Module 426

Version 1.0.0

CPNV

déc. 20, 2024

Table des matières

1 Scrum **3**
- 1.1 Préparation du 1er sprint . . . 3
  - 1.1.1 Durée du sprint . . . 3
  - 1.1.2 Le Planning Poker . . . 3
  - 1.1.3 Tableau de bord Scrum . . . 4
  - 1.1.4 La vélocité . . . 4
- 1.2 Stand-up Meeting (Daily Meeting) . . . 4
- 1.3 Les rôles . . . 4
  - 1.3.1 PO . . . 4
  - 1.3.2 Scrum Master . . . 5
  - 1.3.3 Equipe de développement . . . 5
- 1.4 User Stories (Histoire d'utilisateurs) . . . 5
  - 1.4.1 Comment écrire une User Story . . . 5
  - 1.4.2 INVEST . . . 6
  - 1.4.3 Ecrire des tests d'acceptation . . . 6
  - 1.4.4 Définir les critères d'acceptation . . . 6
- 1.5 Product Backlog . . . 6
  - 1.5.1 Prioriser le Product Backlog . . . 7
- 1.6 Revue de Sprint . . . 7
- 1.7 Rétrospective . . . 8

2 Git **9**
- 2.1 Télécharger la version portable . . . 9
- 2.2 Télécharger Aide Git . . . 9
- 2.3 Liste des commandes les plus couramment utilisées . . . 9

3 Sources **11**
- 3.1 Scrum . . . 11
- 3.2 Git . . . 11

No Module
426

Titre
Développer un logiciel avec des méthodes agiles

Compétence
Appliquer des méthodes agiles pour le développement de logiciels dans les cycles de révision.

Objectifs opérationnels

1. Mettre en oeuvre une fonctionnalité donnée, dans le cadre d'un projet logiciel, avec une méthode agile.
2. Réaliser et tester par étapes les fonctionnalités à l'aide de pratiques agiles dans les cycles de révision prescrits, et présenter en résumé la version logicielle.
3. Mettre en oeuvre, de manière ciblée, des échantillons de développement existants et/ou des composants logiciels testés pour résoudre le problème.
4. Refléter les résultats et le déroulement du travail lors d'un cycle de révision, en déduire les conclusions pour la suite du déroulement.
5. Mettre à disposition les documents du projet et les codes sources du programme dans un système de gestion des versions.
6. Formuler de manière compréhensive le code source programme selon les conventions.

Author
Cindy Hardegger

Email
cindy.hardegger@eduvaud.ch

Mise à jour
déc. 20, 2024

Références
— `Identification du module`
— `DEP`

Table des matières

CHAPITRE 1

Scrum

1.1 Préparation du 1er sprint

Durant le 1er sprint, le product owner présente le sprint backlog (les éléments à faire en priorité qu'il aura sélectionné dans le product backlog). Les membres de l'équipe posent toutes les questions nécessaires à sa compréhension. Il est important que chaque membre de l'équipe puisse avoir une vision limpide de ce qui est demandé dans ce sprint backlog.

Une fois fait, une planification détaillée du sprint est effectuée. Pour ce faire, le sprint backlog sera découpé en tâche (ou unité de travail) représentant entre 2 et 4h de travail (idéalement).

Une fois le sprint accepté par les membres de l'équipe, il ne peut plus changer. Chaque membre fournit son engagement que ce qui a été définit sera effectué durant le sprint.

1.1.1 Durée du sprint

Une courte durée du sprint permet d'avoir des feedbacks plus souvent. C'est un avantage pour avancer plus vite dans un projet. Un sprint peut durer entre 2 semaines et 1 mois.

1.1.2 Le Planning Poker

Technique permettant d'estimer un travail. Chaque membre de l'équipe estime individuellement la charge d'un travail. Cette estimation est mise en commun dans le but de discuter des écarts et de trouver un consensus. La plupart des équipes utilisent des points de complexité pour définir une fonctionnalité.

Pour estimer un élément du sprint, des cartes peuvent être utilisées. Elles représentent 0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?.

Par la suite, l'équipe discute de chaque élément du sprint backlog afin de les découper en unité de travail.

`Planning Poker Feuille`

1.1.3 Tableau de bord Scrum

Comme son nom l'indique, c'est un tableau comprenant des colonnes afin de gérer les différentes unités de travail. Les colonnes sont les suivantes :

— A faire
— En cours
— Terminé

Pour définir l'ordre de priorité des unités de travail, c'est soit le Planning Poker qui le spécifie, soit des dépendances existant entre les unités de travail. Plus une unité est haute dans le travail, plus elle est importante. L'unité est déplacée dans la colonne suivante dès traitement. Chaque membre de l'équipe peut voir rapidement où le sprint en est. Et chaque membre peut choisir sa prochaine unité de travail.

1.1.4 La vélocité

Elle désigne la quantité de travail qu'une équipe peut effectuer durant un sprint. Elle est mesurée à chaque sprint, ce qui permet de déterminer ce qui peut être réalisé. De base, la vélocité s'estime en tenant compte du temps mis à disposition dans un sprint (nombre de personne * temps de travail).

1.2 Stand-up Meeting (Daily Meeting)

Chaque jour, une séance est effectuée. C'est le Scrum Master qui en est garant. Chaque membre se tient debout et répond à trois questions :

— Qu'as-tu réalisé depuis hier ?
— Que vas-tu réaliser aujourd'hui ?
— Quels problèmes rencontres-tu ?

1.3 Les rôles

1.3.1 PO

Le PO (Product Owner) est la personne qui représente la vision du produit pour le Scrum Master et l'équipe de développement. Il centralise à lui tout seul les besoins des utilisateurs. De ce fait, son rôle consiste :

— A créer la vision du produit. Il utilisera des User Stories pour mettre en évidence les attentes des utilisateurs du produit.
— A gérer le Product Backlog. Il l'alimentera grâce aux User Stories et il devra décider des priorités.
— A gérer le prochain Sprint Backlog. C'est lui qui décidera des tâches afin de gérer l'avancement du projet. Attention quand un sprint est commencé, il n'est plus possible de le modifier.
— A gérer le plan de release. Afin que le projet avance, le PO se doit de maintenir un rythme de livraison.

Le PO s'engage dans l'ensemble du processus Scrum en effectuant chaque étape et en collaborant avec l'équipe. L'idéal est quand le PO est le client final. Sur un projet Scrum, il n'y a qu'un seul PO.

1.3.2 Scrum Master

Le Scrum Master a un rôle de facilitateur (il n'est pas un chef de projet !). Il a pour tâche de vérifier que la méthode (Scrum) est bien appliquée, que l'équipe est en mesure de produire et que les obstacles sont levés. Il a les responsabilités suivantes :

- Que le processus fonctionne et soit respecté de tous.
- Servir le PO et l'équipe.
- De ne pas être un chef de projet, il ne prend aucune décision à la place de l'équipe.

Il met en place et anime les réunions suivantes :

- Préparation du sprint (Planning Poker, ...)
- Stand-Up Meeting
- La revue de sprint
- La rétrospective de sprint

1.3.3 Equipe de développement

L'équipe de développement a pour mission la réalisation des Users Stories contenues dans le Sprint Backlog. Elle a les responsabilités suivantes :

- Atteindre les objectifs du sprint en réalisant jusqu'à leur terme les User Stories
- Participer et vivre le Stand-Up Meeting
- Faire vivre le Sprint Backlog en le tenant à jour (A faire, En cours, Terminé)

En général, une équipe se compose de 5-6 personnes. Il est également important que tous les rôles nécessaires y figurent (architecte, testeur, concepteur, développeur, etc). Cependant, chaque membre de l'équipe, malgré leur rôle, porte le nom de développeur.

Chaque membre de l'équipe est capable de reprendre ces responsabilités et de décider la tâche qu'il va développer.

1.4 User Stories (Histoire d'utilisateurs)

Le Product Owner possède une vision du produit. De ce fait, il doit créer les User Stories pour remplir le Product Backlog.

1.4.1 Comment écrire une User Story

Chaque User Story doit être écrite sur une carte (post-it) et utilise un langage naturel en s'aidant des mots-clés suivants :

En tant que ... je veux/dois/peux ... dans le but de ...

Quelques exemples issus d'une application de gestion des mots de passe :

En tant qu'utilisateur, **je peux** visualiser la liste de tous mes comptes **dans le but de** sélectionner un compte en particulier.

En tant qu'utilisateur, **je peux** modifier un compte **dans le but de** maintenir les informations à jour.

En tant qu'utilisateur, **je peux** me connecter **dans le but d**'accéder à l'application de gestion des mots de passe.

La rédaction des User Stories permet de se demander pourquoi réaliser une action. Elles ont pour but de déclencher une conversation entre les utilisateurs. Une User Story doit être confirmée (principe des tests d'acceptation), c'est-à-dire qu'elle est testable (il est possible de la tester directement sur le produit).

1.4.2 INVEST

INVEST permet de juger la qualité d'une User Story, à savoir :

Indépendant : Une User Story est indépendante des autres.

Négociable : Une User Story est adaptable dans le temps.

Valeur : Une User Story a une valeur aux yeux des utilisateurs.

Estimable : Une User Story doit pouvoir être estimée, donc être explicite d'elle-même.

Succint : Une User Story doit être suffisamment petite (tenir en quelques mots).

Testable : Une User Story doit être testable (les tests d'acceptation sont là pour cela).

1.4.3 Ecrire des tests d'acceptation

Un User Story doit être testable. Pour ce faire, un test d'acceptation est mis en place, il a les caractéristiques suivantes :
- N'est pas un test technique
- Est visible pour l'utilisateur
- N'est pas la solution
- N'est pas interne à la fonction testée (se focaliser sur l'aspect fonctionnel de la User Story)

1.4.4 Définir les critères d'acceptation

Pour écrire un test d'acceptation, il faut définir des critères :
- Pré-condition (Etat initial du système avec exécution)
- Quand (Qui est concerné et ce qui est attendu)
- Alors (Résultat)

De plus, il faut au moins écrire un test qui passe et un qui donne une erreur.

Exemple en utilisant la User Story suivante : En tant qu'utilisateur, je peux me connecter dans le but d'accéder à l'application de gestion des mots de passe.

Test OK : L'utilisateur est déconnecté. Il saisit son identifiant XXX et son mot de passe YYY et accède à l'application de gestion des mots de passe.

Test NOK : L'utilisateur est déconnecté. Il saisit son identifiant AAA et son mot de passe YYY, l'accès à l'application de gestion de mots de passe est refusé. Le message d'erreur : « Accès refusé, veuillez-vous authentifiez » s'affiche à l'écran.

1.5 Product Backlog

Il est composé de l'ensemble des fonctionnalités désirées par les utilisateurs qui sont traduites en User Stories.

1.5.1 Prioriser le Product Backlog

Le but est de déterminer les User Stories qui doivent être réalisées en premier afin d'obtenir rapidement de la valeur métier.

Prioriser signifie définir un degré de priorité d'une User Stories par rapport à une autre. Plus la valeur métier sera élevée, plus elle a des chances d'être développée en premier. Cette valeur est définie selon 4 critères :

- Mesure du risque : une User Story est développée au moment où elle permettera d'éviter de lourdes impacts par la suite.
- Amélioration de la qualité : si elle améliore la qualité du livrable, elle doit être développée.
- Dépendance : de manière générale, les User Stories sont indépendances entre elle. Il arrive cependant que certains doivent être développée avant d'autres.
- Donner confiance aux utilisateurs : une user story peut être développée dans le but de donner une idée aux utilisateurs. Par la suite, ces derniers auront d'autres User Stories.

Pour définir la valeur métier, le planning poker est appliqué.

1.6 Revue de Sprint

La revue de sprint sert a présenter le travail effectué (ou non) par le biais d'une démonstration. Même s'il n'y a pas beaucoup a montrer (par exemple lors du 1er sprint), il est important de la maintenir.

Les participants a cette revue sont le PO, le Scrum Master, l'équipe de développement, les utilisateurs invités, les membres du management invités et tout personne ayant reçu une invitation.

La revue de sprint est **informelle**. Elle ne sert pas à juger du travail accompli ou non. Elle sert de base de discussion et de réactions afin d'améliorer la collaboration. Quand un sprint dure 1 mois, la revue ne doit pas dépasser 4h (à adapter selon la durée du sprint).

Une revue de sprint complète est :

- Enoncer le but du sprint à l'assemblée
- La démonstration est effectuée par l'équipe de développement. L'assemble devrait pouvoir tester pour autant que ca ne déborde pas.
- Le PO valide le contenu en identifiant ce qui est terminé et ce qui ne l'est pas.
- L'équipe de développement partage les difficultées rencontrées et les solutions trouvées.
- Le PO récolte le retour des utilisateurs afin d'alimenter son Product Backlog.
- La vélocité est calculé en tenant compte des points d'effort de chaque User Story considérée comme terminée et validée par le PO.
- Le PO discute du contenu restant dans le PO et revoir son plan de release.

A l'issue de la revue de Sprint, le prochain Sprint Backlog doit être mis en place ou révisé dans le cas d'un plan de release.

De plus, la revue de Sprint devrait être une source de motivation pour l'équipe de développement. Elle sert à mettre en valeur le travail effectué. Les personnes présentes voient l'avancement et peuvent apporter un retour immédiat. L'application est fonctionnelle vu que les User Stories sont terminées à 100%. Dans le cas d'une revue de Sprint où rien n'est terminé, l'équipe de développement sera forcée de se remettre en question pour ne pas revivre cela.

1.7 Rétrospective

La rétrospective de Sprint est animée par le Scrum Master. Elle a pour but d'inspecter l'équipe et mettre en place un plan d'action afin de l'adopter dans le Sprint suivant. Sa durée est de 3h pour un sprint d'un mois (à adapter selon la durée du sprint). Son but est que chaque membre s'exprime et que des solutions soient trouvées. Les personnes présentes sont l'équipe de développement, le PO, le Scrum Master et éventuellement un membre de l'équipe de management.

Afin que cette séance se déroule au mieux, il est important que le climat soit favorable. Choisir un lieu propice à la discussion où chaque membre peut se sentir à l'aise. Le Scrum Master fait un flash back des événements passés durant le Sprint. Il note chaque point positif et négatif sur un post-it (vert pour les éléments positifs, orange pour ce qui a fonctionné mais qui peut être améliorer et rouge pour ce qui n'est pas été concluant et qui doit vraiment être améliorer pour les sprints suivants). Le mieux est de faire exprimer chaque membre de l'équipe pour en ressortir les points.

Tous les post-it ne peuvent pas être traités en une fois, c'est pourquoi, le Scrum Master distribue un nombre de points (3 minimum) à chaque personne. Cette dernière distribue ses points sur les post-it rouges. A la fin, les 5 post-it rouges ayant récolté le plus de points sont traités et inclus dans le prochain sprint.

CHAPITRE 2

Git

2.1 Télécharger la version portable

Git Bash Portable

Attention, annuler le téléchargement qui vient automatique et prendre la version portable (en 32bit ou 64 bit)

2.2 Télécharger Aide Git

— `Feuille d'aide de Git`

2.3 Liste des commandes les plus couramment utilisées

Initialisation d'un nouveau dépôt

```
# Initialisation
git init
```

Cloner un dépôt existant

```
git clone git://github.com/schacon/grit.git
```

Le lien est remplacé par le lien du dépôt à cloner

Voir le statut du dépôt

```
git status
```

Ajouter des fichiers

```
# Ajouter tous les nouveaux fichiers ( en rapport avec votre futur commit)
git add fichier1.ext fichier2.ext repo1
```

Publier des modifications sur le dépôt

```
# Publier un changement en local (avant de publier un changement, faire un git status␣
↪pour
# vérifier que tous les fichiers sont inclus)
git commit

# Permet de changer le message ou ajouter un fichier d'un commit
git commit --amend

# Publier le/les changements locaux sur le dépot distant
git push
```

Mettre à jour le dépôt

```
git pull
```

Visualiser l'historique des validations

```
git log
```

Paramètrage de Git

```
# Liste des options disponibles
git config

# Définir son nom
git config --global user.name "Luke Skywalker"

# Définir une adresse mail
git config --global user.email "skywalker@example.com"

# Vérifier les paramètres
git config --list
```

Etiquettes disponibles

[BUGFIX] [TASK] [FEATURE]

CHAPITRE 3

Sources

3.1 Scrum

Livre

- Scrum en action, Guillaume Bodet, Edition Pearson, ISBN 978-2-7440-2468-9
- Scrum - Une méthode agile pour vos projets, Aurélien VANNIEUWENHUYZE, Edition ENI, ISBN 978-2-7460-7867-3

3.2 Git

Site

Git Documentation