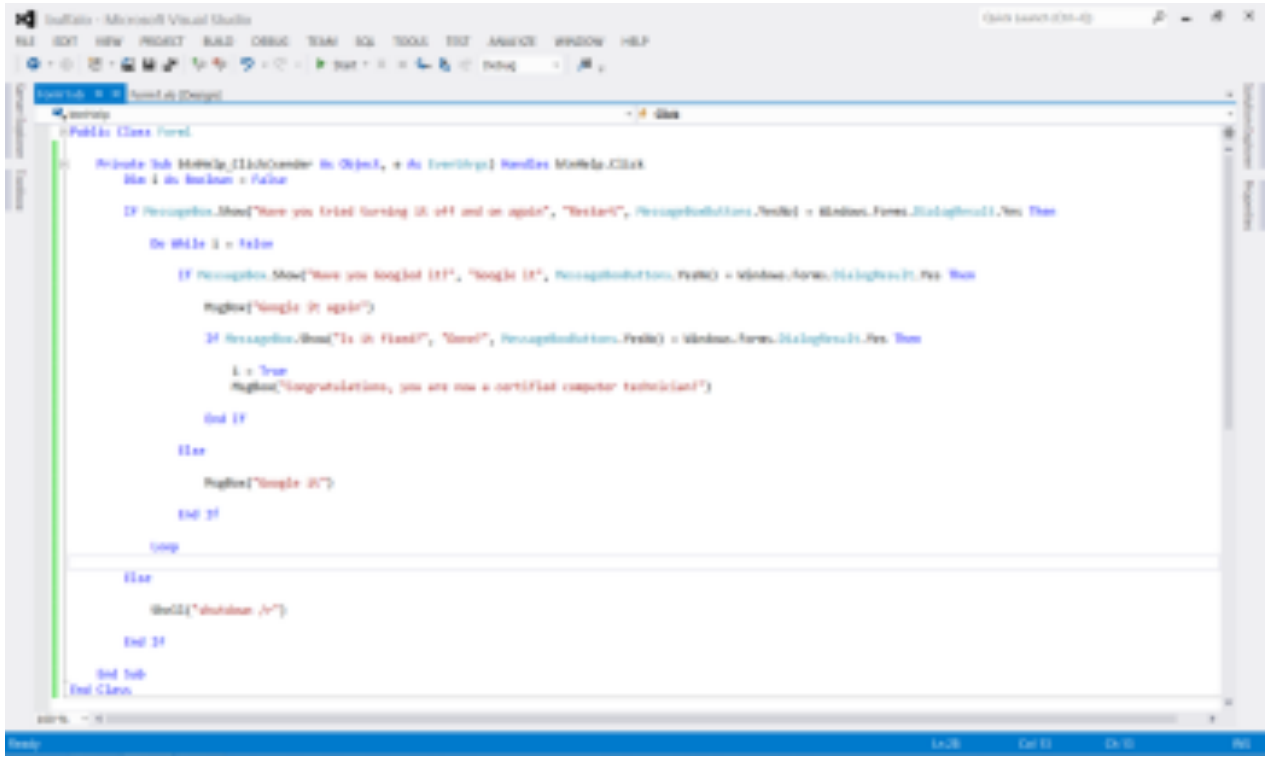
Les environnements de développement

Xcode (Mac OS X)



Les environnements de développement

Visual Studio (Windows)



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

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 provid 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 estimat extension, some parameters can also be estimated. These methods were originally introduced to deal w pertaining mostly to geophysics, but it is now widely recognized that they have a tremendous potential euHeart 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 togeth

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 r
- 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 rese at developing, sharing and integrating patient-specific multi-physics and mu great vessels in normal and pathological conditions to address clinical chall assimilation is thus intended to allow the personalization of the biophysical

Guide de l'utilisateur

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 met assimilation. It is designed to be relevant to a large class of problen numerical models.

To guarantee the highest performance, the library is implemented i provides a Python interface generated by Swig.

Models implemented in Fortran, C, C++, Python, ... can be pluggec Python interface.

Verdandi is provided under the GNU Lesser General Public Li

Scientific context

Data assimilation is the process of combining different sources of i estimate the state of a system. By extension, some parameters can were originally introduced to deal with uncertainties present in mo geophysics, but it is now widely recognized that they have a tremer applications (see euHeart example below).

Whether the system be biological, environmental, mechanical, etc., are always a numerical model, observations and error statistics. Da written independently of the system to which they are applied, and wide class of systems. Therefore methods are generic and can be pu

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
- to provide a framework for perennial development;
- to improve the diffusion and impact of data assimilation algo

Who can be a Verdandi user?

- non-specialists, engineers or researchers, who could directly methods;
- a specialist taking advantage of a modular framework, which and interactions.

Contenu scientifique

Functions

Search for

Support

3. *Optionally intialize a step with* `InitializeStep()`. *This*
4. *Perform a step forward and propagate the state error varian*
5. *Compute the analysis with* `Analyze()`, *whenever observat*
6. *Compute the data assimilation until the model has finished: the simulation is done, false otherwise.*

Reduced Order Unscented Kalmar

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

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

where U – in the group of invertible matrices $\mathcal{GL}_p$ – is of much sr represents the main uncertainties in the system. What is crucial here 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 $(\alpha) = (\alpha$ matrix of these sigma-points denoted by $[V^*] \in \mathcal{M}_{r,p}$ and the m $D_\alpha = diag(\alpha_1 ... \alpha_r) \in \mathcal{M}_r$.

1. Sampling:
   - $C_h = \sqrt{U_h^{-1}}$.
   - $x_h^{(i)a} = x_h^a + L_h C_h I^{(i)}, \quad 1 \leq i \leq p+1$
2. Prediction:
   - $x_{h+1}^f = E_\alpha(\mathcal{M}_h(x_{h+1}^{(*)a}))$
   - $x_{h+1}^{(i)f} = \left| \begin{array}{l} x_{h+1}^f + [\mathcal{M}_h(x_h^a)] D_\alpha [V^*]^T ([V^*] L \\ \text{or} \\ \mathcal{M}_h(x_h^{(i)a}) \end{array} \right.$
   - $L_{h+1} = [x_{h+1}^{(*)f}] D_\alpha [V^*]^T \in \mathcal{M}_{n,p}$
   - $P_{h+1}^f = L_{h+1}(P_\alpha^V)^{-1} L_{h+1}^T$
3. Update:
   - $y_{h+1}^{(i)} = \mathcal{H}_{h+1}(x_{h+1}^{(i)f})$
   - $\{HL\}_{h+1} = [y_{h+1}^*] D_\alpha [V^*]^T$.
   - $U_{h+1} = P_\alpha^V + \{HL\}_{h+1}^T R_{h+1}^{-1} \{HL\}_{h+1} \in \mathcal{M}$
   - $x_{h+1}^a = x_{h+1}^f + L_{h+1} U_{h+1}^{-1} \{HL\}_{h+1}^T R_{h+1}^{-1} (y_i$
   - $P_{h+1}^a = L_{h+1} U_{h+1}^{-1} L_{h+1}^T$

With:
x_h^f forecast state vector;
x_h^a analysis state vector;

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 ?

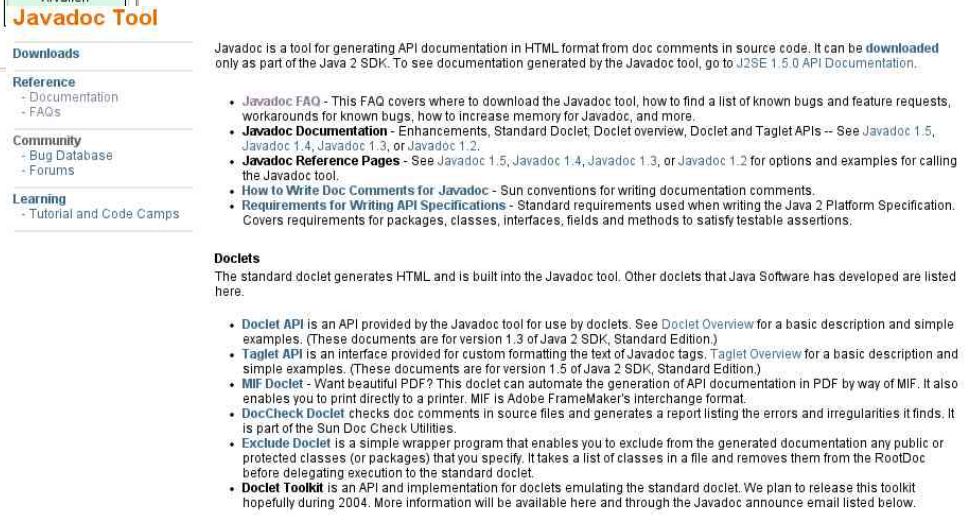
- La seule source d'information absolument juste est le code
- La rédaction de la documentation technique est laborieuse et complexe
- Faciliter la maintenance, le développement d'un code écrit seul et surtout à plusieurs
- La documentation est écrite dans le code, et il est donc relativement facile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



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

Main Page | Classes | Class List | Class Members

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

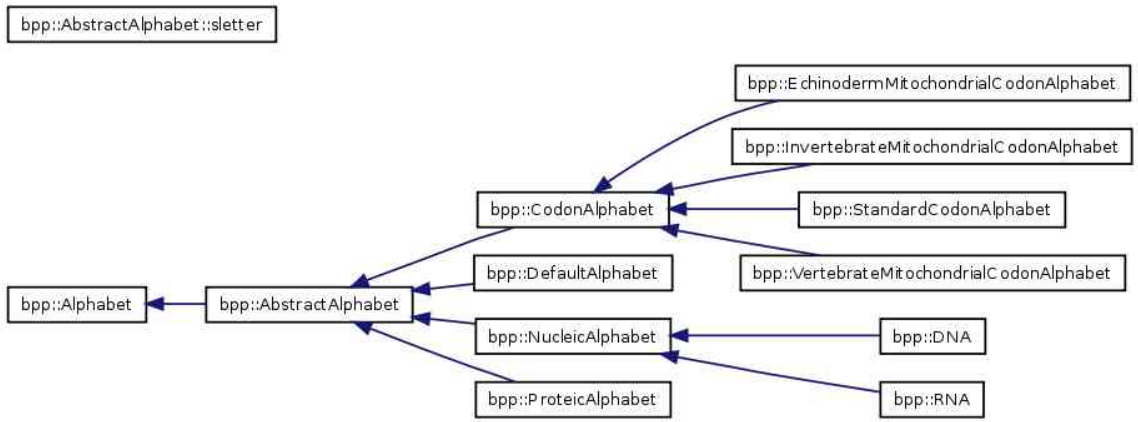
Utils | NumCalc | SeqLib | PhylLib | PopGenLib

Main Page | Classes | Files | Related Pages | Search for

Class List | Class Hierarchy | Class Members

SeqLib Graphical Class Hierarchy

Go to the textual class hierarchy



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?

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 ?

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.

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...

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

Licences de logiciel libre : Typologie

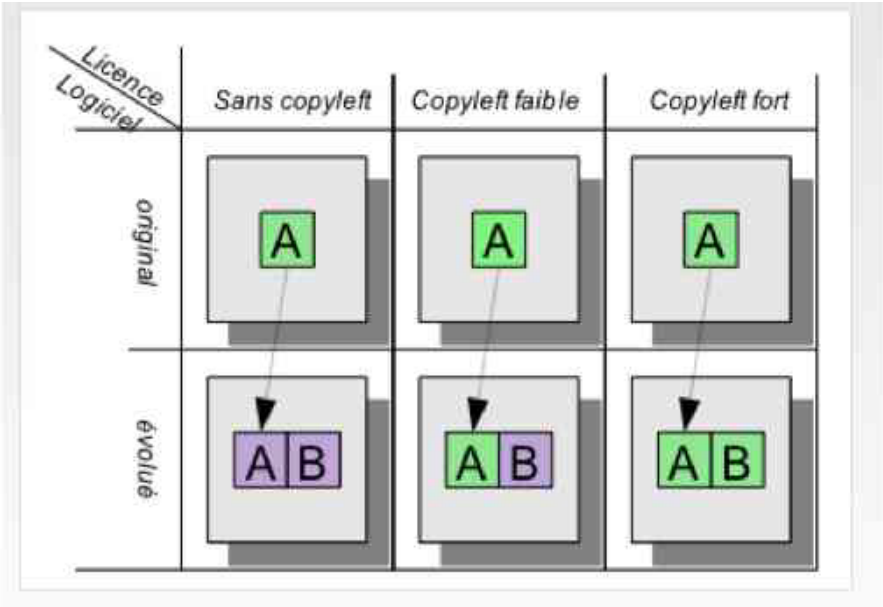
Déterminer le caractère copyleft ou non de la licence

Copyleft :

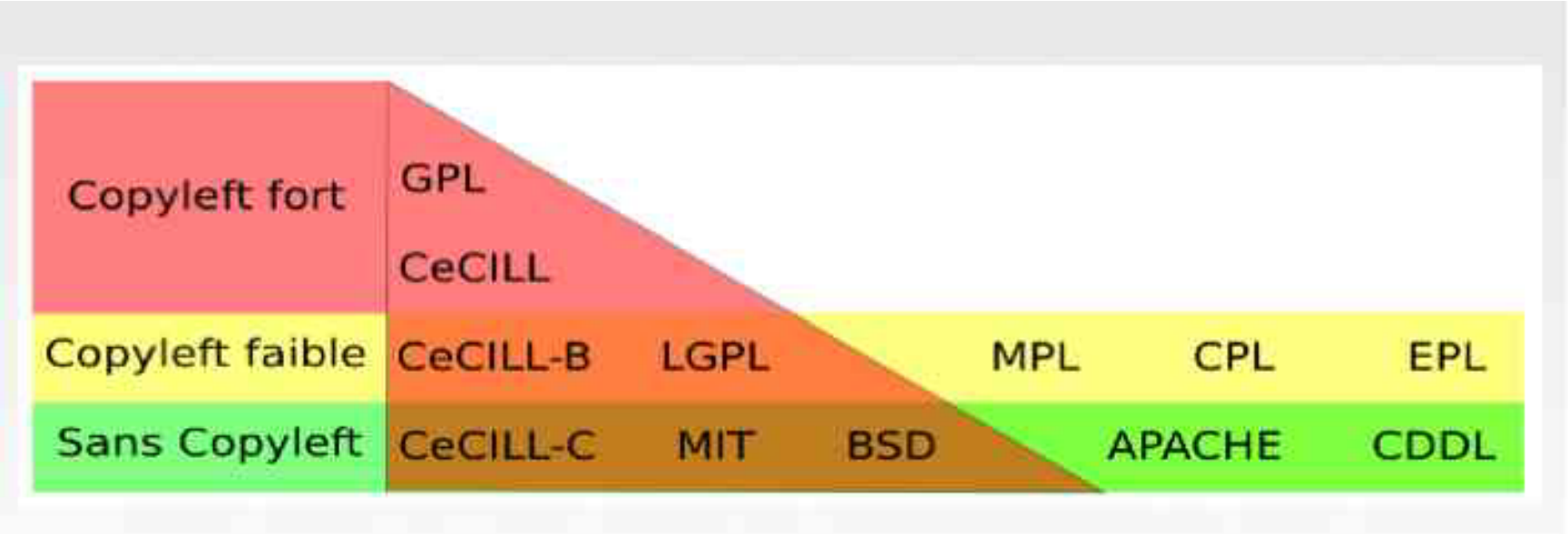
- Exigence de réciprocité : le code source (le logiciel) sera redistribué sous la même licence qu'il a été reçu
- Héréditaire

Portée du copyleft

Copyleft faible ou fort



Licences de logiciel libre : Compatibilité



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

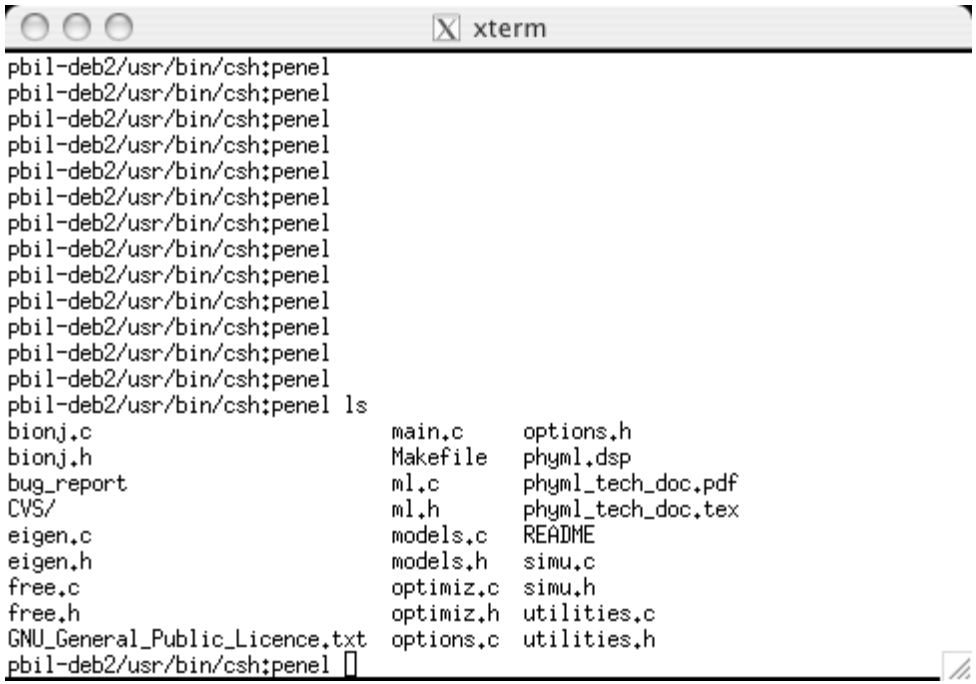
  fichier source suffit pour bénéficier des termes de la licence
- Le dépôt des organismes de type APP (pour un coût un coût dérisoire) apporte une preuve d'antériorité
- Mutualisation de l'effort de développement
- Maximisation des possibilités d'irriguer la société

Licences de logiciel libre : Comment?

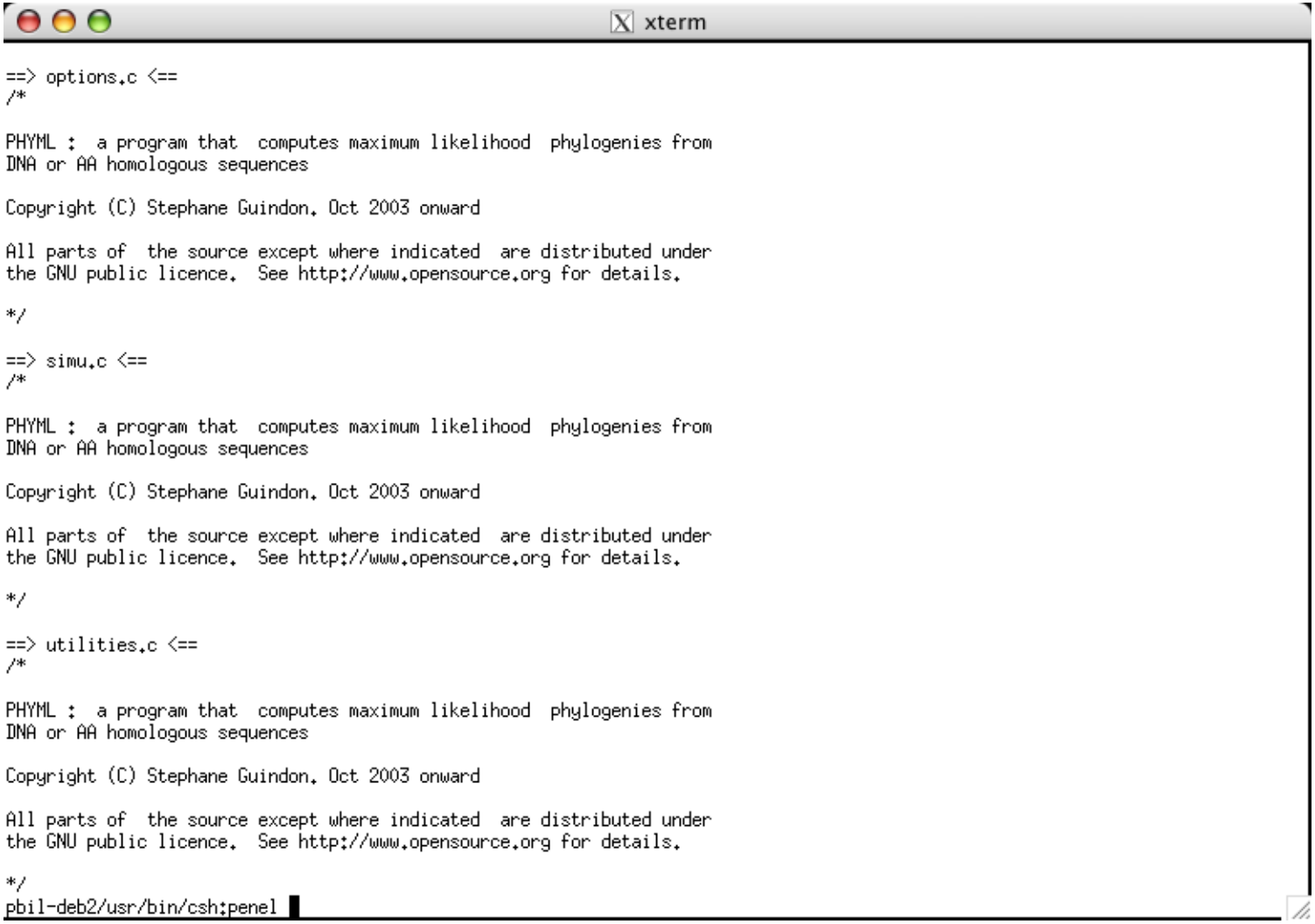
Mention des auteurs et de la licence dans l'en-tête des fichiers

Licence présente dans l'arborescence

Exemple : le programme phyml



Licences de logiciel libre : Comment?



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

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

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}

)
```

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?**

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**

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

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

Travailler en équipe : Gestion de version

Modèle client-serveur :

Programmes :

- CVS
- Subversion

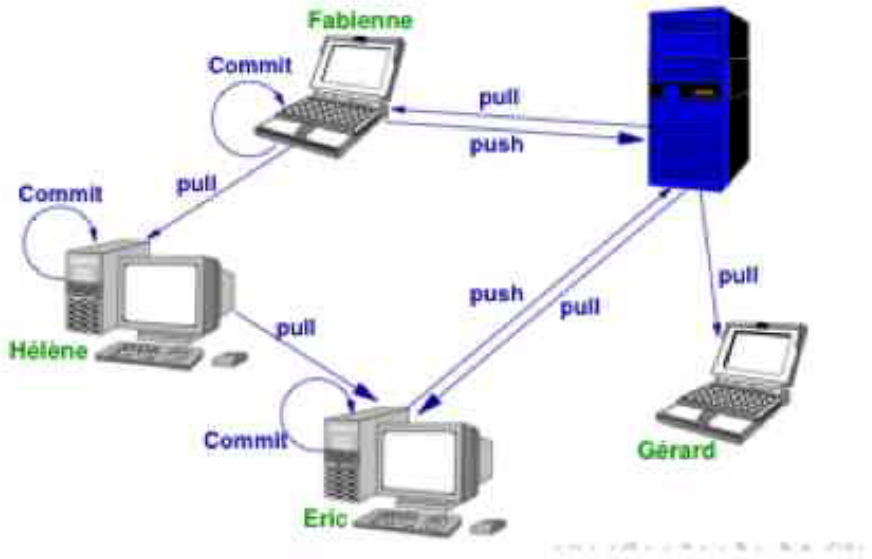
Travailler en équipe : Gestion de version

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

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

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

Travailler en équipe : Les Forges

Exemples :

http://sourceforge.net/

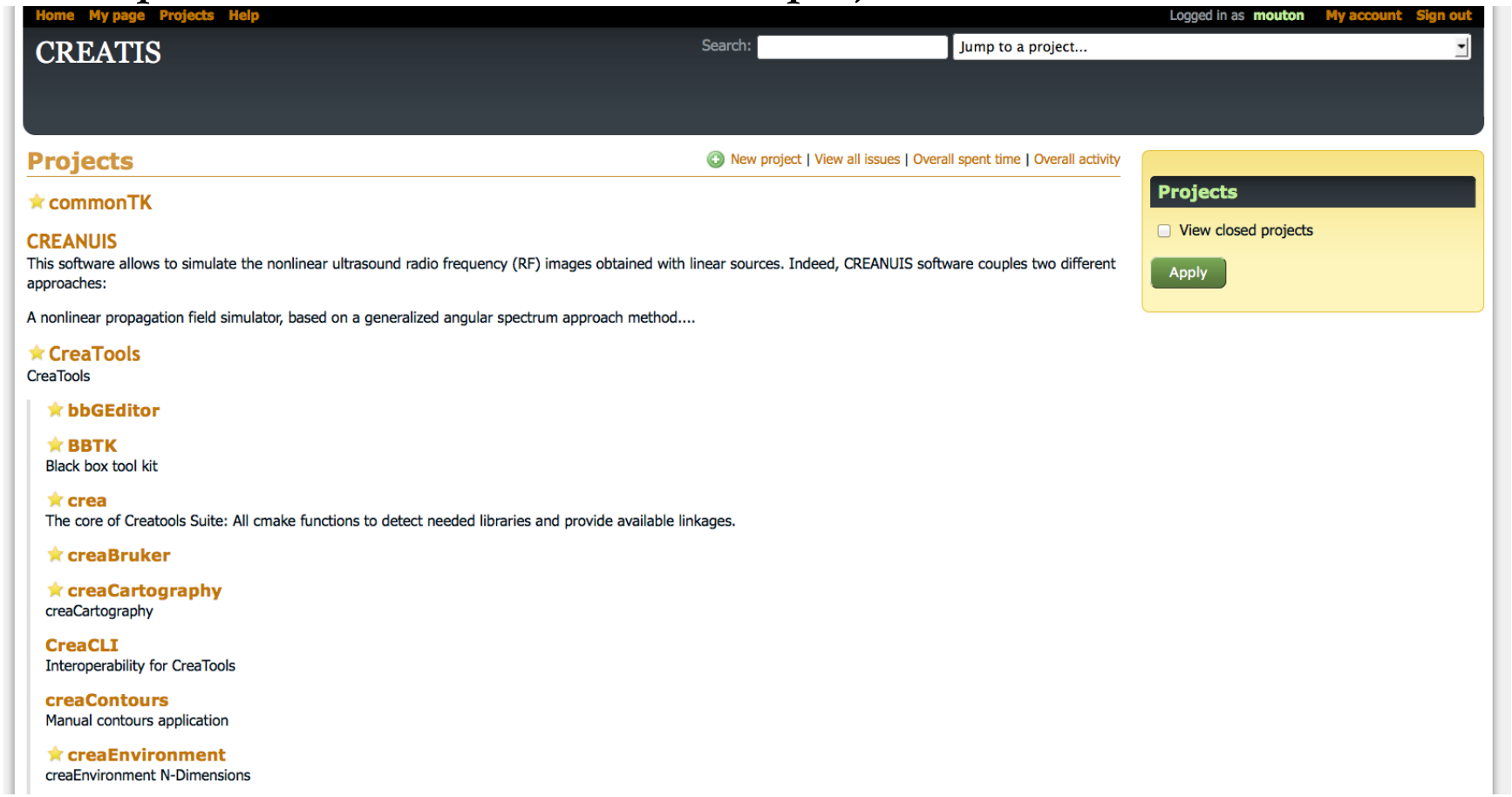
https://github.com/

http://gitorious.org/

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

Travailler en équipe : Les Forges

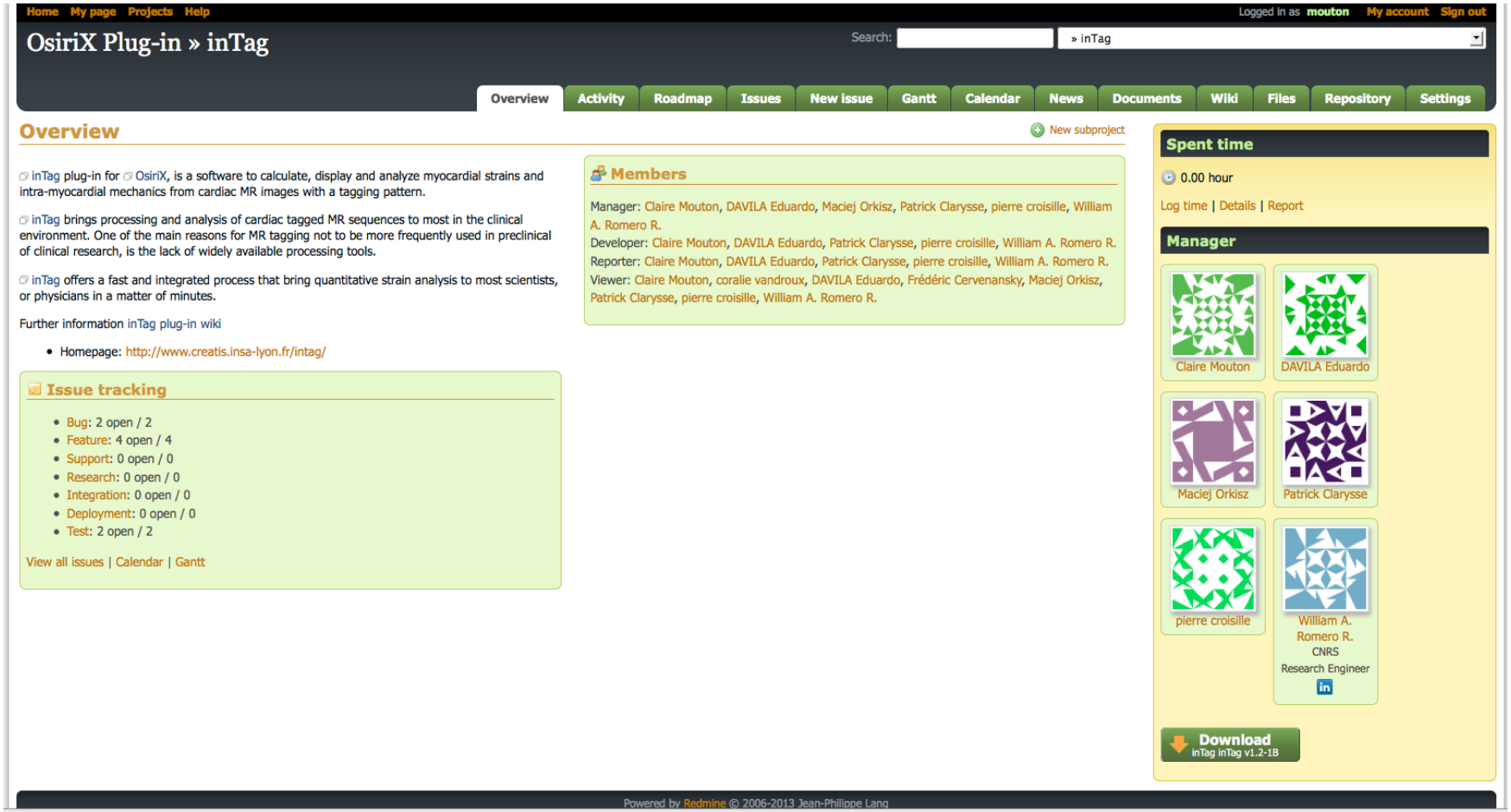
Exemple : Redmine à CREATIS – Les projets



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

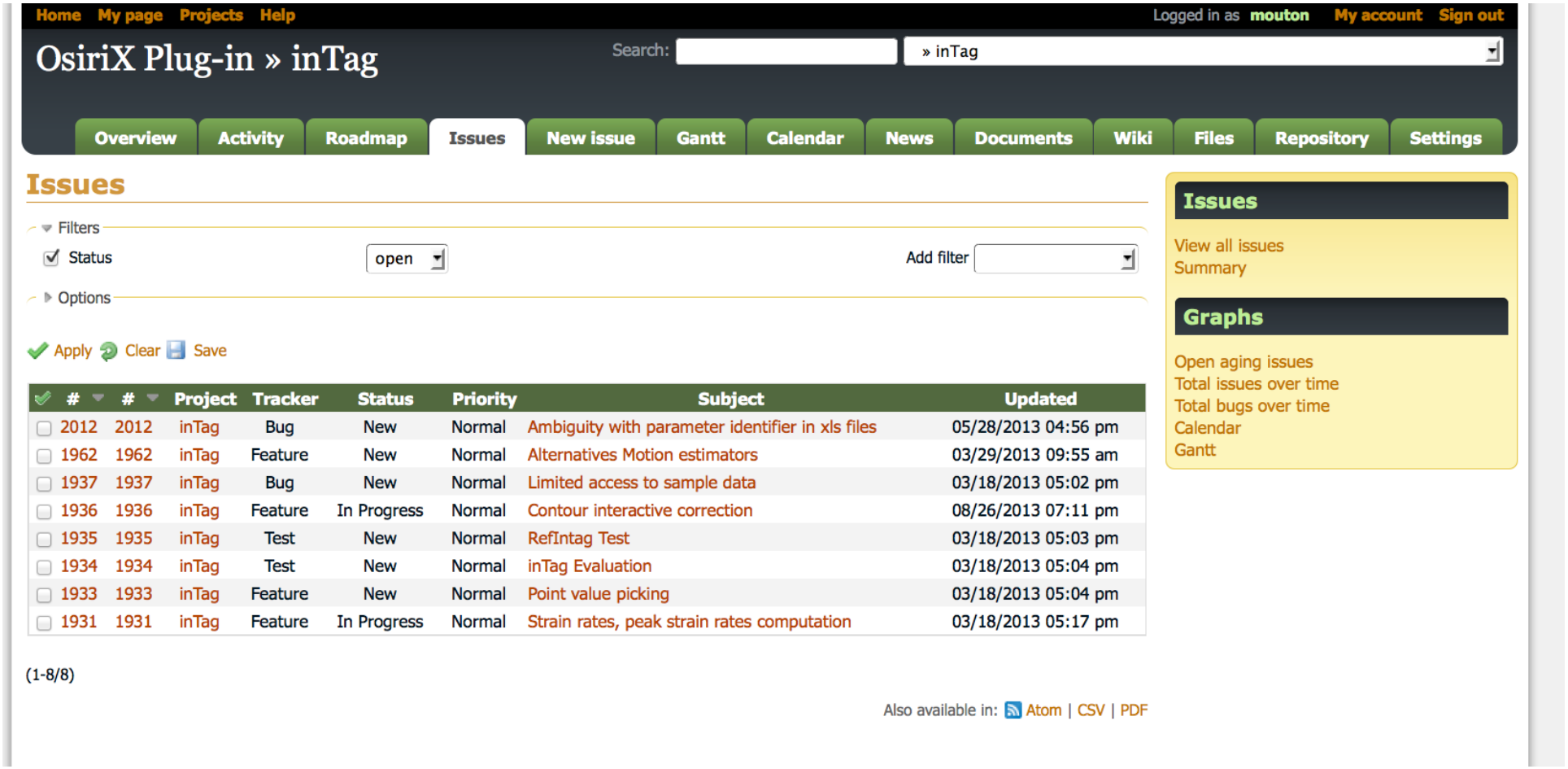
Travailler en équipe : Les Forges

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



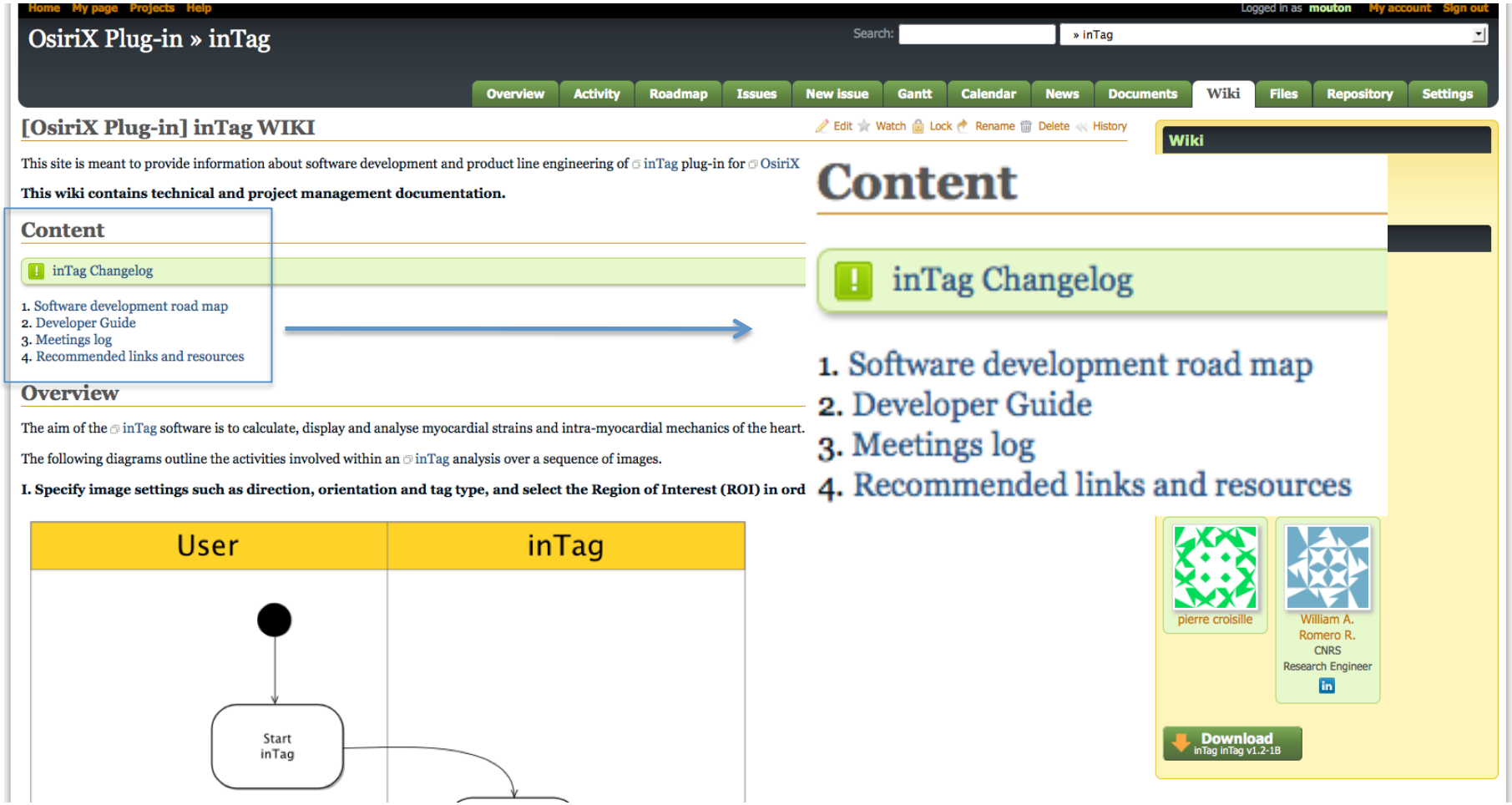
Travailler en équipe : Les Forges

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



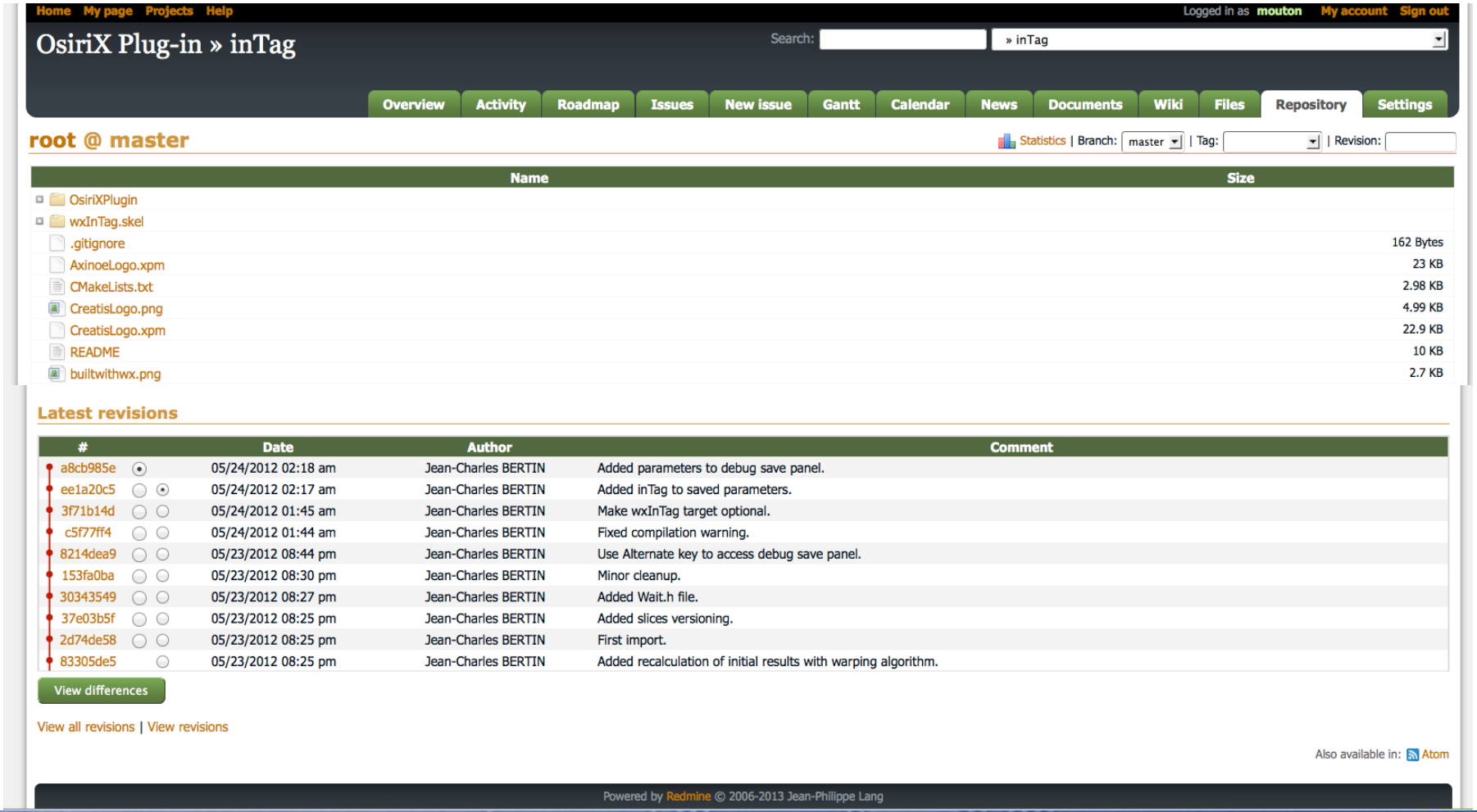
Travailler en équipe : Les Forges

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



Travailler en équipe : Les Forges

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



Fin du cours…

À vous la main!

Des questions avant de plonger dans le code?