Présentation de Doxygen[1]

1. Présentation générale

Doxygen est un système de documentation Open Source pour les programmes écrit en C++, C, Java, IDL mais aussi en PHP, C# et bien d'autres.

Il peut générer une documentation en HTML ou encore un manuel de référence en LaTeX à partir de fichiers sources commentés. Il y a aussi un support pour générer des sorties aux formats HTML compressé, pages de manuel Unix, PostScript, RTF, XML, etc.

Doxygen peut également être configuré pour extraire la structure du code à partir de sources non documentés. Cette fonctionnalité peut s'avérer très utile pour comprendre rapidement le fonctionnement d'un programme dans des projets importants. Les relations entre les éléments sont visualisées à travers des graphes de dépendance, des diagrammes d'héritage et des diagrammes de collaboration qui sont tous automatiquement générés, en utilisant les services de Graphviz. Ces diagrammes ne respectent par contre pas la norme UML, même s'ils sont facilement compréhensibles.

Doxygen est développé sous Linux, mais c'est un outil facilement portable qui tourne sur la plupart des plates-formes Unix. Des exécutables pour Windows 9x/NT et Mac OS X sont également disponibles.

2. Procédure d'installation

Pour installer Doxygen sous un système d'exploitation de type Unix, il faut tout d'abord télécharger la distribution binaire, disponible sur le site http://www.stack.nl/~dimitri/doxygen. Ensuite, il est nécessaire de compiler le logiciel en saisissant les commandes suivantes :

```
$ tar zxvf doxygen-1.x.x.tar.gz
$ cd doxygen-1.x.x
$ ./configure --prefix="/usr/local/doxygen"
$ make
$ make install
```

Doxygen est alors installé dans /usr/local/doxygen. Son exécutable est situé soit dans /usr/local/bin ou dans /usr/local/doxygen/bin.

Néanmoins, pour bénéficier de toute la puissance de Doxygen, il est nécessaire d'installer d'autres programmes. Ce sont tout d'abord Graphviz, disponible sur http://research.att.com/sw/tools/graphviz et ensuite une distribution LaTeX. Graphviz est Open Source et sa procédure d'installation est identique :

```
$ tar zxvf graphviz-1.x.x.tar.gz
$ cd graphviz-1.x.x
$ ./configure
$ make
$ make install
```

[1] Avec tous mes remerciements pour leurs contribution à Sévrine Portal, Jean-Christophe Gay et Romain Séguy

Des distributions complètes de LaTeX sont disponibles sur le site du TeX User Group, http://www.tug.org. Nous ne détaillerons pas ici son installation, car elle est très longue.

Pour les plates-formes Windows, il existe des distributions binaires des trois logiciels précités. L'installation est accessible à tout le monde. Il faudra néanmoins prendre soin d'ajouter à la variable d'environnement PATH des liens vers les répertoires contenant les exécutables des programmes. L'installation pour Windows peut-être trouvée à l'adresse suivante : ftp://ftp.stack.nl/pub/users/dimitri/doxygen-1.4.1-setup.exe

Une autre solution consiste à utiliser Cygwin. L'installation sous Cygwin est légèrement différente de celle sous un Linux natif. Voici la procédure :

```
$ tar zxvf doxygen-1.x.x.tar.gz
$ cd doxygen-1.x.x
$ ./configure --prefix "/usr/local/doxygen"
$ rm install
$ make
$ make install
```

3. Fonctionnalités

- Support des sources C, C++, Java, IDL mais aussi C# et PHP, etc. ;
- Support de la documentation des fichiers, espaces de nommage, classes, structures, unions, templates, variables (même globales), fonctions (même globales), typedefs, énumérations et defines ;
- Compatible avec JavaDoc, Qt-Doc ou encore KDoc ;
- Génère automatiquement des diagrammes de classes sous différents formats graphiques (GIF, PNG, etc.) grâce à Graphviz. Tous les formats sont optimisés pour être facilement lisibles ;
- Utilise Graphviz pour générer des graphes de dépendance, des diagrammes d'héritage de collaboration et des graphes représentant les hiérarchies entre classes ;
- Possibilité de rajouter des balises HTML dans la documentation. Doxygen en tient compte automatiquement lors de la mise en page de la documentation ;
- Possibilité de rajouter de la documentation dans les fichiers d'entête (avant ou après la déclaration d'une entité), les fichiers source (avant ou après la définition d'une entité) ou dans un fichier séparé ;
- Peut générer une liste de tous les membres d'une classe (en incluant tous les membres hérités) avec leur niveau de protection ;
- Détection automatique des sections publiques, protégées et privées ;
- Génération automatique de références vers les classes, fichiers, espaces de nommage et membres documentés ;
- Autorise les références vers la documentation générée pour d'autres projets (ou une autre partie du même projet) ;
- Inclut un outil de recherche rapide pour chercher des chaînes de caractères ou des mots dans la documentation des classes et des membres ;
- L'inclusion de classes non documentées est également supportée, ce qui permet de visualiser rapidement la structure et les interfaces d'une grande partie du code sans connaître les détails d'implémentation ;
- Tous les fragments de code source ont une coloration syntaxique pour faciliter la lecture ;
- Peut gérer des projets importants.

Voici un exemple de code source commenté :

```
#ifndef MACLASSE_HPP
#define MACLASSE_HPP

/*! \brief Description succincte de la classe terminée par un point.
 *
 * \par
 * Un premier paragraphe sans titre décrivant la classe.
 *
 * \par Titre du deuxième paragraphe
 * Un deuxième paragraphe avec un titre.
 *
 * \par Titre du troisième paragraphe
 * Les listes sont automatiquement prises en compte :
 * - premier élément de la liste ;
 * - second élément.
 * Un peu de texte <i>avec</i> du <b>HTML</b>.
 *
 * \author Moi
 */

class MaClasse {

  public:
    //! \brief Description en une ligne du constructeur.
    MaClasse();

    //! \brief Description en une ligne du destructeur.
    virtual ~MaClasse() {}

    /*! \brief Description succincte de la méthode.
     *
     * \param in1 le premier paramètre
     * \param in2 le second paramètre
     * \return description de la valeur de retour
     */
    bool maFonction(int in1, int in2) const;

  protected:
    int att1_;  //!< Un premier attribut.
    int att2_;  //!< Un second attribut.

};

/* ------------------------------------------------------------ */
inline MaClasse::MaClasse() {
  // …
}
#endif
```

Figure 1 : exemple de code documenté pour Doxygen

4. Guide d'utilisation

L'utilisation de Doxygen est très simple. Il faut dans un premier temps générer un fichier de configuration pour le projet en cours :

```
$ doxygen –g mon_nouveau_fichier_de_configuration
```

Il est ensuite nécessaire de modifier le contenu du fichier avec n'importe quel éditeur de texte. De nombreuses options sont incluses par défaut et il faut parcourir tout le fichier de configuration, qui est fortement commenté, et modifier les options que l'on souhaite. Ceci est une opération très rapide et simple.

Une fois ce fichier généré, il est nécessaire de commenter les codes source du programme à commenter si ce n'est pas déjà fait (mais ça devrait déjà l'être, n'est-ce pas ?) en utilisant la syntaxe de son choix décrite dans le manuel utilisateur de Doxygen.

Le code étant maintenant commenté (cf-exemple ci-dessus), la ligne de commande suivante génère la documentation :

$ doxygen mon_nouveau_fichier_de_configuration

5. Points positifs

- Support de multiples formats, permettant de publier la documentation sur Internet (HTML), de partager la documentation avec d'autres logiciels (XML), de réaliser des manuels de référence (PDF/PS), etc. ;
- Grande qualité de la documentation générée grâce aux références et aux diagrammes ;
- Support de multiples langages de programmation ;
- Les sources peuvent être commentés par l'intermédiaire d'un grand nombre de styles différents (style Javadoc, KDoc, etc.), permettant ainsi à chacun de choisir le style de commentaires qui lui convient ;
- Documentation de classes même non documentées pour connaître leur structure, etc.

Annexe : Exemple de style pour C++ et DOxygen

```
/** ------------------------------------------------------------------------ *
  *                                                                          *
  * doxygen.cpp                                                              *
  * Exemple de guide de style avec Doxygen                                   *
  *                                                                          *
  * @author    Mathieu                                                       *
  *                                                                          *
  * ------------------------------------------------------------------------ **/


/** ------------------------------------------------------------------------ *
  *                                                                          *
  * Description de la fonction                                               *
  *                                                                          *
  * @param    inParam1    Description ...                                     *
  * @param    inParam2    Description ...                                     *
  *                                                                          *
  * @return               Description ...                                     *
  *                                                                          *
  * ------------------------------------------------------------------------ **/
int function (int inParam1, int inParam2)
{

    return 0;
}
```