



Bilan du Projet

FactDev

Université Toulouse III – Paul Sabatier

- Florent BERBIE
- Antoine de ROQUEMAUREL
- Cédric ROHAUT
- Andriamihary MananTsoa RAZANAJATOVO

Rédigé par
Équipe FACT

Approuvé par
Frédéric MIGEON

18 avril 2015

Table des matières

| | | |
|----------|----------------------------------|-----------|
| 1 | Le projet | 4 |
| 1.1 | Le logiciel FactDev | 4 |
| 1.2 | Les outils | 6 |
| 2 | La méthodologie | 7 |
| 2.1 | La méthode Scrum | 7 |
| 2.2 | L'intégration continue | 7 |
| 2.3 | La revue de code | 7 |
| 3 | Les résultats | 9 |
| 3.1 | La méthodologie | 9 |
| 3.2 | Le logiciel | 11 |
| A | Backlog product | 13 |
| B | Table des figures | 15 |

Introduction

FactDev est un logiciel de devis et de facturation réalisé dans le cadre de l'UE Projet.

Ce projet s'est fait en réponse à un problème de l'un des membres du groupe : Antoine De Roquemaurel. En effet Antoine, développeur *Freelance*, rédigeait pour ses clients les factures et devis « à la main ». La tâche était répétitive et le risque d'erreurs humaine important :

- erreur dans le calcul des montants
- perte de facture
- plusieurs factures différentes pour un même projet et donc risque de ne pas faire le bon travail demandé par le client

Face à ces difficultés, Antoine a soumis le projet devant répondre aux spécifications suivantes :

- Gestion des clients
- Gestion des projets associées aux clients
- Calculs des tarifs
- Génération de documents
- Recherche

Les autres membres ont vu dans ce projet l'opportunité de développer de nouvelles compétences, aussi bien sur le plan technique qu'organisationnel. D'un point de vue technique avec l'utilisation du *LaTeX* et du *C++* accompagné du *framework Qt*. Sur le plan organisationnel avec la mise en place de la méthode Agile *Scrum*.

Ce document fait état des parties prenantes, des outils et des méthodologies de développement appliquées au projet. Il fait le point sur les résultats obtenus au niveau de la méthodologie, du logiciel et en terme de respect du plan qualité.

1.1 Le logiciel FactDev

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

Il répond aux exigences définies lors de l'introduction en permettant l'enregistrement d'un nouveau client dans la base de données et de projets associés à ce clients. Par exemple, un client X peut demander la réalisation d'un logiciel pour le management de son entreprise et un site web pour promouvoir son activité. On a donc deux projets distincts qui peuvent cependant faire l'objet d'une seule facture. De plus, une facture est fréquemment précédé d'un devis, c'est pourquoi il doit être facile de transformer un devis en facture. Enfin le logiciel devra tenir de certaine réglementation « légale » tel que l'impossibilité de modifier une facture ayant été payée.

1.1.1 Présentation des Parties Prenantes

1.1.1.1 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

1.1.1.2 Encadrant : Frédéric Migeon

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

☎ 05 61 55 (62 46)

✉ Frederic.Migeon@irit.fr

IRIT1 / Niveau 3, Pièce : 361

1.1.1.3 Responsable de l'UE Projet : Bernard Cherbonneau

☎ 05 61 55 (63 52)

✉ Bernard.Cherbonneau@irit.fr

IRIT1 / Niveau 4, Pièce : 413

1.1.1.4 Titulaire : Équipe FACT

Étudiants en M1 Informatique Développement Logiciel à l'Université Toulouse III – Paul Sabatier

Florent Berbie

☎ 06 85 31 92 90

✉ florent.berbie@gmail.com

Antoine de Roquemaurel

☎ 06 54 33 52 93

✉ antoine.roquemaurel@gmail.com

Cédric Rohaut

☎ 06 74 80 12 67

✉ rohaut@icloud.com

Manantsoa Andriamihary Razanajatovo

☎ 06 01 71 53 02

✉ manantsoa.razana@gmail.com

1.2 Les outils

2

La méthodologie

2.1 La méthode Scrum

2.2 L'intégration continue

2.3 La revue de code

La revue de code représente une démarche que nous avons mis en avant dans le plan qualité. L'objectif visé est de tendre vers un projet dont l'intégralité du code a été revu.

La revue de code se fait au moment de l'intégration, c'est pourquoi toutes intégrations nécessitent la création préalable d'une *Pull Request*. Pour cela, une fois le travail correspondant à une *User story* est fini¹, le développeur crée une *Pull Request* via l'outil *Github*. Cette *Pull Request* s'accompagne d'une description plus technique de la *User story* à laquelle elle est liée. La liste des *commits* associés et le code source ajouté et/ou modifié est accessible comme l'on peut le constater ci-dessous.

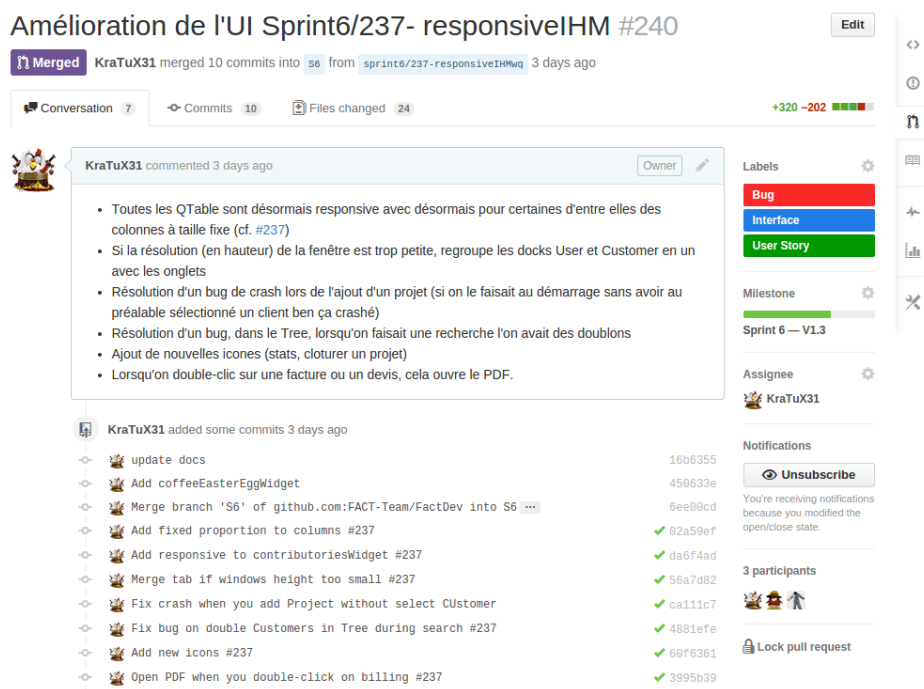


FIGURE 2.1 – Exemple de *Pull Request* du projet *FactDev*

1. selon nos critères définis lors de la présentation de la méthode *Scrum* 2.1

À partir de cette *Pull Request*, les autres membres de l'équipe reçoivent une notification pour indiquer qu'ils doivent procéder à la revue de code. Au moins l'un d'eux doit s'assurer que le code est valide. Un code est dit valide lorsqu'il :

est lisible Le code doit être facile à lire.

est compréhensible Le code doit être facilement compréhensible, avoir un niveau de complexité minimum.

respecte les conventions d'écritures Respect des conventions d'écritures (convention de nommage et de mise en forme).

Cette vérification est aisée via l'outil *Github* qui permet de rajouter des commentaires « *inline* » c'est-à-dire d'ajouter un commentaire aux lignes précises de code à modifier. Ainsi, sans toucher au code, l'on sait précisément l'endroit où l'on doit procéder à des changements.

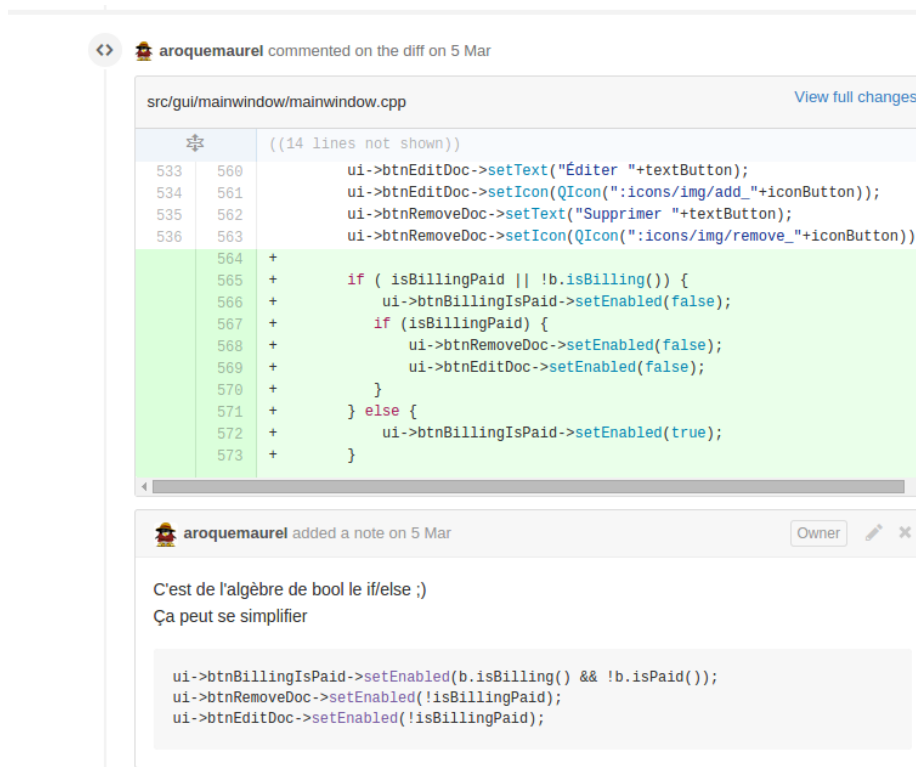


FIGURE 2.2 – Ajout d'un commentaire « inline » lors d'une *Pull Request*

L'on procède également à la vérification de la documentation. Chaque méthode et attribut doit être documenté. Là aussi, il faut que la documentation respecte les conventions d'écritures.

Une fois les remarques faites sur le code et sa documentation, l'on passe aux tests fonctionnels. On se rend donc sur la branche *Git* en question et l'on vérifie que la fonction répond bien à la *User story* et qu'il n'existe aucun bug. Bien entendu, on s'assure que ça n'a pas entraîné de régression sur d'autres parties du logiciel. C'est aussi le moment de proposer des modifications sur le plan ergonomique si besoin est.

Enfin, avant d'intégrer, l'on vérifie que les outils (*Travis CI* et *Coveralls*) ne s'y opposent pas c'est-à-dire que :

- le *Build*² passe, c'est-à-dire qu'il compile sans erreur.
- la couverture de code n'a pas régressé

2. Un *Build* est un artefact logiciel autonome résultant de conversion de fichiers de code source en code exécutable

3

Les résultats

L'UE projet était l'occasion de se positionner dans une situation très proche de celle que nous avons rencontrée en milieu professionnel. Dans ce cadre là, il était nécessaire de prendre des mesures organisationnelles afin de parvenir aux résultats escomptés.

3.1 La méthodologie

La qualité du logiciel *FactDev* est principalement due à la rigueur dans l'application de la méthodologie de développement.

Elles furent définies en même temps que la conception du logiciel durant le *Sprint 0*. Cela comprend les attitudes à adopter par les membres de l'équipe sur le plan organisationnel et techniques avec le respect de conventions qui ont aussi été définies à ce moment là.

3.1.1 Le respect de la méthode Scrum

Au niveau de la gestion du développement par la méthode *Scrum* nous avons tenu un *Backlog*. Dans celui-ci se trouve l'ensemble des *User stories* et *Technical stories* que nous avons à réaliser durant le projet. Ces *Stories* ont été déterminées lors de *Planning Poker* durant les mêlées. C'est durant ces séances que nous déterminions le poids attribué à chacune des *Stories* et comment les répartir sur les six *Sprints* des deux *Releases*.

La répartition des *Stories* s'est faite en fonction de leur poids mais aussi en tenant compte de l'équipe et des événements pouvant survenir. Ainsi le premier *Sprint* a été plutôt léger car il tenait compte du temps d'apprentissage du langage C++ et du framework Qt. Les autres *Sprints* sont en revanche équilibrés. Les deux derniers sont également plus légers sur le plan technique car nous voulions consacrer du temps à la rédaction des documents et à la préparation de la soutenance. De plus, certaines tâches possédaient un fort taux d'inconnu et il était donc difficile de leur attribuer un poids.

Lorsque nous ne pouvions nous voir pour assurer les mêlées quotidiennes, nous nous retrouvions sur une zone de chat « *#irc* » que nous avons créée pour la circonstance. Ainsi, nous faisions le point sur l'évolution du projet et débattions sur des choix de conception.

L'assiduité et le respect de la méthode *Scrum* est visible sur le *Github* de notre projet. En effet, le nombre de *Pull Request* (95), d'*issues* (145), de branches (91) et de *commits* (près de 1200) en est une preuve. Les *Pull Request* mettent également en avant la communication entre les membres de l'équipe.

Autre preuve de notre implication dans le respect de cette méthode, le *BurnUp Chart* ci-dessous :

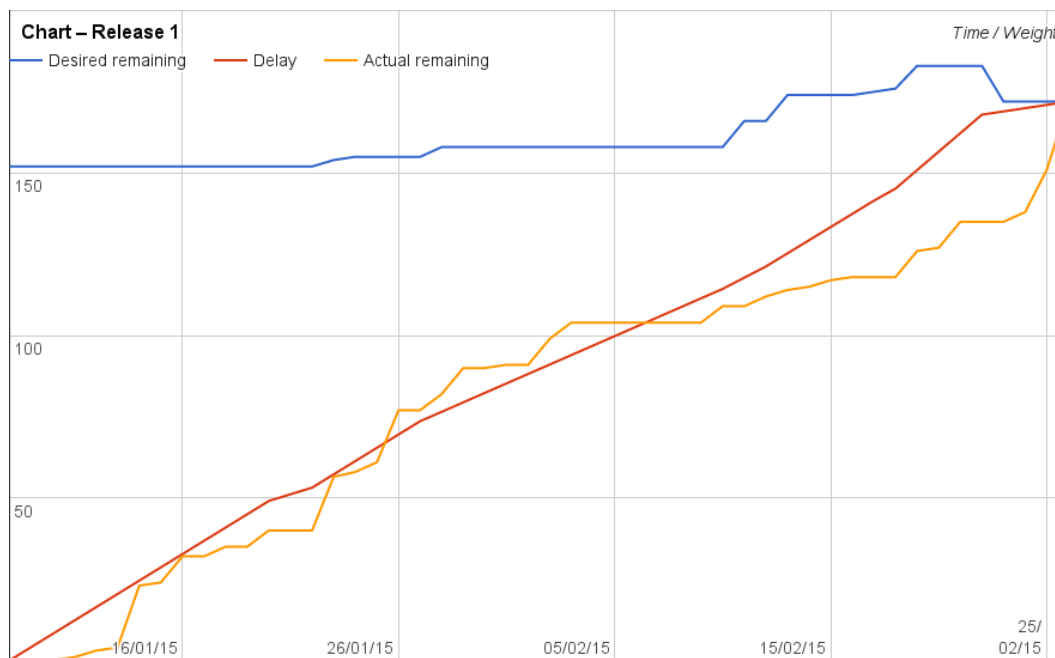


FIGURE 3.1 – Burn Up chart de la première release

La courbe rouge représente l'évolution théorique du projet durant les *Sprints* de la première *Release*. On constate que notre avancé (courbe orange) suit globalement la courbe théorique. Cependant, l'on observe une baisse de notre activité durant la semaine du 5 février qui correspondait à notre période de vacance. De plus, on note également que lors du *Sprint 3* nous nous sommes trouvés en dessous de l'objectif. Cela s'explique par une augmentation du nombre de *Stories* (courbe bleu) qui correspondait à des modifications ergonomiques où à la résolution de bugs. Certaines d'entre-elles étaient sujet à discussion et ont donc été reportées au *Sprint* d'après. Néanmoins nous sommes parvenus à finir nos *Sprints* dans les temps.

Chacun des *Sprints* a été réalisé dans les temps et fut l'objet d'une démonstration auprès de M. MIGEON.

3.1.2 Les outils de qualité du code

La qualité du code peut être mesurée grâce à différents outils.

Dans notre cas nous avons utilisé plusieurs outils tel que *Travis CI* pour l'intégration continue, *Coveralls* pour la couverture de code ou encore *SonarQube* qui fournit des informations diverses sur le code (nombre de lignes de code, respect des conventions de nommage, niveau de complexité, duplication du code).

Travis CI nous a permis de s'assurer que le logiciel compile sur une machine tierce, que les tests unitaires passent et de mettre à jour nos documents (manuel d'utilisateur ou *Doxygen*). Associé à *GitHub* l'on sait après chaque *commit* si le *Build* passe. C'est également lui qui fournit à *Coveralls* notre code à analyser pour vérifier la couverture de notre code. De plus, le fait que l'on ne peut pas intégrer notre code si la couverture de code régresse nous oblige à faire des tests unitaires sur les nouvelles fonctionnalités que l'on a implémenté. Ainsi, à la fin de notre projet, la couverture de code est de 90%. Cette continuité dans l'implémentation des tests unitaires nous a permis de

faire du *Refactoring* régulièrement sans crainte de provoquer des régressions du logiciels. Cela a été particulièrement efficace lors de l'ajout d'un second système de gestion de base de données où l'on devait s'assurer que les méthodes fonctionnent aussi bien sur un système que sur l'autre.

3.2 Le logiciel

Conclusion

Nous tenons avant tout à remercier toutes les parties prenantes de ce projet qui nous ont permis de le mener à bien que se soit l'équipe pédagogique de l'université, le client ou l'équipe de développement.

Le projet FactDev nous a en effet permis de pouvoir mener un projet comme dans le cadre d'une entreprise dans le monde professionnel. Malgré les difficultés que nous avons pu rencontrer comme la difficulté de communiquer, d'exprimer nos points de vues, nos problèmes, des fonctionnalités que nous n'avons pu implanter, de nombreux points positifs en ressortent.

Au delà de l'expérience professionnelle que nous avons acquise grâce à la pédagogie mise en place et du plus dans nos CV, plusieurs points bénéfiques en ressortent. Tout d'abord le logiciel sera sous licence GPL, il sera utilisable et utilisé par son client et éventuellement d'autres personnes dans le futur. Au niveau technique, une multitude de compétences ont été acquises, les principales sont le langage C++ ainsi que les méthodes agiles mais d'autres moins principales mais tout aussi nécessaires ont été vues comme la gestion du projet avec GitHub ou encore les tests avec Travis et Coveralls.

C'est pourquoi cette pédagogie innovante qui nous permet d'acquérir une certaine autonomie est nécessaire dans un monde qui évolue tout les jours notamment celui du numérique avec ses innovations.

A

Backlog product

| Release | Sprint | Statut | Vélocité | | Poids | Prio | #Issue | Type Story | En tant que | Je Souhaite | Afin de |
|-----------|----------|--------|----------|----------|---------|--------|--------|-----------------|-------------|---|--|
| | | | Poids | Nb Story | | | | | | | |
| Release 1 | Sprint 1 | Done | 40 | 10 | 2 | Must | 1 | User Story | Utilisateur | Créer un nouveau client | Ajouter ses informations et prochainement le lier à un dev |
| | | Done | | | 3 | Should | 2 | User Story | Utilisateur | Editer un client | Modifier les informations d'un client |
| | | Done | | | 2 | Would | 3 | User Story | Utilisateur | Supprimer un client | Corriger une erreur, un client ajouté par mégarde |
| | | Done | | | 5 | Must | 11 | User Story | Utilisateur | Afficher la liste des clients | Pouvoir prochainement afficher leurs factures |
| | | Done | | | 5 | Must | 12 | User Story | Utilisateur | Renseigner mes données | |
| | | Done | | | 8 | Could | 14 | User Story | Utilisateur | Afficher les informations d'un client particulier | Obtenir des informations détaillé sur le client |
| | | Done | | | 5 | Could | 20 | User Story | Utilisateur | Chercher un client à partir de son nom | Filter l'affichage des clients |
| | | Done | | | 1 | Would | 22 | User Story | Utilisateur | M'informner sur le développement du projet | |
| | | Done | | | 1 | Must | 16 | Technical Story | - | Création de la fenêtre d'ajout/modification d'un client | |
| | | Done | | | 8 | Would | 15 | User Story | Utilisateur | Afficher un menu contextuel | Editer, Ajouter, Supprimer... |
| | Sprint 2 | Done | 64 | 15 | 5 | Must | 13 | User Story | Utilisateur | Créer un nouveau projet pour un client | pouvoir l'associé à un client |
| | | Done | | | 13 | Must | 7 | User Story | Utilisateur | Créer un nouveau devis | estimer le coût d'un développement |
| | | Done | | | 5 | Must | 30 | User Story | Utilisateur | Afficher les devis d'un projet | Avoir une estimation du coût du projet |
| | | Done | | | 8 | Must | 44 | Technical Story | - | Récupérer un devis dans la bd avec son id | |
| | | Done | | | 1 | Should | 24 | User Story | Utilisateur | Editer un projet existant | modifier ses informations |
| | | Done | | | 3 | Must | 25 | User Story | Utilisateur | Lister les projets d'un client | Afficher la liste des projets d'un client particulier |
| | | Done | | | 5 | Must | 34 | Technical Story | - | User Manual | |
| | | Done | | | 8 | Must | 37 | User Story | Utilisateur | Créer une prestation | |
| | | Done | | | 8 | Would | 47 | Technical Story | | Sécurité des champs | |
| | | Done | | | 0.5 | Could | 50 | Anomaie | | Titres de fenêtres | |
| | Sprint 3 | Done | 59 | 13 | 0.5 | Could | 49 | Anomaie | | Noms docks | |
| | | Done | | | 0.5 | Could | 48 | Anomaie | | Affichage Fax | |
| | | Done | | | 0.5 | Must | 52 | Technical Story | | créer un jeu d'essais de devis et projets | |
| | | Done | | | 3 | Must | 65 | Technical Story | | Insérer dans tripatte | |
| | | Done | | | 3 | Should | 26 | User Story | Utilisateur | Supprimer le projet d'un client | |
| | | TODO | | | 1 | Must | 8 | User Story | Utilisateur | Créer une nouvelle facture | |
| | | TODO | | | 1 | Must | 39 | User Story | Utilisateur | Afficher les factures d'un projet | |
| | | TODO | | | 1 | Must | 41 | User Story | Utilisateur | Editer une facture existante | |
| | | TODO | | | 2 | Should | 45 | User Story | Utilisateur | Signaler une facture comme payée | |
| | | TODO | | | 3 | Could | 27 | User Story | Utilisateur | Editer un devis existant | le générer de nouveau |
| Release 1 | Sprint 3 | TODO | 59 | 13 | 3 | Would | 26 | User Story | Utilisateur | Supprimer un devis | supprimer un devis d'un client |
| | | TODO | | | 3 | | 40 | Technical Story | - | User Manual | |
| | | TODO | | | 5 | | 42 | Technical Story | - | Plan Qualité | |
| | | TODO | | | 13 | | 43 | Technical Story | - | Préparer soutenance | |
| | | TODO | | | 5 | Would | 33 | User Story | Utilisateur | Rechercher un devis ou une facture | Accéder facilement un devis ou une facture spécifique |
| | | TODO | | | 2 | Could | 35 | User Story | Utilisateur | Editer une prestation | Pouvoir l'ajouter à un devis ou une facture |
| | | TODO | | | 8 | Could | | Technical Story | | Sécurité des champs | |
| | | TODO | | | 5 | Should | 28 | User Story | Utilisateur | Générer le fichier .tex d'un devis | pouvoir l'avoir ensuite en .pdf |
| | | TODO | | | 20 | | 38 | Technical Story | - | Déploiement | |
| | | TODO | | | 59 | | | | | | |
| Totaux | 3 | | 163 | 38 | 163 | | | | | | |
| Moyennes | | | 52 | 14 | 4.35135 | | | | | | |

B

Table des figures

| | | |
|-----|--|----|
| 2.1 | Exemple de <i>Pull Request</i> du projet <i>FactDev</i> | 7 |
| 2.2 | Ajout d'un commentaire « inline » lors d'une <i>Pull Request</i> | 8 |
| 3.1 | Burn Up chart de la première release | 10 |