**Projet Génie Logiciel**

**ENSC 2015-2016**

*Documentation technique*

Contenu

[I. Gestion du projet 3](#_Toc438129548)

[a. Contexte et méthode 3](#_Toc438129549)

[b. Planning 3](#_Toc438129550)

[II. UML 4](#_Toc438129551)

[a. Structure 4](#_Toc438129552)

[b. Justifications 5](#_Toc438129553)

[III. Description du programme 5](#_Toc438129554)

[a. Définition des classes 5](#_Toc438129555)

[b. Fiches Form 6](#_Toc438129556)

[1. Fiche Mission 6](#_Toc438129557)

[2. Fiche SelectAstroDelete 7](#_Toc438129558)

[3. Fiche TaskView 8](#_Toc438129559)

[4. Fiche TaskForm 8](#_Toc438129560)

[5. Fiche DayReport 9](#_Toc438129561)

[6. Fiche MissionMap 10](#_Toc438129562)

[7. Fiche LoadMission 10](#_Toc438129563)

[IV. Tests 10](#_Toc438129564)

[a. Tests unitaires 10](#_Toc438129565)

[b. Tests fonctionnels 10](#_Toc438129566)

[c. Résultats 12](#_Toc438129567)

[V. Résultats 12](#_Toc438129568)

[a. Bilan 12](#_Toc438129569)

[b. Evolutivité 12](#_Toc438129570)

[Conclusion 12](#_Toc438129571)

# Gestion du projet

## Contexte et méthode

Ce projet s’inscrit dans l’Unité d’Enseignement de Génie Logiciel, en deuxième année de l’Ecole Nationale Supérieure de Cognitique. L’objectif de ce projet est de réaliser une application sous Windows Form permettant la gestion d’une base martienne abritant des astronautes.

Cette application doit pouvoir être utilisée par les astronautes sur place et par le centre de contrôle sur Terre. Elle doit permettre, entre autre, la gestion des astronautes, de leur emploi du temps et de leurs tâches. Elle doit également être facile et conviviale d’utilisation.

Nous nous sommes servis de GitHub pour travailler ensemble sur ce projet. GitHub est un outil de gestion de développement, qui permet d’accéder au dossier du projet partout et de pouvoir modifier les documents et le code sans conflit. Cela a facilité la répartition des tâches car l’un de nous pouvait se permettre de travailler sur une partie du projet sans modifier ce que l’autre a fait jusque-là.

Nous avons décidé de faire une interface anglaise. Le projet étant dans un contexte international, cela nous semblait pertinent de faire une interface utilisable par le plus grand monde. Néanmoins, pour nous simplifier la tâche et faciliter la communication, le reste du projet est en français.

## Planning

Ce projet a commencé le 7 octobre 2015 et doit être terminé pour le 18 décembre. Le planning détaillé est donné ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| Tâches | Dates | Description | Qui |
| Définition du sujet | 7 – 14 octobre | Définir ce qui est demandé, quelles sont les fonctionnalités attendues et les limites du sujet. Organisation des tâches, définition d’un premier planning. | Jérémy & Florian |
| Définition des classes UML et de l’architecture | 14 octobre – 2 novembre | Etablir l’architecture du projet et des données. Définir l’organisation du fichier XML contenant les données.  Définir l’architecture de l’application. | Jérémy & Florian |
| Rédaction du code | 2 novembre – 13 décembre | Phase de développement de l’interface. Mise en place des fonctionnalités spécifiées et prise en compte des directives formulées lors des définitions et dans le cahier des charges. | Jérémy & Florian |
| Elaboration des tests fonctionnels | 6 – 10 décembre | Création de scénarios visant à tester les fonctionnalités principales et / ou critiques. | Jérémy & Florian |
| Passation des tests fonctionnels | 12– 16 décembre | Passation des tests par plusieurs utilisateurs sélectionnés au préalable. Enregistrement des résultats. | Jérémy & Florian |
| Rédaction du rapport | 6 – 16 décembre | Rédaction de la documentation technique telle que demandée dans le cahier des charges du projet. Reprise des fichiers modifiés pour ajouter des commentaires. | Jérémy & Florian |
| Relecture & Recettage | 16 – 18 décembre | Vérifications pour nettoyer le code et qu’il n’y ait pas de fautes. Vérifier que nos choix et que nos formulations soient cohérentes. | Jérémy & Florian |

# UML

## Structure

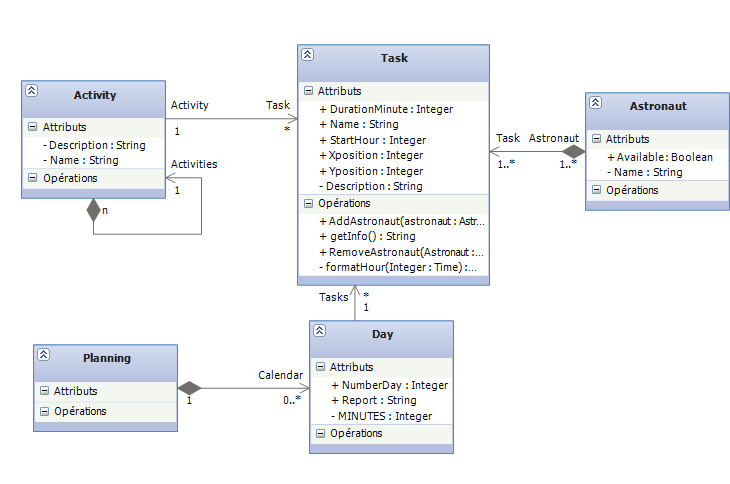


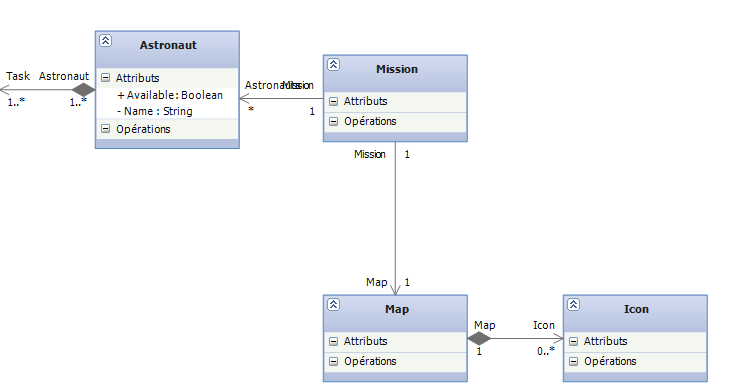
Figure : Structure de l'UML, partie 1

Figure : Structure de l'UML, partie 2

## Justifications

L’UML de notre projet est relativement simple. La classe centrale est la classe *Task* qui correspond à une tâche, à un moment donné. Ce que les astronautes font correspondent à des tâches, et ce qui remplit leur planning. C’est pour cela que c’est, de loin, la classe avec le plus d’attributs et de méthodes. Une *Task* correspond à l’*Activity* que doivent réaliser un ou plusieurs *Astronaut(s)* un certain *Day.* La définition de la *Task* justifie donc les relations qu’elle a avec les autres classes. Une *Task* est définit par son commencement (*StartHour,* qui correspond à des minutes) et sa durée (*DurationMinute*). Elle a un nom, une description (telle que décrite dans le cahier des charges) et une localisation en X et Y pour la repérer sur la carte.

Une *Activity* correspond aux différentes activités telles que définies dans le cahier des charges du projet. Cela inclut par exemple « Repair », « Medical Act » ou « Private ». Comme nous avons une relation de hiérarchie entre certaines activités (comme Science 🡪 Exploration 🡪 Vehicule), nous avons une relation d’association entre *Activity* et elle-même (formalisée par l’attribut *Activities*).

Un *Day* correspond aux jours, tels qu’affichés sur le niveau 1 de l’interface. Chaque jour a donc un numéro précis, *NumberDay.* A chaque jour est associé un rapport, *Report,* tel que décrit dans le cahier des charges. L’attribut *MINUTES* correspond à la durée en minutes du jour Martien, ce qui est essentiel pour d’autres méthodes dans le programme.

# Description du programme

## Définition des classes

Notre programme est composé de 9 classes qui sont des Forms permettant d’interagir avec l’utilisateur et de 7 classes afin de pouvoir modéliser notre solution.

## Fiches Form

### Fiche Mission

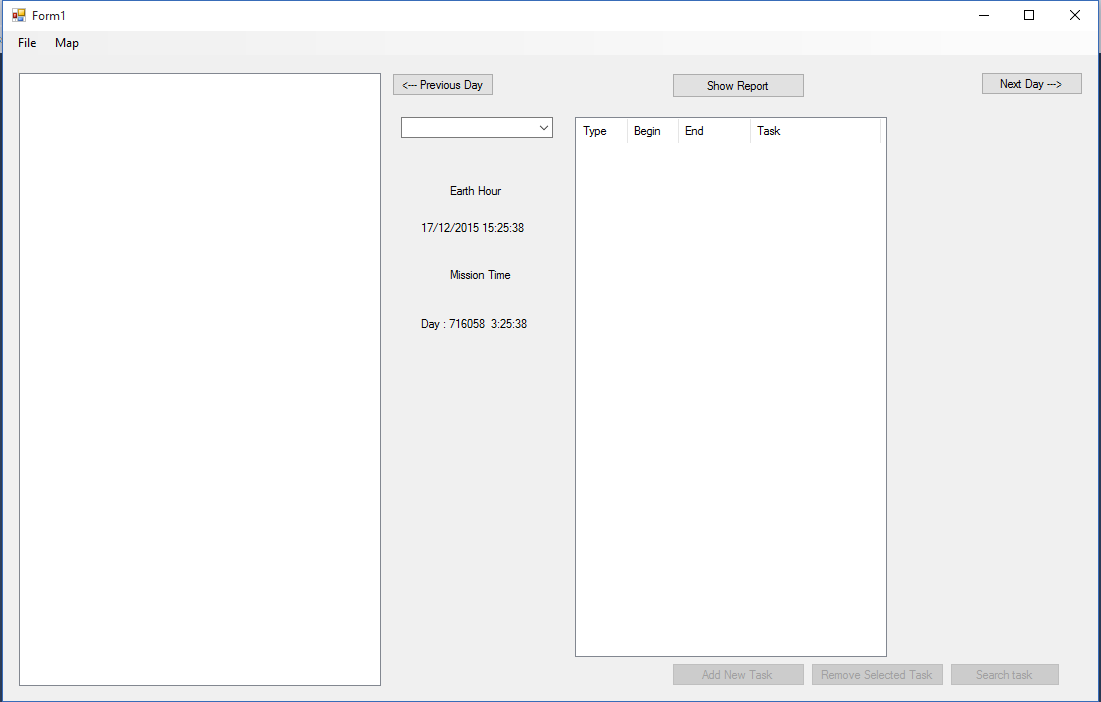


Figure : Fiche Mission, sans planning

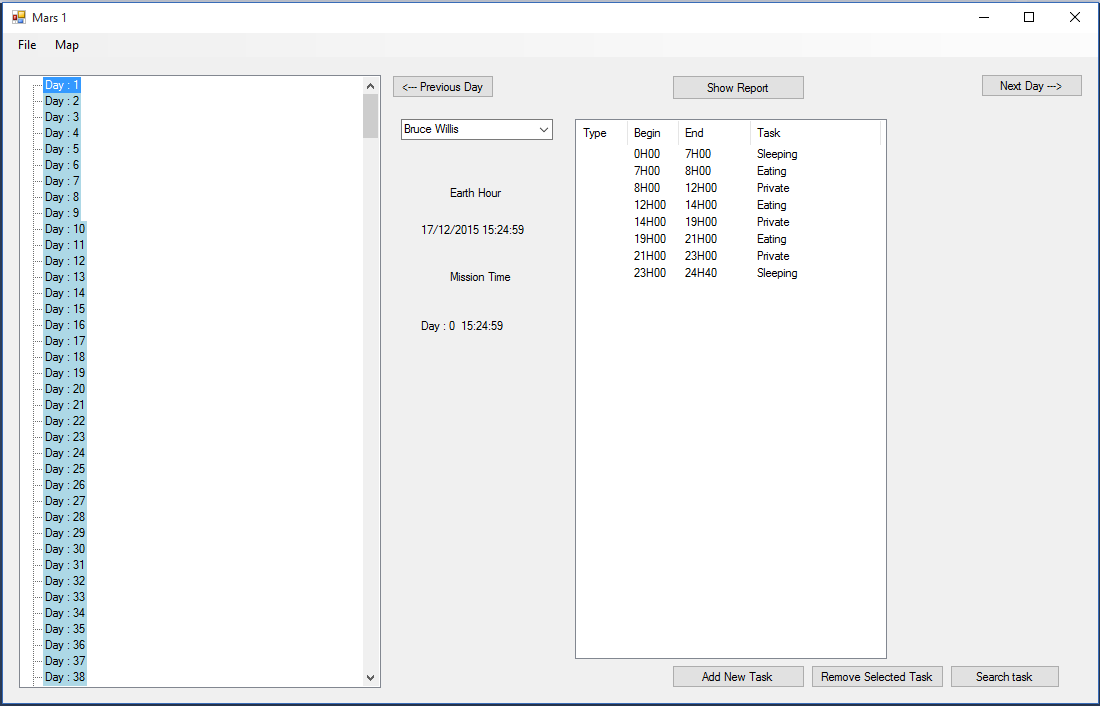


Figure : Fiche Mission, avec planning

Ceci est la principale fiche du programme. Elle permet l’affichage des deux premiers niveaux d’interfaces tels que décrits dans le cahier des charges. Cet affichage se fait via deux *SplitContainer,* chacun contenant un panel (et un niveau).

A gauche, il y a le niveau 1 de l’interface, c’est-à-dire l’affichage des jours. Cet affichage est fait par une *TreeView* (nommée ListCalendar), où chaque item – chaque jour donc – est créé lors du chargement de l’XML de mission. Les jours déjà passés sont affichés en gris, le jour actuel est en vert et les jours futurs sont en bleu. Lorsqu’un jour est sélectionné, le niveau 2 de l’interface correspond à ce jour est affiché sur le panel de droite. Ce panel est constitué des boutons de navigation usuels mais surtout d’une *ListView* affichant les différentes tâches de la journée. Y sont affichées :

* Le type d’activité (TODO : c’est quoi ?)
* Son début et sa fin
* La tâche en question

Double-cliquer sur une tâche affiche la fiche *TaskView*. Cliquer sur le bouton Add New Task affiche la fiche *TaskForm.* Cliquer sur le bouton Remove Selected Task affiche la fiche *SelectAstroDelete*. Cliquer sur le bouton Show Report affiche la fiche *DayReport* associée au jour sélectionné.

### Fiche SelectAstroDelete

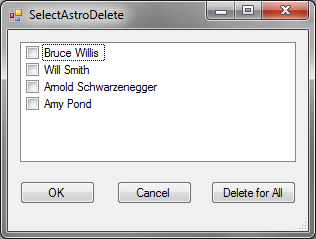


Figure : Fiche SelectAstroDelete

Cette fiche s’ouvre sur confirmation de la suppression d’une tâche depuis le bouton « Delete Selected Task » depuis la fiche *Mission*. Elle permet de gérer la suppression d’une tâche dans l’emploi du temps des astronautes. On peut ainsi supprimer cette tâche pour tous les astronautes avec le bouton Delete for All, ou bien choisir un ou des astronautes manuellement.

### Fiche TaskView

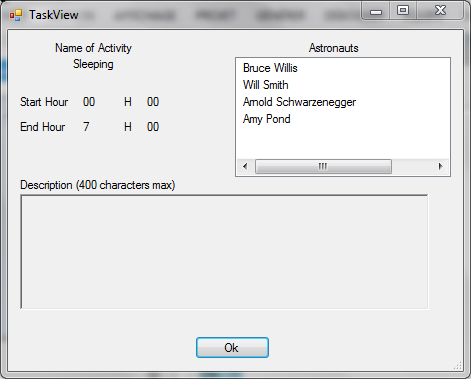


Figure : Fiche TaskView

### Fiche TaskForm

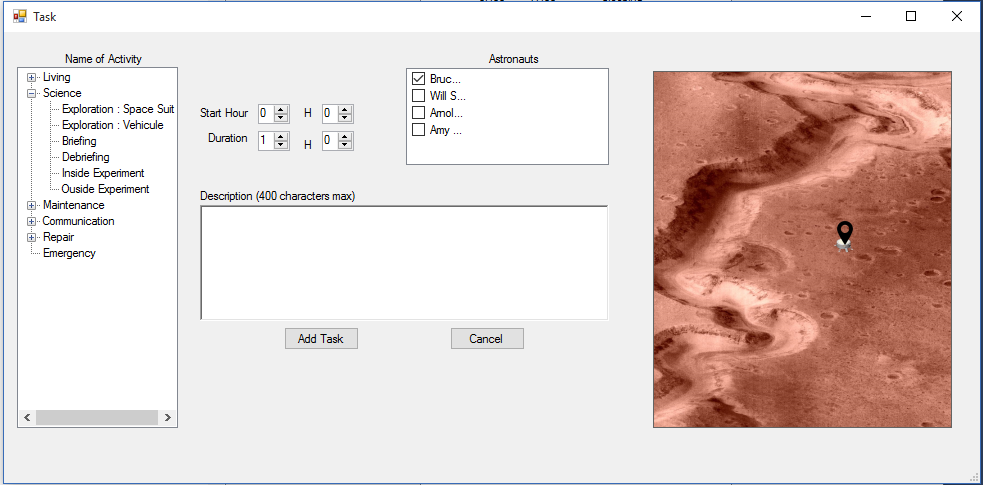


Figure : Fiche TaskForm

### Fiche DayReport

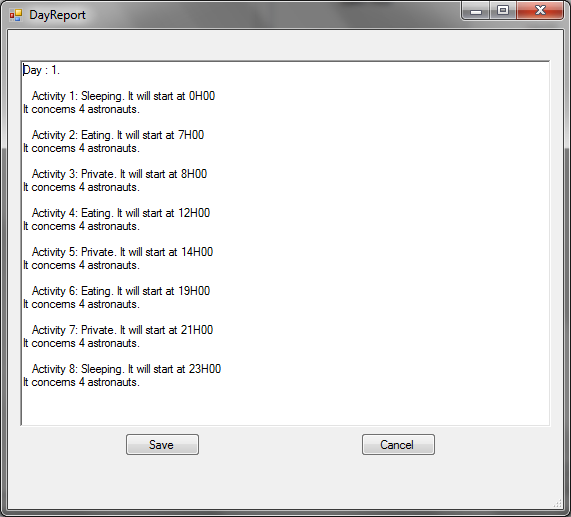


Figure : Fiche DayReport

Cette fiche s’ouvre lorsque l’utilisateur clique sur le bouton « Day Report » dans la fiche *Mission.* Elle correspond à l’attribut « Report » de la classe *Day*. Sur lancement de cette fiche (et à chaque fois que ce rapport est vidé par l’utilisateur), un rapport de base est chargé, tel que montré ci-dessous. L’utilisateur peut le modifier, et ces changements seront sauvegardés.

### Fiche MissionMap

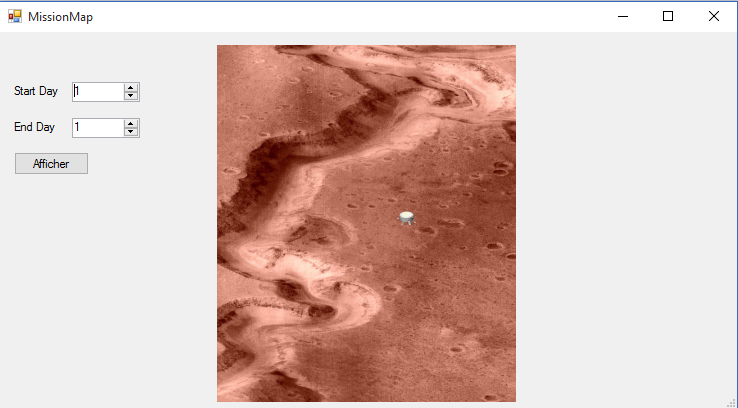


Figure : Fiche MissionMap

### Fiche LoadMission



# Tests

## Tests unitaires

## Tests fonctionnels

#### Test 1 : Gestion de tâches courantes

Ce qui va être testé :

* Lecture du compte-rendu
* Edition du compte-rendu
* Suppression et ajout d’une tâche
* Gestion des horaires
* Limite de caractères sur le descriptif
* Edition d’une tâche
* Sauvegarde

L’utilisateur lance l’application avec le fichier de base pré-installé. Il se rendra au jour en cours. Il ouvrira le compte-rendu pour ce jour. Il l’éditera en ajoutant : *« Communications avec la terre, compte-rendu oral* ». Il validera et sauvegardera le changement.

Il supprimera l’activité « Private » en 21h00 et 23h00 pour tous les astronautes. Il ajoutera une activité nommée « Debriefing » entre 20h30 et 22h00 pour tous les astronautes, sur la station des astronautes et avec ce descriptif :

*“Simple debriefing with the control center of planet Earth. Regular check of various functions, of the vehicles and of the experiments led during the mission. In case of emergencies, contact the control center as soon as possible.*

*Every member has to be present for this task, so that control center can send news to the astronauts' families. Not all of them have to be present for the whole task though, but we ask for some presence.”*

En cas d’erreur sur l’horaire, il ajoutera cette activité entre 21h00 et 22h30. En cas d’erreur sur la description, il ajoutera cette activité avec le descriptif suivant :

*“Simple debriefing with the control center of planet Earth.”*

Une fois l’activité validée, il sera ramené sur le menu principal. L’utilisateur sauvegardera alors la mission en sélectionnant l’option *Save* dans le menu *File* dans le menu de l’application

#### Test 2 : Chargement Recherches

Ce qui va être testé :

* Chargement d’une mission précédemment enregistrée
* Ajout d’une mission en extérieur
* Recherche sur la carte une période donnée
* Recherche sur les jours pour un mot-clé donné sur une période donnée

L’utilisateur lance l’application et va charger la mission précédemment enregistrée en sélectionnant le fichier décrivant la mission et le fichier sauvegardé. Une fois cela effectuer, l’utilisateur vérifiera que les modifications apportées à l’emploi du temps de base aient bien été pris en compte dans le chargement.

Une fois sur le jour en cours, il supprimera l’activité *Private* entre 14h00 et XXhXX pour les deux Astronautes Amy Pond et Bruce Willis. Ensuite il ajoutera sur cette plage horaire une activité *Outside Experiment* en sélectionnant les deux Astronautes. Avant de valider il double cliquera sur la carte à droite et indiquera un lieu autre que la base de la mission.

Une fois cette étape effectué l’utilisateur pourra valider l’ajout de la tâche.

Une fois ce la effectué, l’utilisateur cliquera sur **Map Button**  et sur la nouvelle fenêtre sélectionnera une plage de Jour comprenant celui ou a été ajouté en extérieur. En cliquant sur le bouton ***Afficher*** l’utilisateur verra alors s’afficher un (ou plusieurs) marqueur indiquant la position des missions en extérieurs. En cliquant sur un de ces marqueurs, l’utilisateurs verra apparaître une fenêtre donnant les détails de la tâches.

Une fois cela effectué, l’utilisateur fermera la fenêtre de la carte. Il cliquera ensuite sur le bouton *Recherche* ce qui ouvrira l’interface de Recherche. En écrivant le nom d’une tâche (*Outside Experiment* par exemple) ce qui affichera les tâche portant ce nom classé par ordre chronologique.

#### Test 3 : Erreur à l’ouverture

Ce qui va être testé :

* Gestion des erreurs en cas d’absence du fichier XML d’initialisation dans le dossier de l’application

## Résultats

# Résultats

## Bilan

## Evolutivité

Le projet peut être amené à évoluer de nombreuses façons. Tout d’abord, l’ajout d’une possibilité des tâches est une des choses que l’on pourrait ajouter.

Ensuite la possibilité d’importer des structures missions personnalisée. En effet actuellement la structure des tâches d’une mission est fixe et ne permet l’ajout ou le retrait d’activité. Dans notre code, le chargement prend en compte n’importe quelle architecture d’activité dans le fichier xml de base de mission. Cependant lors de la génération du planning de base, les tâches qui y sont ajoutés sont fixes et ne dépendent pas du fichier XML fourni par l’utilisateur.

En marge de cette option, rajouter une interface afin de pouvoir permettre à l’utilisateur de créer cette architecture d’activités semble être une poste d’évolution pour notre programme.

# Conclusion