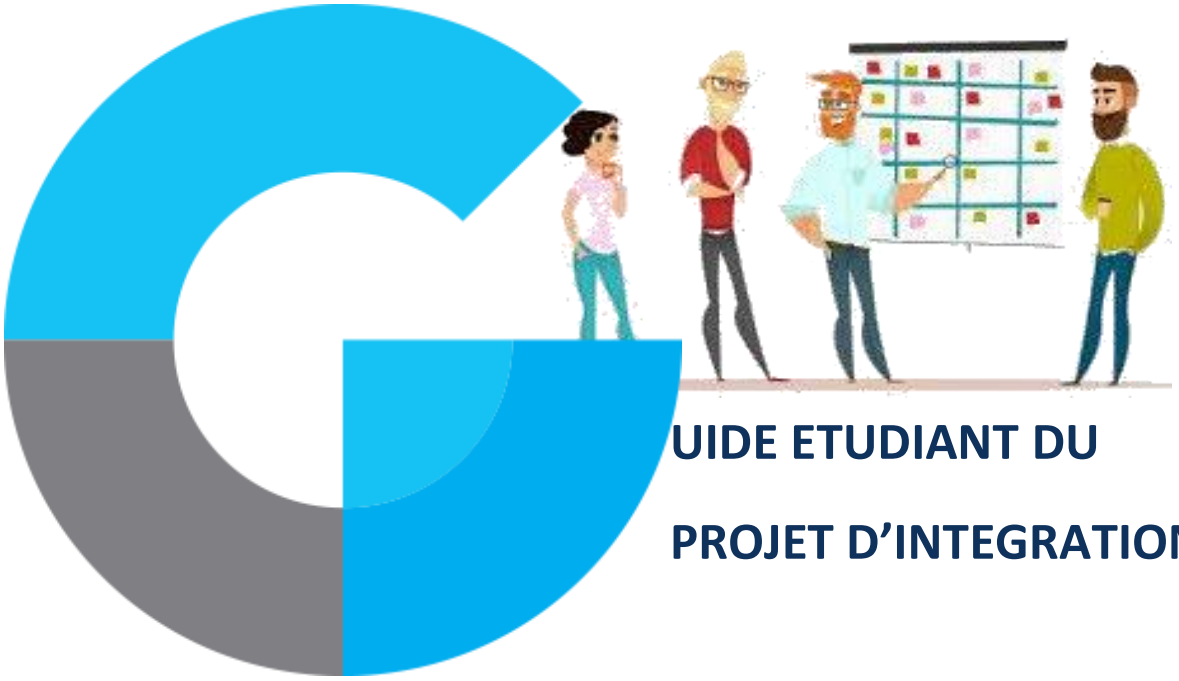


GUIDE ETUDIANT DU PROJET D'INTEGRATION

« Il fût un temps où il suffisait d'avoir un diplôme pour trouver un emploi. »

Nous sommes conscients qu'il ne suffit plus d'avoir un diplôme pour décrocher un travail. Les statistiques confirment cette triste réalité : 30% des jeunes diplômés Tunisiens sont en chômage. En effet, face au nombre important des nouveaux diplômés injectés annuellement dans le marché de travail, les responsables des sociétés, sont aujourd'hui contraint à chercher des compétences techniques et des soft skills plutôt que des diplômes

Dans ce constat, et pour mieux vous guider dans le développement de vos compétences en matière de développement de système d'information, nous avons pensé à vous proposer un guide de projet tuteuré, destiné à mettre en pratique et à compléter les concepts enseignés, non seulement ceux du domaine de l'informatique mais aussi ceux de la conduite de projet. Il favorise l'acquisition d'un "savoir- faire" et d'un "savoir être" dans une optique professionnelle, tout en développant vos qualités d'organisation et de méthode. Il constitue un premier pas vers le travail à réaliser lors de vos stages de PFE, et dans la vie professionnelle.

Objectif général

L'objectif majeur de ce manuel « Projet tuteuré » est de transformer l'espace d'enseignement des étudiants en un espace professionnel. L'étudiant se voit alors émergé dans un environnement professionnel semblable à celui des boites de développement informatique. Le laboratoire informatique est transformé en un espace d'entrainement pédagogique où les mêmes exigences et procédures de travail aussi bien que les méthodologies de conduite de projets informatiques sont appliquées. Il offre ainsi aux étudiants l'opportunité d'ancrer activement les enseignements aux situations professionnelles.

Le projet est également un moment important pour saisir l'imbrication des différentes dimensions abordées au cours de la formation. Il est à la fois un temps de réflexion, d'approfondissement et de « mise à l'épreuve » sur le terrain de certaines notions et connaissances acquises, un temps aussi de découverte d'une situation opérationnelle associée à la formulation et la gestion d'un projet et, in fine, d'acquisition de nouvelles compétences. Il s'agit bien d'une logique d'apprentissage.

Objectifs Spécifiques

A la fin de ce module vous serez capable de :

- Mettre en pratique les savoirs acquis au cours de la formation
- Répondre à une demande réelle en analysant une situation et en proposant une solution adaptée
- Travailler au sein d'une équipe sous contrainte de temps et de moyens
- Appliquer une approche de développement (Agile)
- Partager et diffuser des informations entre les différents membres du même groupe
- Gérer et partager son code à travers un outil de gestion de versionning
- Appliquer des bonnes pratiques de développement
- Développer un composant spécifique de l'application et assurer son intégration continue tout au long du processus du développement
- Rapporter l'état d'avancement de son projet.
- démontrer vos capacités d'initiative, d'autonomie et de responsabilité,

La définition d'un cahier des charges, la solution proposée, la conception, l'architecture de l'application, la composition de l'équipe, l'organisation des tâches, les outils de gestion de versionning, de reporting et d'intégration, la conduite du projet sont des éléments indispensables de la réalisation de votre travail.

Prérequis

Pour pouvoir suivre ces ateliers, vous devriez maîtriser les notions suivantes :

- POO et POOA
- BD et SGBD
- Architecture des logiciels
- Génie logiciel
- Méthodologie de conception

Méthodes d'Évaluation

Les éléments pris en compte pour l'évaluation sont les éléments suivants :

Utilisation d'un outil de travail collaboratif : l'utilisation d'un outil de travail collaboratif (échanges, rendus, saisies du temps de travail, compte rendu de réunions, …) sera prise en compte.

Soutenance : qualité du support (forme et fond), qualité du discours et respect du temps.

DT : qualité du support (forme et font) et respect des contraintes de mise en page.

Documents généraux : qualité du support (forme et font) et respect des contraintes de mise en page.

Une attention particulière sera apportée par les évaluateurs au respect

De la propriété intellectuelle : toute forme de plagiat (même minime) sera sanctionnée par un 0 à la note globale de projet tutoré.

Des dates de rendu : tout retard sera lourdement sanction. La sanction s'appliquera à la note globale du projet.

Du temps de travail : chaque étudiant doit travailler 80h sans quoi sa note sera réduite en conséquence.

Du respect des consignes : le non-respect de l'ensemble des consignes indiquées dans ce document ou transmises ultérieurement par le responsable des projets tutorés sera lourdement sanction. La sanction s'appliquera à la note globale du projet.

Répartition des séances

Atelier	Intitulé
Atelier 1	DESCRIPTION ET PLANIFICATION DU PROJET
Atelier 2	SCRUM : UNE DEMARCHE AGILE POUR GERER VOS PROJETS
Atelier 3	CONCEPTION ET ARCHITECTURE LOGICIEL DU PROJET
Atelier 4	TRAVAIL EN EQUIPE- GESTION DES VERSIONS ET PLANIFICATION DU PROJET
Atelier 5-7	DEVELOPPEMENT DU PROJET AVEC SCRUM
Atelier 8	PRESENTATION TECHNIQUE ET COMMERCIAL DU PRODUIT
Atelier 9	EVALUATION

Atelier 1

DESCRIPTION ET PLANIFICATION DU PROJET

OBJECTIF DU TP 1

L'objectif de ce TP est de vous décrire les finalités du module « Projets tuteurés » et de vous décrire le projet à développer par l'équipe à laquelle vous appartenez.

INTRODUCTION

Ce projet correspond à 126 heures de travail (42H en présentiel et 84H en non-présentiel) et a pour objectif la mise en œuvre des méthodes de conduite de projet. Les compétences visées sont :

- Mettre en œuvre de façon globale, sur une étude donnée, vos connaissances acquises lors votre formation en licence qui permettent de mettre en corrélation enseignements et projet.
- Développer vos « savoir-faire » par une mise en situation propice à l'observation et à l'échange au sein de votre groupe de travail.
- Apprendre à travailler dans un groupe-projet de façon efficace et enrichissante dans une double perspective de développement de votre autonomie et de capacité à travailler et à vous organiser en équipe.
- Prendre conscience de la distance à prendre pour développer une réflexion critique, constructive, pertinente et apprendre à la communiquer à votre « commanditaire ».
- Apprendre à chercher et synthétiser l'information.

Des groupes de **3 à 4 étudiants** sont formés pour réaliser le projet.

Attention, les projets sont un travail en groupe, non encadré !! C'est à dire que votre tuteur ne passe en séance que si vous le sollicitez pour répondre à des questions précises et non faire le travail à votre place. Il ne s'agit ni d'un TD, ni même d'un TP où les enseignants sont là pour vous guider ou vous aider dans les exercices, ... les projets tutorés sont votre travail!

TRAVAIL A FOURNIR

Le travail à fournir par chaque groupe consiste en

- L'analyse, la conception, et le développement d'un projet informatique complet réalisable en 14 semaines. Ce projet impliquera 126h/étudiant.
- Un rapport à la fin de la 12ème semaine ainsi qu'une présentation technique et commercial réalisé par chaque groupe à partir de la 13ème semaine.

Le travail se décompose en 2 parties, l'une globale partagée entre tous les étudiants du même groupe et centrée sur la gestion de projet. Une autre organisée en équipes où chaque membre de l'équipe aura en charge un lot du tâches à développer. Bien entendu, lors de cette deuxième phase, il y aura de fortes collaborations entre les membres d'une même équipe.

DECOMPOSITION GLOBALE ET PLAN DE TRAVAIL

Le but du projet tuteuré consiste à, d'une part, vous initiez à la gestion de projet et, d'autre part, à vous appréhendez les phases d'analyse et de conception d'un projet informatique (la phase des tests et validation du projet n'est pas réalisée). 40% de la note du projet portera sur la gestion du projet, les 60% restant porteront sur les résultats de l'analyse, de la conception, et du développement. Le projet tuteuré se décomposera en 2 grandes étapes.

- **Etape 1** : Préparation et analyse du projet (semaine 2 et 3)
- **Etape 2** : Analyse et conception (le cœur du projet : semaine 3,4 et 5)
- **Etape 3** : Développement du projet

  Les étapes 2 et 3 seront menées parallèlement et guidées par une démarche agile.

L'étape 1

Cette étape va servir à prendre connaissance de votre projet et de vous organiser au sein d'un même groupe de projet.

Il vous est demandé, sans la présence de votre tuteur, de réfléchir au projet. Vous devez établir la liste des taches à faire durant ce projet. Vous devez également préparer une liste de questions précises sur le sujet à laquelle votre tuteur répondra. Un créneau en Laboratoire est prévu pour répondre à vos questions sur le déroulement du projet mais surtout sur votre sujet (atelier 2 et 3). Ce document devra être déposé sur la plateforme dès que votre groupe aura un accès.

Avant de vous engager dans votre projet, vous allez discuter du sujet, avec votre tuteur. Celui-

ci répondra à toutes les questions concernant le sujet et le déroulement du projet (atelier 1 et 2), mais ne vous donnera pas d'information concernant votre organisation. Suite à la remise des projets à travers leurs cahiers des charges, lors de la deuxième semaine, l'ensemble des actions suivantes devront être réalisées pour chaque groupe :

- Préparer et poser toutes les questions nécessaires pour mieux comprendre le projet
- Identifier et comprendre les fonctionnalités du projet. Un diagramme de cas d'utilisation global sera idéal pour illustrer ces différentes.
- Il est à la responsabilité de l'un de l'équipe d'indiquer les présents, les absents et le travail réalisé durant la deuxième semaine.
- Organiser, planifier votre travail pour le reste du projet. Vous devrez finaliser votre décomposition du projet en lots.
- Discuter de l'organisation que vous proposez avec votre tuteur. Celui-ci vous ferra des retours vous permettant de corriger et valider votre décomposition en lots et en tâches.
- A l'aide de l'analyse, un début de décomposition des lots en tâches devra être proposé. L'ensemble devra être consigné dans un document qui devra être communiqué via la plateforme à votre tuteur et déposé sur la forge avant la fin de la 3ème semaine

L'étape 2 & 3

Il s'agit de la suite du projet, et donc, pour ce qui concerne les phases de conception (globale et détaillée) et la phase de développement. Normalement vous avez dû définir votre plan de travail dans le DT (rendu au début de la semaine dernière). Il ne vous reste plus qu'à le suivre. Vous devez donc suivre votre propre organisation, saisir quotidiennement vos heures de travail sur les tâches de la forge et rendre vos livrables aux dates indiquées. Il vous est conseillé de faire un point régulier sur votre travail avec votre tuteur (au toutes les 2 semaines) de manière à ne pas être perdu.

DESCRIPTION DU PROJET PROPOSE

CONCEPTION ET REALISATION D'UN SYSTEME D'INFORMATION DES ETUDIANTS DU DEPARTEMENT INFORMATIQUE

Présentation du projet :

Objectif : L'objectif principal de ce projet est de développer une application web qui permet de gérer le système d'information des étudiants du département informatique. Pour aboutir à cette fin, nous allons tout d'abord effectuer une étude conceptuelle de l'application. Cette dernière nous permettra, en effet, d'accéder facilement à la réalisation de l'application en organisant les idées et en structurant le processus de codage suivant des diagrammes.

Cible et caractéristique : Description de la cible : Étudiants, administrateurs, enseignants. Le projet vise principalement les étudiants du département informatique.

– Cible principale et description : les Étudiants du département.

– Cible secondaire et description : les enseignants et les administrateurs.

Description du contenu : La première page c'est une page d'authentification qui contiendra un formulaire d'authentification pour les étudiants déjà inscris et un onglet registre qui redirige l'internaute vers un formulaire pour l'inscription, la vérification sera faite par la suite, après l'authentification l'étudiant est redirigé vers la page principale, elle contiendra par ailleurs un sidebar à travers lequel on peut effectuer les fonctionnalités que propose le système.

Études de l'existant

Le système existant, s'appuie sur la vérification manuelle des données, ce qui éventuellement entraine plusieurs anomalies :

- Difficulté d'accès aux données ;
- Difficulté de la communication entre administration et étudiant ;
- Gaspillage de temps ;
- Limitation des services estudiantins ;
- Absence d'ergonomie.

Spécification des besoins fonctionnels : L'application que nous allons développer devra regrouper toutes les fonctionnalités nécessaires pour :

- S'authentifier.
- S'inscrire.
- Visualiser des cours.
- Rechercher des cours.
- Visualiser les statistiques des notes.
- Visualiser les statistiques des absences.
- Ajouter des cours.
- Visualiser les emplois du temps.
- Demander un service.
- S'inscrire dans un pfe

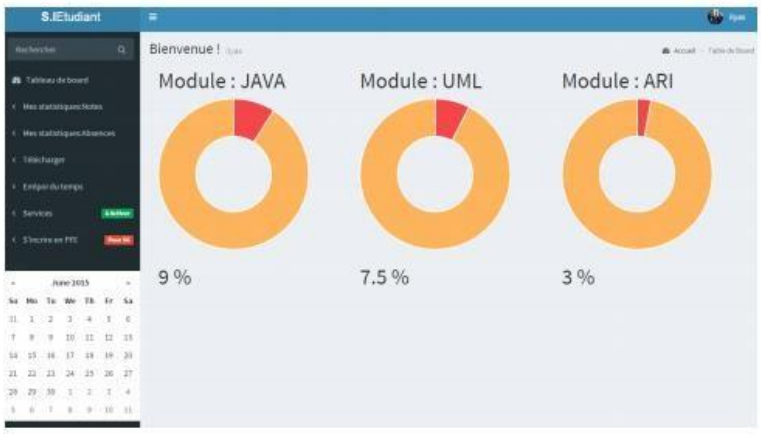
S'authentifier : L'utilisateur devra pouvoir s'authentifier à travers un email et un mot de passe. Le système vérifie l'authentification et le type d'internaute (étudiant, enseignant, administrateur ou un simple internaute).

S'inscrire : Chaque utilisateur pourra effectuer une inscription en saisissant ses informations personnelles et en choisissant son type (Étudiant / Enseignant), afin de permettre au système de vérifier son existences au sein du département.

Visualiser des cours : Chaque étudiant pourra consulter la liste des cours à tout moment et les télécharger.

Rechercher cours par mot-clé : L'étudiant devra pouvoir trouver le plus rapidement possible un cours recherché par un mot clé qui lui est associé

Visualiser les statistiques des notes : L'étudiant devra pouvoir visualiser les notes de chaque module selon les semestres.



Visualiser les statistiques des absences : L'étudiant devra pouvoir visualiser le pourcentage de ses absences dans chaque module.

Ajouter des cours : L'administrateur devra ajouter un cours et lui associer un mot clé. Ce

dernier sera automatiquement affiché chez l'étudiant.

Visualiser les emplois du temps : Chaque internaute pourra visualiser et télécharger l'ensemble des emplois du temps du département informatique.

Demander un service : Chaque étudiant devra pouvoir effectuer une demande d'un service auprès de l'administration, cette demande sera traitée par l'administrateur qui recevra la liste des étudiants ayant demandé un service.

S'inscrire dans un pfe : Chaque étudiant devra remplir une fiche contenant les informations sur les deux binômes d'un pfe. Un administrateur sera chargé d'affecter un encadrant de pfe à chaque binôme.

Spécification des besoins non fonctionnels : Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt ils identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application sont les suivants :

- Sécurité : Les comptes des utilisateurs sont sécurisés par mot de passe (longueur, code système, expiration de sessions, etc)
- Fiabilité : Bon fonctionnement de l'application sans détection de défaillance.
- Performance : L'application répond à toutes les exigences des internautes d'une manière optimale.
- Convivialité : Un design graphique clair et simple pour faciliter l'utilisation à l'utilisateur.
- Portabilité : L'application est multiplateforme. Elle fonctionne sur tous les systèmes d'exploitation et tout type de terminal.
- Responsive Design pour la version mobile de l'application

Atelier 2

SCRUM : UNE DEMARCHE AGILE POUR GERER VOS PROJETS

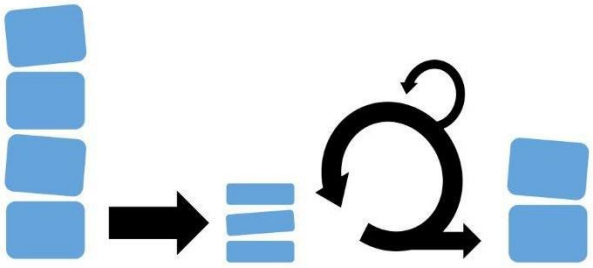
OBJECTIF DU TP 2

L'objectif de ce TP est de vous permettre d'acquérir de solides bases sur ***Scrum*** et ses principes fondamentaux en explorant sa mécanique et les raisons de son efficacité.

Vous pouvez aussi, à la fin de ce TP, et à travers un jeu éducatif, mettre en évidence un certain nombre du manifeste agile décrite exhaustivement au début de l'atelier.

METHODOLOGIE ADOPTE DANS LE DEVELOPPEMENT [http://www.agiliste.fr/guide-de-demarrage-scrum/#D-marrage]

Le **framework Scrum** (« Scrum » signifie « Mêlée » en anglais), ou plus exactement le **cadre méthodologique** Scrum est de loin la méthode Agile la plus utilisée dans le monde. Expérimentée en 1993, elle bénéficie aujourd'hui de nombreux retours d'expérience. Les conférences, communautés, formations, blogs, outils et ouvrages à son sujet ne manquent pas.



Au sujet de Scrum

Parler d'une **« méthode »** concernant Scrum n'est pas ce qu'il y a de plus approprié. Scrum ne se considère pas comme une méthode mais comme un cadre méthodologique. Une méthode dit généralement **« comment »** faire les choses. Scrum ne dit pas comment réussir son logiciel, comment surmonter les obstacles, comment développer, comment spécifier, etc. Il se contente d'offrir **un cadre de gestion de projet Agile** (et c'est déjà beaucoup) : des rôles, un rythme itératif, des réunions précises et limitées dans le temps, des artefacts (product backlog, sprint backlog, graphique d'avancement) et des règles du jeu.

Au sein de ce cadre méthodologique de gestion de projet, les acteurs ajustent empiriquement, au fil des itérations, leur propre méthode en fonction de leur contexte. On

peut qualifier Scrum de simple, pragmatique, transparent et empirique. Scrum ne couvrant que les aspects de gestion de projet, c'est souvent la méthodee Xtreme Programming (XP) qui vient compléter le vide laissé en matière de pratiques de développement. XP apporte ainsi les pratiques de programmation en binôme, de développement piloté par les tests (TDD ou Test Driven Development), intégration continue, etc. Ces dernières jouent un rôle capital pour relever le défi du développement incrémental. Le mouvement Software **Craftsmanship** est également là pour répondre à ces enjeux techniques.

Utilisation de Scrum

Pré requis recommandés

- ✓ Un grand mur libre et dégagé dans l'espace de travail de l'équipe.
- ✓ Blocs de post-it et marqueurs.
- ✓ L'ouvrage « Scrum et XP depuis les tranchées ».
- ✓ Jeu de cartes ou logiciel de Planning Poker.

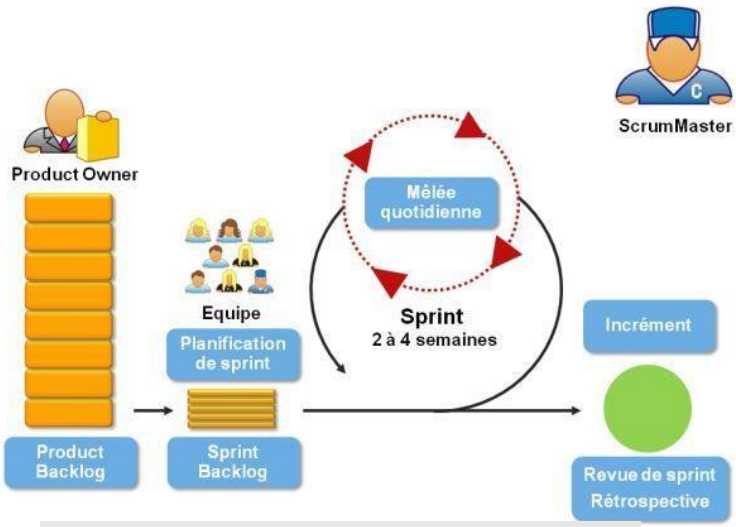


Figure 2 *Processus Scrum (source des icônes des personnages*

Les Rôles en bref

Scrum définit seulement 3 rôles :

- Le Product Owner qui porte la vision du produit à réaliser et travaille en interaction avec l'équipe de développement. Il s'agit généralement d'un expert du domaine métier du projet.
- L'Equipe de Développement qui est chargée de transformer les besoins exprimés par le Product Owner en fonctionnalités utilisables. Elle est pluridisciplinaire et peut donc encapsuler d'autres rôles tels que développeur, architecte logiciel, DBA, analyste fonctionnel, graphiste/ergonome, ingénieur système.
- Le Scrum Master qui doit maîtriser Scrum et s'assurer que ce dernier est correctement appliqué. Il a donc un rôle de coach à la fois auprès du Product Owner et auprès de l'équipe de développement. Il doit donc faire preuve de pédagogie. Il est également chargé de s'assurer que l'équipe de développement est pleinement productive. Généralement le candidat tout trouvé au rôle de Scrum Master est le chef de projet. Celui ci devra cependant renoncer au style de management « commander et contrôler » pour adopter un mode de management participatif.

Vision du produit et product backlog

La première étape consiste à formaliser la **vision du produit** (logiciel) que l'on souhaite réaliser. Cette vision décrit les principaux objectifs, jalons, utilisateurs visés. Elle contribuera à guider et fédérer les acteurs du projet. La suite consiste à établir la liste des **exigences fonctionnelles** et **non fonctionnelles** du produit. Chaque exigence est ensuite estimée par l'**équipe de développement** avec la technique de **Planning Poker**. A la lueur des estimations, la liste ainsi complétée est ordonnancée. Les exigences seront converties en fonctionnalités utilisables selon cet ordonnancement. Le principe étant de convertir en premier les exigences qui apportent le plus de valeur ajoutée (ou ROI) au commanditaire. Il s'agit donc de faire remonter les exigences fonctionnelles de la plus haute valeur ajoutée (ou dont le ROI est le plus élevé) en haut de la liste. Cette liste est appelée le **Product Backlog**. Le Product Backlog servira à piloter l'équipe de développement et pourra évoluer tout au long du projet. **Le changement est non seulement autorisé mais encouragé afin de pouvoir éliminer les idées de départ qui s'avéreront mauvaises et de prendre en compte les nouvelles idées qui arriveront en cours de route.** Cette activité de construction du Product Backlog est collaborative, elle implique le Product Owner et l'équipe de développement.

Pondérer chaque exigence d'une valeur ajoutée n'est pas toujours évident pour une **équipe métier** (MOA) novice. Différentes techniques s'offrent à vous. Le fameux **Planning Poker** (voir paragraphe ci dessous) peut s'avérer utile pour déterminer collectivement les pondérations en « points » de valeur ajoutée. On peut également utiliser des échelles de valeur plus ou moins fine (exemple : « faible », « moyenne », « haute » ou encore les tailles de tee-shirt : XS, S, M, L, XL, XXL).

Estimations des exigences

L'ensemble des fonctionnalités de la liste doivent être estimées par l'**équipe de développement** afin de permettre les futurs engagements de cette dernière. Impliquant plusieurs développeurs, le **Planning Poker**permet de mettre à profit les expériences de chacun et de parvenir rapidement à une estimation optimale et objective. Avant ou pendant les estimations, le Product Owner pourra être sollicité afin de répondre aux questions de l'équipe de développement. A ce stade, le besoin pourra être approfondi, mais sans aller trop loin (il s'agit simplement d'**estimer** le coût de chaque exigence). La conception détaillée se fera pendant les itérations (sprints).

DEMARRAGE

Vous pouvez commencer par déterminer ensemble (**équipe de développement** et **product owner**) la **durée des itérations** ou Sprints (2 semaines maximum). Cette durée devra être la même pour l'ensemble des sprints afin de maintenir un rythme régulier propice aux automatismes et pouvoir construire des indicateurs de pilotage fiables. Un projet démarre généralement par ce qu'on appelle souvent le « **sprint 0** » dédié aux travaux préparatoires du projet (ex : construction du **product backlog** et de la **vision du produit**, préparation des environnements, mise en place de l'intégration continue, définition de l'architecture

générale du projet, initiation des acteurs à Scrum, etc.). Exceptionnellement, la durée de ce « sprint 0 » ne respecte pas forcément la durée fixée précédemment. Mais inutile de le faire durer trop longtemps, souvenez-vous (cf. fiche pratique « introduction aux méthodes Agile »), l'idée est de se lancer sans élaborer au préalable un plan et une architecture millimétrés qui risqueraient de nous enfermer, de nous frustrer, voire de nous coûter cher à courts et longs termes. **L'architecture doit être souple et émerger au fil des sprints**.

Réunion de planification de sprint

Durée maximum : 2 heures par semaine de sprint (autrement dit : 4 heures pour des sprints de 2 semaines).

Phase 1 : Le « Quoi »

Une fois que le **Product Backlog** est suffisamment complet et ordonnancé, on peut planifier un sprint. Le**Product Owner** revoit alors avec l'**équipe de développement** la **vision du produit**, la roadmap, le plan de livraison (jalons et deadline), l'objectif du sprint et le Product Backlog. L'équipe de développement vérifie les estimations, confirme qu'elles sont exactes et sélectionne en haut du Product Backlog les exigences qu'elle se sent capable de convertir en fonctionnalités utilisables d'ici la fin du sprint (il s'agit d'une prévision et non pas d'un engagement « contractuel »).

Phase 2 : Le « Comment »

L'**équipe de développement** fait ensuite l'inventaire des tâches qui permettront de convertir les exigences sélectionnées en fonctionnalités utilisables d'ici la fin du sprint. Toutes les exigences n'ont pas nécessairement besoin d'être découpées en tâches. En cas de manque de temps, l'équipe de développement peut se contenter de découper celles qui seront réalisées au cours des premiers jours du sprint (elle découpera en cours de sprint les autres exigences). Elle doit cependant aller suffisamment loin dans l'effort de conception pour pouvoir vérifier sa prévision. Si elle constate après analyse des exigences sélectionnées, que sa prévision est erronée, elle peut réajuster avec le Product Owner la liste des exigences sélectionnées.

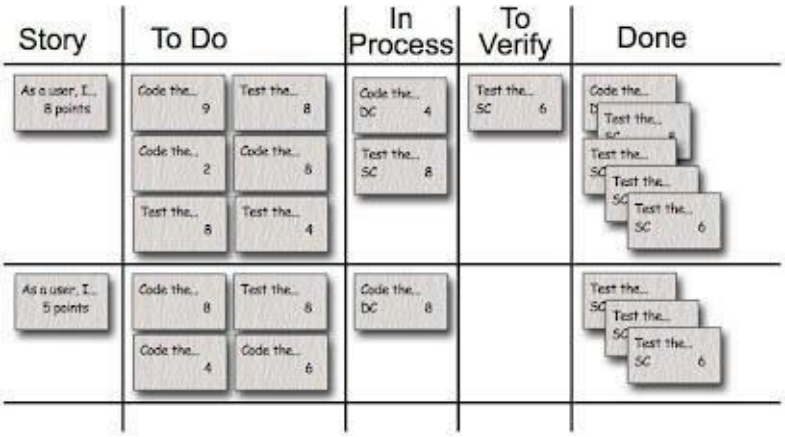


Figure 3 *Exemple de tableau des tâches*

Les **tâches de développement** sont centralisées dans le **Sprint Backlog** et ajoutées au **tableau des tâches**physique (aussi appelé Kanban, même si Kanban veut dire bien plus).

L'idéal est de parvenir à un découpage relativement fin (inférieur ou égal à une journée de travail).

Chacun peut personnaliser les colonnes de son **tableau des tâches**, ou code couleur des post-it. A titre d'exemple, voici la photo du tableau que nous avons utilisé sur l'un de nos premiers projets (au passage, vous pouvez également jeter un œil à la vidéo tournée sur ce projet et au retour d'expérience associé). Les **User Stories** (une **User Story** représente une exigence, voir annexe pour plus de détails) sont les gros post-it larges (rose et orange), les sous tâches des User Stories sont en jaune, on trouve en bleu des tâches imprévues indépendantes. Au dessus du tableau figurent les obstacles courants, un graphique d'avancement appelé **burdown chart** ainsi qu'un calendrier géant avec les dates clefs et les congés de chacun. Sur la droite on trouve des maquettes d'IHM, des documents métier, etc. Cette réunion de planification est l'occasion de préciser ou rappeler à l'équipe la **définition de « terminé »** pour une **user story** ou exigence. Exemple de définition de « terminé » : code commité, testé unitairement, documenté, testé en intégration, revue par un pair, **tests d'acceptation** de la user story passants.

Sprint (2 à 4 semaines)

Au cours du Sprint, l'équipe se concentre sur l'accomplissement des tâches du **Sprint Backlog**. En cas de retard (indiqué par le **Burndown Chart**), des exigences ou tâches seront retirées du Sprint Backlog en cours de route en essayant de préserver l'objectif du sprint (pour cela, il est conseillé d'ordonnancer les exigences au sein du sprint). Et inversement, si l'équipe avance plus vite que prévu, des exigences ou tâches y seront ajoutées. En accord avec le **Product Owner** dans les deux cas.

Les développements se font verticalement et non pas horizontalement par couche. Le but est de développer les fonctionnalités de bout en bout (de la conception aux tests) au fil de l'eau au cours du sprint. Autrement dit d'éviter un mini **cycle en V** au sein du sprint, voire de se retrouver avec une surcharge d'effort de test en fin de sprint. Les développeurs doivent donc éviter de trop paralléliser les exigences et encore moins les tâches de développement. Pour cela, le **pair programming** peut se révéler utile ainsi que la définition d'une limite maximum d'éléments au sein d'une colonne du tableau des tâches (voir notion de **Work In Progress** du Kanban). Si l'équipe de développement n'est pas convaincue, le « jeu du prénom en mode multitâche » peut s'avérer utile.

Le **tableau des tâches** physique rempli de post-it est pratiquement indispensable. Il permet d'avoir une vision claire du travail à accomplir, en cours et terminé. Il peut également s'avérer précieux lors des réunions quotidiennes (voir § suivant), surtout si vous calculez à la main le « reste à faire » du Sprint afin de tracer le graphique d'avancement (voir plus loin le « Burndown Chart »). Le tableau facilite également l'affectation des tâches par l'équipe en ayant une vision d'ensemble du sprint en un coup d'œil. Inutile de se torturer à planifier à l'avance dans le détail l'activité de chaque développeur sur toute la durée du sprint. Il faut se

détacher d'une approche prédictive et des diagrammes de Gantt et renoncer au **style de management « commander et contrôler »**. Ce sont les développeurs qui « tirent » les tâches et non pas le **Scrum Master** qui les affecte.

Pendant le sprint, l'équipe de développement assistera le **Product Owner** dans ses activités d'affinage du Product Backlog. Cette assistance peut consister en des ateliers de conception anticipés, de priorisation ou d'estimation. Il faut compter environ 10% de la capacité à faire de l'équipe de développement pour ces activités.

Figure 4 : Photo d'une mêlée quotidienne

Mêlée quotidienne ou « stand-up meeting »

Durée maximum : 15 minutes.

Cette réunion qui se fait debout (ça évite de s'éterniser) est très importante. Elle permet quotidiennement aux membres de l'équipe de se synchroniser, de remonter les obstacles rencontrés, de s'entraider, de vérifier l'avancement du sprint. Elle contribue également à faire naître l'esprit d'équipe. A condition bien entendu de ne pas transformer cette réunion de « synchronisation » en réunion de « reporting » vers le Scrum Master.

Chaque personne répond à 3 questions :

- ✓ Qu'ai-je fait hier qui a aidé l'équipe de développement à atteindre l'objectif Sprint ?
- ✓ Que vais-je faire aujourd'hui pour aider l'équipe de développement à atteindre l'objectif Sprint ?
- ✓ Est-ce que je vois des obstacles susceptibles de m'empêcher ou d'empêcher l'équipe de développement d'atteindre l'objectif du Sprint ?

Le **Scrum Master** est ainsi immédiatement au courant des obstacles rencontrés, il doit impérativement les prioriser, les suivre et bien sûr s'efforcer de les lever au plus tôt afin de garder l'équipe pleinement concentrée et productive.

La **mêlée quotidienne** se déroule à lieu et heure fixes (devant le tableau des tâches physique de préférence) déterminés par l'équipe de développement. Au début le Scrum Master peut avoir à rappeler qu'il est l'heure de la mêlée et animer cette dernière en rappelant les 3 questions et évitant l'instruction des problèmes ou obstacles en séance afin de ne pas dépasser les 15 minutes imparties. L'objectif du Scrum Master consiste cependant à viser l'appropriation de la mêlée par l'équipe de développement.

Graphique d'avancement (Burndown Chart)

Pour connaître votre avancement, vous allez avoir besoin de tracer le Burndown Chartdu sprint en cours. Ce graphique est simple, il s'agit du tracé de la charge de travail restante (généralement en heures) en fonction du temps (en jours). Pour tracer ce graphique, il suffit de mettre à jour (lors de chaque mêlée quotidienne par exemple) le sprint backlog (voir illustration).

Sprint Backlog

Exigence	Sous Tâche	Reste A Faire								
Exigence X - #102	IHM	8	8	6						
	Mise à jour documentation	2	2	0						
	Services métier	10	4	0						
	Tests automatisés Perf	8	8	8						
Exigence YY - #103	IHM	8	8	8						
	Mise à jour documentation	2	2	2						
	Services métier	10	10	5						
	Tests automatisés Perf	8	8	8						
Exigence XY - #33	IHM	8	8	8						
	Mise à jour documentation	2	2	2						
	Services métier	10	10	7						
	Tests automatisés Perf	8	8	8						
Exigence Y - #365	IHM	8	8	4						
	Services métiers	10	5	0						
	Jours d'itération	Je	Ve	Lu	Ma	Me	Je	Ve	Lu	Ma	Me
	Trajectoire idéale	102	91	79	68	57	45	34	23	11	0
	Trajectoire réelle	102	91	66							

Figure 5 : exemple de Spring Backlog

Vous pouvez également construire un indicateur d'avancement de Release à mettre à jour en fin de chaque sprint (voir outillage proposé via le bouton ci dessous).

Revue de Sprint

Durée maximum : 1 heure par semaine de sprint (autrement dit : 2 heures pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

L'objectif de la revue de sprint est d'inspecter l'incrément produit au cours du sprint écoulé, faire un point sur l'avancement de la Release et adapter au besoin le plan et le Product Backlog. L'équipe de développement présente à tout acteur projet intéressé (à minima le Product Owner idéalement accompagné d'utilisateurs finaux) les nouvelles fonctionnalités développées au cours du sprint. Le Product Owner donne un feedback à l'équipe de développement, il accepte ou refuse les fonctionnalités présentées.

Figure 6 : Photo d'une revue de sprint

L'équipe de développement calcule sa **vélocité** en additionnant les points d'estimation associées aux fonctionnalités acceptées. Une fonctionnalité partiellement terminée ne rapportera aucun point car une telle fonctionnalité n'est pas utilisable. La vélocité ainsi calculée va permettre de mettre à jour le graphique d'avancement de Release et de vérifier

l'avancement de cette dernière. C'est l'occasion de vérifier que le nombre de sprints de la Release demeure adapté ou non.

La livraison à la fin de chaque sprint n'est pas obligatoire.

Rétrospective de sprint

Durée maximum : 45 minutes par semaine de sprint (autrement dit : 1 heure 30 pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

Figure 7: Photo d'une rétrospective

Cette réunion est généralement animée par le « ScrumMaster » qui s'adresse à son équipe. Elle a pour but d'améliorer continuellement le processus de développement de l'équipe en mettant les idées de chacun à contribution. Il existe différentes techniques d'animation de rétrospectives. Voir le livre « Agile Retrospective: Making Good Teams Great » pour en savoir plus.

L'une d'elle consiste à identifier et pondérer les éléments positifs (éléments à cultiver ou source de motivation) du sprint écoulé, puis les éléments à améliorer, puis de définir un plan d'action d'amélioration (en commençant par améliorer les éléments dont la pondération est la plus forte). Si les membres de l'équipe sont à l'aise pour s'exprimer, un simple tour de table permet de remplir le paper-board d'éléments. Dans le cas contraire, on peut avoir recours aux post-it (chacun se voit remettre des post-it vierges et un marqueur et inscrit ses idées dessus pour ensuite les transmettre à l'animateur. Une idée par post-it.). Pour pondérer les éléments, il suffit d'allouer un certain nombre de points (5 par exemple) à chaque membre. Chaque membre peut ventiler ses points à sa guise (exemple : 4 points sur un sujet qu'il considère très important, 1 point sur un autre sujet également important à ses yeux mais quatre fois moins que le premier). Pendant la phase d'inventaire des éléments à améliorer, veillez à ne pas essayer de trouver des solutions avant la phase dédiée au plan d'action d'amélioration. Vous risqueriez de ne pas faire remonter à la surface la majorité des « problèmes ». Ce serait dommage car, j'ai pu constater que parfois, le simple fait d'évoquer un problème sans forcément avoir le temps de trouver une solution entraîne une amélioration au sprint suivant.

Commencer par les points positifs peut être vital lorsque le sprint a été éprouvant. L'équipe aura peut être besoin de se nourrir des éléments positifs avant de s'attaquer aux éléments à améliorer.[La Gestion de Produit Agile expliquée en 15 minutes : http://www.agiliste.fr/lagilite-metier-en-2-mots/]

PENNY GAME : APPRENDRE SCRUM A TRAVERS UN JEU

Objectif du jeu

L'une des principales caractéristiques des processus agile, y compris ***SCRUM***, est le flux rapide des livrables pour les clients. Cette fluidité est assurée à travers un processus itératif qui à chaque itération traite un nombre particuliers de fonctionnalités, classées à priori selon un ordre de priorité. Chaque itération donne naissance à une version du produit (prototype).

Cependant, les communautés de développement, en particulier celles qui sont plus familières avec les techniques traditionnelles (***cascade, modèle en V***), peuvent ne pas reconnaître l'intérêt de développer des fonctionnalités plus petites dans des sprints de développement à court terme.

C'est pour cette raison que nous vous proposons ***Penny Game***, est un jeu éducatif qui met en évidence les avantages majeur de l'adoption d'une démarche Agile dans le processus de développement des logiciels.

De quoi vous avez besoin pour ce jeu

- Vous avez besoin d'un facilitateur (qui comprend le jeu) avec un tableau pour noter les résultats.
- Vous avez besoin de neuf participants pour la forme du jeu ***Penny*** assuré dans le cadre ce TP. Trois d'entre eux représentent une équipe de développement logiciel hypothétique composée d'un ***concepteur***, d'un ***développeur*** et d'un ***testeur***. Les produits de leur développement simulé sont présentés à un quatrième rôle, le ***client (product owner)***. Quatre autres joueurs (minuteurs) affectés à chacun des membres de l'équipe de développement y compris le client. Enfin, un **Scrum Master** gère l'exercice et enregistre les temps dans une feuille de données (voir tableau ci-dessous).
- Vous avez besoin de chronomètres ou de téléphones portables avec chronomètres pour chronométrer le travail de chaque membre de l'équipe.
- Vous avez besoin de 30 pièces de monnaie de différentes valeurs. Vous pouvez utiliser par exemple 10 x 100m, 10 x50, 5 x 2,m 5 x 5m.

Disposition des joueurs : travailleurs de l'équipe

Tous les travailleurs se tiennent autour d'une table. Tous les minuteurs se tiennent derrière leurs 4 membres de l'équipe avec un chronomètre. Le ***product owner*** se tient à côté des travailleurs à la table, et le ***scrum*** master se tient à côté du tableau pour enregistrer les durées et les valeurs de chaque projet.

Tour	Première valeur	Temps de travil				Valeur total du projet
		Concepteur	*Développeur*	*Testeur*	*Product Owner*	
1						
2						
3						
4						
5						

Principe du jeu:

Dans cette version simplifiée, la simulation est exécutée seulement 5 fois, et chaque exécution est appelée ***Tour (Rounde)***. Dans les trois premiers ***Tours***, les 20 pièces sont divisées en groupes ou en lots: soit un lot de 20 pièces (pas de division), 4 lots de 5 pièces, ou 20 lots de 1 pièce. Dans le 4éme tour les 20 pièces sont triées selon leurs valeurs puis divisées en 4 lots de 5 pièces. Dans le 5ème tour des contraintes du processus de développement sont relaxées. Le nombre de de ***sprint*** de chaque tour est équivalent au nombre de lot.

A chaque fois qu'un membre de l'équipe reçoit un lot de pièces, il doit les traiter en retournant chaque pièce. Une fois qu'il a retourné chaque pièce dans le lot, ils passent le lot au travailleur suivant. Une fois que les pièces ont été traitées par tous les travailleurs, elles sont renvoyées au client qui doit toucher le lot pour accepter le travail. À ce stade, le ***Scrum master*** calcule la valeur qui a été livrée (la somme des valeurs des lots de pièces touchées par le client), il doit aussi enregistrer la valeur du premier lot accepté par le client. Chaque minuteur mesure les temps de travail de son travailleur (retournement des pièces).

Dans ce TP, et pour mieux exposer les points forts d'une démarche agile ***Scrum***, nous avons opté pour des itérations de longueur fixe. Vous devriez donc, mesurer la valeur que l'équipe a apportée pendant cette période qui est fixée à **2 minutes**. Si l'équipe finit de traiter un lot de pièces et qu'elle est acceptée par le client, elle peut être traitée à nouveau pour gagner plus de valeur.

Les tours:

L'idée est de varier une chose à chaque tour pour voir comment elle affecte à la fois le temps et la valeur du projet. Pour ceux-ci, vous pouvez suivre les suggestions ci-dessous, ou vous pouvez inventer vos propres idées. Voici ce que vous pouvez faire au moins dans ce TP :

Tour 1: processus classique de développement

Sur un projet classique les pièces vont d'abord toutes être retournées par phase avant d'être passées à la phase suivante (on fini l'intégralité de la conception, avant de passer au développement).

- Les travailleurs ne peuvent utiliser que leur main gauche
- La taille du lot est de 20

- Le nombre total de pièces est 20: 5 x 100m, 5 x 50m, 5 x 20m, 5 x 5m (valeur = 825m)

Tour 2 : Réduire la taille du lot

- Les travailleurs ne peuvent utiliser que leur main gauche
- La taille du lot est de 5
- Le nombre total de pièces est 20: 5 x 100m, 5 x 50m, 5 x 20m, 5 x 5m (valeur = 825m)

Tour 3 : Réduire encore la taille du lot

- Les travailleurs ne peuvent utiliser que leur main gauche
- La taille du lot est de 1
- Le nombre total de pièces est 20: 5 x 100m, 5 x 50m, 5 x 20m, 5 x 5m (valeur = 825m)

Tour 4 : Utilisez uniquement des pièces de monnaie de grande valeur

- Les travailleurs peuvent utiliser les deux mains
- La taille du lot est de 5
- Le nombre total de pièces est 20: 10 x 100m, 10 x 50m (Valeur = 1500m)

Tour 5 : Supprimer l'empêchement

- Les travailleurs peuvent utiliser les deux mains
- La taille du lot est de 5
- Le nombre total de pièces est 20: 5 x 100m, 5 x 50m, 5 x 20m, 5 x 5m (valeur = 825m)

Atelier 3

CONCEPTION ET ARCHITECTURE LOGICIELLE DU PROJET

OBJECTIF DU TP 3

L'objectif de ce TP est d'appliquer les concepts théoriques de la conception et de l'architecture des logicielle afin de présenter une conception détaillée des différents projets présentés dans l'atelier 2 aussi bien qu'une architecture logiciel pour chaque projet.

TRAVAIL DEMANDE

A la fin de ce TP chaque équipe est appelée à :

- Identifier clairement les différentes fonctionnalités assurées par le système sélectionné dans l'atelier 2.
- Dresser le diagramme de classes global aussi bien que les diagrammes de classes raffinés du projet sélectionné dans l'atelier 2.
- Dresser les diagrammes de séquences décrivant l'acheminement des différents messages dans les cas d'utilisation les plus importants.
- Développer le modèle relationnel relatif aux données de votre future application.
- Proposer une architecture globale de votre future application. Un diagramme de composant et un diagramme de déploiement seront appréciés.
- Développer les IHM les plus importantes pour votre future application

N'hésiter pas à demander de l'aide à votre enseignant à chaque fois que vous en avez besoin.

- Un rapport dument rédigé doit être remis au plus tard au début de l'atelier 4.