INSA de Rennes
Quatrième année Informatique

Rapport bilan de planification

Projet VaR

Benjamin Bouguet - Paul Chaignon
Eric Chauty - Ulysse Goarant

Hamdi Raissi - Ivan Le Plumey
Quentin Giai Gianetto

Mai 2014

Table des matières

Chapitre 1

Introduction

En décembre 2013, nous avions établi une estimation du temps de travail à réaliser sur chaque partie du logiciel, que ce soit du développement ou des tests, chaque point avait été détaillé. Durant les quelques mois qui ont suivi, un décompte du temps passé sur chaque étape du développement ou des tests a été effectué.

Aujourd'hui nous sommes en mesure d'effectuer un bilan de notre planification initiale. Nous reprendrons les fonctionnalités précédemment détaillées dans le rapport de planification, en passant en revue sur les éventuelles différences observées entre la durée estimée et la durée réelle.

Nous avions également effectué des prévisions de temps de travail sur la conception, la page HTML, la réalisation du rapport final et la préparation à la soutenance. Nous n'en parlerons pas dans ce document car ces dernières sont secondaires dans les enseignements à retenir des erreurs de planification. En effet, ces étapes se sont toutes bien déroulées ou ne sont pas encore terminées.

Chapitre 2

Développement

Le développement est composé de sept grandes parties :

- les données
- les calculs
- le backtesting
- l'IHM
- les graphiques
- les rapports
- les sessions

Nous dresserons dans cet ordre le bilan des différentes phases du développement. Comme prévu, nous avons eu l'avantage que beaucoup de nos tâches ont été indépendantes les unes des autres. Ceci nous a permis d'avoir une plus grande flexibilité dans la répartition des tâches et leur réalisation. Cependant, nous avons eu un surplus de travail (conflits) lors de la fusion des différentes branches de développement.

2.1 Gestion des données

Le calcul de la Value-At-Risk s'effectuant sur des données financières, il est nécessaire de commencer par implémenter la gestion des données.

2.1.1 Actifs

Les actifs constituent les éléments de base manipulées dans le logiciel. La première étape a été l'implémentation du modèle obtenu lors de la phase de conception. C'est à dire créer la classe et les méthodes associées.

Notre logiciel est capable d'importer des données provenant de Yahoo Finance aux formats CSV et texte. L'importation est consitutée d'un ensemble de traitements sur le fichier des actifs et résulte en la création d'un nouveau fichier ne contenant que les données sélectionées.

Tâches	Durée prévue	Durée réelle
Modélisation des actifs	10h	11h

TABLE 2.1 – Durées des tâches liées aux actifs

Nous avons passé presque exactement le temps prévu sur l'importation des données. Cependant nous aurions pu être plus rapide. En effet, nous avons réalisé cette fonctionnalité au cours de trois versions ; les deux dernières constituant des améliroations de la première. Avec plus de discussions entre nous avant le développement, nous aurions peut-être pu nous passer des deux dernières versions.

2.1.2 Portefeuilles

Les portefeuilles sont constitués d'une composition d'actifs. Il était donc nécessaire d'avoir préalablement implémenté la gestion des actifs avant d'envisager l'implémentation de la gestion des portefeuilles.

Tâches	Durée prévue	Durée réelle
Modélisation des portefeuilles	20h	19h30

TABLE 2.2 – Durées des tâches liées aux portefeuilles

Aucun souci particulier n'ayant été rencontré durant la réalisation des différentes classes nécessaires à la gestion des portfeuilles, nous avons presque égalé le temps prévu, notre estimation était donc bonne.

2.2 Importation et exportation

Tâches	Durée prévue	Durée réelle
Importation	35h	4h30
Exportation	20h	9h

TABLE 2.3 – Durées des tâches liées à l'importation/exportation

Comme expliqué dans le rapport final, nous avons modifié la spécification associé à l'importation et l'exportation des données. Les données sont maintenant toutes importées et exportées en une seule fois sans que l'utilisateur puisse choisir lesquels.

En conséquence, ces fonctionnalités ont été bien plus rapide que prévu à développer.

Cependant nous avions prévu de passer plus de temps sur l'importation que l'exportation et, même si les deux ont été développées en même temps, l'exportation a été difficile à mettre en place. Cela est dû à la nécessité d'utiliser une librairie externe pour écrire l'archive. Nous avons donc installé et utilisé la librairie QuaZip.

2.3 Calculs

Les fonctionnalités de calcul contiennent principalement le calcul de la Value-at-Risk selon les différentes méthodes. Pour calculer cette dernière, nous avons

utilisé le logiciel de statistiques R. Dans un premier temps, nous avons cherché à interfacer notre logiciel avec R en utilisant RInside, conformément aux spécifications. Cependant, cela s'est révélé être très compliqué [1] malgré toutes nos recherches et essais. Nous avons dû trouver une autre solution.

Les tests de corrélation nous ont été fournis sous forme de scripts R par notre encadrant. Il nous a d'abord fallu les tester (ce qui correspond à la tâche de test des scripts externes) puis les appeler via R. Finalement, nous avons passé moins de temps que prévu sur l'interfaçage avec R. Notre solution risque cependant d'être moins stable dans le temps que si nous avions utilisé une librairie externe comme RInside.

Enfin, les calculs statistiques se résument à de simples appels à des fonctions fournies par R. C'est pourquoi cette tâche est relativement courte.

Tâches	Durée prévue	Durée réelle
Interfaçage avec R	20h	14h30
Modélisation GARCH	15h	11h30
Méthode historique	5h	9h
Méthode Riskmetrics	10h	2h
Tests de corrélation	10h	?
Tests scripts externes R	5h	4h
Calculs statistiques	5h	?

TABLE 2.4 – Durées des tâches liées aux calculs

Bien que le temps passé sur la méthode historique excède celui prévu initialement, cela se doit au fait qu'un code en commun à d'autres classes a été développé. De plus, des dépendances avec d'autres branches ont nécessité plusieurs modifications. En elle-même, la méthode historique a été rapide à développer.

Comme nous le pensions, la réalisation de la méthode Riskmetrics a été plus rapide, car grandement inspirée de la modélisation GARCH. Plus de travail en préparation sur les notions mathématiques à implémenter nous a permis de développer cette fonctionnalité encore plus rapidement que prévu.

De même que précédemment, la méthode GARCH s'est révelée plus rapide à réaliser, une connaissance moins floue sur Qt et les notions mathématiques utilisées, aurait pu donner une meilleure évaluation.

2.4 Backtesting

Le développement du backtesting a nécessité préalablement le fonctionnement du calcul de la VaR selon les différentes méthodes de calcul.

Tâches	Durée prévue	Durée réelle
Backtesting	30h	3h

TABLE 2.5 – Durées des tâches liées au backtesting

1. Le rapport final explique ce problème plus en détails et, rapidement, les démarches que nous avons entrepris pour essayer de le résoudre.

Cette étape a été complètement sur-évaluée. Ceci pourrait peut être s'expliquer de nouveau par le manque de certaines notions mathématiques, mais aussi par la bonne réalisation des calculs de VaR précédemment. Le backtesting étant totalement basé dessus, et ne fait que lancer des tests de comparaisons entre les méthodes de calcul de VaR.

2.5 Rapports

De manière générale, les rapports résument les informations générées par les autres modules du logiciel. Il n'était cependant pas nécessaire que le module correspondant au rapport à générer soit d'abord implémenté. Le développement en parallèle du module de la génération des rapports et des modules correspondants d'où sont extraites les données, a donc été possible.

Tâches	Durée prévue	Durée réelle
Rapport au format DOCX	10h	23h
Rapport au format PDF	10h	0h
Rapport du calcul de la VaR	5h	2h
Rapport de statistiques générales	10h	2h
Rapport de backtesting	5h	2h
Rapport des tests de corrélation	5h	2h

TABLE 2.6 – Durées des tâches liées aux rapports

Dans le rapport de spécification, nous avions prévu d'utiliser la librairie libopc pour générer les rapports en PDF ou DOCX. Cela s'est révelé être une mauvaise idée, cela était bien trop complexe. Nous avons opté pour l'utilisation et la réalisation d'un programme Java, auquel notre logiciel envoie un ensemble de paramètres représentant les données à ajouter dans le rapport.

Cela explique donc pourquoi nous avons dépassé le temps prévu pour la réalisation des rapports en DOCX, car ce fût les premiers réalisés. Avec la librairie Java que nous avons utilisé, la conversion du format DOCX au format PDF est très simple. Nous avons donc fait cela en même temps que la génération du format DOCX.

Le développement de notre propre programme Java pour faire l'interface avec la librairie de génération nous a permis de gagner du temps sur la suite de la génération. En effet, grâce à ce programme, la génération de nouveaux rapports est très simple à mettre en place. Elle nécessite uniquement de modifier le *template*[2].

2. Plus d'explications sur le fonctionnement de notre programme sont données dans le rapport final.

2.6 Graphiques

Les graphiques nécessitent principalement l'implémentation de la gestion des données. L'implémentation des matrices de corrélation nécessite de plus celle des calculs correspondants.

Tâches	Durée prévue	Durée réelle
Matrice de corrélation	15h	6h
Histogramme	15h	4h

TABLE 2.7 – Durées des tâches liées aux graphiques

2.7 IHM

L'IHM contient toutes les actions que l'utilisateur peut effectuer, les formulaires que l'utilisateur devra remplir ainsi que les éléments graphiques tels que les tableaux. La durée des tâches comprend aussi la gestion des messages d'erreurs ainsi que leur affichage dans l'IHM.

Tâches	Durée prévue	Durée réelle
Exporter des actifs	5h	?
Importer des actifs	10h	4h
Backtesting	5h	?
Calcul de la VAR	5h	?
Ajouter/Modifier portefeuille	10h	3h
Gestion des sessions	10h	?
Générer rapports	5h	5h
Calculs statistiques	5h	?
Exporter un portefeuille	10h	?
Importer un portefeuille	5h	?
Supprimer un portefeuille	5h	?
Listing des portefeuilles	10h	1h30
Afficher le contenu d'un portefeuille	10h	14h
Afficher les rapports d'un portefeuille	5h	5h

TABLE 2.8 – Durées des tâches liées à l'IHM

Pour la plupart des éléments graphiques, de simples intéractions entre les objets Qt ont été suffisantes. C'est pour cela que le temps prévu s'est révélé trop important.

L'affichage du contenu des portefeuilles été plus compliqué. Nous aurions pu nous limiter à une simple lecture à la base de données puis à un affichage dans un tableau, mais ceci n'était pas très adapté. Nous sommes partis sur la

réalisation d'un QAbstractItemModel, car c'est plus propre comme cela. Par exemple, nous n'avons pas à gérer explicitement le bon nombre de ligne ou de colonnes à afficher.

2.8 Session

Cette fonctionnalité ajoute la sauvegarde dans la base de données des portefeuilles, rapports et actifs. Les modifications sont aussi sauvegardées (changement de nom, suppression d'un rapport, changement de l'emplacement du fichier d'un rapport). Cette tâche a nécessité que la gestion des actifs et des portefeuilles soit en partie déjà réalisée.

Tâches	Durée prévue	Durée réelle
Sauvegarde de la session	10h	6h30
Restaurer la session	10h	10h

TABLE 2.9 – Durées des tâches liées à la session

Nous avons devéloppé cette fonctionnalité vers la fin du projet. Nous maîtrisions donc mieux Qt et la librairie pour SQLite. Nous avons donc gagné un peu de temps sur cette tâche.

Chapitre 3

Tests

Comme expliqué dans le rapport final, nous avons effectué les tests au fur et à mesure des branches de développement. L'IHM étant plus difficile à tester automatiquement, celle-ci a été testée (manuellement) en suivant des scénarios. Nous avons établis ces scénarios au fur et à mesure que les fonctionnalités étaient développées.

Tâches	Durée prévue	Durée réelle
Tests de la gestion des données	20h	35h
Tests des calculs (modèle et VaR)	20h	5h + ?
Tests IHM	20h	5h + ?

TABLE 3.1 – Durées des tâches liées aux tests

Nous avons passé plus de temps que prévu sur les tests uniatires et fonctionnels automatiques. En effet, nous avons notamment dû configurer l'outil d'intégration continue, Travis CI, qui nous permet maintenant de connaître les résultats des tests après tout changement. Cet outil nous assure cependant une plus grande qualité logicielle et simplifie le travail pour la relecture des branches.

Les tests de l'interface graphique et des calculs ne sont pas encore complètement finis donc nous ne pouvons qu'estimer le temps qui y sera finalement consacré.

Chapitre 4

Diagramme de GANTT

Chapitre 5

Conclusion

Le tableau ci-dessous permet de comparer la planification initiale du projet global avec le temps réellement nécessaire à sa réalisation :

Tâches	Durée prévue	Durée réelle
Développement	375h	?
Tests	60h	?
Total	435h	?

TABLE 5.1 – Durées des tâches principales

Dans ce document, nous avons comparé les temps prévus à la réalisation des différentes parties de notre logiciel, en expliquant dans la mesure du possible les raisons pour lesquelles nous avons observé des différences entre la durée prévue et la durée effective. Globalement, le temps passé sur la réalisation de ce projet est inférieur au temps prévu. Cela ne signifie pas que moins de choses ont été réalisées, mais plutôt qu'elles avaient été sur-évaluées en temps de développement ou que nous avons été plus efficace.

Cependant nous avons quand même dû réduire légèrement les possibilités de notre logiciel [1]. En effet, nous avions complètement surestimé le temps que nous pouvions consacrer à ce projet chaque semaine. Nous avions notamment sous-estimé le temps nécessaire pour les autres matières et projets à travailler au cours du second semestre.

1. Voir les changements apportés aux spécifications dans le rapport final.