Académie de Nice - Lycée général et technique Les Eucalyptus
Section de Technicien Supérieur (STS)
Systèmes numériques option Informatique et Réseaux (SNIR)

Épreuve U62 – Projet informatique – Méthodologie et attendus

Table des matières

Annexes **3**

A Rappel méthodologique **3**

A.1 Cyle de développement basé sur la méthode UML2 . . . 3

A.1.1 Analyse du cahiers des charges . . . 3

A.1.2 Cas d'utilisation - use cases . . . 4

A.1.3 Diagrammes de séquences . . . 4

A.1.4 Diagrammes d'activité . . . 4

A.1.5 Diagrammes de classes . . . 5

A.1.6 Diagrammes de déploiement . . . 5

A.1.7 Diagrammes d'état/transition . . . 5

A.1.8 Codage . . . 6

A.2 Contraintes de codage . . . 6

A.3 Conventions de nommage . . . 7

B Les revues de projet **8**

B.1 Revue 1 à +20 heures . . . 8

B.2 Revue 2 entre +50 et +60 heures . . . 8

B.3 Revue 3 à +100 heures . . . 9

C Modèle de présentation pour l'épreuve U62 du BTS **11**

C.1 Présentation - Tips and Hints . . . 11

C.2 Présentation générale . . . 11

C.3 Présentation individuelle . . . 11

D Dossier technique à réaliser **13**

D.1 Dossier d'études – Analyse des besoins . . . 13

D.2 Dossier d'études – Réalisation . . . 13

D.2.1 Modèle UML . . . 13

D.2.2 Codage . . . 13

D.2.3 Test . . . 14

D.2.4 Documentation utilisateur . . . 14

D.2.5 Présentation et démonstration . . . 14

D.3 Produit livrable - CDROM . . . 14

E Gestion des versions - SUBVERSION (Optionel) 15

Table des figures

1 Arborescence du Référentiel Subversion pour le projet 15
2 Arborescence du Référentiel Subversion pour cas d'utilisation 16

Annexes

Annexe A Rappel méthodologique

A.1 Cyle de développement basé sur la méthode UML2

Chaque document ou partie de document est soumis à des lectures croisées.

1. analyse du cahier des charges et du domaine applicatif : détermination des services, du dictionnaire de données (nom, rôle, domaine de définition, contraintes), contraintes fonctionnelles ; définition des protocoles (types, codage, message,etc.) de communication avec l'extérieur ;
2. diagramme d'objets : exemples de données utilisées dans le système ;
3. diagramme de déploiement du système : équipements nécessaires et leur inter connexions (voir OSI) ;
4. diagramme des cas d'utilisation : services rendus par le système, inter actions avec l'extérieur (acteurs primaire et secondaires) ;
5. diagramme de déploiement pour chaque cas d'utilisation : sous-ensemble du diagramme de déploiement système ;
6. diagramme de séquences systèmes ; traduction graphique de la description textuelle ;
7. diagramme de classes du domaines : basé sur le dictionnaire de données établi à l'étape 1, détermination des caractéristiques (*i.e.* propriétés, attributs, type,etc.) et opérations de chaque classes, relation entre classes (type de relation, navigabilité, cardinalité, rôles, contraintes, clef) ;
8. diagramme d'états des classes du domaine : comment évolue l'état (ensembles des attributs) d'une instance de classe ; uniquement pour les classes ayant plus de deux états ;
9. diagramme d'activités et/ou de séquences de chaque cas d'utilisation ; diagramme d'activités : flux de travail (work flow) et de flux de données (data flow), de séquences : dialogues entre les objets dans le temps ;
10. test de validation des cas d'utilisation ; cette étape permet de code la classe contrôleur associée au cas d'utilisation ;
11. diagramme de classes d'analyse : apparition des classes frontière (IHM, communication), entités (persistance, table dans un SGBDR) et contrôleur (diagramme d'activités et d'états) ; définition des test unitaires de chaque classe (compilation conditionnelle en C/C++) ;
12. diagramme de classes de conception : en fonction des librairies disponibles sur la plate forme d'exécution, mise en place de l'héritage et du polymorphisme ;
13. codage et tests unitaires ; classe d'analyse ;
14. test d'intégration : classe du domaine, groupe de classe du domaine ;
15. test de validation : cas d'utilisation ;
16. recette : acceptation du système par le client.

A.1.1 Analyse du cahiers des charges

- notés les contraintes : physique, temporelle, ergonomie, d'interface, de développement, etc ;
- lors de la lecture, synthétiser le cahier des charges par un tableau comme celui-ci :

sujets	verbes	compléments	adjectifs	valeurs	N° page

  - sujets et compléments sont des classes potentielles ;
  - verbes sont des cas d'utilisation ou des méthodes de classes ;
  - adjectifs dénotent d'éventuellement des attributs de classes ;
  - valeurs indiquent des types pour les attributs et éventuellement des attributs.

  Ce tableau est utilisé pour la rédaction du **dictionnaire de données** et donc pour la création du **diagramme de classes du domaine** ;
- noté les protocoles de communication : se procurer la documentation des protocoles, chercher les librairies pour le codage ;

A.1.2 Cas d'utilisation - use cases

Un cas d'utilisation est accompagné de son diagramme, de sa description et des interfaces avec les acteurs (IHM, communication, persistance). Un cas d'utilisation correspond à un service rendu par le système et est déclenchée par un acteur [1] ; voici un modèle de description à respecter [2] :

- objectif : quel service est rendu par le cas d'utilisation ;
- acteur : le rôle de chaque acteur primaire et/ou secondaire ;
- pré condition : état du système pour permettre la réalisation du cas d'utilisation ;
- description du scénario nominal : scénario dans lequel les alternatives et les exceptions ne sont pas prises en compte ; « tout est bien dans le meilleur des monde » ;
- description des scénarii alternatifs : sous-cas d'utilisation ⇢ « extend » ;
- description des scénarii d'exception : mauvais fonctionnement dû à un problème technique lié au diagramme de déploiement ;
- post condition : état du système à la fin du cas d'utilisation ; généralement les données qui ont été changées dans le système par le cas d'utilisation.

Définir le plan de validation du cas d'utilisation ; diagramme de déploiement à mettre en œuvre, cas de tests (stimuli, résultats attendus).

A.1.3 Diagrammes de séquences

1. diagramme de séquence système (assimilable au manuel utilisateur). Les acteurs principaux à gauche, les acteurs secondaires à droite ; une seul objet : **`Systeme`** ;
   Définir les protocoles (ensemble de messages, séquencement des messages) d'échanges de données entre les acteurs et le système. Élaborer les IHM graphiques.
   Raffiner le plan de validation du système.
2. diagramme de séquence du domaine : le diagramme de séquence système détaillé autrement dit l'instance du système est remplacée par des instances des classes du domaine (voir section A.1.5 page suivante point 1 page ci-contre) ;
   Raffiner les protocoles d'échanges ;
3. diagramme de séquence d'analyse : les classes du domaine sont remplacées par leur classes contrôleur, entité et frontière (voir section A.1.5 page suivante point 2 page ci-contre) ;
   Raffiner les protocoles d'échanges ;
4. diagramme de séquence de conception : les classes d'analyse héritent éventuellement de classes fournies par le langage de programmation ou les bibliothèques (voir section A.1.5 page suivante point 3 page ci-contre) ;
   Raffiner les protocoles d'échanges.

A.1.4 Diagrammes d'activité

Un diagramme d'activité pour chaque cas d'utilisation ; déroulement des étapes séquentielles ou parallèles mettant en évidence les classes du domaine et les acteurs ;

Dessiner, à main levée, les interfaces graphiques pour chaque action du diagramme d'activités. Dessiner également le graphe de navigation entre les différentes échanges – fenêtres, boîtes de dialogue, page HTML, autres – entre le système et les acteurs.

Dans le cas de communication, spécifier les messages émis et reçus : rôle, type, contenu.

Raffiner les plans de validation du système.

1. si le système propose une IHM, un cas d'utilisation est déclenché par l'action sur un objet graphique (widget) tel qu'un bouton, un élément de menu etc. de la fenêtre principale. Le service peut être également déclenché par un message reçu par une machine distante
2. ce schéma est souvent proposé par les outils de modélisation UML, par exmple sous BOUML, utiliser l'assistant : `Tools > Use case wizard` .

A.1.5 Diagrammes de classes

1. diagramme de classes du domaine : éclatement du système en classes ; Une classe de domaine peut correspondre à un acteur (principale ou secondaire), à une donnée (caractérisée par un état et un comportement) manipulée par le système.
   Compéter le dictionnaire de données – nom, signification, domaine de définition, utilisé par, contraintes (volume, temporelle,etc) –.
   Élaborer le test de validation de chaque classe du domaine.
2. diagramme de classes d'analyse (intégration du modèle M (modèle) V (vue) C (contrôleur) :
   - une classe contrôleur par cas d'utilisation et sous-cas d'utilisation : *include*, *extend* ; cette classe agrège les classes contrôleur des classes du domaine participantes au cas ;
   - une classe d'exception (suffixe *Exception*) par cas d'utilisation ;
   - une classe d'exception pour chaque scénario ; d'exception ; cette classe hérite de la classe d'exception du cas d'utilisation dans laquelle elle peut être lever ;
   - pour chaque classe du domaine :
     - une classe contrôleur (suffixe *Ctrl*). Cette classe est composée des classes frontières et de la classe entité ; le constructeur de cette classe initialise les instances des classes frontière et entité, autrement dit les paramètres formels du constructeur regroupent les paramètres formels des constructeurs des classes composantes.
     - une classe frontière (suffixe *Bny*) pour chaque acteur relié à la classe du domaine.
       - classe frontière (suffixe *IhmBny*) pour interface avec acteur humain ;
       - classe frontière (suffixe *ComBny*) pour interface avec une partie opérative ;
       - classe frontière (suffixe *Bny*) pour interface avec les autres types d'acteur ;
     - une classe entité (suffixe *Ent*) pour stocker les attributs de la classe du domaine. On y ajoute les accesseurs et mutateurs ; dans le cas de persistance dans une base de données relationnelle, une table est créée pour la classe ;
     - une classe d'exception (suffixe *Exception*) ; cette classe gère les cas d'erreurs engendrés par la classe ;

   Élaborer le plan de validation de chaque classe d'analyse.
3. diagramme de classes de conception : Les classes frontières et entités non nécessaires sont supprimées. On hérite de classes fournies par les librairies du langage de programmation (*e.g.* C++ STL) ou de classes fournies par des bibliothèques existantes (*e.g.* : LibSerial, Mysql++, Borland) ; Ce diagramme est un rétro ingénierie du code.
   Éliminer les plans de validation des classes d'analyse dérivées de classe de bibliothèques. **Note** : Ré utiliser les classes développées lors des TPs.

A.1.6 Diagrammes de déploiement

1. un diagramme de déploiement pour le système ;
2. un diagramme de déploiement pour chaque cas d'utilisation ; sous ensemble du diagramme de déploiement système ;
3. un diagramme de déploiement pour chaque cas de test ; diagramme de déploiement de cas d'utilisation plus environnement de test.

Élaborer des procédures de validation des configurations matérielles ; développer des scripts pour automatiser la validation.

A.1.7 Diagrammes d'état/transition

Uniquement pour les classes dont le comportement présente plus de deux états.

Raffiner les plans de validation des classes.

A.1.8 Codage

1. pour affiner la compréhension du système et tester les inter actions avec les acteurs , écrire un shell script pour chaque cas d'utilisation nommé **nomCasUtilisation.sh** implémentant le diagramme de séquence où d'activité du cas d'utilisation. Utiliser les commande Linux pour les algorithmes, les commandes **echo**, **read**, **cat** en conjonction avec les redirections et les fichiers `/dev/ ? ? ?` pour les entrées/sorties ;
2. convertir le shell script **nomCU.sh** en C/C++ **nomCU.c** en programmation impérative : remplacer les classes par des structures, utiliser les librairies standard C ;
3. raffiner le diagramme de classes du domaines (nom, attributs, opérations) ;
4. voir section précédente pour les classes d'analyse et de conception ;
5. pour les méthodes complexes, donner l'algorithme sous forme de pseudo code ;

Utiliser la méthode de génération avec les scripts shell associés de l'enveloppe du code fournis lors des travaux pratiques.

A.2 Contraintes de codage

- aucun « hard coding » n'est admis !
  - aucun littéral n'apparaît dans le corps des fonctions ;
  - seuls les chiffres 0 et 1 sont acceptés.
  - l'utilisation de fichier de configuration est fortement recommandée ; éventuellement les paramètres de l'application peut être stockés dans un SGBDR ;
  - l'utilisation des macros du pré processeur C est acceptable ; en C++, la création d'un classe abstraite contenant des constantes (`const static`) de classes est envisageable ;
- en programmation impérative, chaque fonction est codée dans un fichier source portant son nom et incluant son programme de test (la macro `NDEBUG` contrôle la prise en compte du `main()`) ; les fonctions sont regroupées dans des librairies statiques (`.a`, `.lib`) ;
- une classe C++ est codée dans deux fichiers différents :
  - une fichier de déclaration dit d'entête suffixé par `.h` ;
  - une fichier de définition suffixé par `.cpp` ;
- une classe Java est codée dans un seul fichier suffixé par `.java` ;
- chaque classe possède son programme de test unitaire (*i.e.* `main()`) dans son fichier de définition ; en C/C++, la compilation conditionnelle avec une macro `NOM_DE_DE_LA_CLASSE_UT` contrôle la prise en compte du `main()` ; la fonction `assert()` est utilisée pour la œuvre des matrices de test. Il est rappelé qu'un programme de test unitaire est autonome (aucune inter action avec un utilisateur) et qu'il n'affiche que les erreur détectées par `assert()`. Il peut être nécessaire de mettre en œuvre une émulation d'un diagramme de déploiement de test.
- l'utilisation de solution de test unitaire tel que Cunit ou Junit est conseillé ;
- chaque fichier source débute par un préambule similaire à celui-ci :

```
/*****************************************************************
* Projet : nomProjet
* Module : nomModule
* Auteur : nomEtudiant
* Nom du fichier : $HeadURL$
* Date de création :
* Modifié par :$LastChangedBy$
* Date de modification :$Date$
* Révision : $Revision$
* ***************************************************************
* Description :
*
*****************************************************************/
```

Les commentaires peuvent variés en fonction du langages de programmation. Les mots encadrés par le caractère $ sont des mots clé utilisés par Subversion, ils sont substitués automatiquement à chaque envoi du fichier dans le référentiel (*i.e.* check in, commit) :

- HeadURL : nom du fichier dans le référentiel,

  - LastChangedBy : nom de la personne ayant modifié le fichier en dernier,
  - Revision : numéro de version dans le référentiel ;
- le code source doit être indenté pour visualiser clairement sa structure ;

A.3 Conventions de nommage

Une convention de nommage dans la programmation informatique est un ensemble de règles destinées à choisir la séquence de caractères à utiliser pour les identifiants dans le code source et la documentation.

Les raisons pour lesquelles on utilise une convention de nommage (par opposition à l'autorisation accordée aux programmeurs de choisir n'importe quelle séquence de caractères) sont les suivantes :

- rendre le code source plus facile à lire et à comprendre avec moins d'efforts ;
- améliorer l'apparence du code source (par exemple, en interdisant les noms trop longs ou les abréviations) ;

Le choix de conventions de nommage peut prêter à d'énormes controverses, les partisans de chaque convention tenant la leur pour la meilleure, les autres étant inférieures. Voici celles à appliquer pour ce projet inspirées des conventions pour le langage de programmation Java utilisées par la société Oracle :

- un identifiant d'artéfacts (fichier, classe, variable, ...) utilise seulement les caractères [a-z], [A-Z], [0-9] et le point '.' : Ne pas utiliser de tiret '-', de souligné '_', d'espace, ou d'autres caractères ($, *, accents, ...) ;
- un nom de classe écrit en minuscule mais commence par une majuscule. Si le nom est constitué de plusieurs mots, la première lettre de chaque mot est une majuscule, exemple : **`MaClasse`**,
- une classe controleur est suffixe par **`Ctrl`**, exemple : **`MaClasseCtrl`**,
- une classe frontière est suffixe par **`Bny`**, exemple : **`MaClasseBny`**,
- une classe entité est suffixe par **`Ent`**, exemple : **`MaClasseEnt`**,
- un nom de variable ou d'attribut est écrit en minuscule. Si le nom est constitué de plusieurs mots, la première lettre de chaque mot est une majuscule, exemple : `maVariable`,
- un nom de fonction (ou méthode) est écrit en minuscule. Les verbes sont à privilégier. Si le nom est constitué de plusieurs mots, la première lettre de chaque mot est une majuscule, exemple : `maMethode`. Cette règle ne s'applique pas aux constructeurs (C++, Java) et destructeurs (C++) des classes.
- un nom de constante (*i.e.* static final) ou de macro C (*i.e.* define) est en majuscule. Si le nom est constitué de plusieurs mots, les mots sont séparés par le caractère « souligné (_) », exemple : `TOURNER_A_DROITE`,

Annexe B Les revues de projet

Après le lancement du projet (à +20 heures), à mi-projet (entre +50 et +60 heures), et durant la phase finale du projet (à +100 heures), un bilan doit mettre en évidence :

- ce qui a été réalisé ;
- ce qui reste à réaliser ;
- les ajustements éventuels, techniques ou relatifs au planning.

Outre l'intérêt des revues de projet pour accompagner l'étudiant dans une partie importante de sa formation, elles permettent de constater avec lui son niveau d'implication et l'avancement du projet. Elles permettent à l'équipe pédagogique de définir des étapes privilégiées pour construire l'**appréciation globale**, **objective** et **partagée**, qui accompagnera le dossier réalisé par l'étudiant.

C'est aussi un **moyen** qui permet à l'équipe pédagogique de constater les besoins des étudiants, et donc de proposer des éléments de formation complémentaires ou de remédiation.

B.1 Revue 1 à +20 heures

Objectif

La première revue de projet a pour objectif de vérifier la compréhension du travail demandé et la mise en œuvre du travail par les différents membres de l'équipe. Elle permet d'envisager quelques pistes de solutions. Elle se déroule de manière informelle avec le professeur référent.

Attendues – Informations à fournir

- le dictionnaire de données est rédigé ;
- les protocoles de communications sont définis ;
- les contraintes sont clairement identifiées ;
- la documentation des parties opératives a été synthétisé ;
- une ébauche de planning.

B.2 Revue 2 entre +50 et +60 heures

Objectif

La deuxième revue de projet permet de vérifier les solutions retenues ainsi que les essais qui permettent d'atteindre progressivement le fonctionnement désiré de la réalisation. Cette revue fait l'objet d'une présentation orale individuelle (avec support multimédia) et se déroule en présence d'un professeur de spécialité et d'un professeur de SPC.

Attendues – Informations à fournir

- descriptions des scénarii nominaux et alternatives de chaque cas d'utilisation (scénarii d'exception omis) ;
- diagrammes de séquence système pour chaque cas d'utilisation cohérents avec les descriptions ;
- dictionnaires de données : définition, protocole de communication entre modules fonctionnels ;
- diagramme de classes du domaines ;
- interface homme-machine complet (graphique et texte) avec la navigation sous la forme d'un diagramme d'activités ;
- analyse complète pour au moins un cas d'utilisation avec ébauche du code ;
- une analyse de la problématique liée à la physique appliquée.

Contenu de la revue
Conseils de réalisation de la présentation multi média :

1. une présentation de 15 mn ; Contraintes de réalisation :
   - utiliser Libreoffice Draw ;
   - aucun modèle graphique (pas d'arrière plan) ;
   - utiliser la totalité de l'espace d'un transparent : marges minimum ;
   - bas de page : numéro de diapositives, votre nom, nom de projet, nom du module ;
   - le contenu est essentiellement constitué de dessin ;
   - les phrases longues sont interdites (une ligne maximum) ;
   - une dizaine de diapositives.

   Modèle proposé pour le contenu de la présentation :
   (a) titre du projet, noms des membres (couleurs associées) ;
   (b) présentation générale du système : parties opératives, fonctionnalités ; préciser les parties/fonctionnalités qui vous sont propres ;
   (c) la problématique, sous forme d'un dessin, faisant apparaître les éléments (avec leur définition), les protocoles de communication (contrainte, format), les contraintes (physique, temporelle, mécanique), l'aspect de physique appliquée ;
   (d) les cas d'utilisation de votre responsabilité (éventuellement interaction avec d'autres cas d'utilisation) ;
   (e) le diagramme de déploiement (avec composants) qui vous concernent (*i.e.* uniquement les nœuds que vous utilisez), y incorporer les informations de communication : configuration, protocole (trame, fonction modbus,...)
   (f) pour un cas d'utilisation (CU) choisi (donner son nom)
      i. diagramme de séquence système et/ou activité cohérent avec la description textuelle ;
      ii. interface homme-machine ; l'enchaînement est montré par un diagramme d'activités ;
      iii. diagramme de classes du domaine propre à ce CU : être capable de justifier les classes et les associations ;
      iv. diagramme de séquence du domaine et/ou activité : toutes les classes du domaines doivent apparaître, toutes les méthodes des classes du domaines doivent apparaître, utiliser les cadres d'interaction « référence » pour diminuer la complexité ;
   (g) les test de validation (diagramme de déploiement, stimuli, résultats attendus) ;
   (h) le planning GANTT : tâches réalisées, tâches futures ; on doit voir les phases de développement : analyse, conception, test (unitaires, d'intégration, de validation), les dates des revues 1 et 2, la date de remise du dossier, etc ;
   (i) les problèmes rencontrés.
2. une séance de questions/réponses portant essentiellement sur la problématique et les diagrammes UML.

B.3 Revue 3 à +100 heures

Objectif
La troisième revue de projet permet d'évaluer le niveau d'avancement du projet, d'élaborer une procédure de recette globale de la réalisation et l'intégration de sa partie dans ce qui sera présenté, lors de l'épreuve, devant la commission d'interrogation. Cette revue fait l'objet d'une présentation orale individuelle (avec support multimédia) et se déroule en présence d'un professeur de spécialité associé à un autre professeur de spécialité ou un professeur de SPC, en fonction de la spécificité du projet.

Attendues – Informations à fournir

Le candidat présente un cas d'utilisation fonctionnel (tester et valider) ; il doit intégrer une IHM (texte ou graphique), une logique de contrôle, une communication réseau et une persistance de données (fichier ou SGBDR).

La documentation du code, au format HTML, généré par Doxygen est à livrer.

Contenu de la revue

- une présentation de 10 mn ; 4 transparents maximum :
  1. le titre de la démonstration, le diagramme de déploiement (avec composants) mise en œuvre intégrant (y inclure les images des POs), une synthèse des protocoles de communication (type, trame,etc.), une présentation de la persistance des données (*e.g.* DTD XML, schéma relationnel) ;
  2. contexte du programme : positionnement dans le diagramme des cas d'utilisation, diagramme de classes implémenté (codé), diagramme de séquence et/ou d'activité codé ;
  3. planning présentant le déroulement dans le temps du développement du programme démontré et le travail restant à faire ;
- une démonstration de 10 mn
  1. présentation de l'environnement opérationnel réel : diagramme de déploiement/composants, inter connexions ;
  2. scénario de la démonstration ;
  3. exécution ;
- une séance questions/réponses de 20 mn : portant essentiellement sur le code ;

Annexe C Modèle de présentation pour l'épreuve U62 du BTS

C.1 Présentation - Tips and Hints

- pas de fioriture ;
- charte graphique simple : fond blanc, texte noir (à améliorer s'il reste du temps) ;
- pas de phrases ; le texte donne une direction au discours mais ne se suffit pas à lui même ;
- beaucoup de dessin simple (schématique) ;
- l'étudiant montre qu'il connait et maitrise son sujet ; éventuellement des notes sur feuille de papier numérotés écrit en GROS ;

C.2 Présentation générale

Objectif

- rappel du cahier des charges : celui-ci est connu du jury ;
- qui a fait quoi ;

Contenu

1. titre - noms/numéros des étudiants (couleurs différentes)
2. sommaire
3. la problématique : pourquoi le système est développé, quel(s) problème(s) sont résolu(s) par le système ; un dessin, pas de texte ;
4. besoins – QUOI, QUI (sans UML) engendrés par la problématique ; faire apparaître la répartition par étudiant et par module ;
5. architecture matérielle : photo des parties opératives, inter connexion (physique et protocoles utilisés) ; sans UML ; faire apparaître la répartition par étudiant et par sous systèmes ;
6. architecture logicielle (synoptique) : description générale des sous systèmes, fonctionnalités, introduction des sigles, responsabilité de chaque étudiant (le must : chemise de couleur correspondante) ; échange de donnée ; toujours pas d'UML ;
7. état d'avancement ; total, par sous système, par phase de développement (analyse, conception, tests unitaires, tests d'intégration, etc.) ; utilisation du diagramme des cas d'utilisation en mettant pour chaque cas d'utilisation son état d'avancement en pourcent ;
8. analyse générale – UML
   (a) cas d'utilisation ; mise en évidence des sous système et des responsabilités ;
   (b) Diagramme de déploiement : machine, SE, inter connexion (sans trop de détails) ;
9. Problèmes rencontrés : documentation des parties opérative, absente des parties opérative, étudiant absent etc ;
10. démonstration générale : diagramme des cas d'utilisation démontré, diagramme de déploiement, scénario sous forme d'un diagramme de séquence ou d'activité.

C.3 Présentation individuelle

Objectif

Il s'agit, pour le candidat, dans les différentes activités de conduite d'un projet, de présenter son travail, de montrer son savoir-faire et de justifier ses choix.

Contenu

1. nom du projet, nom du sous système, sigle du sous système – nom étudiant (couleur) ;
2. sommaire ;
3. problématique/objectif du sous système ; dictionnaire de données : expliquer (dessin) les données traitées et les algorithmes mathématiques mis en œuvre (i.e. cycle, trajectoires théorique/réelle, poussée,...) ; justification des choix ; modèle logique de la BD/format des fichiers d'échange (XML DTD, example) ou de paramètres ; sans UML ;
4. contraintes de développement : système d'exploitation : nom, version ; langage(s) de programmation, EDI (nom, version), SGBD (nom, version), bibliothèques (nom, version), documentation (bouml, doxygen), autres (HTML, XML,...) ;
5. Analyse UML
   (a) Diagramme de déploiement détaillé : photo parties opératives, inter connexion (physique, protocole détaillé, *i.e*. paramètres voie série,, adresse IP, port, structure trame, fonction modbus,...) ;
   (b) Cas d'utilisation généraux avec mise en évidence des cas d'utilisation sous sa responsabilité ;
   (c) Ses cas d'utilisation détaillés : préciser ceux qui ont été réalisés ;
   (d) planning (GANTT) ;
   (e) diagramme de séquences système de chaque cas d'utilisation réalisé ; y faire apparaître les IHMs et/ou les messages de communication ;
   (f) diagramme de classes du domaine : uniquement les classes utilisées dans le sous système ;
   (g) diagramme d'activité/séquence de chaque cas d'utilisation réalisé ; y faire apparaître les IHMs et/ou les messages de communication ;
6. Conception
   (a) architecture temps réelle/multi tâche (thread) : thread, sémaphore, mutex, signaux, mémoire partagée, etc ; diagramme d'activités (nœud d'action : send signal, accept event, accept time event ; nœud de bifurcation, d'union) ;
   (b) diagramme de classes de conception (héritage bibliothèque) : ingénierie inversée du code ;
   (c) diagramme de séquence/activité des cas d'utilisation : représente les lignes de code ;
   (d) diagramme d'état d'une classe (à choisir)/Code : attention le code doit correspondre !
7. Test
   (a) technique de test unitaire : simulation, diagramme de déploiement ;
   (b) test d'intégration et de validation : diagramme de déploiement, simulation ;
8. Problèmes rencontrés : documentation des parties opérative, absente des parties opérative ;
9. Démonstration : diagramme des cas d'utilisation démontré, diagramme de déploiement, scénario sous forme d'un diagramme de séquence ou d'activité.

Démonstration

1. montrer le système physique mise en œuvre correspondant au diagramme de déploiement/composant ;
2. vérifier le diagramme de déploiement : ping, script, telnet (vérification port), script ;
3. exécution – expliquer ce qui va se passer : action/résultat attendu, résultat réel ;

Annexe D Dossier technique à réaliser

À l'issue du projet, l'équipe d'étudiants remet au centre d'examen un dossier technique unique représentatif de l'ensemble du projet. Ce dossier comprend une partie commune à tous les membres de l'équipe et la partie personnelle traitée par chacun d'entre eux. Le sujet du projet a précisé la répartition des Tâches professionnelles entre celles qui devaient être réalisées de façon commune et celles qui devaient être réservées à une action individuelle. Dans les 30 pages au maximum qui sont allouées à chaque étudiant, et dans le cadre de son autonomie de réflexion et d'action au sein du projet, il est souhaitable qu'une partie de ce qu'il rédige puisse montrer sa participation à une réflexion commune. L'autre partie contiendra les éléments qui permettront d'évaluer son action individuelle. Le dossier technique du projet réalisé par un groupe d'étudiants peut donc être constitué comme suit :

- Partie commune : (de 20 à 30 pages)
  - introduction, situation du projet dans son contexte industriel ;
  - dossier de spécifications ;
  - dossier d'étude préliminaire et plan de tests des performances au regard du cahier des charges. Suivant la nature du projet et ses points d'entrée, certains éléments de ce dossier peuvent être présents dans les parties personnelles ;
  - éléments nécessaires à la recette de la maquette ou du prototype final ;
  - résultats des essais de la maquette ou du prototype final ;
  - conclusion par rapport au cahier des charges fourni par le donneur d'ordre : test intégration, procédure et résultats de la recette.
- Partie personnelle : (de 20 à 30 pages)
  - situation de la partie personnelle dans l'ensemble du projet ;
  - dossier d'étude et de réalisation détaillée, tests unitaires. En fonction des spécificités du projet et des contraintes de documentation imposées par le cahier des charges, des documents annexes peuvent être joints sous forme électronique (annexes techniques, programmes complets, manuel d'utilisation, notice de maintenance, sources complets,. . .).

D.1 Dossier d'études – Analyse des besoins

- protocole de communication ;
- dictionnaire de données ;
- cas d'utilisation ;
- diagramme de séquence système pour chaque cas d'utilisation ;
- diagramme de classes du domaine ;
- diagramme de séquence et d'activité du domaine pour chaque cas d'utilisation ;
- diagramme de déploiement système et cas d'utilisation.

D.2 Dossier d'études – Réalisation

D.2.1 Modèle UML

- diagramme de classes d'analyse ;
- diagramme de classes de conception ;
- diagramme de séquence/d'activité d'analyse ;
- diagramme de séquence/d'activité de conception ; rétro ingénierie du code ;
- format des données des fichiers de configuration, de persistance, . . .

Schéma de la base de données relationnelle, modèle logique de données basé sur le digramme de classes de domaine.

D.2.2 Codage

- fichiers sources (*e.g.* C++, Java, SQL, HTML, CSS, XML, autres) ;

D.2.3 Test

- plan de validation du système, des classes du domaine ;
- programme de test unitaire [3] des classes d'analyse contrôleur et frontières non IHM) ;
- programme de test pour chaque classe du domaine mettant en œuvre le plan de validation de la classe ;
- programme de test pour chaque cas d'utilisation mettant en œuvre le plan de validation du système ;
- résultats des tests.

D.2.4 Documentation utilisateur

- manuel d'installation du système et éventuellement des sous systèmes ;
- manuel d'utilisation du système et éventuellement des sous systèmes ;
- notice de maintenance du système et éventuellement des sous systèmes.

D.2.5 Présentation et démonstration

- présentation réalisée avec LibreOffice ; la présentation utilisée est au format PDF ;
  1. besoins : cas d'utilisation, diagramme de séquence système ;
  2. configuration matérielle et réseau : diagramme de déploiement ;
  3. contraintes : matérielles, logicielle, de développement, temporelles ;
  4. architecture : diagramme de classes du domaine, diagramme de séquence/activité ;
  5. conception : diagramme de classes de conception ;
  6. problèmes rencontrés ;
  7. conclusion.
- démonstration.

D.3 Produit livrable - CDROM

Cette section décrit les livrables à mettre à la disposition du client sous forme papier et informatique, L'équipe de projet doit remettre au jury :

- les artefacts listés dans les sections précédemment ;
- les programmes d'installation ;
- les ressources externes utilisées : programme de test, bibliothèques logicielles, etc ;
- bibliographies : site Web, livres, etc.

3. `int main (int argc, char** argv)`

Annexe E Gestion des versions - SUBVERSION (Optionel)

Un référentiel est créé par l'enseignant sur le serveur. Le travail en cours de chaque module (*i.e.* sous-système) est enregistré dans un répertoire appelé `trunk`. Une opération « commit » est effectuée en fin de chaque séance de TP avec un commentaire long détaillant les activités réalisées dans la séance.

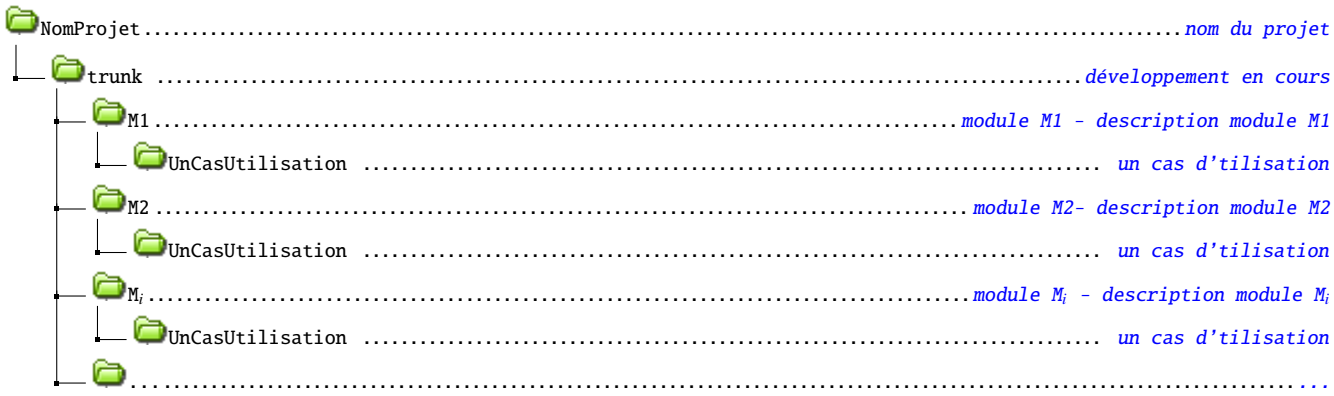

Figure 1: Arborescence du Référentiel Subversion pour le projet

Pour chaque cas d'utilisation, une arborescence de ce type est conseillée :

Figure 2: Arborescence du Référentiel Subversion pour cas d'utilisation