Système multi-agents appliqué à la simulation de gestion de planning logiciel

Maxime CHASTE,

Pierre-Philippe BERENGUER

Master 2 IAGL – IDL

Université de Lille 1

Le 04/02/2015

Introduction

La gestion de projet est une tâche complexe. Plusieurs paramètres entre en compte tels que les facteurs humain, et matériel. Des outils de simulation permettent d'aider les prises de décision concernant les allocation de ressources au projet. Toutefois, des incertitudes découlent de l'apparition d'événements aléatoires durant l'avancement. Ainsi la projection des réalisations, et leurs délais exactes ne représentent pas un résultat réaliste.

Une application de simulation doit pouvoir aider à la mise en place du planning d'avancement de projet. Cette application doit reproduire les incertitudes sur les tâches à réaliser, et leurs durées. Qui plus est, un gestionnaire de projet doit prendre en compte les modifications des délais des tâches pour adapter les allocations de ressources.

L'objectif ici, est de proposer une technique de simulation de planning dans l'environnement du développement logiciel. Cette technique repose sur la mise en place d'un système multi-agents, offrant une intelligence artificielle (I.A) dans la simulation.

Système multi-agents

En informatique, un système multi-agents (SMA) est un système composé d'un ensemble d'agents, situés dans un certain environnement et interagissant selon certaines relations. Un agent est une entité caractérisée par le fait qu'elle est, au moins partiellement autonome. Cela peut être un processus, un robot, un être humain, etc.

Y. Demazeau, en 1997, propose une mnémonique pour représenter les systèmes multi-agents intitulée *« Voyelles » :*

- (A)gents : qui sont-ils ?
- (E)nvironnement : comment est-il construit ?

- (I)nteractions : qui peut faire quoi ?
- (O)rganisation : modèle social, modèle de communication ?
- (U)tilisateur : quelle est notre place en tant qu'utilisateur de l'outil de simulation ?

Modèle : planification de projet

L'approche adoptée est dite de *« Bottom-up »*, nous cherchons à modéliser individuellement chaque agent dans leur environnement dans le but d'observer des résultats au niveau macroscopique.

Unité de temps

L'unité de temps utilisée dans la simulation s'appelle *WorkUnit*. Cette unité permet d'unifier la mesure temporelle au cœur de notre simulation est correspond à un tour dans l'avancement. Cela est comparable par exemple à la valeur *one-man-week* répandue en planification logicielle. Une tâche nécessitant trois *WorkUnits* de travail, auquel on alloue un seul développeur sera réalisée au terme de trois itérations. Si toutefois, trois développeurs sont alloués à la tâche elle aboutira au bout d'un seul tour.

Simulation

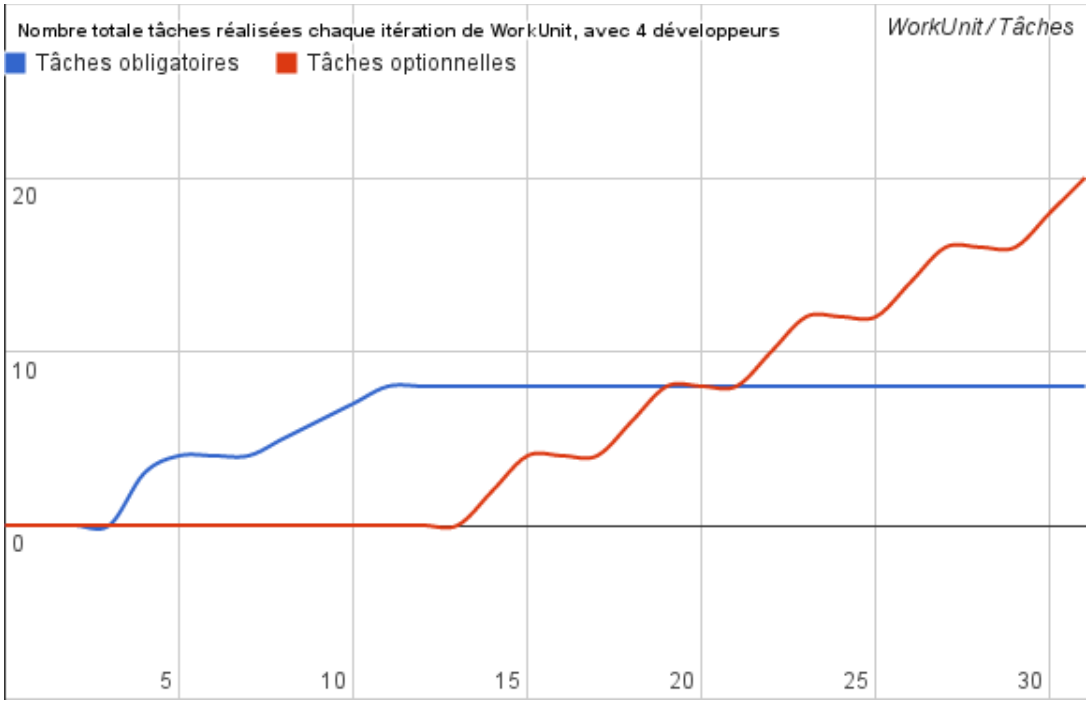
Un agent dans notre simulation est un développeur. Cet agent est attribué à une tâche. De plus, il dispose d'une capacité de travail, appelée *WorkUnit*. Chaque développeur alloué à une tâche disposera d'une capacité équivalent à un *WorkUnit*. En revanche lorsqu'il est alloué à une nouvelle tâche, cette capacité est divisée par deux, pendant un tour. Ceci modélise, le temps d'adaptation, ou apprentissage nécessaire à un développeur pour travailler sur une nouvelle implémentation.

Une tâche, représente une itération dans l'avancement du projet. Un projet est une série hiérarchique de tâches, mises à la suite des autres, ou en parallèle. Ainsi une tâche se caractérise par un ensemble d'agents développeur qui lui sont attribués. Et, une tâche peut être configurée de sorte qu'elle accepte une quantité maximale de développeur.

L'environnement de simulation regroupe, manipule les agents développeurs, et les tâches à accomplir. Il permet d'appliquer les stratégies du systèmes aux entités instanciées.

Stratégies

La première stratégie sur laquelle repose la simulation est d'embaucher un employé. L'algorithme assigne un développeur libre à une tâche qui n'est pas terminée. Mais, les développeurs sont alloués en priorité aux tâches obligatoires, si toutes les tâches obligatoires ne peuvent plus accueillir de développeurs, alors nous assignons les ressources aux tâches optionnelles. Comme observable sur le schéma suivant, les ressources sont allouées en priorités aux tâches obligatoires, et ensuite aux tâches optionnelles :



Dans cette simulation, le nombre de tâches effectuées augmentent au cours du projet, en priorisant les tâches obligatoires

La seconde stratégie est de mettre à jour la durée d'une tâche. A chaque itération dans la simulation, la durée de toutes les tâches restantes sont réévaluées. Ceci consiste à augmenter ou réduire la quantité restante de *WorkUnits*. Cette méthode tente de se rapprocher du monde réel ou l'estimation des délais est mise à jour au cours du projet.

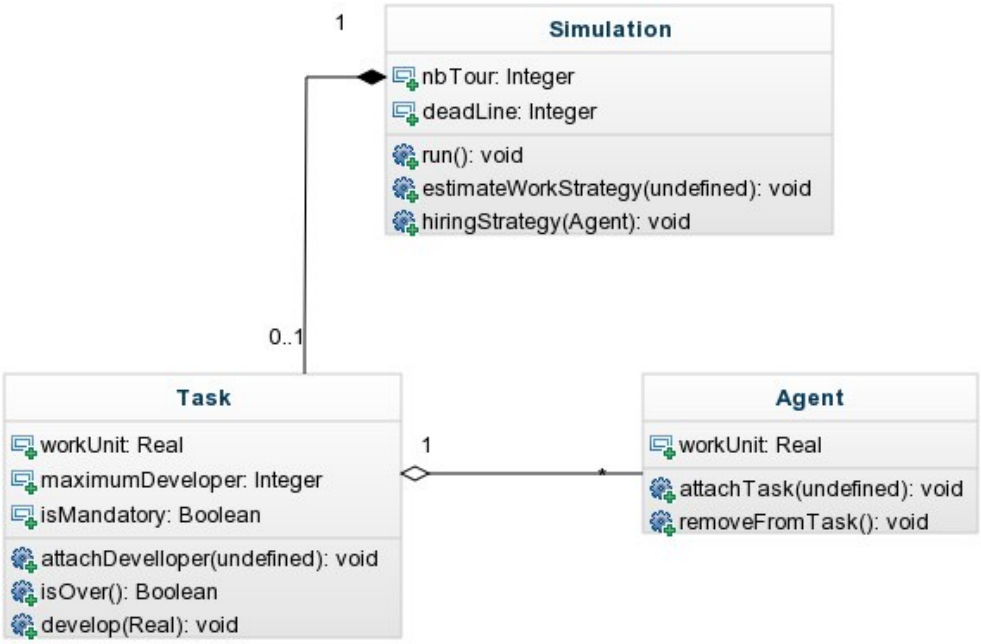
Implémentation

La durée ou *WorkUnit* est une valeur réelle. La durée d'une tâche est majorée ou minorée à chaque tour de la simulation de façon aléatoire via la méthode *Simulation.estimateWorkStrategy(Task task)*.

La simulation est une suite d'itérations non équitables. Lors de chaque itération, *Simulation.hiringStrategie(Agent agent)* permet d'allouer les employés aux tâches lorsqu'ils sont libres, en privilégiant les tâches obligatoires.

La simulation se termine lorsque la durée du projet est écoulée. Le résultat final de la simulation permet d'affirmer ou non que les tâches obligatoires (*MandatoryTask)* sont terminées. Les tâches optionnelles (*OptionnalTasks)* qui sont terminées sont également énumérées.

En revanche la durée maximum de la simulation est une valeur entières. Nous représentons la fin du projet à 20 *WorkUnits* par exemple.



Img 1: UML : Simulation multi-agents pour la planification de tâches à développer

L'exécution est basée sur un modèle probabiliste appelé méthode Monte-Carlo. La stratégie d'embauche (*Simulation.hiringStrategy)* s'applique sur les tâches, elles-mêmes triées aléatoirement. Ainsi aucune tâche n'est favorisée plus qu'une autre. Cependant l'algorithme prend en compte la hiérarchie, de sorte à commencer par les tâches obligatoires (*Task.isMandatory).* Enfin, les développeurs qui ne sont pas encore assigné, ou dont la tâche est terminée, sont affecté également de façon aléatoire. Ceci permet de disposer d'un modèle disposant d'une intelligence (I.A) équitable.

Utilisateur

L'outil développé permet de paramétrer chaque entité de la simulation. D'abord, il faut configurer la durée du projet. Ensuite, il faut définir le nombre de tâches obligatoires, respectivement optionnelles. Ces tâches sont configurées en saisissant, d'une part le nombre de développeurs maximum qui peuvent y contribuer, et d'autre part l'estimation de *WorkUnits* nécessaires.

Résultats

Comparaison de résultats de planification

Voici un premier exemple permettant de simuler une planification de projet. Le but ici, est de comparer les délais nécessaires en fonction du nombre de développeurs.

Configuration de la simulation :

- délais maximale de 30 *WorkUnits*
- 8 tâches obligatoires, avec une estimation respective de 5 *WorkUnits* à la création, et pouvant accueillir 3 développeurs maximum
- 40 tâches optionnelles, avec une estimation respective de 3 *WorkUnits* à la création, et pouvant accueillir 3 développeurs maximum

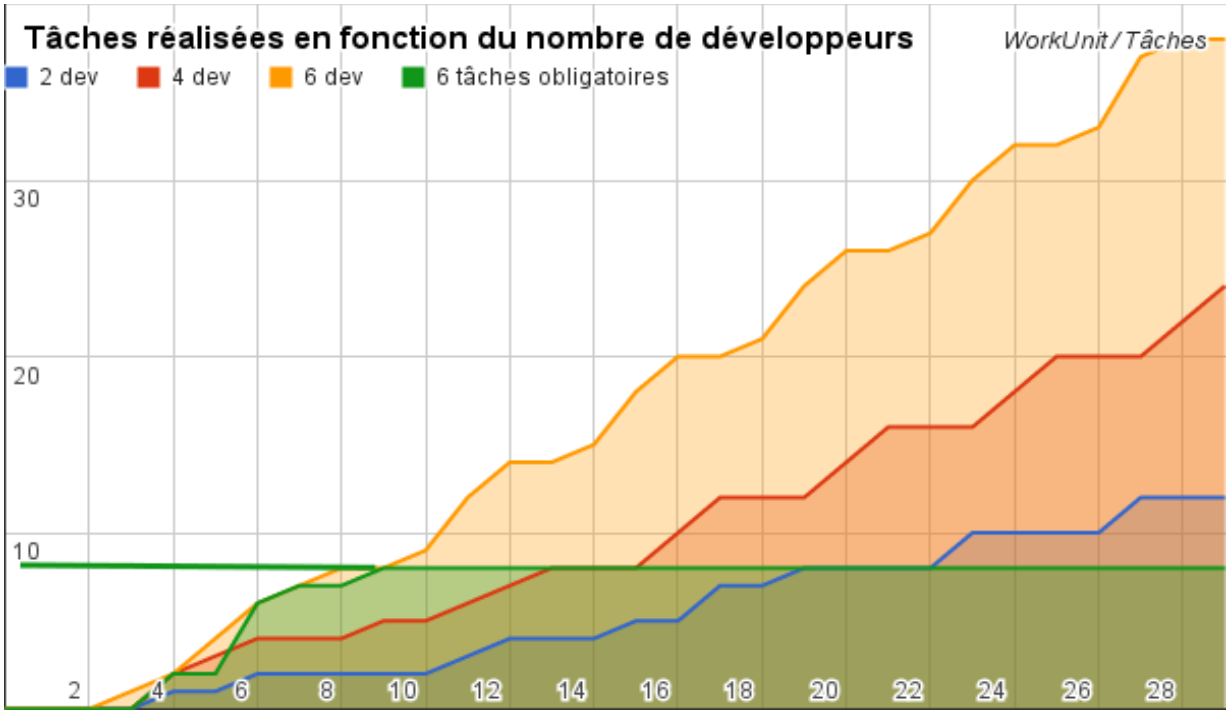
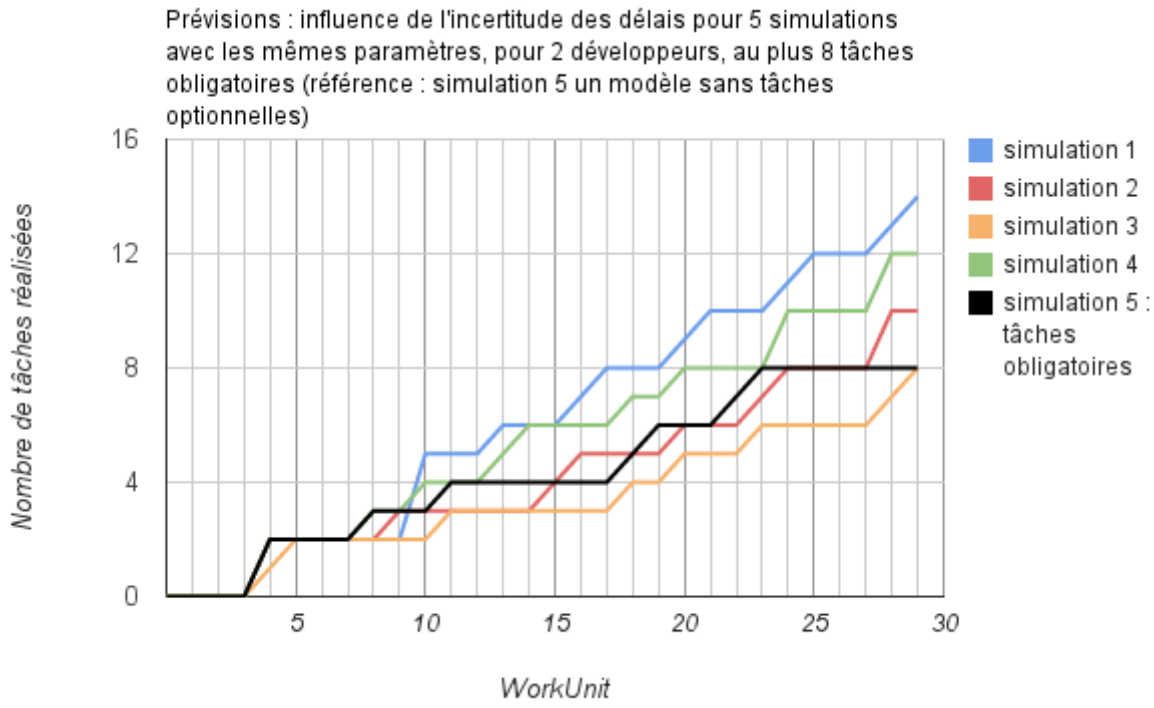


Illustration 1: Ligne verte indicateur du nombre de tâches minimal, les tâches réalisées au cours du temps sont plus nombreuses avec un plus grand nombre de développeurs

Interprétation : chaque simulation permet d'avoir une estimation du nombre de tâches terminée à un instant *t*. Nous observons qu'avec 6 développeurs les tâches obligatoires sont terminées au bout de 9 *WorkUnits*. En revanche, avec seulement 2 développeurs, elles se terminent au bout de 20 *WorkUnits*.

Influence de l'aléatoire

A chaque itération de la simulation, c.a.d lorsqu'un *WorkUnit* arrive à son terme, les délais sont mis à jour de façon aléatoire pour simuler les incertitudes dans l'avancement d'un projet réel. Sur le schéma suivant, les simulations ont été exécutées avec les mêmes paramètres. Au minimum 8 tâches devaient être implémentées avant la fin du projet.



Interprétation : pour des simulations avec les mêmes configurations nous n'obtennons pas les mêmes prévisions. Le nombre de tâches réalisées à un instant *t* n'est pas régulier. Au cours de la 3e simulation, les tâches n'aboutissent qu'au terme du 29e *WorkUnit*. La signification dans un environnement de production réel serait d'être vigilant concernant les délais estimés et d'être pessimiste face aux résultats de simulation, autrement dit de considérer la simulation n° 3 comme plus sûre.

Conclusion

L'outil implémenté permet de prendre connaissance des délais nécessaires pour implémenter une ou plusieurs tâches. La durée raccourcie plus le nombre de développeurs augmentent. De plus l'incertitude dans la planification de projet est observée dans les résultats de simulations : avec les mêmes environnements et configuration, les simulations n'obtiennent pas les mêmes délais.

Les systèmes multi-agents (SMA) permettent de représenter des modèles complexes avec simplicité. En respectant des patterns simples, et en utilisant très simplement de l'héritage, les agents peuvent être étendus. Il est également facile d'intégrer de nouvelles stratégies au sein de la simulation.

L'émergence de délais non prévus manuellement est une aide précieuse pour l'établissement d'un planning de développement pour des logiciels ambitieux. L'utilisation de l'aléatoire est également au centre des algorithmes mis en exercices. Mais, le modèle pourrait être largement amélioré.

Pour se rapprocher d'un environnement réel, les agents développeurs pourraient être allouées à plusieurs tâches. De mêmes les capacités pourraient évoluer au cours de la simulation si un agent reste sur une tâche à long terme.

Dernièrement, un outil de simulation de planification ambitieux serait de proposer un modèle cognitif. Les agents développeurs disposeraient de leur propres moyens de perception, décision, et moyen de communication.