Міністерство освіти і науки України
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»

М.А. Демиденко

УПРАВЛІННЯ ПРОЕКТАМИ ІНФОРМАТИЗАЦІЇ ЗА МЕТОДОЛОГІЄЮ SCRUM

Навчальний посібник

Дніпро
НГУ
2017

УДК 338.22.021.4
БКК 32.973-018
Д30

Рекомендовано вченою радою як навчальний посібник по дисципліні "Управління проектами інформатизації" для студентів напряму підготовки студентів напряму підготовки 6.051, 8.051 Економіка , 6.050102, 8.050102 "Економічна кібернетика", (Протокол № 20 від 26.12.2017).

Рецензенти:

Мещеряков Л.І..- д-р техн. наук, проф.(ДНВЗ "Національний гірничий університет"; кафедра «Програмного забезпечення комп'ютерних систем»)
Паршина О.А. д-р економ. наук, проф.(Дніпровський університет імені Альфреда Нобеля, завідувач кафедри економіки і моделювання бізнес-процесів)

Демиденко М.А.
Д30 Управління проектами інформатизації за методологією SCRUM : навч. посіб. / М.А. Демиденко ; Нац. гірн. ун-т. — Електрон. текст. дані. – Д. : 2016. – 80 с. – Режим доступу: http://nmu.org.ua (дата звернення:) – Назва з екрана.

ISBN 978-912-350-293-9

Викладено основи управління проектами інформатизації. Розглянуто основні поняття і методологічні основи управління проектами інформатизації за методологією SCRUM .

Матеріал викладено на рівні, доступному студентам, які знайомі з курсом інформатики для економістів. Методи, що розглядаються в посібнику, ілюструються великою кількістю прикладів. Посібник має на меті навчити студентів застосовувати методи SCRUM для обґрунтування оптимальних управлінських рішень при виконанні складних інформаційних проектів.

Посібник базується на досвіді викладання дисципліни «Управління проектами інформатизації» в ДВНЗ «Національний гірничий університет», призначений для студентів вищих учбових закладів і може бути корисним для економістів, плановиків, менеджерів та маркетологів.

УДК 338.22.021.4
БКК 32.973-018

ISBN 978-912-350-293-9

Demydenko Mykhailo
Management of informatization projects in the SCRUM methodology: manual / M.A. Demydenko; National Mining University - D.: 2017. - 80 p.

The basic concepts and methodological bases of the management of informatization projects under the SCRUM methodology are considered.

The material is presented at a level accessible to students who are familiar with of informatics for economists. The methods discussed in the manual are illustrated by many examples. The manual aims to teach students to apply SCRUM methods to substantiate optimal managerial decisions when performing complex information projects.

The manual is based on the experience of teaching the discipline "Management of Informatization Projects" at the National Mining University, intended for students of higher educational institutions and may be useful for economists, planners, managers and marketers.

ORCID: 0000-0001-6298-2079

доц. к.т.н. Демиденко М.А.
каф. Електронної економіки та економічної кібернетики
ДНВЗ “Національний гірничий університет”

ЗМІСТ

Вступ

Методологія Scrum була запропонована японськими і американськими дослідниками. Scrum ефективно використовується для управління інформаційними проектами, тобто управління розробкою інформаційних систем та програмного забезпечення. Особливістю розробки програмного та інформаційного забезпечення є те, що розробка виконується покроково, поступово удосконалюючи новий продукт, при цьому узгоджуючи з замовником функціональність продукту. Тому основна ідея методології Scrum - ітеративний підхід до планування та виконання проекту. На відміну від стандартного підходу (наприклад в будівництві), коли проект спочатку планується «від» і «до», а результат отримуємо «в кінці шляху», даний спосіб (для інформаційних систем) дозволяє в короткими ітераціями у часі з мінімальними витратами отримати готовий продукт. В перебігу проекту виконавець отримує зворотний зв'язок від клієнта, на основі якого здійснюється циклічне нарощування функціональності і вдосконалення продукту. Основна характеристика Scrum - гнучкість. Даний підхід дозволяє оперативно реагувати на зміни у вимогах замовника і швидко адаптувати продукт до них. На сьогоднішній день Scrum - добре опрацьована методологія. Популярність Scrum зростає з кожним днем, у тому числі в Україні.

В навчальному посібнику подано матеріал, який викладається студентам спеціальності “Економічна кібернетика” в дисципліні “Управління проектами інформатизації”.

В посібнику подано чотири теми які розкривають базовий матеріал по дисципліні.

В першій темі подано основні термінологічні формулювання, показано місце методології Scrum в розробці інформаційних систем. Подано стислий опис алгоритму функціонування Scrum. Розглянуто організаційні моменти по-

будови ефективної команди розробки інформаційних систем за методологією Scrum.

В другій темі обговорюється методологія планування в організації Scrum. Розглянуто методи документування та організації ефективного планування роботи команди розробки інформаційного забезпечення .

В третій темі розглянуто питання підвищення ефективності роботи команди Scrum, забезпечення якості продукту, методи роботи з командою, яка географічно розподілена у просторі.

В четвертій темі розглянуто розрахунок економічної ефективності інформаційної системи, яка розробляється за методологією Scrum.

В посібнику додано перелік літературних джерел які необхідно використовувати для поглибленого вивчення дисципліни і самостійної роботи.

Тема 1. Основи Scrum

В темі розглянуто історія розвитку SCRUM, основні принципи, поняття, визначення , організаційна структура розробки інформаційних систем в Scrum.

1.1. Що таке Scrum

Управління інформаційним проектом можна охарактеризувати як діяльність з планування, організації, моніторингу, забезпеченню і управлінню ресурсами та їх опрацюванню, щоб досягнути конкретні цілі і завдання проекту ефективним і дієвим способом.

Scrum («штовханина») - методологія управління проектами, яка застосовується при розробці інформаційних систем, а також для гнучкої розробки програмного забезпечення.

Scrum це методологія яка застосовує різноманітні технологічні прийоми і процеси розробки інформаційних продуктів яка дозволяє підвищити ефективність управління розробкою продукту.

Коли Джефф Сазерленд створив методологію Scrum в 1993 році, то він запозичив термін «штовханина» із дослідження японських фахівців Такеучі і Нонака, яке було опубліковано в Harvard Business Review [1]. У цьому дослідженні, Такеучі і Нонака порівняли роботу в високопродуктивних, багатофункціональних командах до сутичок і штовханин навколо м'яча, які виникають в грі регбі.

Scrum є провідною методологією швидкої розробки інформаційних продуктів і використовується найуспішнішими компаніями світу які входять до переліку Fortune 500 , Toyota, Amazon, Google, Morning Star и Microsoft.

Scrum розроблено для того, щоб змінити спосіб за яким розробляються складні інформаційні проекти, в результаті чого рамки Scrum є гнучкими і його принципи можуть бути застосованими не тільки для інформаційних проектів, але і у більш широкій сфері для проектів в інших галузях промисловості.

Scrum ґрунтується на філософії ї емпіризму і теорії управління емпіричними процесами. Емпіризм стверджує, що знання приходять із досвідом та прийняттям рішень на основі набутих практичних вмінь. Scrum використовує ітеративний, покроковий підхід для поступової розробки інформаційного продукту, якій бере за основу ряд принципів емпіризму. В основі управління емпіричними процесами лежать три основні принципи: прозорість, перевірка та адаптація. Модель емпіричних процесів методології Scrum показано на рис 1.1.

Прозорість: Важливі аспекти процесу розробки інформаційного продукту повинні бути видимими для працівників відповідальних за результат. Ці аспекти мають сприйматися і трактуватися однаково співпрацівниками, що дозволить їм мати єдине бачення інформаційного проекту, що розробляється. До прикладу: всі учасники процесу, розробники, що виконують роботу, і ті, що оцінюють її результат у вигляді продукту, повинні користуватися загальною термінологією і загальними стандартами, що дозволить усім розділяти єдине розуміння змісту розробки.

Перевірка: Користувачі Scrum повинні перевіряти його документацію, а також контролювати процес просування до мети проектування для своєчасного виявлення небажаних відхилень від плану та норм і стандартів. Однак перевірка не повинна бути настільки частою, щоб заважати роботі. Перевірки приносять найбільшу користь, якщо проводяться кваліфікованими інспекторами на робочих місцях.

Адаптація: Якщо за результатами перевірки інспектор робить висновок, що один або більше аспектів процесу відхиляються від допустимих норм, і що продукт, який перебуває у процесі розробки, буде неприйнятним, необхідно

негайно внести зміни або у процес, або у робочі матеріали. Зміни повинні вносится як найраніше для зменшення ризику подальшого відхилення від норми.

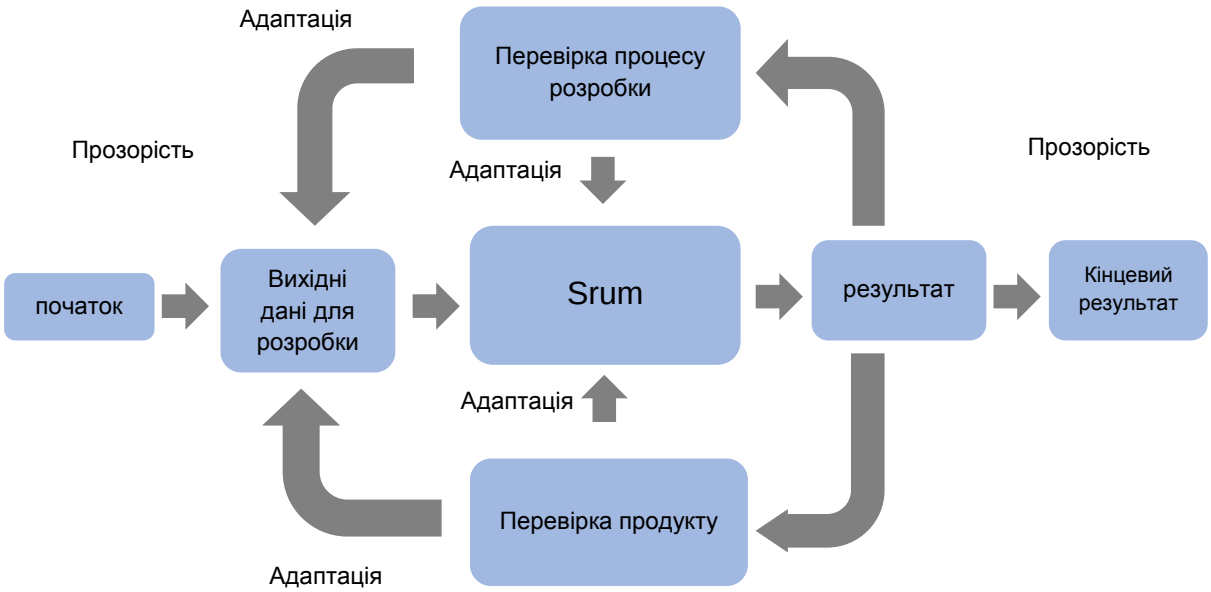


Рисунок 1.1 Модель процесів Scrum

Принципи Scrum

В основу Scrum технологій покладено декілька принципів, які вказують на найбільш важливі аспекти в цій методології:

- Особистості, здібності, таланти розробників та взаємодія між розробниками є важливішими, ніж процеси та інструменти розробки інформаційної системи;
- Працездатна інформаційна системи важливіша, ніж повна документація для цієї системи ;
- Співпраця із замовником важливіше, ніж контрактні зобов'язання;
- Вміння реагувати на зміни вимог та необхідність коригування планів , важливіше ніж проведення робіт за планом.

Практична реалізація цих принципів можлива за умови виконання таких вимог:

- Впроваджувати ранні, ще не достатньо досконалі, але працездатні версії інформаційного продукту для забезпечення вимог замовника;
- Підтримувати запровадження змін, які підвищують конкурентоспроможність інформаційного продукту ;
- Регулярно постачати замовнику вдосконалений інформаційний продукт (кожен місяць або тиждень або ще частіше);
- Підтримувати щоденне спілкування між замовником і розробниками протягом всього проекту;
- Проектом мають займатися мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;
- Метод який рекомендовано для передачі інформації - це особиста розмова (обличчям до обличчя);
- Спонсори, розробники і користувачі повинні мати можливість підтримувати постійний темп розробки інформаційного проекту на невизначений термін;
- Забезпечувати постійне поліпшення технічної досконалості та зручності дизайну інформаційного продукту;
- Простота - мистецтво НЕ робити зайвої роботи;
- Самоорганізовані команди розробляють кращу архітектуру, функціональність і дизайн інформаційної системи;
- Забезпечувати поліпшення ефективності роботи команди розробників з урахуванням мінливих обставин і вимог проекту.

1.2. Історія створення Scrum

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака. Вони відзначили, що проекти, над якими працюють невеликі, багатофункціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як

«підхід регбі». Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'95. Швабер та Сазерленд протягом наступних років працювали разом, щоб обробити та описати весь їхній досвід та найкращі практичні зразки впроваджені в інформаційній індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM [2]. Згодом Scrum набув широкої популярності і з його використанням розробляються інформаційні проекти в ведучих компаніях Microsoft, Google, SAP, Oracal.

1.3. Стислий опис алгоритму функціонування Scrum в розробці інформаційних проектів.

Алгоритм Scum можна представити наступними шістьма кроками.

1. **Розділіть колектив розробників проекту** на маленькі, багатофункціональні, команди які можуть самоорганізуватися .

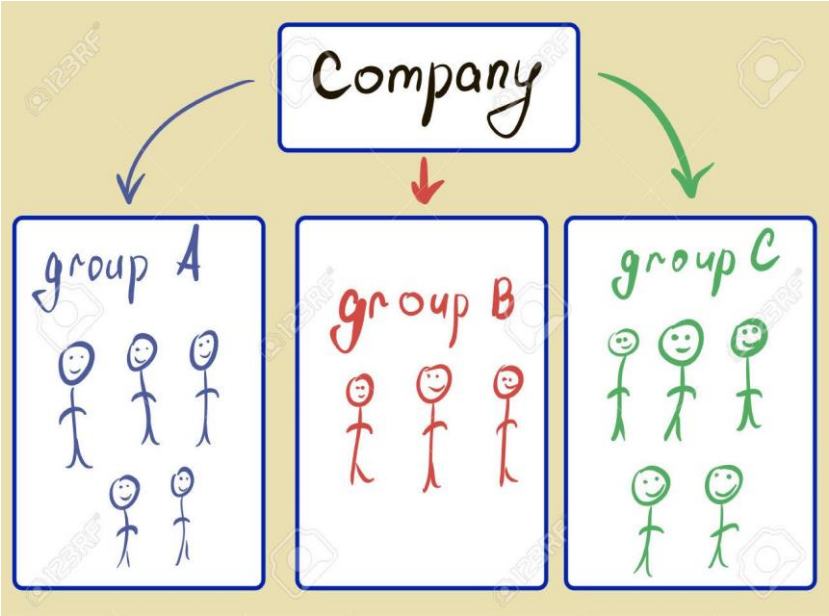


Рисунок 1.2. Розподіл колективу розробників проекту на групи

2. Розділіть роботу по реалізації інформаційного проекту на маленькі, конкретні компоненти - задачі. Відсортуйте цей список за пріоритетами (важливістю) та оцініть відносний обсяг роботи по кожному елементу.

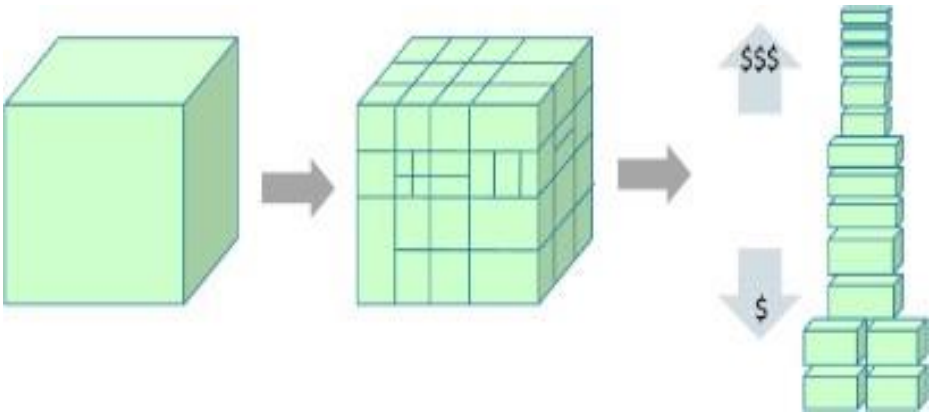


Рисунок 1.3. Розподіл завдань по важливості

3. Розділіть час відведений на розробку інформаційного проекту на короткі проміжки часу фіксованої довжини – ітерації- спринти (зазвичай 1-4 тижні) .

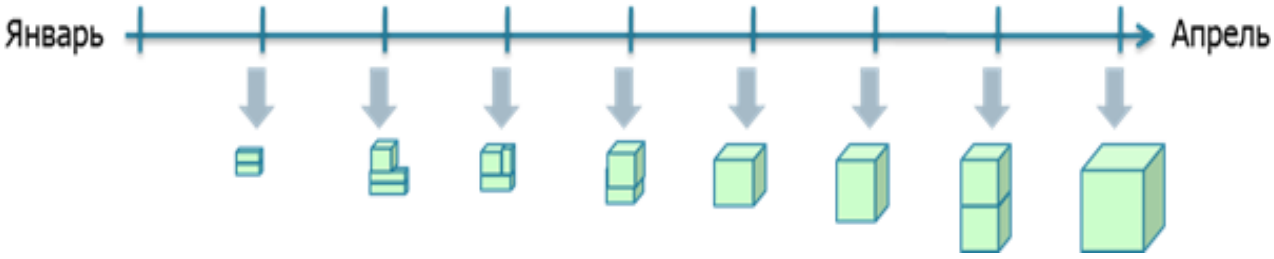


Рисунок 1.4. Розподіл часу на виконання інформаційного проекту

4. **Визначте обсяг задач для поточної ітерації-спринта , виконайте їх розробку.** Після закінчення кожної ітерації має проводитися демонстрація потенційно готового до використання продукту.

5. Проаналізуйте і оптимізуйте план доробки задач і корегуйте пріоритети наступних задач спільно з клієнтом, ґрунтуючись на даних, які отримані при демонстрації після кожної ітерації. *Оптимізуйте процес розробки інформаційно-*

го продукту за допомогою проведення ретроспективного аналізу після кожної ітерації.

6. Перейти до пункту 4.

Таким чином, **замість** великої команди **яка** , **працює** довго над **чимось** великим , у нас невелика команда **яка** , **короткими ітераціями** працює над невеликими **шматочками** розробляючи великий проект. Стислий алгоритм функціонування подано у фільмі.

1.4. Учасники Scrum

Учасник Scrum є Команда розробників, Власник продукту, який представляє замовника інформаційної системи і Scrum-майстер який виконує роботу керівника і консультанта.

Рисунок. 1.5. Учасники Scrum

1.4.1. Склад Команди

Команда (sprint goal). Команда складається з 3-9 людей що виконують роботу (аналізують, виконують дизайн, пишуть код, тестують, готують документацію і таке інше). У Scrum, команда є самокерованою.

Scrum Команда складається із Власника Продукту , команди розробників (Development Team) і Scrum-майстра (Scrum Master). Scrum команди є самоорганізованими і багатофункціональними. Самоорганізовані Команди самі обирають, як їм найкращим чином виконати роботу, та не чекають вказівок від людей, що не входять до їх складу. Багатофункціональним Командам властиві всі необхідні навички, щоб виконати роботу і вони не залежати від людей, що не входять до їх складу. Командна модель Scrum створена для оптимізації гнучкості, креативності та продуктивності.

Scrum команди створюють продукт покроково та ітераційно, що збільшує можливості для зворотного зв'язку.

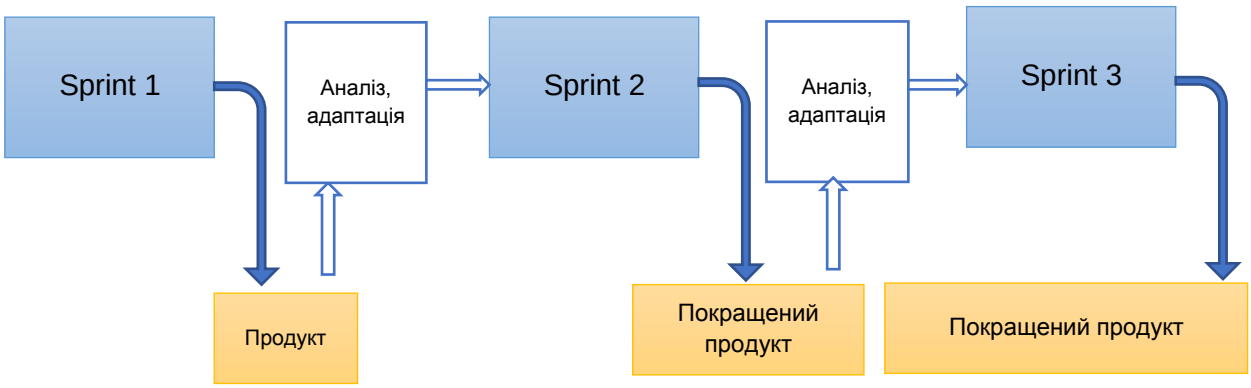


Рисунок. 1.5. Покроковий, ітераційний процес розробки в Scrum

1.4.2. Власник продукту

Власник Продукту представляє зацікавлені сторони та є голосом клієнта. Власник Продукту є відповідальним за досягнення максимальної відповідності вимогам замовника, продукту та роботи, що виконується командою розробників. Способи, завдяки яким це досягається, можуть відрізнятися в залежності від організацій, Scrum команд та окремих осіб.

Обов'язки Власник Продукту такі:

- Управляє очікуваннями замовників і всіх зацікавлених осіб;
- Координує і визначає пріоритети в Журналі продукту;
- Надає зрозумілі вимоги команді;
- Взаємодіє з командою і замовником;
- Відповідає за приймання готового продукту в кінці кожної ітерації;
- Власник продукту ставить завдання команді, але він не має права ставити завдання для конкретного члена проектної команди протягом спринту.
- Знає 5 рівнів планування в методології Agile. Згідно з нею, при виконанні проекту планування здійснюється безперервно. Рівень 1: *Стратегічний план.* Для інформаційного продукту потрібен стратегічний план розробки і впровадження продукту. Рівень 2: *Дорожня карта* це короткостроковий або довгостроковий план випуску виробником будь-якого продукту. Найчастіше це нова версія або розвиток вже відомого продукту. На основі дорожньої карти, ринкових умов і стану продукту Власник продукту може планувати версії інформаційного продукту. Рівень 3: *Планування спринту* разом із командою розробників. Рівень 4: *Планування* роботи команди і *узгодження* журналу продукту і організація допомоги команді в досягнені мети спринту. Рі-

вень 5: Проведення щоденної наради для перевірки і узгодження просування робіт для досягнення мети спринту.

Власник Продукту є єдиною особою в команді, яка відповідає за журнал вимог до продукту (Журнал продукту, Product Backlog). В журналі продукту міститься перелік робіт які треба виконати під час спринту. В журналі вимог роботи впорядковані за їх вважливістю (пріоритетом).

Журнал продукту включає в себе наступні пункти:

- Чітке визначення елементів журналу продукту;
- Впорядкування елементів журналу продукту для максимального досягнення цілей та поставлених завдань;
- Відповідальність за цінність роботи, що виконується командою розробників;
- Забезпечення доступності, прозорості та зрозумілості журналу продукту, а також відображення тих елементів, над якими Scrum команді доведеться працювати найближчим часом;
- Відповідальність за розуміння командою розробників вимог журналу продукту на належному рівні.

Власник продукту може виконувати перераховані вище функції сам, або ж довірити їх виконання членам команди розробників, однак підзвітним вважається саме Власник продукту.

Власником продукту завжди призначається одна особа, а не група. Власник продукту може представляти інтереси групи в журналі продукту, проте члени команди, які бажають змінити пріоритетність вимог у журналі продукту, повинні в першу чергу переконати в цьому Власника продукту.

Для успішного виконання Власником продукту своїх обов'язків всі члени організації повинні поважати його рішення. Всі рішення Власника продукту є

видимими через вміст і впорядкування Журналу продукту. Члени команди мають виконувати вимоги власника продукту і ніхто інший не може змінювати ці вимоги..

1.4.3. Керівник Scrum (Scrum-майстер)

Scrum-майстер - це керівник, який є відповідальним за спроможність команди виконати поставлені цілі і подолати складнощі які виникають у процесі розробки інформаційного проекту.

Scrum-майстер несе відповідальність за те, щоб Scrum був гарантовано зрозумілим всім учасникам та функціонував ефективно. Scrum-майстер стежить за тим, щоб всі учасники команди дотримувалися теоретичних засад, практик та правил Scrum. Scrum-майстер є слугою-лідером для команді розробників

Scrum-майстер також допомагає особам, що не входять до складу Scrum команди зрозуміти, які з їх взаємодій зі Scrum командою є корисними, а які ні. Scrum-майстер допомагає внести зміни в такі взаємодії для збільшення цінності продукту, що розробляється Scrum командою.

Scrum-майстер допомагає Власнику продукту у наступному:

- Виявляє методи ефективного управління Журналом Продукту;
- Інформує членів команди розробників про бачення, цілі та елементи Журналу продукту;
- Вчить команду розробників створювати лаконічні та зрозумілі елементи Журналу продукту;
- Здійснює довгострокове планування розробки продукту;
- Розуміє та практикує гнучкі методи розробки та управління;
- На вимогу чи за необхідності виступає ведучим нарад Scrum.

Scrum-майстер допомагає команді розробників у наступному:

- Вчить команду розробників самоорганізації та багатофункціональності;
- Вчить та веде за собою команду розробників при створенні продуктів із високою цінністю;
- Усуває перешкоди, що виникають у процесі роботи команди розробників;
- За необхідності проводить наради Scrum;
- Проводить необхідні тренінги для Scrum команди в тих організаційних областях, в яких Scrum є ще не до кінця впровадженим та зрозумілим.

Scrum-майстер допомагає Організації- розробнику інформаційної системи, у наступному:

- Планує етапи впровадження Scrum в межах організації;
- Допомагає співробітникам компанії та зацікавленим особам зрозуміти та впровадити Scrum та принципи емпіричної розробки продукту;
- Виступає ініціатором змін, що посилюють продуктивність Scrum команди;
- Співпрацює з іншими Scrum-майстрами для оптимізації використання Scrum в межах організації.

1.4.4. Команда розробників

Команда розробників потенційно готових частин продукту в кінці кожного спринту. Команда складається з 3-9 людей, що виконують роботу (аналізують, виконують дизайн, розробляють інформаційний продукт, тестують, готують документацію і таке інше). Scrum, команда є самокерованою.

Команди розробників є структурованими та уповноваженими організацією розробником інформаційної системи, самостійно організовувати та керувати процесом своєї роботи.

Команди розробників мають відповідати наступним вимогам :

- Команди самоорганізовані і самі вирішують як правильно втілювати завдання Журнал продукту в нову версію інформаційної системи, що розробляється;
- Команди розробників є багатофункціональними, тобто володіють усіма навичками, необхідними для розробки версії продукту;
- Scrum не існує ніяких інших посад у команді розробників, окрім Розробника, незалежно від виду роботи, що виконується. У цього правила немає винятків;
- Окремі члени команди розробників можуть володіти спеціалізованими знаннями у різних областях, однак відповідальність лежить на усій команді розробників;
- У команди розробників немає структурних підрозділів, які б виконували окремі функції, наприклад, підрозділ тестування або бізнес-аналізу.

Розмір команди Оптимальний розмір команди розробників повинен бути невеликим, для того щоб команда залишалася простою в управлінні, і в той же час достатньо великим, щоб вона могла виконати значний обсяг роботи.

Якщо в команді розробників менше трьох осіб, то взаємодія між ними зменшується, і в результаті продуктивність команди знижується. На певному етапі спринту у невеликої команди може виявитися дефіцит необхідних знань, що унеможливить завершення робот над версією продукту.

Якщо ж у команді більше дев'яти осіб, то керування стає складнішим і вимагає більше зусиль і часу для координації роботи членів команди, ніж це

передбачає ефективне управління виробничою діяльністю. Великі команди розробників викликають зайві складності в управлінні процесом.

Власник продукту і Scrum-майстер не враховуються при підрахунку кількості членів команди розробників.

Послідовність розробки інформаційного продукту і взаємодія власника продукту, Scrum-майстра та команди розробників показана на рис. 1.6

Рисунок 1.6. Розробка інформаційного продукту, взаємодія учасників Scrum

1.5. Висновки

- Scrum (Scrum) – це підхід для розробки та підтримки функціонально складних продуктів
- Scrum було розроблено в 90-х -2000-х роках. з того часу він набув широкої популярності і з його використанням розробляються інформаційні проекти в ведучих компаніях світу.

- Цінності Scrum містять ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти та створювати команди, в яких приємно і цікаво працювати. Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити.
- Цей посібник містить визначення Scrum. Визначення полягає в описі учасників Scrum, їх ролей в розробці інформаційного продукту, нарад та документації Scrum, а також правил, що забезпечують зв'язок між ними.

1.6. Контрольні запитання

1. Сформулюйте визначення технології проектування інформаційних систем SCRUM.
2. Сформулюйте алгоритми функціонування технології SCRUM.
3. Яким є склад команди SCRUM?
4. Поясніть місце і роль власника продукту в розробці проекту.
5. Роз'яснить обов'язки Scrum- майстра.
6. Яким є оптимальний обсяг команди SCRUM?
7. Що таке є цінності SCRUM і в чому вони полягають?
8. Поясніть історію створення SCRUM.
9. Чому виникла необхідність створення SCRUM?

Тема 2. Методологія планування та організації Scrum

В темі викладено методи ефективного планування робіт і співпраці колективу розробників в SCRUM.

Планування і організація з Scrum виконується покроково, ітераційно. Схема організації Scrum показана на рис. 2.1 . Великий проект розбито на невеличкі завдання які розробляються протягом декількох ітерацій, які називаються спринтами. Весь проект складається з послідовності спринтів. Всі спринти мають однакову тривалість, приблизно від одного до чотирьох тижнів. Тривалість спринту обирає команда розробників . Протягом кожного спринту працівники додають нові функціональності в інформаційний продукт, виконують відповідні завдання і просувають інформаційний проект до завершення.

Завдання які виконуються протягом кожного спринту сортуються відповідно до важливості, яку визначає Власник продукту.

Відсортовані по важливості завдання, додаються до Журналу продукту – Product backlog (журнал запитів на виконання робіт).

Команда розробників організує нараду (до 4годин), на якій визначаються затрати людино-годин на виконання і здійснюється планування робіт, яке має назву планування спринту. Протягом цієї наради Власник продукту інформує про завдання, які він хоче, аби були виконані. Тоді команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог протягом спринту.

В процесі виконання завдань Спринту, щоденно проводяться наради, які тривають не більше 15 хвилин. Під час цих нарад визначаються завдання на поточний день.

Наприкінці спринту члени команди проводять ретроспективну нараду. Під час наради підбивають підсумки завершених робіт і визначають як можна покращити спринт у майбутньому. Визначають, що заважало виконанню робіт і як покращити швидкість виконання робіт в спринті.

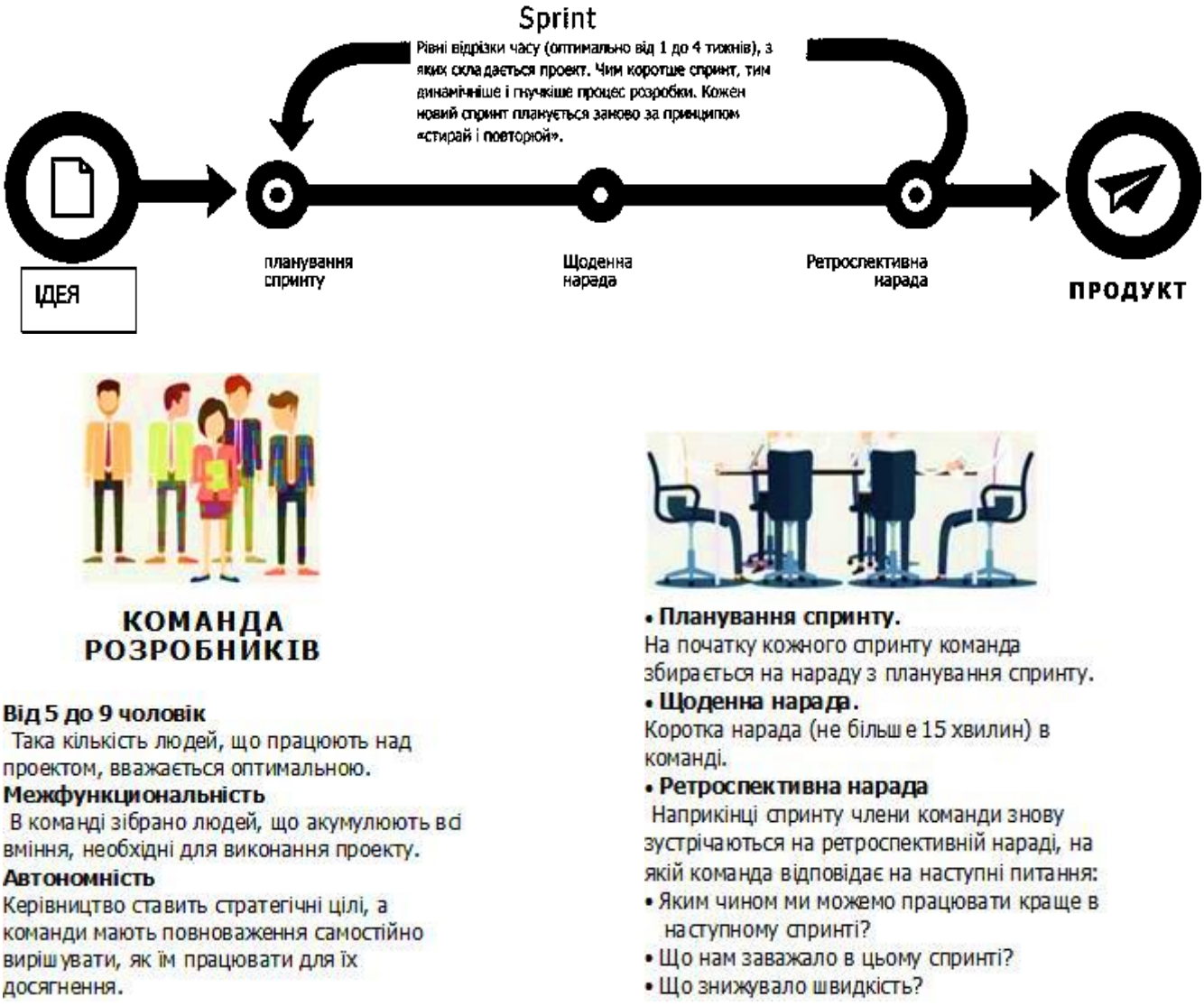


Рисунок 2.1. Схема організації Scrum

2.1. Планування спринтів

Мета планування спринтів – забезпечити своєчасність виконання командою робіт по розробці інформаційного продукту.

В результаті планування має бути підготовлено:

- Ціль спринту.
- Список учасників команди (і ступінь їх зайнятості, якщо вона не стовідсоткова).
- Журнал продукту - Sprint backlog (із завданнями, які увійшли в спринт).
- Дата демонстрації.
- Місце і час проведення щоденного Scrum'а.

Обсяг робіт та пріоритети завдань визначаються Власником продукту. Оцінка трудовитрат - це прерогатива команди. Завдяки взаємодії команди і Власника продукту в ході планування спринту виробляється оптимальна координація робіт.

Найчастіше Власник продукту починає планування наступного спринту з опису основних цілей і найбільш значущих завдань-історій. Після цього команда проводить оцінку трудовитрат всіх завдань-історій, починаючи з найважливіших. У процесі планування у команди виникають питання і пропозиції, що до обсягу наступних робіт. У деяких випадках час, який знадобиться на виконання завдань-історій, не співпадатиме з очікуваннями Власника продукту, тому, він може переглянути пріоритет для завдань-історій або змінити обсяги робіт. У свою чергу, команда виконає переоцінку трудовитрат на виконання робіт. Таким чином планування виконується поступово в декілька ітерацій, за допомогою яких узгоджуються всі деталі процесу виконання робіт у наступному спринті.

2.2. Журнал Продукту- Product backlog

Журнал продукту (Product backlog) - це основа Scrum'а. Журнал продукту є списком вимог, завдань - історій, функціональностей, які впорядковані за ступенем важливості. При цьому всі вимоги описані таким чином, щоб замовник інформаційного продукту правильно їх зрозумів (табл. 2.1).

Елементи "історіями" (user story), а іноді елементами Журналу продукту. Опис кожної нашої історії включає в себе наступні поля:

- **ID** - унікальний ідентифікатор - просто порядковий номер. Застосовується для ідентифікації історій у разі їх перейменування.
- **Назва** - короткий опис історії. Наприклад, "Перегляд журналу своїх транзакцій". Воно повинно бути однозначним, щоб розробники і Власник продукту (власник продукту) могли приблизно зрозуміти, про що йде мова, і відрізнити одну історію від іншої. Зазвичай від 2 до 10 слів.
- **Важливість** - ступінь важливості даного завдання, на думку Власника продукту. Число , яке вказує на важливість. Наприклад, 10, або 150. Чим більше значення, тим вище важливість.
- **Попередня оцінка** - початкова оцінка обсягу робіт, необхідного для реалізації історії в порівнянні з іншими історіями. Вимірюється в story point'ах. Приблизно відповідає числу "ідеальних людино-днів". Запитайте вашу команду: "Якщо зібрати команду з оптимальної кількості людей, тобто не надто велику і не дуже маленьку (найчастіше з двох осіб), закритися в кімнаті з достатнім запасом їжі і працювати ні на що не відволікаючись, то, скільки днів тоді знадобиться на розробку завершеного, протестованого продукту, готового до демонстрації і релізу? ". Якщо відповідь буде "Для трьох осіб, закритих в кімнаті, на це буде потрібно 4 дні", це означає, що початкова оцінка становить 12 story point'ов.

- **Як продемонструвати** - коротке пояснення того, як завершена задача буде продемонстрована в кінці спринту. По суті, це простий тестовий сценарій типу "Зробіть це, зробіть те - повинно вийти те-то".
- **Примітки** - будь-яка інша інформація: пояснення, посилання на додаткові джерела інформації, і т.д. Зазвичай вона представлена у формі коротких тез.

Таблиця 2.1

Журнал продукту

ID	Назва	Важливість	Попередня оцінка трудомісткості	Як продемонструвати	Примітки
1	Перегляд журналу особистих транзакцій	10	8	Увійти в систему; перейти на сторінку транзакцій; покласти гроші на рахунок; повернутися на сторінку транзакцій; перевірити, що нова транзакція з'явилася в списку.	Щоб уникнути великих запитів до бази даних, варто скористатися посторінковим виведенням інформації. Дизайн такий же, як і у сторінки перегляду користувачів.
2	…	….	….	….	….

Зазвичай Журнал Продукту зберігається в Excel таблиці з можливістю спільного доступу (кілька користувачів можуть редагувати файл одночасно). Офіційно документ належить Власнику продукту , але інші користувачі також можуть його редагувати, тому що розробникам досить часто доводиться заглядати в Журнал продукту, щоб щось уточнити або змінити оцінку трудовитрат.

2.2.1. Додаткові поля для Журналу продукту

Додаткові поля в Журналі продукту використовуються для того, щоб допомогти Власнику продукту визначитися з його пріоритетами.

- **Категорія** – за допомогою цього поля Власник продукту може легко вибрати всі пункти категорії і встановити їм відповідний пріоритет.
- **Компоненти** – це поле вказує, які компоненти (наприклад, база даних, сервер, клієнт) будуть задіяні при реалізації історії. Поле "компоненти" виявиться корисним, якщо над проектом працюють кілька Scrum команд, наприклад, одна, яка працює над панеллю керування і інша, яка відповідає за клієнтську частину. В даному випадку це поле істотно спростить

для кожної з команд процедуру вибору історії, за яку вона могла б взятися.

- **Ініціатор запиту -** Власник продукт зберігає інформацію про всіх замовників, зацікавлених в даній задачі. Це потрібно для того, щоб тримати їх у курсі справи про хід виконання робіт.
- **Ідентифікатор в системі обліку дефектів** - в описі історії необхідно зберігати посилання на всі дефекти, які було виявлено.

2.3. Організація планування Спринту

Спринт, планується під час наради із планування спринту. План дій розробляється при спільній роботі всієї Scrum команди.

Для спринту тривалістю в місяць часові рамки зустрічі становлять вісім годин. Для більш коротких спринтів на планування виділяють менше часу, пропорційно загальній тривалості спринту. Приміром, для двотижневого спринту планування займе не більше чотирьох годин.

Приклад розкладу з планування спринту:

Час на планування спринту: з 09:00 до 13:00 (після кожної години перерва на 10 хвилин) ;

9:00 - 10:30. Власник продукту роз'яснює мету спринту і розповідає про бізнес-процеси з журналу продукту. Обговорюється час і місце проведення демонстрації результатів спринту.;

10:30 - 11:00. Команда проводить оцінку часу, який буде потрібно на розробку бізнес-процесів і, при необхідності дробить їх на більш дрібні. У деяких випадках Власник продукту може змінити пріоритет їх виконання. З'ясовуємо всі питання по виконанню завдань;

11:00 - 12:00. Команда визначає список історій, які увійдуть в наступний спринт. Щоб перевірити наскільки це реально, обчислюємо продуктивність команди розробників;

12:00 - 13:00. Домовляємося про час і місце проведення щоденного Scruma (якщо вони змінилися в порівнянні з минулим спринтом). Після чого приступаємо до розбиття списку історій на завдання.

Нарада планування спринту складається із двох частин, тривалість кожної з яких становить половину загальної тривалості наради. Під час двох частин планування спринту члени команди відповідають на наступні питання: відповідають на наступні питання:

- Що буде розроблено, що стане результатом роботи наступного спринту;
- Як поліпшити роботу спринту.

Частина перша: Що планується зробити в цьому спринті? У цій частині команда розробників намагається спланувати розробку елемента інформаційної системи. Власник продукту надає команді розробників впорядковані в журналі продукту вимоги, і вся Scrum команда намагається досягти єдиного розуміння щодо роботи, яку потрібно виконати протягом спринту.

Результатом цієї зустрічі є журнал продукту, в якому зазначено останній розроблений продукт, можливості команди розробників та останній показник її продуктивності. Кількість елементів із журналу продукту, які команда здатна виконати до закінчення спринту, визначається самою командою. Тільки команда розробників може реально оцінити обсяг роботи, який вона в змозі завершити до закінчення спринту.

Після того, як команда розробників спрогнозує елементи журналу продукту, які вона виконає в поточному спринті, Scrum команда приступає до формування цілей спринту.

Ціль спринту– це мета, якої буде досягнуто в результаті спринту завдяки реалізації журналу продукту, і яка вказує команді розробників чому вона працює саме над цим покращенням функціональності продукту.

Частина друга: Як буде виконано обрану роботу? Після того, як обсяг роботи спринту визначено, команда розробників вирішує яким чином протягом спринту втілити окрему функціональність у “завершений” продукт.

Елементи журналу продукту, обрані для виконання під час найближчого спринту, разом із планом їх розробки називають журналом завдань спринту (sprint backlog).

Як правило, команда розробників починає планувати систему і роботу, завдяки якій журнал продукту можна перетворити на працюючий план розробки продукту (елемента інформаційної системи). Під час планування спринту команда розробників планує такий обсяг роботи, який вона в змозі виконати за спринт. До закінчення цієї зустрічі робота, запланована командою розробників на перші дні спринту, розбивається на вимоги, які можна виконати за день або ж менше. команда розробників сама організовує свою роботу, плануючи поетапність виконання вимог із журналу спринту як під час наради із плануванню спринту, так і, за необхідності, протягом усього спринту.

Власник продукту може бути присутнім на другій частині планування спринту, щоби мати можливість пояснити завдання із журналу продукту і, за необхідності, допомогти знайти альтернативи. Якщо ж команда розробників вирішує, що у неї надто багато, чи надто мало роботи, вона може повторно обговорити з Власником продукту вимоги з журналу спринту. команда може запросити людей зі сторони, щоб вони порадили щось із технічної або ж експертної точки зору.

До закінчення наради із планування спринту команда розробників повинна пояснити Власнику продукту і Scrum майстру яким чином вона працюва-

тиме в якості самоорганізованої команди, з метою досягнення цілей спринту і створення очікуваного продукту.

Ціль спринту надає певної гнучкості роботі команди розробників щодо розробки функціональності під час спринту. Ціль спринту може бути важливим етапом для наступних цілей в розробці кінцевого продукту.

Вибір історій, які увійдуть до спринту.

Основне в плануванні спринту - процедура вибору завдань-історій, які увійдуть до Спринту. Точніше, вибір історій, які потрібно скопіювати з журналу продукту в журналу спринту.

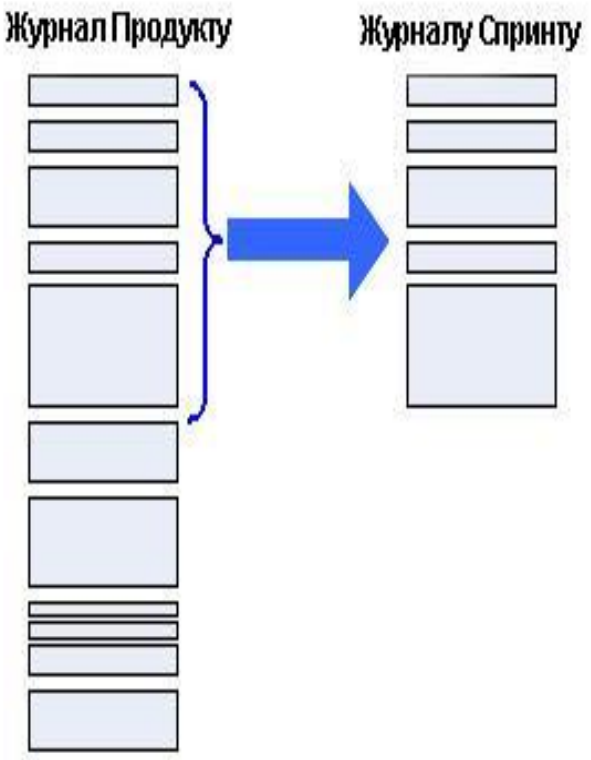


Рисунок 2.2 Вибір історій в журнал спринту з журналу продукту

Кожен прямокутник представляє завдання-історію, розташовування якої в списку відповідає рівню її важливості. Найбільш важлива історія знаходиться нагорі списку. Розмір історії (тобто трудоємність у story point'ax) визначає розмір кожного прямокутника. Висота блакитний дужки позначає прогнозовану

продуктивність команди, тобто кількість історій, які команда збирається завершити в наступному спринті.

Саме команда вирішує, скільки історій увійде в спринт. Ні Власник продукту, ні хто-небудь ще не визначає кількість історій.

У зв'язку з цим, виникають два питання:

1. Яким чином команда вирішує, які історії потраплять в спринт?
2. Як Власник продукту може вплинути на їхнє рішення?

Перед початком роботи команда має перелік робіт які треба виконати. На рисунку 2.2 показано такий перелік. Команда розробників має оцінити свою продуктивність по виконанню робіт на час спринту і вибрати роботи, які можна виконати за цей час В першу чергу до виконання вибираються роботи, які мають більшу важливість, вищий пріоритет.

2.3.1. Оцінка витрат часу за допомогою розрахунків story point і фокус-фактору

Розглянемо методику яка базується на розрахунку продуктивності команди

Продуктивність є мірою "кількості виконаної роботи". Вона розраховується як сума попередніх оцінок трудовитрат (кількість story point) на виконання всіх історій, які входять до плану спринту.

Згідно Рис 2.3 попередня оцінка продуктивності команди розробки буде становити 18 Story Points. Чи отримали ми прогнозовану продуктивність? Ні! Тому що наша одиниця виміру - це story points, яка, в нашому випадку приблизно дорівнює "ідеальному людино-дню".

Ідеальний людино-день - це максимально продуктивний день, коли ніхто і ніщо не відволікає від основного заняття. Такі дні - рідкість. Крім того, потрібно брати до уваги, що в ході спринту може бути додана незапланована робота, людина може захворіти і т.д.

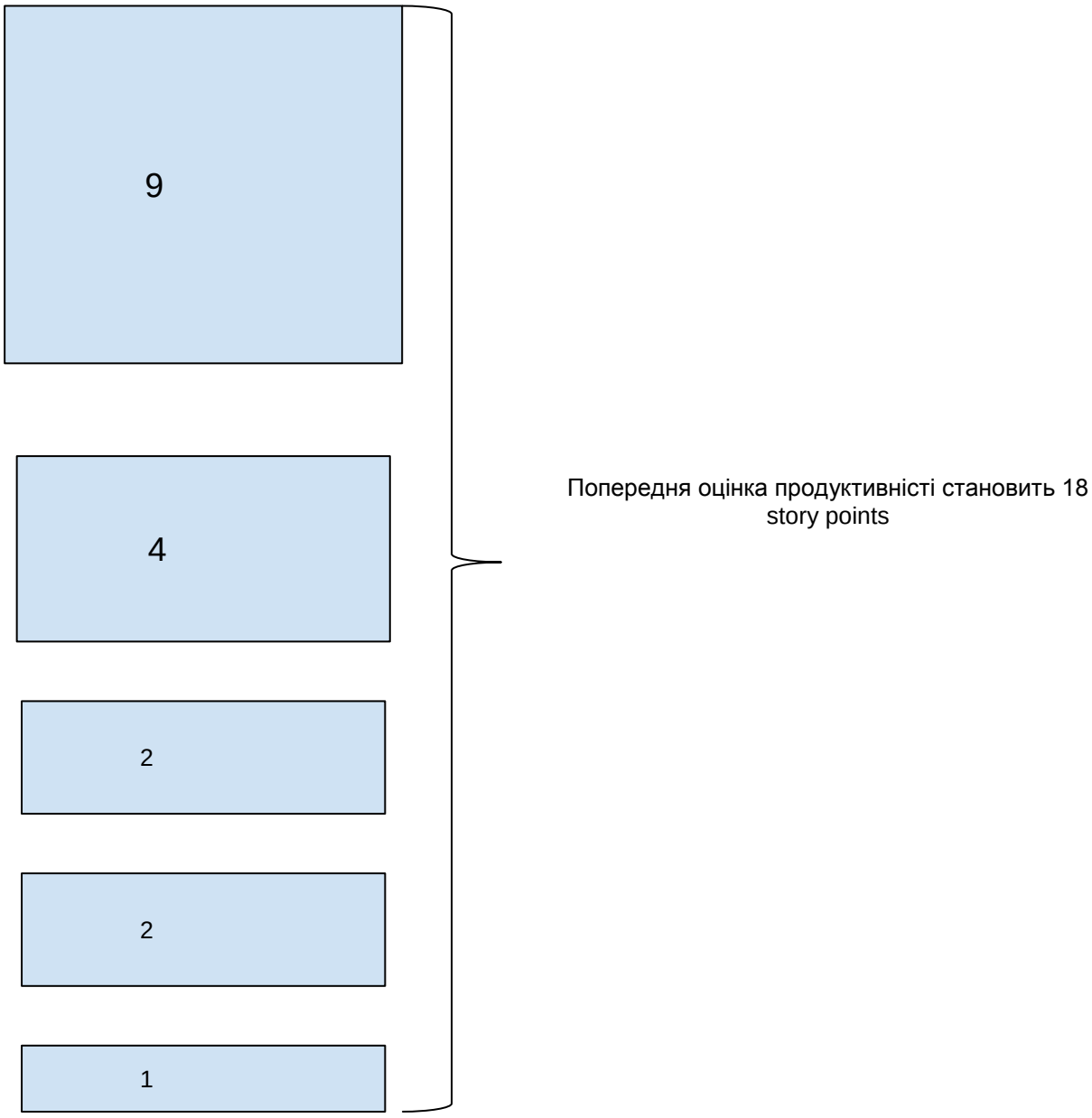


Рисунок 2.3. Попередня оцінка продуктивності

Тому реальна продуктивність буде значно відрізнятися від попередньої оцінки Для того щоб більш точно розрахувати продуктивність додаємо поняття фокус-фактору.

Фокус-фактор - це коефіцієнт того, наскільки команда сфокусована на своїх основних завданнях. Низький фокус-фактор може означати, що команда очікує неодноразового втручання в свою роботу або припускає, що оцінки продуктивності надто оптимістичні.

Розрахунок Фокус-фактору виконується за формулою

$$\text{Фокус-фактор} = \frac{\text{Реальна продуктивність}}{\text{Наявні людино-дні в спринті}}$$

За реальну продуктивність приймається сума початкових оцінок для тих історій, які були завершені в ході останнього спринту.

Припустимо, під час останнього спринту командою з трьох чоловік реалізовано 15 Story Point'ов. Тривалість спринту була 2,5 тижні, що становить 37,5 людино-днів. Необхідно спрогнозувати продуктивність команди на майбутній спринт. Беручи до уваги, що члени команди брали відгули тоді отримаємо 44 людино-днів.

Розрахуємо фокус Фактор

Фокус-фактор = 15/37,5 = 0,4.

Тоді розрахунок story points для 44 людино - днів має вигляд

Story Points = 44*0,4=17,6

Таким чином для виконання задач поточного спринту буде необхідно 17,6 Story Points, це на 2,6 Story Points більше, ніж для розрахунку без урахування фокус фактору.

Після проведення розрахунків ми визначили, що під час наступного спринту продуктивність команди розробників буде 17,6 Story Points. Тепер ми маємо додавати в журнал спринту із журналу продукту стільки завдань, щоб їх сумарна кількість Story Points на виконання була якомога ближчою до 17,6.

Розглянемо вплив Власника продукту на організацію і планування спринту.

Команда розробників і Власник продукту планують роботу разом, і для кожної історії в спринті враховують три параметри, які дуже тісно пов'язані між собою. Такими параметрами є :

- Визначення важливості та пріоритетів завдань які увійдуть до поточного спринту ;
- Обсяг робіт які треба виконати під час спринту ;
- Трудовитрати на виконання завдань спринту.

Обсяг робіт та пріоритети завдань визначаються Власником продукту. Оцінка трудовитрат - це прерогатива команди (рис. 2.4). Завдяки взаємодії команди і Власника продукту в ході планування спринту виробляється оптимальне співвідношення всіх трьох змінних.

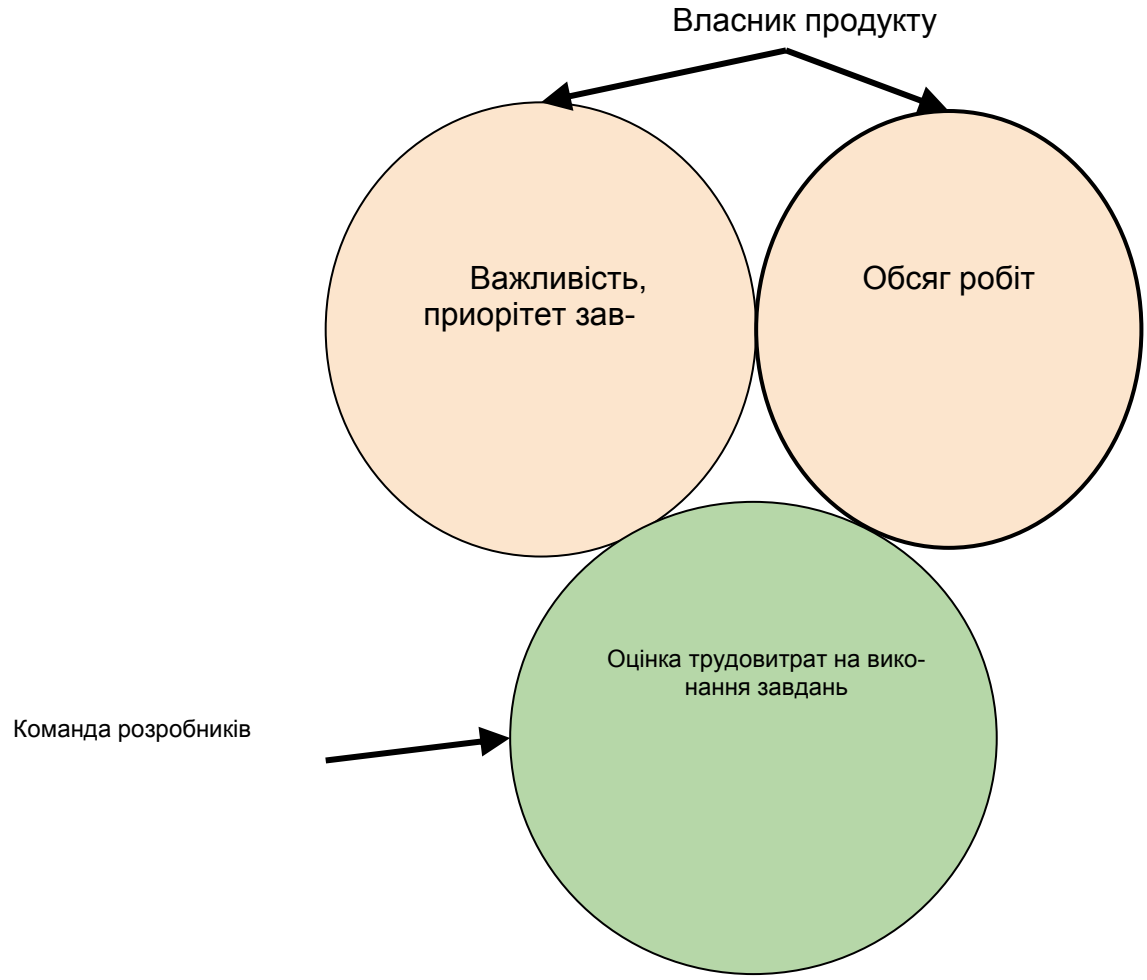


Рисунок 2.4. Параметри історій в спринті

Найчастіше Власник продукту починає планування наступного спринту з опису основних цілей і найбільш пріоритетних історій. Після цього команда проводить оцінку трудовитрат всіх історій, починаючи з найважливішої. У процесі оцінки у команди виникають питання з приводу обсягу майбутніх робіт. Іноді відповіді на ці питання приводять до перегляду всіх оцінок трудовитрат для даної історії.

У деяких випадках час, який знадобиться на виконання історії, не співпадає з очікуваннями Власника продукту. Тому, він ініціює перегляд пріоритетів для історії або змінить обсяг роботи. Такі зміни , у свою чергу, змусять команду розробників зробити переоцінку трудовитрат і так далі. Шляхом поступових доопрацювань і узгоджень команда розробників і Власник продукту виробляють оптимальний план виконання робіт на поточний спринт.

2.3.2. Оцінка витрат часу з допомогою гри в planning poker

Оцінка витрат часу на виконання завдань спринту є командною роботою, і найчастіше, всі члени команди беруть участь в оцінці кожної історії.

Під час планування не визначено, хто буде виконувати ту чи іншу частину роботи. Реалізація історій зазвичай вимагає участі різних фахівців (дизайн інтерфейсу користувача, кодування, тестування, і т.д.).

Для того, щоб кожен учасник команди міг видати якусь оцінку, він повинен більш-менш розуміти, в чому суть цієї історії. Отримуючи оцінку від кожного члена команди, ми переконуємося, що всі розуміють, про що йде мова. Це збільшує ймовірність взаємодопомоги під час спринту. А також це збільшує ймовірність того, що найбільш важливі питання з цієї історії привернуть увагу розробників якомога раніше.

При оцінці історії спільними зусиллями різнобічне бачення проблеми призводить до сильного розкиду оцінок. Такі розбіжності краще виявляти і обговорювати якомога раніше.

Якщо попросити всіх оцінити історію, то зазвичай людина, яка розуміє її краще за інших, поверне оцінку першим. На жаль це сильно впливає на оцінки інших людей.

Але існує прекрасна практика, яка дозволяє цього уникнути. Вона називається planning poker (придумана Майком Коном).

Кожен член команди отримує колоду з 13-ти карт, таких же, як на рис. 2.5. Всякий раз, коли потрібно оцінити історію, кожен член команди вибирає карту з оцінкою (в story point'ах), яка, на його думку, підходить, і кладе її на стіл сорочкою вгору. Коли все члени команди визначилися з оцінкою, карти одночасно розкриваються. Таким чином, члени команди змушені оцінювати самостійно, а не "списувати" чужу оцінку.

Якщо виходить велика різниця в оцінках, то цю різницю обговорюють і намагаються виробити спільне розуміння того, що повинно бути зроблено для реалізації цієї історії. Можливо, вони розіб'ють завдання на більш дрібні. Після цього команда оцінить історію заново. Цей цикл повинен повторюватися до тих пір, поки оцінки не зійдуться, тобто не стануть приблизно однаковими.

Дуже важливо нагадувати всім членам команди, що вони повинні оцінювати загальний обсяг робіт, а не тільки "свою частину", тобто, тестувальник повинен оцінювати не тільки роботи з тестування.

Зауважте, послідовність значень на картах - нелінійна. Ось, наприклад, між 40 і 100 нічого немає. Чому так?

Це потрібно, щоб уникнути появи помилкового почуття точності для високих оцінок. Якщо історія оцінюється приблизно в 20 story point'ов, то немає сенсу обговорювати чи повинна вона бути 20, або 18, або 21. Все, що нам потрібно знати, це те, що її складно оцінити. Тому ми приблизно призначаємо їй оцінку в 20.

Рисунок 2.5. Карти для planning poker

Якщо у вас виникло бажання більш детально переоцінити цю історію, то краще розбийте її на більш дрібні частини і оціните вже їх!

Шахраювати, викладаючи карти 5 і 2, щоб отримати 7, не можна. Ви повинні вибрати або 5 або 8 - сімки немає.

Є ще кілька спеціальних карт:

- 0 = або "історія вже готова" або ж її оцінка "пара хвилин роботи".
- ? = "Я поняття не маю. Абсолютно ".
- Чашка кави = "Я дуже втомився, щоб думати. Давайте зробимо перерву".

Для автоматизації planning poker розроблено декілька програм які знаходяться в додатках Google Play, які можна завантажити на мобільні пристрої для використання.

2.4. Проведення щоденних нарад

Щоденні наради – це 15-хвилинні зібрання для команди розробників з метою синхронізації дій та створення плану роботи на найближчі 24 години. Це робиться для того, щоб перевірити, що було зроблено з часу проведення попередньої щоденної наради та запланувати роботу, яку можна виконати за наступні 24 години.

Наради відбуваються в обумовленому місці в один і той же час для уникнення плутанини. Під час таких нарад кожен учасник команди розробників розповідає колегам про наступне: Що було зроблено з часу минулої зустрічі? Що планується зробити до наступної зустрічі? Що йому заважає у виконанні запланованих завдань?

Члени команди розробників використовують щоденні наради для оцінки прогресу просування до цілі спринту, а також оцінки прогресу у виконанні роботи із журналу спринту. Щоденні наради підвищують імовірність того, що команда розробників досягне цілі спринту. Часто команда розробників зустрічається відразу ж після щоденної наради для перепланування залишку роботи по спринту. Щодня команда розробників повинна вміти пояснити Власникові продукту та Scrum-майстру, яким чином вона працюватиме в якості самоорганізованої команди для досягнення мети та створення очікуваного покращення функціональності продукту за час, що залишився до завершення спринту.

Scrum-майстер несе відповідальність за те, щоб команда розробників не пропускала такі наради, однак відповідальність за проведення щоденної наради лежить на команді розробників. Scrum-майстер вчить Команду розробників утримувати час проведення щоденної наради в межах 15-хвилин.

Щоденні наради покращують процес спілкування всередині команди, зводячи до мінімуму потребу в інших нарадах, допомагають визначити та усу-

нути перешкоди на шляху до успішної роботи, сприяють швидкому прийняттю рішень, а також підвищують компетентність цілої команди розробників по проекту. Щоденні наради можуть проводитися віртуально, за допомогою Skype, відеоконференцій, засобів колективного використання робочого столу комп'ютера, але практика проведення нарад показує, що реальна зустріч під час наради учасників команди є більш ефективною і продуктивною. Як тільки ви зберете всю команду разом, результат з'явиться негайно. Уже після першого спринту команда погодиться, що це була хороша ідея - зібратися всім в одному місці.

"Зібратися разом" означає:

1. В межах чутності: кожен в команді може поговорити з будь-яким іншим членом команди без крику і не встаючи з-за свого столу.
2. В межах видимості: кожен член команди може побачити будь-якого іншого. Кожен може бачити дошку завдань.
3. Автономно: якщо раптом вся ваша команда підніметься і почне раптову і дуже жваву дискусію про архітектуру системи, нікого з не членів команди не виявиться досить близько, щоб йому це завадило. І навпаки. "Автономно" не означає, що команда повинна бути повністю ізольована. У просторі, розділеному на секції, цілком може вистачити окремої секції для команди, з досить високими стінами, щоб не пропускати більшу частину шуму ззовні.

2.5. Застосування дошки завдань для управління Scrum

Застосування дошки завдань э елементом Канбану (яп. рекламний щит, білборд), який представляє собою методологію візуального управління процесом розробки інформаційної системи, яка визначає що розробляти , коли розробляти, і в якій кількості розробляти. Для управління і контролю виконанням завдань журналу спринту за Канбаном використовується дошка завдань майс-

тер. Вигляд дошки завдань показано на рис. 2.6. Таким чином, дошка є наочним представленням журналу спринту.

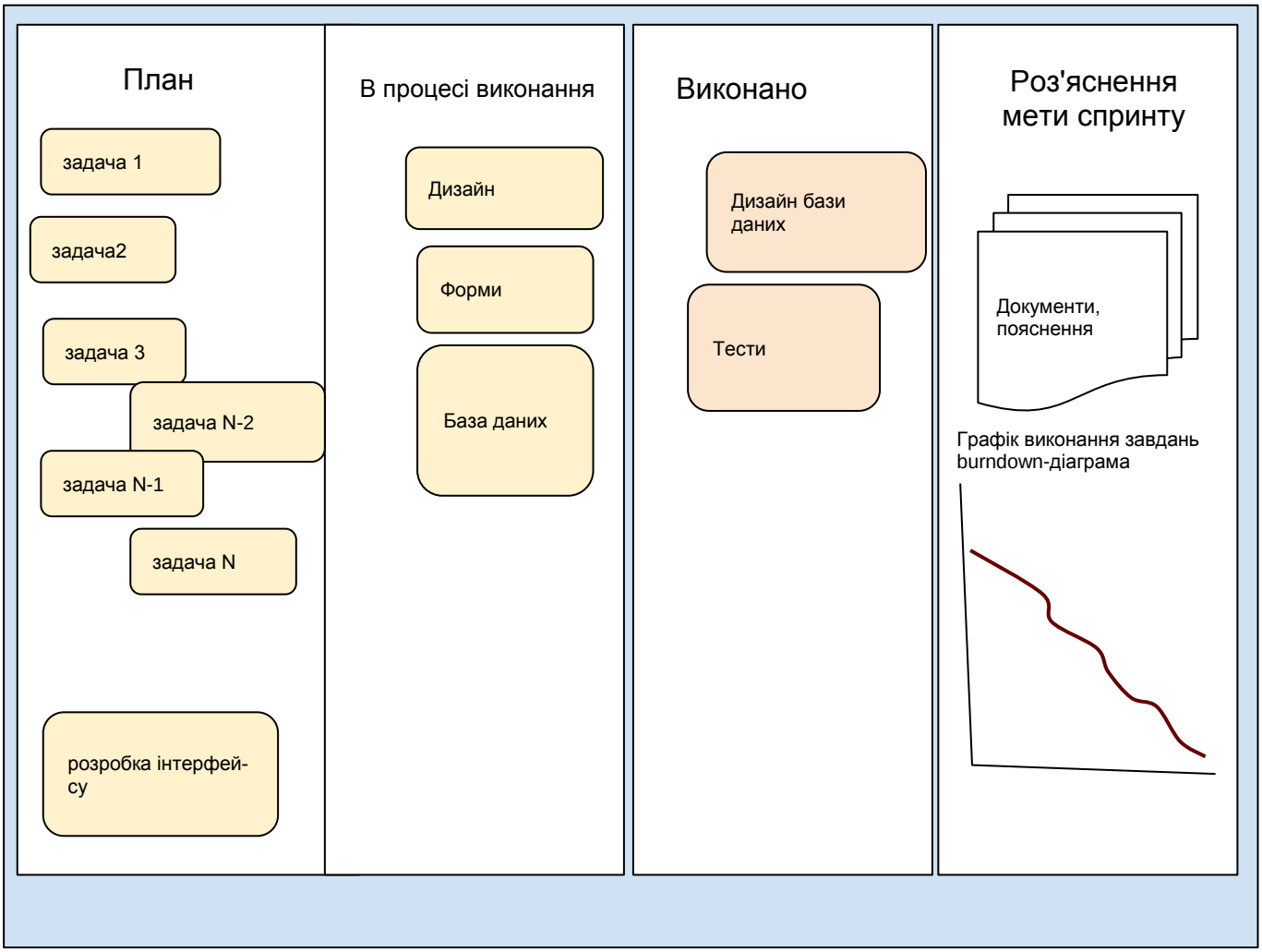


Рисунок 2.6. Дошка завдань

Для кожного завдання яке увійшло у спринт підготовляється стікер. На стікері записується назва завдання і кількість Story points, які треба витратити на виконання цього завдання.

Дошка завдань розділена на чотири частини. В першій частині яка має назву “План” містяться завдання які заплановано до виконання у спринті. В другій частині знаходяться завдання які виконуються в поточний час. Завдання які вже виконано переноситься в третю частину із назвою “Виконано”. В четвертій

частині дошки знаходяться документи які роз'яснюють цілі спринту, графік виконання завдань спринту, в якій вносяться дані після щоденної наради.

На початку спринту всі завдання знаходяться у частині “План”. В міру виконання завдань вони поступово будуть переходити у частину “В процесі виконання”. Виконані завдання переносяться у третій розділ дошки і такими завданнями вже ніхто не буде займатися у цьому спринті.

Побіжний погляд на дошку завдань повинен дати можливість будь-якій людині зрозуміти, наскільки успішно просувається ітерація спринту. Scrum-майстер несе відповідальність за те, щоб команда приймала відповідні заходи при виявленні перших тривожних симптомів.

Зазвичай дошка завдань оновлюється під час щоденної наради. У міру того, як кожен член команди розповідає про те, що він зробив за вчорашній день і чим буде займатися сьогодні, він переміщує стікери на дошці завдань. Як тільки розповідь стосується якогось незапланованого завдання, то для нього клеїться новий стікер. При оновленні тимчасових оцінок, на стікері пишеться нова оцінка, а стара закреслюється. Іноді переміщенням стікерів займається Scrum-майстер.

У деяких командах прийнято, що всі члени команди оновлюють дошку завдань перед кожною зустріччю. Це теж добре працює.

Незалежно від того, який формат журналу спринту ви використовуєте, намагайтеся залучити всю команду в підтримку журналу спринту в актуальному стані.

В деяких випадках журнал спринту веде тільки Scrum-майстер. Він повинен кожен день обходити всіх членів команди і дізнаватися про оцінки виконання завдань у часі і по суті роботи.

Недоліки цього підходу в тому, що:

- Scrum-майстер витрачає занадто багато часу на "паперову роботу", замість того, щоб займатися підтримкою команди і усуненням перешкод;
- Члени команди не в курсі стану спринту, оскільки їм не потрібно піклуватися про журнал спринту . Цей брак зворотного зв'язку зменшує загальну гнучкість і сконцентрованість команди.

Тому робота всієї команди над журналом спринту і дошкою є більш доцільною.

Відразу ж після щоденної наради, хтось із членів команди підсумовує оцінки всіх завдань на дошці і ставить нову точку на графіку виконання завдань спринту (burndown-діаграмі).

Розглянемо роботу з графіком виконання завдань спринту. Графік має вигляд показаний на рис. 2.7.

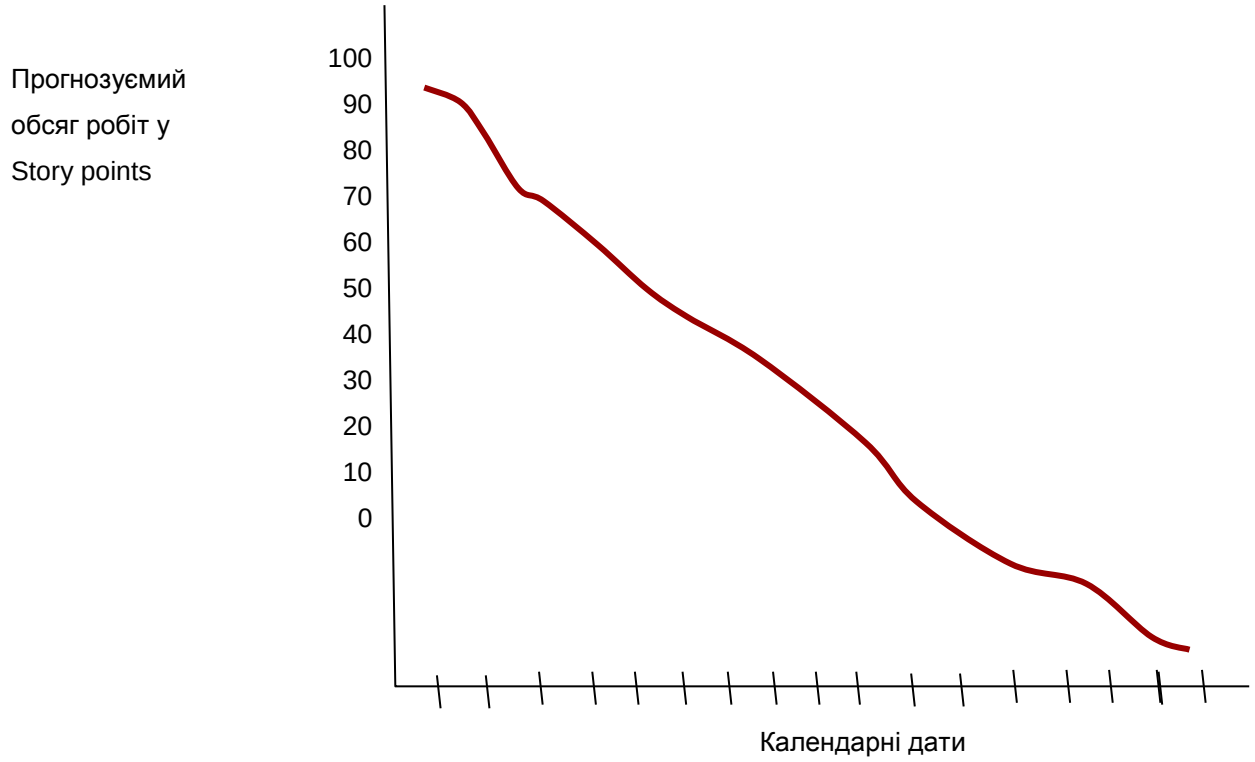


Рисунок 2.7. Графік виконання завдань спринту (burndown-діаграма).

Припустимо, на виконання робіт спринту виділено 100 Story points. Наступного дня, на щоденній нараді, з'ясовується скільки Story points виконано за добу і на діаграмі відмічається відповідна точка.

По мірі просування проекту на діаграмі відмічаються точки виконання робіт і у будь який час можна побачити обсяг роботи яка виконана та яку ще треба виконати у цьому спринті. Графік має вигляд спадної кривої.

2.6. Проведення демонстрацій

Демонстрація, яка дозволяє обговорити результати виконаних робіт, обмінюватися досвідом і дати оцінку результатам роботи окремих учасників команди (рис.2.8). Ретельно підготовлена і проведена демонстрація має позитивний вплив на виконання інформаційного проекту.

Цей вплив можна підсумовувати наступним чином:

- Демонстрація має проходити в дружній атмосфері, тому різні команди можуть вільно спілкуватися між собою і обговорювати нагальні питання і обмінюватися досвідом;
- На демонстрації зацікавлені сторони обмінюються важливими відгуками та думками;
- Інші команди розробників дізнаються, чим займається ваша команда;
- Позитивна оцінка роботи надихає команду;
- Проведення демонстрації змушує команду дороблятиі завершувати виконання завдань. Демонстрації дозволяють зменшити кількість завдань, які знаходяться на завершальному етапі але все ще не завершені.

Рисунок 2.8. Проведення демонстрації

Якщо команду змушувати проводити демонстрації , коли вони мають не завершений і не працюючий продукт, їм буде не по собі. Команда буде запинатися і спотикатися, показуючи функціональність розробленого продукту. Людям буде шкода цю команду, а декого може навіть розлютити те, що вони втратили час на цій демонстрації. Це дуже неприємно. Але це діє, як гірка пілюля. У наступному спринті команда намагатиметься все доробити. Команда знає, що демонстрацію доведеться проводити не дивлячись ні на що, і завдяки цьому шанси побачити завершену роботу значно зростають.

Основні поради з підготовки та проведення демонстрації можна сформулювати наступним чином:

- Scrum-майстер має якомога чіткіше сформулювати мету поточного спринту;
- Якщо на демонстрації присутні люди, які нічого не знають про ваш продукт, то треба приділити пару хвилин, щоб ввести їх в курс справи;

- Не витрачайте багато часу на підготовку демонстрації, особливо на створення ефектної презентації. Сконцентруйтеся на показі тільки реально працюючої задачі;
- Слідкуйте, щоб демонстрація проходила в швидкому, динамічному темпі;
- Бажано щоб демонстрація була зосереджена на сутності завдань, які розв'язуються, а не на технічних деталях;.
- Треба приділити увагу тому "що ми зробили", а не на тому "як ми це робили";
- Якщо це можливо, дайте аудиторії самій спробувати розроблений продукт;
- Не потрібно показувати багато зроблених виправлень, достатньо згадати про них, але демонструвати їх не варто, тому що це забере багато часу і знизить увагу до більш важливих завдань.

2.7. Висновки

Планування Scrum виконується покроково. Схема Scrum має вигляд ітераційного процесу. Одна ітерація має назву спринт (sprint). Послідовність спринтів складає ітераційний процес Scrum.

Мета планування полягає в тому, щоб, забезпечити команду інформацією для ефективної роботи протягом декількох тижнів.

Журнал продукту (Product backlog) - це основа Scrum'а. Журнал продукту є списком вимог, історій, функціональності, які упорядковані за ступенем важливості.

Спринт, планується під час наради із планування спринту. План дій розробляється при спільній роботі всієї Scrum команди. Під час планування спринту члени команди відповідають на наступні питання: Що буде розроблено, що стане результатом роботи наступного Спринту. Як поліпшити роботу Спринту.

Власник продукту визначає важливість і перелік історій які увійдуть в розробку в наступному спринті. Команда розробників визначає трудомісткість завдань і обговорює з власником продукту деталі виконання робіт.

Оцінка витрат часу на виконання завдань виконується за допомогою розрахунків story point і фокус-фактору. Оцінка витрат часу може доповнюватися методологією яка має назву planning poker.

Scrum передбачає проведення щоденних нарад тривалістю не більше 15 хвилин у чітко визначеному місці проведення наради. Щоденні наради покращують процес спілкування всередині команди, зводячи до мінімуму потребу в інших нарадах, допомагають визначити та усунути перешкоди на шляху до успішної роботи, сприяють швидкому прийняттю рішень, а також підвищують компетентність цілої команди розробників по проекту.

Для управління і контролю виконанням завдань журналу спринту. використовується дошка завдань.

Дуже важливим і обов'язковим елементом управління розробкою інформаційної системи є проведення демонстрації наприкінці спринту. Демонстрація дозволяє обговорити результати виконаних робіт, обмінятися досвідом і дати оцінку результатам роботи окремих учасників команди. Ретельно підготовлена і проведена демонстрація має позитивний вплив на виконання інформаційного проекту.

2.8. Контрольні запитання

1. Яка мета планування спринту?
2. Що містить правильно підготовлений план спринту?
3. Роз'яснить як ви розумієте схему організації спринту, проілюструйте відповідь графічно.
4. Для чого потрібен журнал продукту ?
5. Роз'яснить основні елементи журналу продукту.

6. Яку роль відіграє власник продукту в плануванні спринту?
7. Роз'яснить як би ви організували планування спринту?
8. Як розрахувати витрати часу на розробку із застосуванням методу Story point?
9. Як розраховується витрати часу на розробку за методом Фокус-фактору?
10. Розрахуйте із допомогою Фокус-фактору таке завдання: командою з чотирьох чоловік реалізовано 20 Story Point'ов. Тривалість спринту була 2 тижні, що становить 40 людино-днів. Враховуємо, що члени команди брали відгули, тоді отримаємо 50 людино-днів. Необхідно спрогнозувати продуктивність команди на майбутній спринт.
11. Для чого потрібні щоденні наради і що вони покращують в роботі команди?
12. Роз’ясніть як застосовується дошка завдань для управління роботою команди?
13. Для чого слугує демонстрація спринту?
14. Що треба врахувати при підготовці і демонстрації спринту?

Тема 3 Підвищення ефективності впровадження Scrum

В темі розглянуто сучасні методики підвищення ефективності роботи і взаємодії розробників інформаційного продукту із застосуванням SCRUM.

3.1. Проведення ретроспективи Спринту

Ретроспектива проходить наприкінці спринту відбувається після демонстрації, перед подальшим плануванням спринту. Команда демонструє результати виконаних робіт та покращену функціональність продукту всім зацікавленим особам. Залучається максимальна кількість глядачів. Усі члени команди беруть участь у демонстрації. Тривалість ретроспективи обмежується у часі (наприклад, 4-ма годинами) в залежності від тривалості ітерації і змін у продукті.

Ретроспектива спринту дає Scrum команді можливість перевірити себе та створити план покращень, які можна було б внести під час наступного спринту. Метою ретроспективи спринту є:

1. Перевірка того, наскільки успішно пройшов спринт, беручи до уваги злагодженість роботи команди, процеси та використані інструменти;
2. Визначення та впорядкування тих елементів роботи, які були виконані успішно, і тих, які могли б бути виконані краще;
3. Розробка плану із впровадження покращень у процес роботи Scrum команди.

Scrum-майстер заохочує Scrum команду переглянути процеси розробки в рамках процесів та практик Scrum, щоб зробити її більш ефективною у наступ-

ному спринті. Під час кожної ретроспективи спринту Scrum команда шукає шляхи для покращення якості продукту, за необхідності уточнюючи визначення “завершеності”.

Під час ретроспективи розглядаються і вирішуються наступні питання:

- Власник продукту визначає що зроблено і що не зроблено під час спринту;
- Команда розробників аналізує що було зроблено правильно, які проблеми виникнули і як їх було розв'язано;
- Власник продукту разом з командою аналізує виконання завдань журналу продукту і роблять висновки яким чином можна було покращити роботу щоб зробити її швидше і якісно;
- Команда аналізує, що треба зробити щоби більш ефективно виконати завдання в наступному спринті;
- В результаті проведення ретроспективи формується перелік завдань які їдуть наступний журнал спринту.

Перед завершенням ретроспективи спринту Scrum команда повинна визначити практичні та дієві чинники покращення процесу роботи, які вона реалізує у наступному спринті. Впровадження цих змін у наступному спринті якраз і є адаптацією до перевірки самої Scrum команди. Хоча зміни можуть бути внесені в будь-який час, ретроспектива спринту є спеціалізованою нарадою, присвяченою виключно перевірці і аналізу зробленого та розробці заходів покращення роботи команди.

Інформація, яка напрацьовується в ході ретроспектив, зазвичай вкрай важлива. Можливі шляхи вирішення проблем, знайдених командою на ретроспективі, можуть виявитися корисними не тільки для неї самої, але і для інших команд.

Для того, щоб акумулювати результати ретроспектив і передавати їх іншим командам обирають одну людину, яка бере участь у всіх ретроспективах всіх Scrum команд в ролі "сполучної ланки". Ця особа під час ретроспективи пропонує досвід інших команд для застосування і покращення роботи над інформаційними продуктами.

3.2. Робота з версіями інформаційного проекту

Особливістю роботи з інформаційними проектами є застосування версій [3]. Життєвий цикл продукту включає в себе декілька версій які є поступовим удосконаленнями функціональності інформаційної системи. Тому планування має враховувати наявність декількох версії продуктів які будуть розроблені поступово у часі. Планування повинно бути сконструйовано таким чином, щоб передбачати перебіг подій більше ніж на одну версію інформаційного продукту.

В певний час замовнику треба представити першу версію інформаційного продукту. Перша версія має включати всі основні і найбільш важливі функції, які необхідні для роботи інформаційної системи. Додаткові функціональності продукту додаються у подальших версія системи.

Для такого планування можна застосувати наступний алгоритм .

1. Розглядаємо завдання журналі продукту і виконуємо їх групування за важливістю для функціонування інформаційної системи . Надаємо кількісну оцінку важливості за вибраною шкалою для цих груп. (Наприклад оцінка за сто-бальною школою);
2. Сортуємо групи завдань з їх важливість;
3. Виділяємо групи завдань, які обов'язково увійдуть у першу версію інформаційної системи, групи завдань, які бажано включити до першої версії але при несприятливих обставинах можуть бути перенесені в наступні версії та групи завдань які будуть реалізовані у наступних версіях;

4. Виконуємо оцінку витрат часу на розробку першої і другої групи завдань за важливістю. В оцінці витрат часу приймають участь власник продукту і команда розробників. Вони спільно обговорюють завдання і отримують обґрунтовану кількісну оцінку продуктивності команди. Для розрахунків використовуються методики які розглянуті параграфі 2.3.1.
5. Плануємо спринт;
6. Виконуємо роботи заплановані у спринті. Оцінюємо реальну продуктивність команди розробки. Якщо за непередбачуваних обставин продуктивність команди є нижчою ніж заплановано виконуємо коригування завдань. Команда разом із власником продукту змінює перелік завдань, які буде включено до наступних спринтів. Перегрупування виконується наступним чином: завдання першої групи за важливістю, які не можуть бути виконані переміщуються у другу групу, а ті що не можуть бути виконаними у другій групі переходять до третьої групи. Таким чином вдається узгодити вимоги замовника по функціональності інформаційного продукту і реальну продуктивність команди розробки системи. В результаті замовник інформаційної системи має змогу обрати, чи отримати йому найважливішу функціональність у першій версії системи або отримати систему у цілому із затримкою у часі.

Приклад типової ситуації подано у [3]. Власник продукту працював разом з командою над проектуванням заплутаного екрану для пошуку складних даних. Запропонований варіант виявився далеким від ідеального, але ніхто не зміг запропонувати нічого кращого, і в цілому цей екран влаштовував всіх. Під час чергової наради, присвяченого планування спринту, всі погодилися з тим, що потрібно розробляти екран пошуку відповідно до запропонованого прототипом.

Робота просувалася успішно протягом першої половини двотижневого спринту цієї команди. На сьомий день вранці власник продукту оголосив, що

напередодні ввечері на нього зійшло осяяння. Він продемонстрував розробникам начерк більш вдалого, екрана пошуку. Він не мав практично нічого спільного з варіантом, який власник продукту запропонував раніше. Безумовно, новий варіант був кращий від попереднього. Ніхто навіть не намагався це оскаржити.

Команді залишалося лише зважитися на одне з двох.

- Зупинити поточний спринт і почати новий спринт, повністю зосередившись на новому варіанті екрану пошуку, визнаному більш вдалим, ніж попередній;
- Продовжити роботу над колишнім варіантом екрану пошуку, визнаним цілком задовільним сім днів назад.

Новий варіант екрану пошуку був явно краще, тому розробникам слід було негайно переключитися на його реалізацію. Однак методологія Scrum допомогла їм сформулювати це питання дещо по-іншому: поставити замовнику продукт з цілком задовільним варіантом екрану пошуку, маючи в розпорядженні сім днів для розробки додаткових функціональних можливостей, або поставити замовнику продукт тільки з більш вдалим варіантом екрану пошуку?

Остаточне рішення залишалося за власником продукту, але у виробленні цього рішення приймала участь вся команда. Колективно команда вирішила завершити реалізацію цілком задовільного варіанта екрану пошуку. Вдосконалений варіант екрану пошуку був реалізований через дев'ять місяців в наступній версії продукту.

3.3. Забезпечення якості інформаційного проекту

Якість - це «ступінь відповідності характеристик продукту вимогам замовника , їх досконалість. Тому для якість означає декілька понять, серед яких основні:

- Якість елементів проекту;

- Якість процесу, виробництва цих елементів проекту.

Необхідно розуміти різницю між якістю і гатунком. Гатунок - це категорія, що призначається продуктам або послугам, що мають одне і те ж функціональне призначення, але різні технічні характеристики .

Наприклад, програмний продукт може бути високої якості (відсутність очевидних дефектів), але низького ґатунку (обмежена кількість функцій), або ж низької якості (безліч дефектів, погано організована документація для користувачів), але високого ґатунку(безліч функцій). Менеджер проекту і команда управління проектом відповідають за визначення та забезпечення необхідних рівнів як якості і гатунків.

Розвиток технологій і обчислювальної техніки кардинально змінив обставини розробки інформаційних продуктів. Нині став доступним і недорогим доступ до Інтернету, інформація стала надлишковою і часто безкоштовною, обчислювальні потужності значно подешевшали. Тому можливості в експериментуванні по досягненню високої якості продукції значно розширилися (рис.3.1). Зокрема вартість помилок які трапляються в розробці знижується. Тому розробники мають можливість експериментувати з метою досягнення найвищої якості продукту.

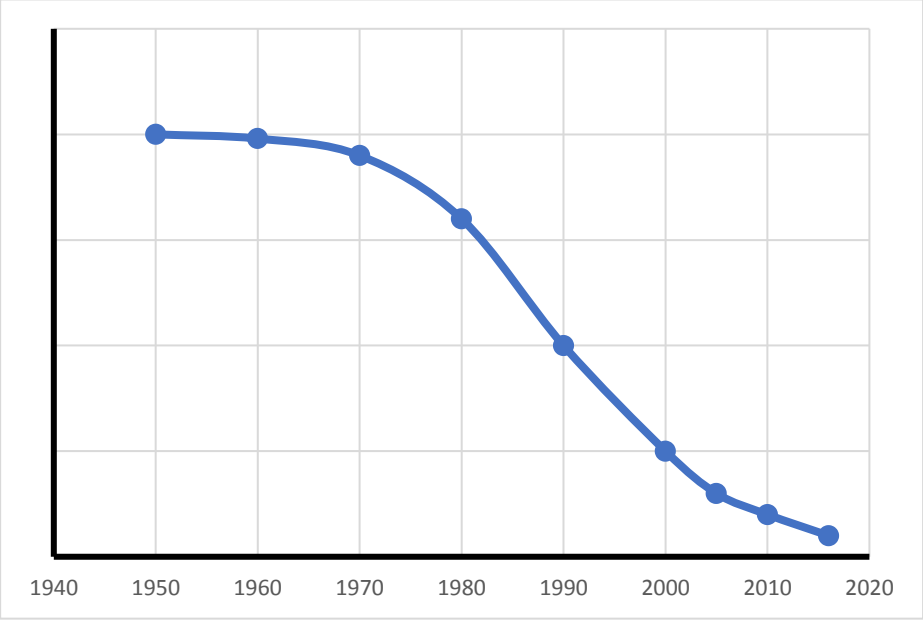


Рисунок 3.1. Вартість експериментування

Якість інформаційних продуктів можна оцінити за допомогою тестування. Для Scrum-команд тестування є одним з центральних видів діяльності і частиною процесу розробки, а не чимось таким, що відбувається вже після того, як розробники завершають свою роботу. Замість того щоб намагатися тестувати якість після того, як продукт вже створений, методологія Scrum вбудовує якість в процес розробки і в продукт в міру просування його розробки. Якість неможливо "додати в продукт згодом". Тому потрібно вдосконалити сам процес розробки і вбудувати якість в створюваний продукт.

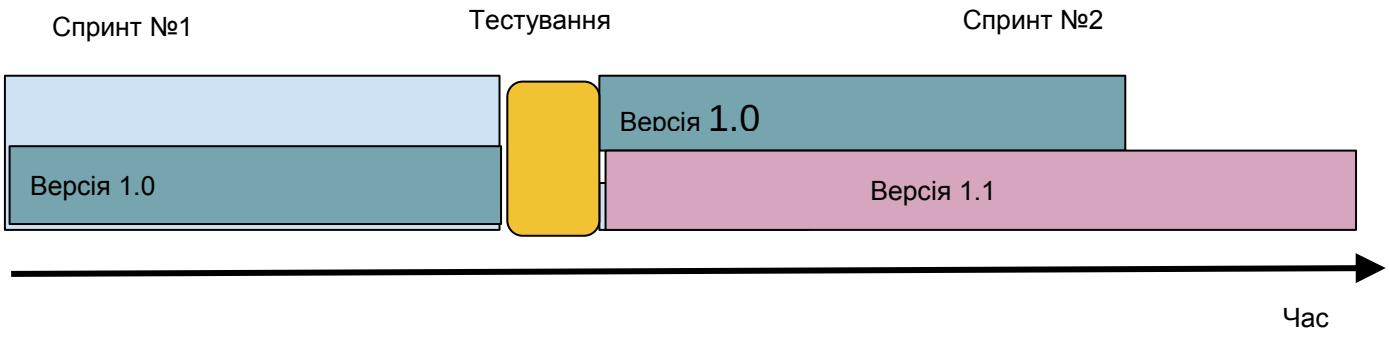
В ідеальному випадку результатом кожного спринту повинна бути система, або її частина потенційно готова до використання. Досвід розробки інформаційних систем показує що така система такий підхід не працює Зазвичай в результати спринту будуть міститися помилки і недоробки. Для виявлення помилок залучають тестувальників, які не є частиною команди. Вони перевіряють інформаційну систему тими видами тестів, про які Scrum-команда не могла навіть і подумати, чи на які у неї не було часу, або відповідного обладнання. Тестувальник моделює роботу користувачів у різноманітних ситуаціях виявляючи недоліки системи. На рисунку 3.2 показана робота команди розробників, тестувальників і користувачів.

Рисунок 3.2 Тестування інформаційної системи.

Приймальне тестування пов'язано з незручностями і затримками у розробці продукту. Тому необхідно зменшити кількість часу, який для нього необхідно. Цього можна досягти наступними способами:

- Максимально поліпшити якість версії продукту, який створює команда розробників;
- Підвищити ефективність ручного тестування ,тобто знайти кращих тестувальників, забезпечити їх кращими інструментарієм ;
- Автоматизувати тестування.
- Заучити тестувальників у Scrum-команду;
- Зменшити обсяг робіт , які виконуються за спринт. Це автоматично призведе до підвищення якості, зменшить тривалість приймального тестування і кількість помилок. Зрештою, це повинно підняти продуктивність всієї команди, адже вона зможе сконцентруватися на нових завданнях, замість того, щоб постійно витрачати час на виправлення старого коду, який постійно ламається.

Практична реалізація підвищення якості і усунення недоліків реалізується наступним чином: коли команда закінчує спринт, вона переходимо до наступного, але враховує, що в наступному спринті буде потрібно витратити час на виправлення помилок минулого спринту (рис.3.3). Якщо наступний спринт виявляється перевантажений роботою над виправленням дефектів минулого, то команда намагається зрозуміти причину такої кількості дефектів і виробити



спосіб підняти якість.

Рисунок 3.3 Підвищення якості за рахунок усунення недоліків.

Як наслідок, команда визначає довжину спринту достатньою, щоб встигнути впоратися з обсягом роботи по виправленню недоліків минулого спринту.

Поступово, за кілька місяців, кількість роботи з усунення дефектів минулих спринтів зменшується. При плануванні спринту, щоб врахувати той час, який є запланований на усунення дефектів, команда встановлює зменшене значення фокус-фактора. Згодом команда починає краще визначати потрібне значення фокус-фактора. У цьому допомагає статистика реальної продуктивності команди розробки у минулих спринтах.

3.4. Робота з географічно віддаленими командами

Розвиток Інтернету і систем віртуальних комунікацій дає можливість географічно віддаленим командам та працівникам працювати спільно за Scrum технологією.

Цей вид діяльності, який називають віртуальним робочим місцем (ВРМ), може бути реалізований вдома або ж на території клієнта, тобто скрізь, де забезпечується доступ до мережі зв'язку. Віртуальне робоче місце містить два основних компоненти — робоче місце співробітника і корпоративну мережу підприємства, до якої підключається співробітник для виконання своїх функціональних обов'язків.

Взаємодія між робочим місцем співробітника і корпоративною мережею здійснюється через Інтернет.

Основні характеристики віртуальної форми організації такі:

- відкрита розподілена структура;
- гнучкість;
- пріоритет горизонтальних зв'язків;
- високий статус інформаційних і кадрових засобів інтеграції.

Ключовою перевагою віртуальних форм є можливість вибирати і використовувати найкращі ресурси, знання і здібності з найменшими витратами часу. Основні конкурентні переваги віртуальних підприємств:

- швидкість виконання ринкового замовлення;
- можливість зниження сукупних витрат;
- можливість повнішого задоволення потреб замовника;
- можливість гнучкої адаптації до змін навколишнього середовища;
- можливість знизити бар'єри виходу на нові ринки.

Поряд із переліченими щойно перевагами віртуальні підприємства мають і деякі слабкі місця:

- Така система може працювати лише з мотивованими і сумлінними працівниками;
- Треба приділяти велику увагу безпеці передачі конфіденційної інформації через мережу Інтернет.

В організації географічно віддалених команди Існує дві основні стратегії :

1. Робота з віддаленими Scrum командами ;
2. Робота з окремими віддаленими членами команди Scrum

Робота з віддаленими командами показана на рис. 3.4

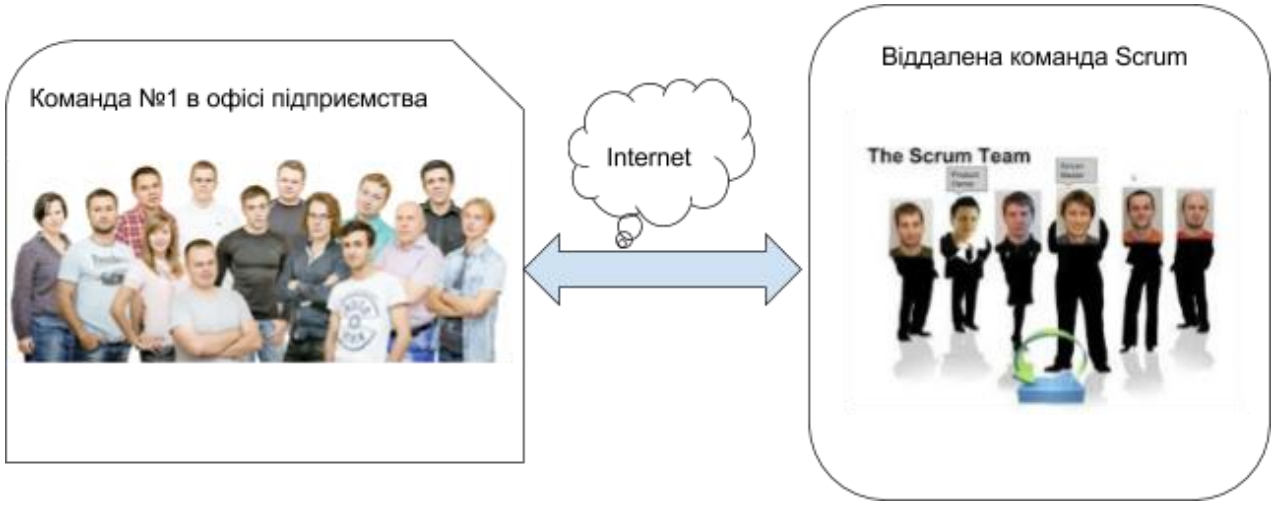


Рисунок 3.4 робота з віддаленою командою.

Робота з окремими віддаленими членами команди показана на рис. 3.5.

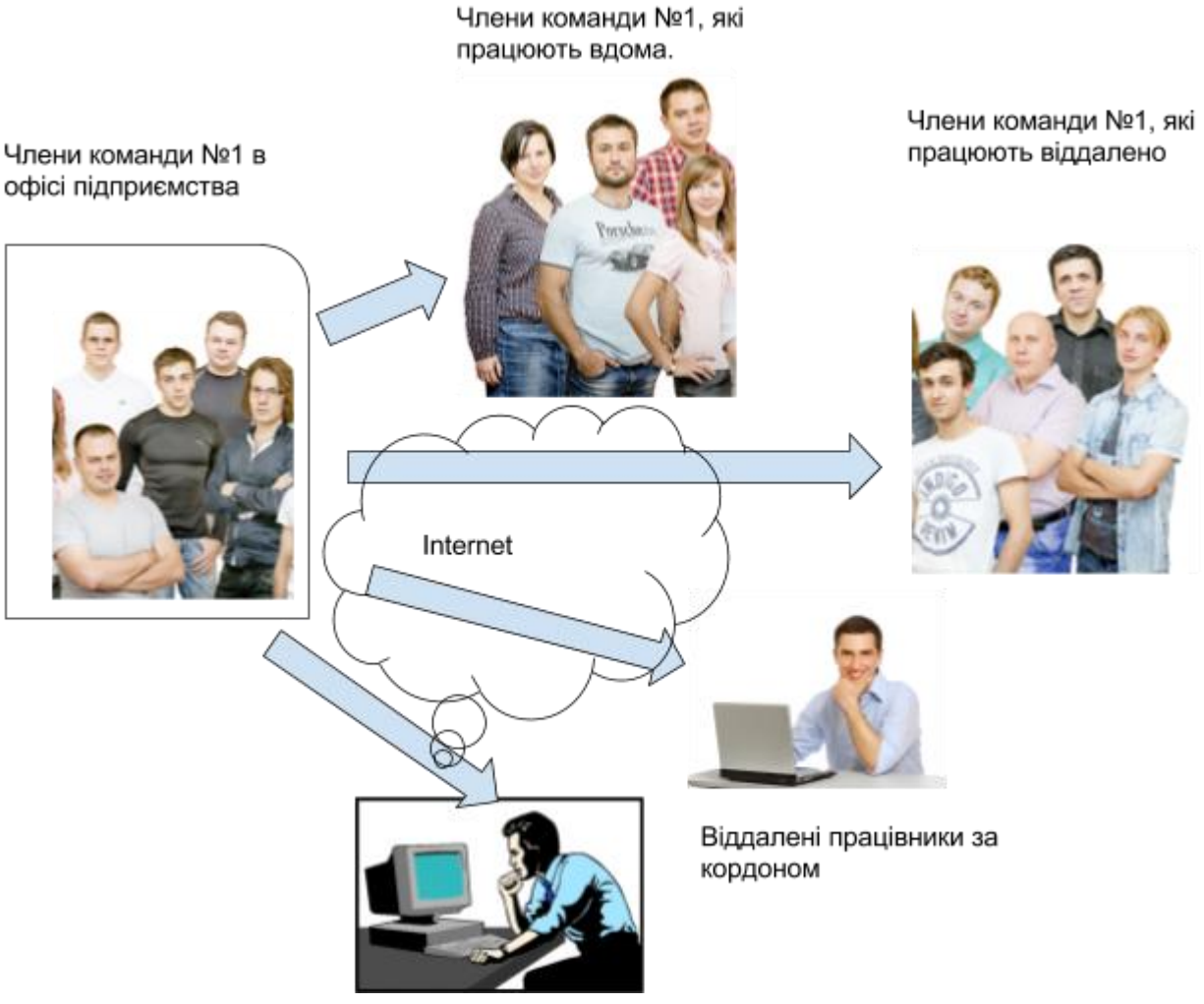


Рисунок 3.5. Робота з віддаленими працівниками команди

Географічне віддалені команди і члени команд беруть участь в щоденному Scrum, використовуючи Skype - конференції (іноді відео конференції). Вони доступні в чаті протягом всього дня.

Для ефективної роботи віддалених команд треба вжити наступні заходи:

- Web-камера і навушники з мікрофоном мають бути на всіх робочих станціях членів команди.
- Кімната для проведення телеконференцій, обладнана web-камерами, мікрофонами, комп'ютерами з усім необхідним ПЗ - для загального доступу до робочого столу під час проведення телеконференцій .
- "Віддалені вікна". Великі монітори в кожному офісі, на яких завжди можна бачити, що відбувається в інших офісах. Віртуального вікна між двома відділами за допомогою, яких можна спостерігати інші команди і віддалених працівників на своєму робочому для того, щоб створити відчуття, що всі розробники знаходяться разом.

При дотриманні вказаних вимог робота з віддаленими командами і працівниками є достатньо ефективною і перспективною формою організації розробників інформаційних проектів.

Які існують ефективні програмні засоби для роботи з віддаленими командами? Аналіз існуючого програмного забезпечення показує що до таких засобів відносяться наступні розробки :Wizergos, Jell, geekbot , Status Hero, Google Docs, JIRA, Google Hangouts, Skype, ScrumDo , Xamun, WebStandup. Програмні продукти дозволяють організувати зустріч між членами команди в різних часових поясах (асинхронна зустріч) і забезпечити безпосереднє спілкування (синхронні зустрічі). Крім того, програмні засоби забезпечують спільну роботу з документами, а також планування спринтів для таких команд.

3.5. Досвід роботи з Scrum

Своїм досвід Scrum діляться з нами розробники інформаційних систем. Карманова Наталія працювала у якості Scrum-майстра і учасника команди розробки в міжнародній IT компанії «111 хвилин». Їх команда розробляла продукти для мобільних пристроїв. Одним з таких продуктів є додаток “Sightseen”, який призначений для туристів і мешканців міста Амстердам. Додаток працює на мобільних пристроях Apple і дозволяє знаходити туристичні місця відповідно до запитів користувачів.

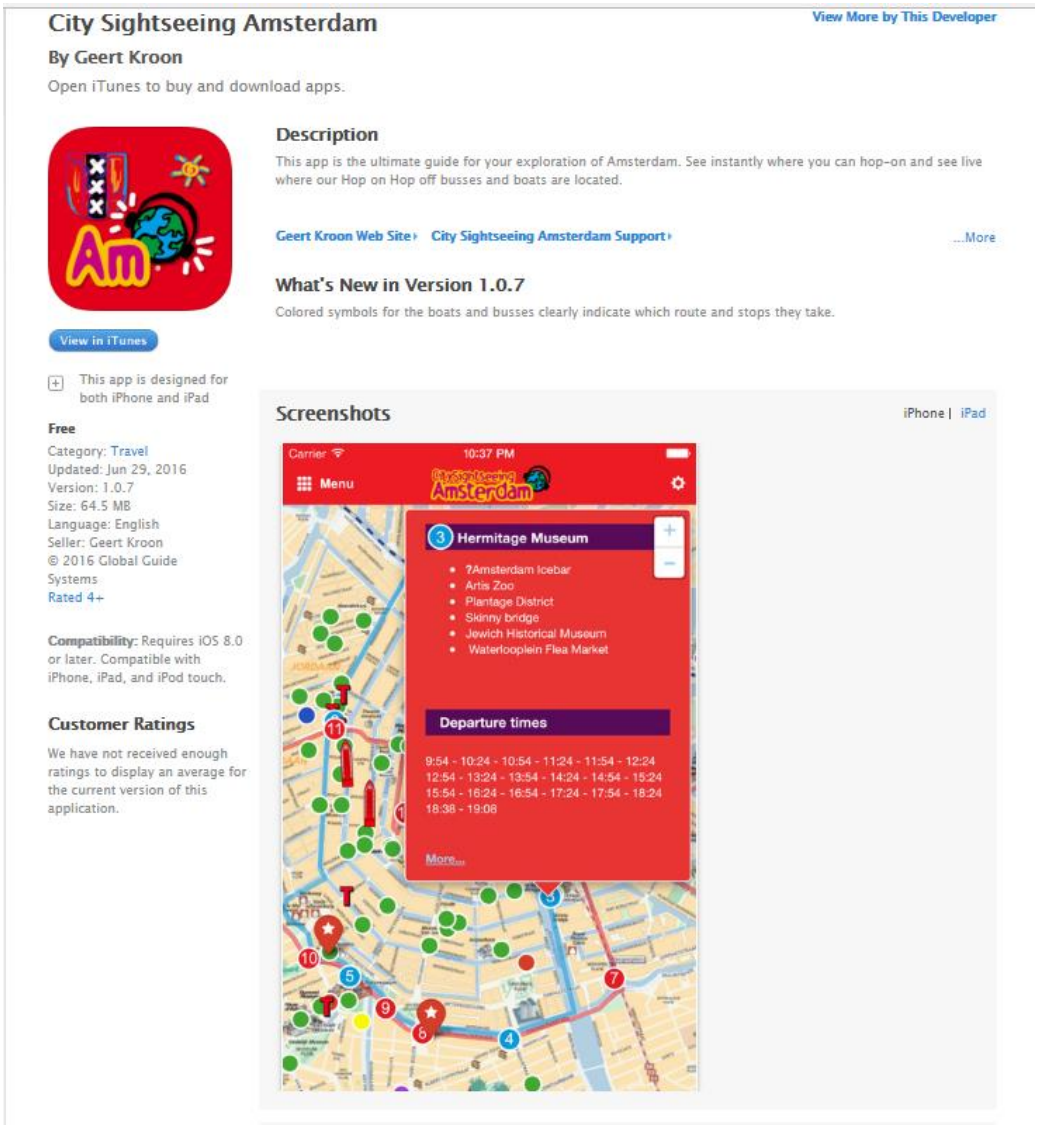


Рисунок 3.6. Додаток Sightseen Amsterdam

Досвід роботи Наталія сформулювала у вигляді відповідей запитання.

Коротко про продукт Sightseen Amsterdam

1. “ Sightseen Amsterdam” - додаток, який дає змогу знайти цікаві місця у Амстердамі. Цікавий як туристам, так і місцевим мешканцям.

2. *Принципи розподілу ролей учасників гнучкої розробки*

Певного принципу не існує. Команда формується виходячи із завантаженості працівників, а також із вимог кожного проекту індивідуально (наприклад, якщо замовники англомовні та хочуть спілкуватись з розробником та менеджером протягом проекту, то розробник і менеджер відповідно повинні володіти достатнім рівнем англійської мови)

3. *Який розмір і структура команди?*

Розмір і структура команди залежить від складності та терміну виконання проекту. Розмір команди залежить від складності і трудомісткості проекту. Стандартний склад команди: менеджер, тестувальник, дизайнер, програміст (також до вартості послуг входить системний адміністратор, але в команді він не фігурує, так само як і сейлс менеджер)

4. *Чим умотивовані розробники для самоорганізації? Які види мотивації корисні, які - шкідливі?*

Самоорганізація означає свободу вибору розробника, використовувати інструменти, які він вважає найдоцільнішими. На простому прикладі, коли клієнт приходить до сервісу, наприклад, до ресторану, він не вчить повара як йому готувати замовлення, тому що нічого не тямить у цьому, він може лише попросити повара не додавати гірчицю до салату та не сильно прожарити стейк. А якщо клієнт розуміється на готовці (клієнт сам може бути поваром), то він може обговорити процес приготування з поваром та вислухати його аргументи, чому він готовитиме страву тим, а не іншим чином.

Також, повертаючись до розробки ІС, якщо розробник застосовуватиме застарілі інструменти, то він нашкодить як проектові так і собі, і ризикує стати неактуальним = непотрібним на ринку праці ІТ.

a. *Роль Scrum-майстра, як лідера процесу*

Scrum-майстер - він же проект менеджер слідкує за життєдіяльністю проекту від самого початку і до кінця:

Етап Передпродажної оцінки (Presale):

- робить аналіз вимог продукт замовника
- проводить оцінку проекту на стадії Естімету разом із командою і сейлзом

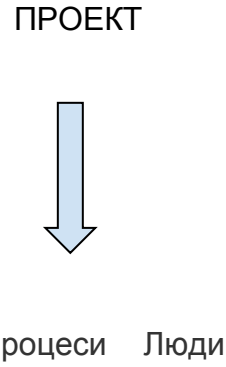
Етап розробки продукту (Development):

- обробляє документацію (Product backlog -> Scope of Work, Schedule) та передає її команді у зручному вигляді (дадаючи у трекер задач)
- створює сприятливу атмосферу для усіх членів команди (слідкую за тим, щоб усі були забезпечені необхідною інформацією, документацією, доступами, залагоджує конфлікти і тд)
- проводить мітинги
- спілкується із замовниками (processes changes, deliver reports, paymnets etc)
- слідкує за виконанням задач та дотриманням графіку
- знає нюанси та бізнес логіку продукту, що завжди стає у нагоді для програміста, який не поглиблюється у такі деталі
- приймає рішення у нестандартних ситуаціях

5. *Як бути хорошим Scrum-майстром?*

Основна задача - дати змогу виконавцям ефективно застосовувати свої знання та вміння.

Необхідні якості: лідерство, дипломатичність, розуміння процесів та вміння керувати ними.



6. *На що звертати увагу при виборі Scrum-майстра?* На його комунікаційні здібності, лідерські якості
7. *Які характеристики правильного Власника продукту?*
   Чіткий опис своїх вимог, розуміння того, що команда знає як реалізувати продукт.
8. *Якими інструментами, техніками, навичками корисно володіти?*
   Треба бути добрим знайомим з стандартами якій викладено в “PM BOK Guide and Standards”
9. *Планування спринту (ітерації розробки)*
   Спринти - це задачі розбиті на проміжки часу. Вони містять у собі пріоритети (пріоритети для замовника і для розробника), тобто певні задачі незручно виконати на перших ітераціях, але замовнику це конче потрібно (щоб показати інвестору наприклад)
10. *Щоденні зустрічі і в Scrum проекті*
    Щодня збирається зустріч - мітинг, яка має на меті обговорення поточних задач та проблем
    Також, команда збирається під час проекту за необхідності, а також перед кожним спринтом, щоб обговорити задачі, риски та інше

11. *Демонстрація результатів спринту і завершення ітерації*

    По завершенню ітерації замовник отримує Демо версію продукту, яку він вивчає і надає відгук (фідбек) в якому можуть міститися зауваження.

12. *Ретроспектива (аналіз виконаної роботи) процесу Scrum і як виконуються безперервні поліпшень інформаційного продукту*

    Безперервність поліпшення - це робити висновки з помилок, та вчитися на досвіді, щоб не повторювати помилок знову на інших проектах. Ретроспективний аналіз виконуємо на завершенні спринту.

3.6. Висновки

Важливим інструментом підвищення ефективності роботи команди є ретроспективи . Ретроспективи дозволяють виявити недоліки в роботі і усунути їх при подальшому виконанні інформаційного проекту.

Аналіз і усунення недоліків дозволяють підтримувати високу якість продукту вже під час розробки.

Основними характеристиками організації роботи віддалених команд і працівників є такі: відкрита розподілена структура, гнучкість, пріоритет горизонтальних зв’язків, високий статус інформаційних і кадрових засобів інтеграції.

Перевагою віртуальних форм роботи з географічно віддаленими працівниками є можливість вибирати і використовувати найкращі ресурси, знання і здібності з найменшими витратами часу. Основні конкурентні переваги це швидкість виконання замовлення, можливість зниження сукупних витрат, можливість повнішого задоволення потреб замовника.

До недоліків слід віднести недостатньо тісний особистісний контакт між працівниками.

3.7. Контрольні запитання

1. З якою метою проводяться ретроспективи спринту ?
2. Які питання розглядаються під час ретроспективи з спринту?
3. Яка роль Scrum-майстра в підготовці ретроспектива спринта?
4. Роз'ясніть алгоритм планування спринту при роботі з версіями інформаційного продукту.
5. Сформулюйте те як ви розумієте поняття якості інформаційного продукту.
6. Як ви розумієте поняття тестування інформаційного продукту.
7. Вкажіть заходи за допомогою яких можна прискорити та покращити тестування інформаційного продукту.
8. Які ви вбачаєте переваги і недоліки в роботі з географічно віддаленими командами?
9. Роз'ясніть які існують стратегії співпраці з географічно віддаленими командами ?
10. Які технічні і програмні засоби ви би застосували для організації співпраці з географічно віддаленими командами?

Тема 4. Розрахунок економічної ефективності інформаційного проекту.

В темі розглянуто сучасні методики обрахування економічного ефекту від впровадження інформаційного проекту.

Розрахунок економічної ефективності проекту, який застосовує технологію Scrum часто виконується ще до впровадження в експлуатацію інформаційної системи. Тому під час розрахунків оцінюється ефективність інвестиції в даний інформаційний проект порівняно з можливими інвестуваннями цих грошей в інші проекти, або у виробництво продукції. Інвестор або замовник розробки інформаційного проекту має бути впевненим в тому, що гроші які будуть вкладені в проект повернуться йому з прибутком швидше ніж при інвестиціях інші проекти.

Розрахунок економічної ефективності інформаційної системи це важливий елемент в розробці інформаційного продукту. Помилки на цьому етапі розробки коштуватимуть достатньо дорого. План розробки інформаційної системи вимагає щоб рішення про економічну ефективність були розглянути і прийняти у відповідний час і відповідній фазі розробки, навіть якщо ці рішення ґрунтуються на обмежених знаннях.

Методологія Scrum стверджує, що ми не повинні робити передчасні рішення, якщо навіть процес розробки диктує таку необхідність. Тому ми маємо прийняти рішення у відповідний момент якій показано на рис. 4.1. Якщо ми приймаємо передчасно відповідальні рішення то ми знаходимося на експоненційний частині графіку вартості прийняття рішення і помилка буде коштувати достатньо дорого. Тому рішення має бути прийняте тоді коли вартість не ухвалення рішення стане більшою ніж вартість прийняття рішення (рис. 4.1). Тому доцільно не поспішати і з’ясувати всі аспекти економічної діяльності підприємства, структури управління, ефективності бізнес процесів і вже на основі отрима-

них даних виконати економічно і науково обґрунтований розрахунок економічної ефективності інформаційної системи.

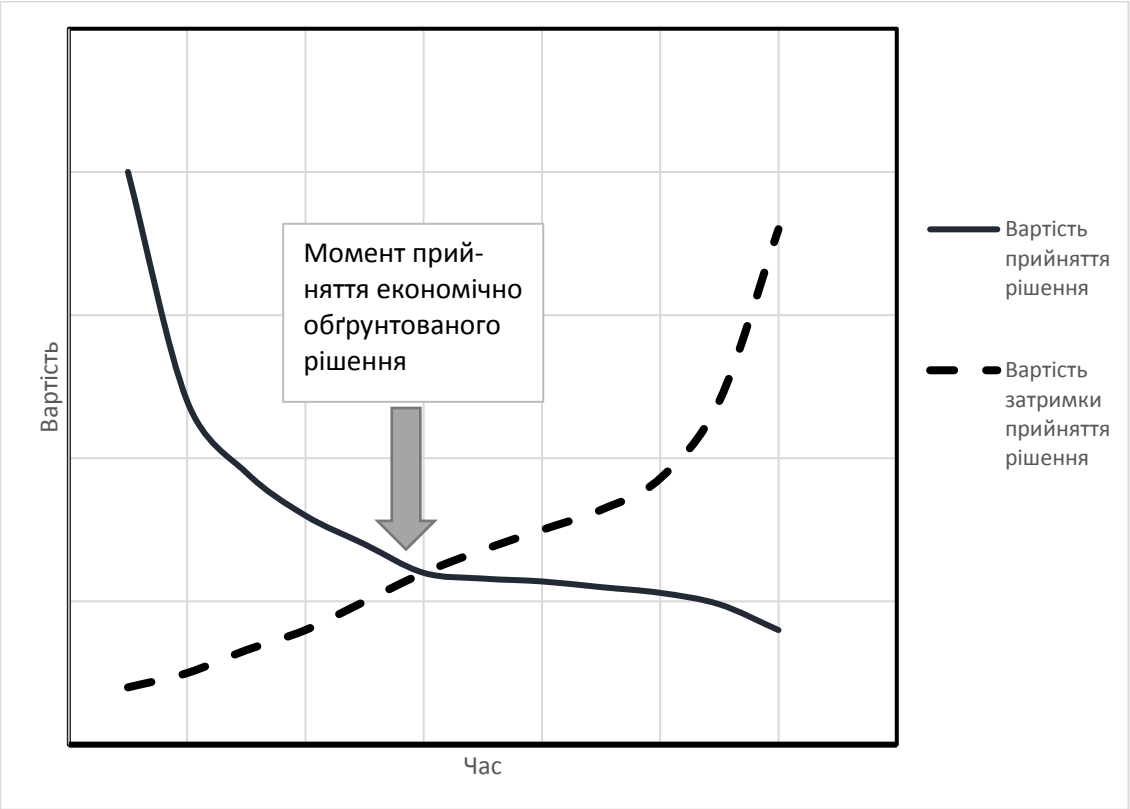


Рисунок 4.1. Вартість прийняття правильного рішення

Впровадження інформаційних систем на виробничих підприємствах забезпечує економічний ефект, який виражається в:

- Підвищення ефективності виробничих потужностей;
- Економії оборотних активів;
- Зниження виробничого браку;
- Зниженні товарно-матеріальних цінностей на складах;
- Зниження транспортно-заготівельних витрат;
- Скорочення витрат на адміністративно-управлінський апарат;

- Впровадженні наскрізного і своєчасного оперативно-виробничого планування та обліку виробництва, що дозволяє знизити втрати безпосередньо в цехах (втрата матеріалів, обсяг незавершеного виробництва, зменшення переналадок обладнання за рахунок обґрунтованого підвищення серійності, зниження простоїв, зменшення частки понаднормових робіт і т.п .);
- Зниження неврахованих недостач;
- Зниження необґрунтованої видачі матеріалів в цеху;
- Зниження витрат допоміжних матеріалів;
- Зниженні собівартості продукції за рахунок скорочення загальновиробничих і адміністративних витрат;
- Зниженні дебіторської заборгованості за рахунок всебічного, персоніфікує-ванного контролю;
- Зниженні необґрунтованих знижок на відвантажений товар;
- Скороченні часу виконання замовлень, при раціональнішому завантаженні устаткування;
- Зниженні обсягів непродуктивного праці;
- Зменшенні простоїв і часу зберігання матеріалів які підлягають обробці;
- Забезпеченні оперативності отримання та достовірності даних по руху матеріальних потоків, запасам і витратам матеріалів на всіх етапах виготовлення продукції;
- Вдосконаленні системи обліку та звітності, спрощення та впорядкування виробничого документообігу.

Разом з тим внесок інформаційної системи в загальний прибуток підприємства важко точно оцінити, тому що цей внесок здійснюється через обробку інформаційних потоків, вдосконалення документообігу і бізнес процесів на виробництві.

Тому доцільно за допомогою експертних оцінок визначити, яку долю в прибутку підприємства забезпечує інформаційна система. Для цього треба залучити фахівців інформаційного відділу, бухгалтерії, виробничого відділу, керівництво підприємства та інших підрозділів, які доцільно залучити з погляду на особливості цього підприємства. Практика показує що інформаційна система забезпечує від 0,5% до 5% від загального прибутку підприємства. Але ці цифри мають бути уточнені для конкретного замовника.

Для розрахунку доля інформаційної системи треба розрахувати показник cash-flow.

Cash flow (чистий грошовий потік) - це різниця між позитивним грошовим потоком (надходження грошових коштів) і негативним грошовим потоком (витрачання грошових коштів) в аналізованому періоді. В якості такого періоду використовують рік .

Приклад розрахунку cash-flow показано в таблиці 4.1. Цей показник розраховано за формулою Cash flow = ((рядок 1 + рядок 2 + рядок 3) - (рядок 4 + рядок 5 + рядок 6 + рядок 7 + рядок 8)).

Розрахунок Cash-flow Таблиця 4.

№	Показник	Результат за 2015, грн.	Результат за 2016, грн.
1	**Надходження від продажу продукції**	613379000	819267000
2	**Погашення дебіторської заборгованості**	30507000	0
3	**Інші грошові надходження від операційної діяльності**	155672000	320947000
4	**Собівартість продукції**	336288000	520539000
5	**Витрати на збут**	6949000	12429000
6	**Адміністративні витрати**	42789000	38189000
7	**Інші витрати у рамках операційної діяльності**	105794000	15820000
8	**Податки на прибуток та проценти за користування позиками**	32226000	35058000
9	**Cash flow**	307 738 000,00₴	553 237 000,₴

Приймемо, що частка інформаційні системи в прибутку підприємства дорівнює, наприклад, 1% і далі виконуємо розрахунок економічної ефективності системи. Для 2015року — 3077380 грн, для 2016року — 5532370 грн.

Першим показником в розрахунку економічної ефективності є чиста поточна вартість - NPV.

NPV (англ. Net present value, NPV), показує величину грошових коштів, яку інвестор очікує отримати від проекту, після того, як грошові надходження окуплять його початкові інвестиційні витрати і періодичні грошові витрати, пов'язані із здійсненням проекту. Розрахунок NPV робиться за формулою:

$$NPV = \sum_{t=1}^{n} \frac{\text{ЧГП}_t}{(1+i)^t} - IB_t$$

Де $-\text{ЧГП}_t$ чисті грошові потоки на t-м році розрахунку;

i – ставка дисконту;

IB – інвестиційні витрати;

t – поточний рік реалізації проекту;

n – тривалість проекту у роках.

Якщо NPV позитивна, то проект можна рекомендувати для фінансування. Якщо NPV дорівнює нулю, то надходжень від проекту вистачить лише для відновлення вкладеного капіталу. Якщо NPV менше нуля - проект збитковий і він відхиляється.

Для даних таб.4,1

$NPV = NPV_{2015}/(1+i) + NPV_{2016}/(1+i)^2 - IB$

нехай *i* =0,24 , інвестиції становлять 5000000 грн.

$NPV = 3077380/(1+0,24) + 5532370/(1+0,24)^2 - 5000000 = 6079813.47 - 5000000 = 1079813.47$

Таким чином NPV становить 1079813.47 грн. що є позитивного величиною, яка вказує на прибуток який буде отримано за рахунок впровадження інформаційної системи. Висновок по цьому показником інформаційна система прибуткова і має бути запроваджена.

Далі розраховуємо рентабельність інвестицій за допомогою коефіцієнта рентабельності.

Коефіцієнт рентабельності (ARR) - розраховується як відношення середнього рівня (середньорічні) чистого грошового потоку до інвестиційних витрат:

$$ARR = \frac{\sum_{t=1}^{n} ЧГП_t / n}{IB},$$

де n - кількість років, протягом яких реалізується проект.

ARR = ((3077380 + 5532370) / 2) / 5000000 = 0,861

Якщо коефіцієнт рентабельності виразити у відсотках то він складає 86.1%

Наступним показником ефективності інформаційної системи є період часу, за який система поверне гроші, які в неї були вкладені.

Період окупності (ПО) показує з якого моменту часу (року, місяця) проект починає приносити прибуток або через скільки років вкладені інвестиції повернуться (окупляться):

Період окупності розраховується за формулою

$$ПО = \frac{IB}{\sum_{t=1}^{n} ЧГП_t / n}.$$

Для нашого прикладу періоду окупності складає

ПО = 5000000/((3077380 + 5532370) / 2) = 1,16 року.

По виконаним розрахункам роблять висновок про доцільність впровадження і економічну ефективність інформаційної системи.

У випадку інформаційної системи, яка наведена у прикладі період окупності системи дорівнює 1,16 року, коефіцієнт рентабельності достатньо високий

0.861, чиста поточна вартість складає 1079813.47 грн. Висновок розрахунку: інформаційна система економічно вигідна, доцільно впроваджувати систему на підприємстві.

4.1. Контрольні запитання

1. Що має підтвердити розрахунок економічної ефективності інформаційної системи;
2. Назвіть основні чинники, за рахунок яких інформаційна система підвищує ефективність діяльності підприємства;
3. Маємо два підприємства А і В. Підприємство А протягом двох років було збитковим. Структура управління не ефективна, інформаційна система використовується для нарахування заробітної платні. Підприємство має високу собівартість продукції. Підприємство В протягом двох років показувало рентабельність на рівні 24%, інформаційна система використовується для управління виробництвом, але є застарілою. На якому підприємстві і чому доцільно впроваджувати інформаційний проект постанови впровадженню сучасної ERP системи?
4. Роз'яснить економічний зміст показника NPV (чиста поточна вартість)..
5. Роз'яснить економічний зміст і розрахунок коефіцієнту рентабельності інформаційної системи.
6. Роз'яснить економічний зміст і формули розрахунку періоду окупності інвестицій у інформаційну систему.
7. Розрахувати показники економічної ефективності інформаційної системи для підприємств. Вихідні дані: cash-flow за два роки становлять за перший рік — 5 млн.грн., за другий рік — 6 млн. грн. Інвестиції в запровадження інформаційної системи дорівнюють 8 млн. грн. Ставка дисконту — 0,25. Розрахувати чисту поточну вартість, коефіцієнт рентабельності , період окупності інвестицій.

Заключне слово

Scrum – є однією з найпопулярніших методологій гнучкої розробки інформаційних систем. Scrum акцентує свою увагу на якісному контролі процесу розробки.

Суть методології полягає в хорошій самоорганізованості команди, яка постійно взаємодіє із замовником, випускаючи інформаційний продукт на кожній ітерації розробки. В цьому підході є такі ролі: Scrum майстер, Власник продукту, Команда розробників. Також у Scrum використовуються такі поняття: спринт, журнал продукту, журнал спринту, діаграма виконання задач, планування спринту, щоденна нарада, огляд підсумків спринту і ретроспективна нарада.

Схема Scrum має вигляд ітераційного процесу. Одна ітерація має назву спринт (sprint). Послідовність спринтів складає ітераційний процес Scrum. Ітерація (спринт) у Scrum триває від 2 до 4 тижнів. Методологія добре підходить для невеликих і середніх команд розробників, що працюють в умовах невизначених умов і завдань, які часто змінюються. В команді розробників повинно бути добре розвинене спілкування, як між членами команди, так і з замовником. Команда має бути автономною — її члени і тільки вони вирішують, яким чином досягти поставлених цілей. Всередині неї панує атмосфера взаємного збагачення ідеями, досвідом.

Планування Scrum виконується покроково. Для підвищення ефективності роботи доцільно використовувати дошку завдань яка є елементом методології Канбан.

Оцінка витрат часу на виконання завдань виконується за допомогою розрахунків story point і фокус-фактору . Оцінка витрат часу може доповнюватися методологією яка має назву planning poker.

Ретроспективні наради дозволяють визначити і усунути недоліки в організації Scrum підвищити ефективність роботи команди.

Найбільш ефективно команда працює тоді, коли співробітники знаходиться в одному приміщенні, але сучасні засоби обміну інформацією дозволяють добре організувати роботу географічно віддалених команд.

Scrum є універсальною методологією, яка застосовується не тільки для розробки інформаційних і програмних продуктів, але і в проектах інших галузей промисловості.

Scrum має привабливі переваги. Scrum орієнтований на клієнта, адаптивний. Scrum дає клієнту можливість робити зміни у вимогах в будь-який момент часу (але не гарантує того, що ці зміни будуть виконані). Можливість зміни вимог приваблива для багатьох замовників.

Scrum досить простий у вивченні, дозволяє економити час за рахунок відтермінування некритичних робот. Scrum дозволяє отримати потенційно робочий продукт наприкінці кожного спринту.

Scrum призначений для команди, яка здатна вирішити необхідні завдання з мінімальною координацією роботи членів команди. Це особливо привабливо для малих компаній і стартапів, тому що виключає необхідність найму і спеціалізованого навчання персоналу та керівників.

Scrum має недоліки. В Scrum не створюють план реагування на ризики. Таким чином, унеможливлюють формальну (юридичну або адміністративну) протидію порушенням правил Scrum. Використання багатофункціональної команди дозволяє знизити витрати на координацію, але це приз-

водить до підвищення витрат на відбір персоналу, його мотивацію, навчання. За певних умов на ринку праці, формування повноцінної, ефективної Scrum команди може бути неможливим.

Розрахунок економічної ефективності інформаційної системи це важливий елемент в розробці інформаційного продукту. Під час розрахунків оцінюється ефективність інвестиції в даний інформаційний проект порівняно з можливими інвестуваннями цих грошей в інші проекти, або у виробництво продукції. Інвестор або замовник розробки інформаційного проекту має бути впевненим в тому, що гроші які будуть вкладені в проект повернуться йому з прибутком швидше ніж при інвестиціях інші проекти. Помилки на цьому етапі розробки коштуватимуть достатньо дорого. План розробки інформаційної системи вимагає, щоб рішення про економічну ефективність були розглянути і прийняти у відповідний час і відповідній фазі розробки, навіть якщо ці рішення ґрунтуються на обмежених знаннях. Тому рішення має бути прийняте тоді коли вартість не ухвалення рішення стане більшою ніж вартість прийняття рішення. Доцільно не поспішати і з'ясувати всі аспекти економічної діяльності підприємства, структури управління, ефективності бізнес процесів і вже на основі отриманих даних виконати економічно і науково обґрунтований розрахунок економічної ефективності інформаційної системи. Основні формули розрахунку і приклади надано в четвертому розділі посібника які дозволяють виконати економічно і науково обґрунтований розрахунок економічної ефективності інформаційної системи.

Перелік рекомендованої та використаної літератури

1. Takeuchi, Hirotaka and Nonaka, Ikujiro. 1986. The new new product development game. Harvard Business Review 64:1:137-146 (Jan/Feb), reprint no. 86116.
2. *Agile Project Management with Scrum*, Ken Schwaber, Microsoft Press, January 2004, 163pp,
3. Кон, Майк. Scrum: гибкая разработка ПО. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 576 с.
4. Демиденко М. А. Управління проектами інформатизації / Михайло Андрійович Демиденко. – Дніпропетровськ: Національний гірничий університет, 2014. – 108 с. – (НГУ)
5. Демиденко М. А. Системи підтримки прийняття рішень / Михайло Андрійович Демиденко. – Дніпропетровськ: Національний гірничий університет, 2016. – 106 с. – (НГУ).
6. Donald G. Reinertsen "Managing the Design Factory" Free Press; First Edition, 1997
7. Mary Poppendieck, Tom Poppendieck "Implementing Lean Software Development" Pearson Education, 2007.
8. Mike Cohn "Agile Estimating and Planning" -Pearson Education, 2005.
9. Джоэл Спольски "Джоэл о программировании"- Символ-Плюс, 2006.
10. Mary Poppendieck, Tom Poppendieck "Lean Software Development. An Agile Toolkit" - Addison-Wesley, 2003.
11. Barry Boehm, Richard Turner, "Balancing Agility and Discipline"- Addison-Wesley Professional, 2003.
12. Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum" - Upper Saddle River, NJ ,2003.
13. Фредерик Брукс "Мифический человеко-месяц" - Символ-Плюс, 2000.

14. Ken Schwaber, "Agile Project Management with Scrum" - Microsoft Press; 1 edition, 2004
15. Кент Бек "Экстремальное программирование"- Питер,2006
16. Том ДеМарко, Тимоти Листер "Человеческий фактор" - Символ-Плюс, 2006.
17. Х. Книберг и М. Скарин “Kanban и Scrum: выжимаем максимум” - InfoQ.com, 2010
18. H. Kniberg “Lean from the Trenches Managing arge-Scale Projects with Kanban” - 2010
19. Schwaber, Ken. *Agile project management with Scrum*. Microsoft press, 2004.
20. Kniberg, Henrik. *Scrum and XP from the Trenches*. Lulu. com, 2015.
21. Руководство к Своду знаний по управлению проектами (PMBOKGuide, 5 edition), PMI, 2013.
22. Practice Standard for Project Risk Management, PMI, 2009.
23. Рассел Д.Арчибальд, Управление высокотехнологичными программами и проектами, АйТи, ДМК Пресс, 2010
24. Моделювання економіки: навч. пос. / А.С. Корхін, І.Ю Турчанінова, – М-во освіти і науки України, Держ. вищ. навч. заклад «Нац. гірн. ун-т». – Д. : ДВНЗ «НГУ», 2016. – 104 с. http://ir.nmu.org.ua/handle/123456789/149850
25. Методичні вказівки до лабораторних робіт з дисципліни "Моделювання економіки"/ А.С. Корхін, І.Ю Турчанінова, – М-во освіти і науки України, Держ. вищ. навч. заклад «Нац. гірн. ун-т». – Д. : ДВНЗ «НГУ», 2015. – 104 с.
26. Mykhailo Demydenko Research of implementation of educational module" e-commerce and web-marketing" for economics students/ Использование междисциплинарных исследований с целью внедрения новых учебных программ/модулей и/или новых учебных методик в области электронной коммерции : материалы междунар. конф. проекта «ECOMMIS» TEMPUS, 2–4 апр., 2012 г., г. Берлин.–Берлин: Берлинский технический университет; ECM-Office; Д.: Национальный горный университет

Предметний покажчик

Державний вищий навчальний заклад
«Національний гірничий університет»
Витяг з протоколу № 20
засідання Вченої ради університету

«26» грудня 2017 року

Склад Вченої ради затверджений рішенням
Конференції трудового колективу від 26.10.2015 р.
(протокол № 2)
На засіданні були присутні 64 з 90 членів Вченої ради.

Розгляд результатів експертизи рукопису навчального посібника «Управління проектами інформатизації за методологією Scrum» (автор: Демиденко М.А.) на предмет надання грифу «Рекомендовано вченою радою як посібник».

СЛУХАЛИ: директора науково-методичного центру НГУ, канд. техн. наук, професор кафедри транспортних систем і технологій Салова В.О., який відзначив, що:
1. На розгляд подано матеріали:
– рукопису навчального посібника «Управління проектами інформатизації за методологією Scrum»;
– дві рецензії фахівців у сфері управління проектами;
– програму дисципліни «Управління проектами інформатизації»;
– обґрунтування доцільності видання начального посібника.
2. Рецензії на навчальний посібник «Управління проектами інформатизації за методологією Scrum» позитивні.
3. Експертизою встановлено:
– зміст навчальної книги синтезує та узагальнює відомості про управління проектами інформатизації за методологією SCRUM і відповідає освітньо-професійній програмі підготовки бакалаврів з економіки та програмі дисципліни «Управління проектами інформатизації»;
– за побудовою, методами структурування навчального матеріалу, дидактичним опрацюванням змісту, обсягом навчального посібника відповідає чинним вимогам;
– навчальний посібник «Управління проектами інформатизації за методологією Scrum» апробований в начальному процесі протягом п'яти років, що є підставою для надання навчальному посібнику «Управління проектами інформатизації за методологією Scrum» грифу «Рекомендовано вченою радою як посібник».

УХВАЛИЛИ: надати навчальному посібнику «Управління проектами інформатизації за методологією Scrum» (автор: Демиденко М.А.) гриф «Рекомендовано вченою радою як навчальний посібник для бакалаврів та магістрів спеціальностей 6.051, 8.051 Економіка.

Вчений секретар О.А. Данилова

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ І.К. 02070743

Навчальне видання

Демиденко Михайло Андрійович,

УПРАВЛІННЯ ПРОЕКТАМИ ІНФОРМАТИЗАЦІЇ ЗА МЕТОДОЛОГІЄЮ SCRUM

Навчальний посібник

У редакції автора

Підписано до друку . Формат 30 × 42/4.
Папір офсетний. Ризографія. Авт. Арк. 2,1.
Обл.–вид. арк. 13,7. Тираж 100 прим. Зам._№ 96/12.

Підготовлено до друку та видруковано
у Державному вищому навчальному закладі "Національний гірничий університет”
Свідоцтво про внесення до Державного реєстру ДК No 1842.
49600, м. Дніпро, просп. Д. Яворницького, 19.