



# *Plan d'Assurance Qualité Logicielle*

FactDev



Université Toulouse III – Paul Sabatier

— Florent BERBIE  
— Antoine de ROQUEMAUREL  
— Cédric ROHAUT  
— Andriamihary Manantsoa RAZANAJATOVO

5 février 2015

---

# Table des matières

---

<b>1</b>	<b>Description du Projet</b>	<b>4</b>
1.1	Objet du projet . . . . .	4
1.2	Présentation des Parties Prenantes . . . . .	4
1.3	Production . . . . .	5
<b>A</b>	<b>Table des figures</b>	<b>9</b>
	<b>Index</b>	<b>10</b>

# Avant-Propos

---

Le Plan d'assurance Qualité Logicielle a pour objectif la définition et le suivi des dispositions à prendre dans le cadre du projet *FactDev* afin d'en assurer la qualité, une bonne gestion et d'atteindre les résultats attendus.

À cet effet, le Plan d'Assurance Qualité Logicielle fixe les droits, devoirs et responsabilités de chaque partie prenante en vue d'assurer l'atteinte de ces objectifs.

Il constitue un outil de travail et un référentiel commun à tous les acteurs pour leur donner une vision similaire du projet, mais il constitue également le cahier des charges de la qualité, réalisé en collaboration avec le client puis approuvé par celui-ci. Il constitue enfin la définition des procédures à suivre, des outils à utiliser, des normes à respecter, de la méthodologie de développement du produit et des contrôles prévus pour chaque activité.

Ainsi, d'un commun accord sont déterminés ces différents aspects du projet qui constituent un contrat entre le titulaire et le client et toutes les autres parties prenantes. Ce contrat prend effet dès son acceptation par le client et les personnes concernées et peut être, si les circonstances l'obligent, amené à être modifié au cours du projet. Dans ce cas, toute évolution future sera soumise à l'acceptation du client car au terme du projet, le Plan d'Assurance Qualité Logicielle constituera l'un des documents de résultat du projet.

# 1

## Description du Projet

---

### 1.1 Objet du projet

Le logiciel a pour but de faciliter la création de devis et la conversion de ces devis en factures.

Pour cela il sera possible d'enregistrer des clients dans une base de données et d'offrir une gestion de ces derniers. Un client peut avoir un ou plusieurs projets avec, pour chacun, un ou plusieurs devis ou factures.

### 1.2 Présentation des Parties Prenantes

#### 1.2.1 Client : Antoine de Roquemaurel

*Développeur Freelance, et membre de l'équipe de développement.*

☎ 06 54 33 52 93

🌐 <https://antoinederoquemaurel.github.io>

✉ [antoine.roquemaurel@gmail.com](mailto:antoine.roquemaurel@gmail.com)

#### 1.2.2 Encadrant : Frédéric Migeon

*Maître de conférence à l'Université Toulouse III – Paul Sabatier*

☎ 05 61 55 (62 46)

✉ [Frederic.Migeon@irit.fr](mailto:Frederic.Migeon@irit.fr)

IRIT1 / Niveau 3, Pièce : 361

#### 1.2.3 Responsable de l'UE Projet : Bernard Cherbonneau

☎ 05 61 55 (63 52)

✉ [Bernard.Cherbonneau@irit.fr](mailto:Bernard.Cherbonneau@irit.fr)

IRIT1 / Niveau 4, Pièce : 413

## 1.2.4 Titulaire : Équipe FACT

*Étudiant en M1 Informatique Développement Logiciel à l'université Toulouse III – Paul Sabatier*

**Florent Berbie**

☎ 06 85 31 92 90

✉ [florent.berbie@gmail.com](mailto:florent.berbie@gmail.com)

**Antoine de Roquemaurel**

☎ 06 54 33 52 93

✉ [antoine.roquemaurel@gmail.com](mailto:antoine.roquemaurel@gmail.com)

**Cédric Rohaut**

☎ 06 74 80 12 67

✉ [rohaut@icloud.com](mailto:rohaut@icloud.com)

**Manantsoa Andriamihary Razanajatovo**

☎ 06 01 71 53 02

✉ [manantsoa.razana@gmail.com](mailto:manantsoa.razana@gmail.com)

## 1.3 Production

### 1.3.1 Méthode de développement

#### 1.3.1.1 Définition et pertinence de la méthode Scrum

Le développement du projet se fera selon la méthode Agile *Scrum*, comme cela a été convenu avec notre encadrant.

Cette méthode, basée sur les stratégies itératives et incrémentale, permet de produire à la fin de chaque *Sprint* (incrément/itération) une version stable et testable du logiciel. Les différents événements associés à *Scrum* accroissent la communication grâce à des réunions quotidiennes aussi appelées « *mélées* ». Ceci permet une meilleure cohésion, une meilleure coopération et une meilleure homogénéité du travail fourni par les membres de l'équipe. A cela s'ajoute la présence d'artefacts, c'est-à-dire des éléments à réaliser avec des ordres de priorité et qui contribuent à améliorer la productivité.

Dans le cadre de ce projet, la méthode *Scrum* s'avère être pertinente pour plusieurs raisons : Premièrement, dans la mesure où nous avons proposé un sujet et spécifié les fonctionnalités de celui-ci, la méthode *Scrum* se veut adaptée. En effet, les fonctionnalités que nous avons proposé permettent de définir les limites de notre première version *Release* livrable.

L'ajout de fonctionnalités en fonction des attentes du client pourront être implémentées au fur et à mesure des différents sprints. Cela a pour avantage de fournir un travail continu, d'assurer un suivi avec le client pour répondre au mieux à ses besoins. De plus, la durée du projet étant relativement courte, il serait difficile de revenir sur notre conception préalable alors qu'ici chaque sprint permet de s'assurer que le projet avance dans la bonne direction.

Outre les avantages qu'apporte la méthode *Scrum* à ce projet, l'équipe avait la volonté d'évoluer

vers une méthode qui diffère de celles qui ont pu être abordées en cours. Cette volonté est d'ailleurs confortée par le désir de découvrir de nouvelles technologies (C++ et Qt jusque-là peu connus par la majorité du groupe).

### 1.3.1.2 Application de la méthode Scrum et critères de qualité

Dans le cadre de notre projet, la durée d'un *Sprint* a été défini à deux semaines. Un *Sprint* est constitué de *User Stories* et de *Technical Stories* qui sont préalablement définies lors de réunions quotidiennes aussi appelées *mêlées*. Ces *mêlées* sont quotidiennes, nous profitons de notre temps libre dans l'emploi du temps pour les organiser. Pour les jours où ce n'est pas possible, nous avons mis en place un salon de discussion IRC.

La définition des *Technical Story* indique brièvement la fonction que l'on doit réaliser. Les *User Story* sont quant à elles toujours posées sous cette forme :

**En tant que** [Personne(s) utilisant le logiciel]  
**Je souhaite** [La fonctionnalité que je désire avoir]  
**Afin de** [Objectif de la fonctionnalité]

Chaque *Story* possède un poids, c'est-à-dire une valeur indiquant sa complexité et/ou le temps nécessaire afin de la mettre en œuvre. Le poids de chacune des *Story* est déterminé durant les « mêlées » au moyen d'un « Planning poker ». Ceci permet à chacun d'exprimer la complexité d'une *Story* à réaliser. Chacun justifie le choix du poids qu'il a affecté et après un éventuel débat, on affecte cette valeur à la *Story*. Celle-ci ne peut être affectée qu'une fois que l'ensemble des membres de l'équipe se sont mis d'accord. Elle possède également une priorité précisant l'importance d'intégrer cette *Story* ans le sprint. Cette priorité se fait au moyen de quatre niveaux d'importance :

- *Must* : La *Story* doit obligatoirement être réalisée lors du sprint
- *Should* : La *Story* devra être réalisée (dans la mesure du possible)
- *Could* : La *Story* pourra être réalisée car elle n'a aucun impact sur les autres tâches
- *Would* : La *Story* ne sera pas nécessairement faite et sera alors reportée au prochain *Sprint*

Une *Story* est considérée comme finie lorsqu'elle est fonctionnelle d'un point de vue utilisateur c'est-à-dire :

- Les tests unitaires (pour la base de données ou pour les modèles) sont validés
- Les tests d'intégrations sont validés
- Lorsque chaque méthode est documentée

À la fin du *Sprint*, on présente à notre client, Monsieur Frédéric MIGEON, afin qu'il le valide. Un *Sprint* fournit toujours :

- une démonstration des nouvelles fonctionnalités logicielles
- des tests unitaires
- une documentation du code (au format HTML et PDF)
- un manuel d'utilisateur à jour des nouvelles fonctionnalités

Notre projet sera réalisé en deux *Releases*, c'est-à-dire qu'il y aura deux versions livrables. Chacune de ces versions est composée de trois *Sprint* ayant chacun un poids moyen de 50 pour environ 10 *Stories*.

Les versions livrables du logiciel sont définies dans un carnet de sprint (*Sprint Backlog*) qui référence l'ensemble des *Stories* des différents *Sprint*. Ce carnet est susceptible d'évoluer en fonction des besoins du client, de nouveaux choix de conception ou encore d'éléments n'ayant pas été prévu.

## 1.3.2 Organisation et rôles dans l'équipe de développement

La méthode *Scrum* possède des rôles qui lui sont propres : le *Scrum Master*, le *Product Owner* et l'équipe de développement.

### 1.3.2.1 Scrum Master : Florent Berbie

Le Scrum Master aura pour mission principale de guider les développeurs dans l'application de la méthode Scrum. Il veillera à ce que la méthode soit comprise de tous et appliquée de façon correcte. Il aura le rôle de meneur lors de phases importantes d'application de la méthode telles que le planning poker ou encore les mêlées quotidiennes.

### 1.3.2.2 Product owner : Antoine de Roquemaurel

Le Product owner est la seule personne responsable du carnet de produit et de sa gestion. Ce dernier comprend l'expression de tous les items associés à une priorité (l'importance pour le client). La compréhension de ceux-ci ainsi que la vérification du travail fourni est sous la responsabilité du Product owner.

### 1.3.2.3 Équipe de développement : Florent Berbie, Antoine de Roquemaurel, Cédric Rohaut, Andriamihary Razanajatovo

Afin d'assurer une plus grande cohésion entre les membres de l'équipe, un même niveau d'implication et un travail de plus grande qualité nous avons définis des rôles spécifiques :

### 1.3.2.4 Directeur qualité : Cédric Rohaut

Le responsable qualité sera garant de la transposition des exigences du client sous forme de solutions techniques au sein du logiciel final. En d'autres termes, il veillera à ce que le logiciel apporte une solution technique optimale pour le client. De plus il devra vérifier et valider la qualité du logiciel au travers de tests, du respect des conventions et de l'aspect général du logiciel.

### 1.3.2.5 Directeur documentation : Andriamihary Razanajatovo

Le responsable documentation sera chargé de la révision de l'ensemble des documents avant leur remise ou leur soumission aux parties prenantes concernées. Il veillera à la qualité des aspects fondamentaux, à savoir le fond et la forme, des documentations à fournir.

#### 1.3.2.6 Directeur technique Qt, C++ : Antoine de Roquemaurel

Le responsable technique Qt, C++ sert de support à l'équipe en cas de problèmes techniques liés au développement sur la plate-forme Qt. De par ses connaissances acquises dans ce domaine, il sera le plus à même à aider les membres de l'équipe projet ayant des difficultés avec cette technologie. De plus, dans le but de minimiser les couplages lors de la conception, il veillera à ce que le patron MVC soit correctement utilisé. Ainsi, le code sera clairement découpé en trois parties : modèle, vue et contrôleur ce qui permettra de s'assurer des contrôles d'interactions entre nos composants logiciels.

#### 1.3.2.7 Autres rôles et responsabilités de chaque développeur et organisation/fonctionnement général

Dans un souci d'assurance qualité, chaque responsable est en mesure de présenter les difficultés qu'il a eu à gérer, de présenter les différentes solutions possibles à ce problème et de justifier le choix de la technique adoptée.

Lors de la réalisation d'un cas d'utilisation (« issue » sur Github), celui-ci est assigné à un développeur. Une fois que ce dernier considère sa tâche comme finie, il indique (via une « Pull Request ») aux autres membres de l'équipe que la tâche est soumise à la validation et à l'intégration. Un des autres membres vérifie que la fonction est conforme à sa description, que le code est facilement compréhensible et correctement commenté. Dans le cas où la revue de code ne donne pleine satisfaction, le membre effectuant cette revue et les autres membres de l'équipe pourront ajouter des commentaires pour débattre de la fonctionnalité, de l'implémentation de la fonction ou des technologies employées pour y résoudre. Chacun pourra alors soumettre sa vision du problème et la manière avec laquelle il aurait résolu le problème.



# A

## Table des figures

---

# Index

---

Devis, 3

Facture, 3

Outils, 3

Qt Creator, 3

Parties prenantes, 3