

УДК 004.051

Чубіч А.І.¹, Киричек Г.Г.²

¹ студ. гр. КНТ-526 НУ «Запорізька політехніка»

² канд. техн. наук, доц. НУ «Запорізька політехніка»

ВПРОВАДЖЕННЯ KANBAN-ДОШКИ ПРИ ОРГАНІЗАЦІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Kanban - це метод поліпшення процесів розробки, що використовується гнучкими командами. Основною метою методу є отримання готового якісного продукту вчасно. Kanban починається з візуалізації, щоб процеси були на очах у команди. Для цього використовують дошки: фізичні та електронні. Дошка - це обов'язковий елемент для гнучкої методології. Кожен член команди отримує до неї доступ в будь-який час і бачить, на якому етапі перебуває завдання. Дошка дозволяє зробити робочий процес відкритим і зрозумілим для всіх фахівців, що важливо, коли у команди немає одного формального керівника [1]. Зазвичай команда складається з таких спеціалістів: власник продукту (Product Owner), програмісти і головний розробник, спеціалісти з тестування (QA) і головний QA (QA Lead), скрам-

майстер (Scrum Master) та бізнес-аналітик (Business Analyst) [2].

Після затвердження проектний рішень, прийнятих на етапі проектування, команда приступає до розробки. Весь функціонал розподіляється між програмістами, які розробляють алгоритми, пишуть вихідний код, виконують компіляцію і налагодження коду. У проекті є план процесу робіт. Спочатку ми його аналізуємо і поділяємо дошку на стовпці, які відображають етапи [3].

Виділимо наступні статуси для завдань: To do, In Progress, Code Review, Ready QA Testing, QA Under Test, Ready UAT Testing, UAT Under Test, UAT Done, Expected та Done (рис.1).

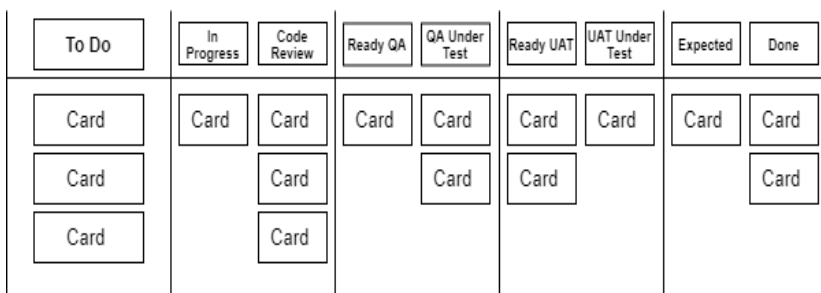


Рисунок 1 – Схема дошки за методологією Kanban

Виділимо 5 основних типів завдань, які будуть створюватись під час роботи над проектом: завдання (task), історія (story), епік (epic), поліпшення (improvement) та баг (bug). На початку циклу розробки створюється епік - це велике завдання, яке повністю описує окремий новий функціонал. Кожен епік складається з історій - описів вимог. Кожна історія є окремою вимогою, яка реалізується в системі. Історії можуть не входити до складу епіків, а будуть окремими самостійними об'єктами, які описують невеликий новий функціонал або зміни поточного. Крім того, створюються нові поліпшення (improvements) - це вимога до оновлення або невеликої зміни частини функціоналу. Покращення можуть додаватися вже після початку розробки, а також на етапі тестування. Тестувальники вносять свої пропозиції, які можуть спростити роботу з системою та зробити її більш комфортною для кінцевого користувача.

Для нових створених задач автоматично призначається статус To do. Цей статус означає, що задача створена, але поки не розпочата робота над нею. Розробник або тестувальник, на якого вона призначена, змінює статус на In Progress, коли приступає до виконання. Програміст, який розробляє або виправляє функціонал, описаний в завданні, вносить зміни на dev-оточенні.

Після цього статус завдання потрібно змінити на Code Review. Після встановлення цього статусу, на задачу автоматично призначається головний розробник, який перевіряє розроблений код. Якщо текст коду задовольняє вимогам, зміни додаються до тестового оточення. Задачі призначається статус Ready QA Testing [3].

Розробник може призначити задачу на тестувальника, який її створював або вже працював з нею. Також, можна залишити непризначеною (Unassigned) задачею і вільний тестувальник зможе почати роботу над нею. Коли починається етап тестування, QA-спеціаліст змінює статус на QA Under Test. Якщо під час перевірки, виникають питання або тестувальник знаходить баги, вони оформлюються в окремих задачах зі статусом To do. Коли тестування задачі закінчено, проблеми вирішені та функціонал відповідає поставленим до нього вимогам, тестувальник переміщає задачу у Ready UAT Testing статус.

Коли починається цикл UAT тестування, задачі переводяться спеціалістами у UAT Under Test статус та призначаються відповідні тестувальники. UAT тестування – це приймальне, призначене для користувача тестування. Будь-яка розробка або доопрацювання програмного забезпечення проходить завершальну стадію UAT-тестування. При цьому тестування проводиться бізнес-користувачами прийнятої системи. Але на практиці часто його проводять QA спеціалісти. Перевірки проводять на спеціальному оточенні, яке містить у собі всі зміни та доповнення, що відбувалися на етапах розробки та QA-циклу. Після успішного завершення перевірки, задачі переводяться до UAT Done.

Також передбачені статуси Expected та Done. As Designed призначається створеному багу, якщо він визнається не багом, а особливою можливістю (feature). Тобто, мається на увазі, поведінка системи, яка явно не відображена у вимогах, але є правильною з точки зору бізнес-логіки. Статус Done використовується для багів, які вирішено не виправляти. Окрім цього, цей статус виставляється для задач типу Task, коли вони виконані тестувальником або розробником. Цикл змін статусів відображено на рисунку 2.

Розробка за системою Kanban дає більше гнучкості, якщо під гнучкістю розуміти частоту зміни пріоритетів. На відміну від Scrum-розробки, де є чітке розподілення на спринти, під час яких потрібно виконати чітко запланований об'єм роботи, для Kanban вимоги до системи можуть змінюватись декілька разів на день, як і пріоритети задач. Окрім цього, не прийнято робити оцінку часу, який необхідний для виконання задачі. Це опціонально, команда вирішує сама, адже кожен з учасників самостійно організовує свою роботу.

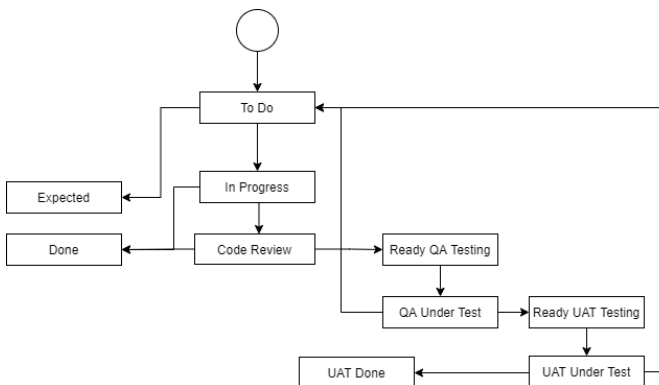


Рисунок 2 – Цикл змін статусів для задач

Висновки. Канбан-дошка та правила розроблено таким чином, щоб допомогти використовувати наявні ресурси найбільш ефективним способом, швидко знаходити проблеми і вирішувати їх в стислі терміни. Центральне місце в Kanban займає візуалізація процесу, яка допомагає відстежити стан задач, які виконуються на даний момент, хто над яким завданням працює і скільки часу потрібно для його завершення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Стелман Е. Осягаючи Agile: Цінності, принципи, методології. / Е. Стелман, Д. Грін. – М.: Манн, Іванов і Фербер, 2019. – 448с.
2. Мартин Р. Быстрая разработка программ. Принципы, примеры, практика. / Р. Мартин, Дж. Ньюкирк, Р. Косс. – М.: Вильямс, 2004. – 752 с.
3. Кон М. Scrum: гибкая разработка ПО. / М. Кон. – М.: Вильямс, 2011. – 576 с.