Annexe à la spécification du 3ème délivrable

Ce document présente les consignes de documentation de la fin du projet COMP.

Documentation à rendre

La documentation à rendre au client consiste en un *rapport de fin de projet* (voir plus loin) et des *annexes techniques* (voir plus loin).

Le *rapport de fin de projet* et les *annexes techniques* sont consignés dans un répertoire unique (mais pouvant contenir des sous-répertoires), archivé et déposé sur la page Moodle du module avant le **dimanche 27 janvier à minuit** (n'hésitez-pas à le déposer le vendredi pour profiter de votre week-end). Le répertoire sera archivé en utilisant un **compresseur d'usage courant**. C'est le client qui juge ce qui est d'usage courant (ex. zip ; ne pensez même pas à 7zip). Le répertoire s'appellera **« projet COMP auteur1 auteur2 … ».**

Le client ne doit avoir à se poser aucune question en recevant vos documents ; ni comment ils sont organisés, ni comment les ouvrir, encore moins où trouver le décompresseur qui va bien. Il ne doit pas non plus avoir à renommer vos répertoires qui porteraient des noms trop peu discriminants (ex : projet_comp ou mon_projet). Ces recommandations sont évidemment valables à chaque fois que vous devez transmettre un dossier électronique.

1. Rapport de fin de projet

Le *rapport de fin de projet* comporte une *description technique* de votre produit (voir plus loin), plus un *bilan de gestion de projet* (voir plus loin). Ce rapport fait au total au moins 10 pages, et plus seulement si il est intéressant. Le rapport de fin de projet constitue un **unique fichier PDF** nommé **« rapport COMP auteur1 auteur2 … »**.

1.1. Description technique

La *description technique* de votre produit comporte la description du *schéma d'exécution* (voir plus loin) et de *l'architecture du logiciel* (voir plus loin), plus une justification de pourquoi ces choix, et une évaluation de ces choix, maintenant que les choses sont faites, les referiez-vous pareillement ? Il doit aussi comporter un commentaire sur l'évaluation de votre produit ; comment l'avez-vous validé, que couvrent vos tests ? Il s'agit ici d'un schéma de validation ; il faut le dire avec des mots, et ne pas se contenter de lister les cas de test. Ces mots peuvent être schématiques, mais pas ambigus. Faites comme si vous vous adressiez à un client que vous voulez convaincre que votre produit est intéressant.

1.1.1. Schéma d'exécution

La description du *schéma d'exécution* doit expliquer *schématiquement*, mais *sans ambiguïté*, comment un programme WHILE s'exécute en terme de votre langage cible.

Schématiquement : on n'a pas besoin des déclarations sauf si elles contiennent une décision d'implémentation qui vous semble cruciale. Il suffit que cela soit compréhensible par un homme de l'art.

Sans ambiguïté : vos schémas de traduction n'ont pas tous marché du premier coup. Cela signifie que dire qu'on produit du Java, du C++, du MIPS ou du Python ne suffit pas. Maintenant que le projet s'achève, vous savez ce qu'il faut dire pour ne pas refaire vos erreurs.

On ne peut pas se contenter de dire que le programme est sa spécification. On ne peut pas communiquer avec un client ou un donneur d'ordre sur cette base ! On doit être capable de discuter les choix de développement sans lire le programme, et même avant que le programme n'existe ! Là c'est plus facile puisque le programme est fait, mais normalement c'est fait avant, et c'est plus difficile.

1.1.2. Architecture logicielle

Une fois qu'on a décidé de comment exécuter un programme WHILE en terme du langage cible, il reste à décider de *l'architecture du logiciel* qui va traduire un programme WHILE en un programme cible.

Pour une part, l'architecture est dictée par l'utilisation de xText qui agit comme un *framework*. Cependant, xText ne dit pas tout. Par exemple, il est clair que le programme cible contient des parties qui dépendent du programme source (code généré), et d'autres qui n'en dépendent pas (prélude, postlude et libwh ; où sont définies ces dernières, sous quelle forme ?

Vous avez pu étoffer le run-time de WHILE (la libwh). Comment est rédigée cette libwh ; en WHILE, en langage cible ? Comment est-elle liée à l'exécutable que produit le compilateur ? Faites-vous attention à ne pas recompiler la libwh à chaque fois que vous compilez un programme ? Quel est le flot des fichiers créés soit à la main soit automatiquement ? Des informations sont extraites presque dès le début et servent jusqu'à la fin (ex. les noms des symboles lus dans le source WHILE) ; comment cette information est-elle transmise d'un bout du compilateur à l'autre ?

Ne pas oublier de mentionner les outils de productivité comme Makefile ou un équivalent, les scripts de test, etc.

Si vous aviez à diriger le travail de quelqu'un qui reprendrait le développement de ce logiciel, vous devriez lui dire quels sont les composants et comment ils sont implémentés et reliés entre eux. Certains sont des packages, d'autres des classes, d'autres des fichiers. Ce n'est pas à lui de deviner ! À nouveau, cela devrait être fait avant le développement, mais c'est plus facile à faire maintenant, après que vous soyez passés par la phase d'essais-erreurs.

1.2. Bilan de gestion de projet

Le *bilan de gestion de projet* rappelle les *étapes du développement*, et comporte un *rapport d'activité individuel* par participant (voir plus loin).

1.2.1. Étapes du développement

Comme la description technique, le bilan de gestion de projet comporte une analyse rétrospective ; qu'est-ce qui a bien marché, qu'est-ce qui n'a pas marché ? Comment referiez-vous maintenant ? Faites comme si vous vous adressiez à un client que vous voulez convaincre que votre gestion de projet est intéressante. N'oubliez-pas cependant que le client vous a suivi de façon hebdomadaire et qu'il a pris note de votre progression.

1.2.2. Rapports d'activité individuels

Les *rapports d'activité individuels* vous permettent de décrire votre rôle dans le développement du produit. En principe, on y retrouve des mots de l'architecture du logiciel ou du schéma d'exécution selon que vous étiez plutôt codeurs ou plutôt concepteurs. Sont à mentionner toutes les activités pertinentes pour le développement du projet : concevoir, coder, tester, documenter, … Quand un travail s'est fait à plusieurs que chacun mentionne le travail et les autres qui y participaient.

2. Annexe technique

L'annexe technique est constituée de l'ensemble des documents techniques de votre développement : programmes et documentations. Normalement, c'est quelque chose comme un répertoire. Cela devrait contenir un README bien en évidence pour expliquer ce qu'on doit y trouver.