PyUt
Python UML Tool
Rapport général

Groupe Burgbacher
EI5, eivd

27 février 2002

Table des matières

Résumé

PyUt est un petit éditeur UML graphique qui permet l'édition, l'impression et l'export de diagrammes de classe sous différents formats. Ce document présente les choix, la répartition du travail, et les responsabilités des membres du groupe.

1 Informations générales

Nom du projet PyUt - Python UML Tool
Site web de référence *http://pyut.sf.net*
Sources du projet Disponible par CVS (cf 3.1 page 4)
Professeur J. Zuber <*jzuber@aicos.com*>
Membres du groupe

- Laurent Burgbacher <*lb@alawa.ch*>
- Nicolas Dubois <*nicdub@gmx.ch*>
- Cédric Dutoit <*dutoitc@vtxnet.ch*>
- Nabil Hamadi <*hamadi12@yahoo.fr*>
- Dève Roux <*droux@eivd.ch*>
- Philippe Waelti <*pwaelti@eivd.ch*>

2 Choix

2.1 Langues du projet

Afin que ce projet puisse continuer et bénéficier du soutien de la communauté libre, la langue principale du projet est l'anglais. Par contre, ce document est en français car il n'est pas jugé utile à cette communauté.

2.2 Langages et environnements

2.2.1 Candidats

Les langages candidats sont :

- Python
- Java
- C++

2.2.2 Langage choisi

PyUt est développé en Python 2.2[Pyt]

Ses caractéristiques principales sont :

- Portable (Linux, Unix, Windows, MacOS)
- Interprété
- Orienté objet
- Dynamiquement typé
- Types de haut niveau natifs
  - Liste, pile
  - Dictionnaire (table de hachage)
- Supporte le paradigme fonctionnel
- Grande bibliothèque standard couvrant entre autres le réseau, XML, l'accès à des bases de données.
- Gratuit

Raisons de ce choix :

- Développement rapide
- Simplicité et puissance expressive du langage
- Portablilité
- Taille des librairies obligatoires < 3 Mo
- Apprentissage d'un nouveau langage intéressant
- Librairie graphique complète disponible (wxPython, voir 2.2.4)

2.2.3 Langages rejetés

Java
Raisons du rejet :

- Lourdeur
- Taille des runtimes obligatoires > 10 Mo

C++
C++ aurait pû être un bon choix également. La librairie graphique choisie est d'ailleurs écrite en C++. Mais la vitesse de développement est plus grande en Python, et le temps étant une ressource critique dans ce projet, nous avons préféré écarter C++.

2.2.4 Librairie graphique

Notre choix s'est porté sur *wxPython* 2.3.2[wxP]. Cet environnement est portable sous Windows, Unix et Linux. L'adaptation MacOS est en cours.

Raisons de ce choix :

- Portable
- Complet
- Bien documenté
- Libre (gratuit, sources disponibles)
- Bien connu de deux membres du groupe

3 Outils

3.1 CVS

CVS est un outil de gestion de versions, libre et portable. Il est déjà utilisé par plusieurs membres du groupe. Il est intégré à la suite d'outils *Cygwin*[ddC] qui est libre, et fait partie des utilitaires standards de Linux.

L'accès aux sources se fait par les lignes de commandes suivantes :

```
cvs -d :pserver:anonymous@cvs.pyut.sf.net:/cvsroot/pyut login
cvs -z3 -d :pserver:anonymous@cvs.pyut.sf.net:/cvsroot/pyut co pyut
```

Ceci crée un répertoire `pyut` dans le répertoire actuel, et récupère la hiérarchie des sources (y compris la documentation).

3.2 Sourceforge

Sourceforge[Sou] est un site web qui fournit gratuitement des outils de développement en ligne pour les concepteurs de logiciels libres. Ainsi, les sources du projet sont stoquées sur leur serveur (cf 3.1). Le site web du projet (cf 1 page 2) est également hébergé par Sourceforge.

3.3 LaTeX

Toute la documentation sera produite en LaTeX[pL] (ce document également). Plusieurs membres du groupe l'utilisent déjà.

Raisons du choix :

- Portabilité
- Format libre
- Qualité du résultat
- Particulièrement adapté à la création de rapports
- Export possible en HTML, pdf, postscript, text

4 Responsabilités

Chef de groupe Laurent Burgbacher
Chef de groupe remplaçant Philippe Wälti
Documentation Cédric Dutoit
Tests Nabil Hamadi
Qualité Dève Roux
Composants Philippe Wälti
Configuration Nicolas Dubois

5 Groupes

Dès le départ, nous avons formé trois groupes de deux personnes pour travailler sur les différentes parties de l'application.

5.1 Structure de données

Il s'agit de la modélisation et de la conception de la structure des données que manipule le programme.

- Dève Roux
- Laurent Burgbacher

5.2 wxOGL

wxOGL est la partie dessin vectoriel de wxPython (cf 2.2.4 page 3).

- Philippe Wälti
- Nabil Hamadi

5.3 Interface graphique

Design et implémentation de l'interface graphique utilisateur.

- Cédric Dutoit
- Nicolas Dubois

5.4 Répartition du travail

- Laurent Burgbacher
  - Implémentation d'une partie de la structure de données interne
  - Interface entre les différentes parties du programme (mediator)
  - Plugins dynamiques
  - Rapport général
- Nicolas Dubois
  - Dialogue d'édition des classes
  - Installeur Windows de PyUt
- Cédric Dutoit
  - Fenêtre principale
  - Base des menus
  - Dialogue d'édition des liens UML
  - Documentation utilisateur et aide en ligne
  - Site web du projet (cf section 1 page 2)
- Nabil Hamadi
  - Implémentation des classes UML graphiques
  - Tests automatisés
- Dève Roux
  - Implémentation d'une partie de la structure de données interne
  - Chargement/sauvegarde des documents en XML
  - Impression
  - Export bmp/jpeg/ps
- Philippe Wälti
  - Implémentation des liens UML graphiques
  - Interface entre les différentes parties du programme
  - Interface entre les liens UML et leur dialogue d'édition

6 Annexes

6.1 Documentation utilisateur

La documentation utilisateur a été produite en LaTeX afin de pouvoir être convertie facilement en HTML et PDF.

6.2 Documentation développeur

Python fournit un moyen puissant de documenter le code : les docstrings. Suivant la déclaration d'une classe ou méthode, le développeur peut ajouter une chaîne de caractères pour documenter la partie en question. Cette documentations peut-être consultée dans l'interpréteur Python en invoquant l'attribut `__doc__` de la classe/méthode en question.

L'utilitaire `pydoc` de la distribution standard est un petit serveur web. Il permet de consulter la documentation de manière claire, en l'extrayant dynamiquement du code. Ainsi, contrairement à Java (où il faut régénérer la documentation à l'aide de l'utilitaire `javadoc`), la documentation reste à jour par rapport au code.

Nous avons modifié `pydoc` (voir `bin/pydoc.py` dans la distribution de PyUt) afin qu'il mette en évidence des tags à la manière de `javadoc`. Il s'agit des tags suivants, qui sont utilisés pour chaque méthode :

author : auteur de la méthode
version : version de la classe (fixé par CVS (cf section 3.1 page 4))
since : version où apparait la méthode
param : paramètre d'entrée de la méthode
return : valeur de retour de la méthode
todo : indication de chose restant à faire pour la méthode

De plus, les premières lignes, sans tags, expliquent le but de la classe/méthode.

6.3 Installeur

Deux programmes sont utilisés pour créer l'installeur Windows. Tout d'abord, *py2exe*[Py2] qui convertit des fichiers sources Python en un programme exécutable. Puis, *InnoSetup*[Inn] qui est un générateur d'installeur. Basé sur un script généré en partie automatiquement par un wizard, il crée un seul exécutable qui installera tout le programme, à la manière de *Install Shield*. L'avantage de *InnoSetup* est qu'il est gratuit.

Sous linux, c'est le standard `rpm` qui est utilisé. PyUt est donc également fournit sous forme d'un paquetage `rpm` généré par les outils standards de Python (distutils library).

7 Conclusion

La première chose qui vient à l'esprit est le manque de temps. Nous regrettons de ne pas avoir eu quatre périodes par semaine, au lieu des deux prévues. En effet, il est difficile de se remettre dans le projet en si peu de temps, surtout lorsque qu'il faut chaque fois réinstaller les outils nécessaires sur les postes à disposition. Bien sûr, l'exercice est très intéressant. Le développement à six collaborateurs pose des problèmes d'organisation, mais le résultat final dépasse les problèmes rencontrés. Il est motivant de voir l'application progresser par le biais du travail des autres membres de l'équipe.

Nous avons beaucoup appris durant ce projet, que ce soit au niveau purement informatique (Python, wxPython, CVS, design patterns), au niveau de la communication au sein du groupe et de l'utilisation des compétences de chacuns. Le groupe est resté très soudé, certaines séances se poursuivant durant le repas de midi. L'auto-apprentissage a été très important, et les connaissances particulières des membres furent bien partagées.

Nous nous sommes heurtés à des problèmes difficiles, comme le manque de documentation de la librairie wxOGL (partie graphique). Puis, les vacances ont porté leurs fruits pour certains

d'entre nous, permettant des séances de travail plus longues et profitables.

Nous somme tous partants pour continuer ce projet, que ce soit dans le cadre de l'école, ou de notre côté. Nous pensons qu'il serait dommage de ne pas y donner suite, car les perspectives futurs sont intéressantes (reverse-engineering Python, génération de code...). Peut-être que le projet de groupe du deuxième semestre pourrait nous permettre continuer dans la lancée du premier ?

Références

[ddC] Environnement de développement Cygwin. *http://www.cygwin.com*.

[Inn] InnoSetup. *http://www.innosetup.com*.

[pL] Le projet LaTeX. *http://www.latex-project.org*.

[Py2] Py2exe. *http://startship.python.net/crew/theller/py2exe*.

[Pyt] Site Python. *http://www.python.org*.

[Sou] Site Sourceforge. *http://www.sf.net*.

[wxP] Site wxPython. *http://www.wxpython.org*.