Projet de développement

Documentation
Soutenance...

Philippe Collet

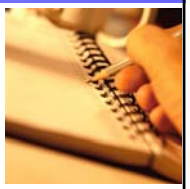
Licence 3 Informatique

2011-2012

Université Nice SOPHIA ANTIPOLIS

Plan

- **Documentation : principes et outils**
- **Contenu attendu pour le projet**
- **Modalités de soutenance**



Pourquoi documenter (le code) ?

- **Ce que l'on décrit bien, se conçoit bien !**
- **Pour le prochain programmeur ou nous même ?**
- **Est-ce du temps perdu ?**
  … Et jusqu'à quel point documenter ?
- **Documenter, (une fonction)**
  … n'est pas commenter! (le code)

Problématiques

- **Programmeur :**
  - Historique: qui/quand/quoi
  - Algorithme
  - Borne d'utilisation
  - Paramètres
  - Valeur retournée
  - Structure du module
  - Mode d'utilisation
  - Code d'erreur
- **Mainteneur :**
  - Historique: qui/quand/quoi
  - Documentation utilisée à jour
  - Que fait la fonction
  - Quel module utilise quelle fonction
  - Effet de cascade de la modification en cours

Garder synchrone code et documentation

- **Une documentation obsolète est une documentation inutile**
- **Documentation interne et extraction :**
  - Commentaires formatés
  - Graphe de dépendance entre les classes, les fichiers
  - Hiérarchie de classes
- **Mais la documentation externe a des avantages**
  - Lisibilité accrue
  - Synthèse du logiciel

Exemple : Javadoc

- **Extracteur spécialisé java (très simple)**
  - Cohérence avec les conventions Java
  - Lancement possible sur toute une hiérarchie
  - Production de html
  - Extensible avec des « doclets » (changement du format de sortie)

getImage
public Image getImage(URL url, String name)

Returns an Image object that can then be painted on the screen. The url argument must specify an absolute URL The name argument is a specifier that is relative to the url argument. This method always returns immediately, whether or not the image exists. When this applet attempts to draw the image on the screen, the data will be loaded. The graphics primitives that draw the image will incrementally paint on the screen.

Parameters:
url - an absolute URL giving the base location of the image
name the location of the image, relative to the url argument

Returns:
the image at the specified URL

See Also:
Image

Exemple : Javadoc (source)

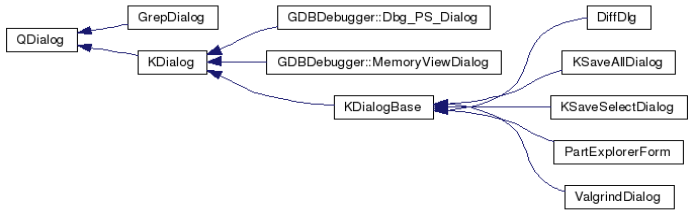
```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the image
 * @param  name the location of the image, relative to the url argument
 * @return      the image at the specified URL
 * @see         Image
 */
public Image getImage(URL url, String name) {
        try {
            return getImage(new URL(url, name));
        } catch (MalformedURLException e) {
            return null;
        }
}
```


Exemple : Doxygen

- **Multi-langages**
  - C++, C, Java, Objective-C, Python, IDL (Corba et Microsoft)
  - partiellement PHP, C#
- **Multi-plates-formes**
  - Unix, Windows
- **Multi-sorties**
  - HTML, LaTeX (PDF), RTF, Man, XML, Aide Windows
- **Traitements plus élaborés :**
  - Fichier de configuration, assistant d'aide à la configuration
  - Génération de graphes avec graphviz/dot
  - Génération d'un moteur de recherche (pour site web avec PHP)

Doxygen : graphes

- **Hiérarchie de classes**
- **Graphes d'inclusion**
- **Graphes de collaboration**
- **Graphes d'appel**



Exemple : PhpDocumentor

- **Outil de documentation écrit en PHP**
  - semblable à l'outil de javadoc
  - travaille par analyse des commentaires et du code
  - Open source (licence LPGL)
- **Fonctionnalités**
  - Génération de documentation du code réalisé en objet ou procédural
  - Formats d'exports : HTML, PDF, CHM ou docbook
- ***www.phpdoc.org***

Exemple : PhpDocumentor

```
class class1 {
  /**
    * example of documenting a method, and using optional description with @return
    * @return string de-html_entitied string (no entities at all)
    */
   function bar($foo)   {
      return strtr($foo,array_flip(get_html_translation_table(HTML_ENTITIES)));
   }

   /**
    * example of using @return with a class name
    * @param integer even or odd integer
    * @return Parser|false phpDocumentor Parser object or error
    */
   function &factory($number)   {
      $returnval = true;
      if ($number % 2)      {
          $returnval = new Parser;
      } else      {
          $returnval = false;
      }
      return $returnval;
   }
}
```


Application à votre projet : soutenance

Université Nice SOPHIA ANTIPOLIS

Résultat attendu

- **Chaque projet doit rendre accessible, avant la soutenance, les informations suivantes :**
  - **Redmine :**
    - Wiki : au moins la page principale éditée pour synthétiser les grands choix de conception, les fonctionnalités livrées, les problèmes rencontrés, etc.
    - Jalons (milestones / version) déterminés dans le projet (a priori ou a posteriori, l'essentiel étant de communiquer sur ces jalons)
    - Identification des tâches individuelles ou en binôme : ticket (fonctionnalités, bugs corrigés...)
    - Chaque ticket fermé correspondant à du développement doit correspondre à une version sur le subversion
    - Les réunions doivent aussi faire l'objet d'un compte-rendu (ticket réunion spécifique), que ce soit pour une réunion avec prise de décision au sein du groupe ou avec l'encadrant.

Résultat attendu

- **Subversion :**
  - Tout le code doit être documenté raisonnablement (fichiers source en C, toute classe en Java, pages et classes PHP) : entête du module/classe, commentaire sur les fonctions/opérations les plus importantes (pas forcément tous les getters/setters)
  - De la même manière, les fonctionnalités les plus critiques doivent être forcément testées (plus une fonctionnalité est critique ou vous a posé de problèmes, plus on s'attend à la voir fortement testée). Etre capable de passer l'ensemble des tests écrits sur votre projet fait partie des résultats attendus (il doit donc y a voir de la documentation pour expliquer comment faire). De plus, il est aussi attendu que l'équipe puisse argumenter ses choix de test (pourquoi telle ou telle partie a été plus ou moins testée et comment).
  - Chaque commit doit avoir un commentaire associé (ce commentaire n'a pas a être très long, surtout si la fermeture d'un ticket correspond au commit)
  - La dernière version exécutable et testable doit être identifiée le jour de la soutenance (pour être justement exécutée et testée)

Soutenance

- **Chaque soutenance dure 25 minutes**
- **15 minutes de présentation : chaque présentation doit faire intervenir tous les membres de l'équipe de façon équivalente. Le plan suivant est conseillé :**
  - Introduction et rappel du sujet
  - Fonctionnalités réalisées : bilan à gros grain de ce qui a été fait, est resté en chantier, a été écarté par rapport à ce qui était attendu (démo possible mais pas obligatoire)
  - Grands choix de conception et approche suivie : les jalons et tickets doivent vous servir à facilement établir, au fur et à mesure de votre avancement, les éléments à mettre en avant dans cette partie
  - Problèmes rencontrés et solutions apportées (correction, changement dans la conception, abandon d'une fonctionnalité...)
  - mini-démo finale (attention au timing)
- **10 minutes de question :**
  - Chaque membre du projet doit être capable de répondre aux questions sur les choix de conception généraux, les parties qu'il a développées, etc.

Questions

http://deptinfo.unice.fr/twiki/bin/view/Linfo/ProjetDevt2012Soutenance