**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ**

***Циклова комісія програмування***

**Реферат**

**з курсу «Основи інформатики, програмування та алгоритмічні мови»**

**На тему: “Scrum-підхід до організації процесу розробки програмного забезпечення”**

***Лук’яненко Дар’я Владиславівна***

**Студентів групи: 2П-19**

**Викладач Марченко С. В.**

**Черкаси-2020**

**Зміст:**

**1. Вступ**

**2.Методологія Scrum**

**3. Висновок**

**4. Література**

**1.Вступ**

Планування та реалізація проекту - досить складаний процес. Поза сумнівами, невід'ємна частина випуску успішного програмного продукту - грамотна організація робочого процесу. Застарілий метод - нескінченні наради і доповіді - тільки посилює проблему, так як відволікає від ведення головної справи. Спростити роботу над проектом допоможуть онлайн інструменти. Зокрема так звані менеджери задач. Вони покликані допомагати у вирішенні подібних сучасних проблем. Сьогодні існує ряд нових сфер бізнесу, що характеризуються коротким циклом розробки і виведення продукції на ринок, орієнтацією не на функціонал, а на потреби споживача. Тому все більше фахівців звертають свою увагу на технологію під назвою Scrum. В даний час, Scrum є однією з найбільш популярних методологій гнучкої розробки програмних засобів [1].

**2. Методологія Scrum**

Scrum, найбільш широко використовуваний Agile процес, успішно використовується в розробці програмного забезпечення протягом останніх 20 років. Хоча Scrum в основному практикується в комерційному програмному середовищі, методологія успішно застосовується до освіти, виробництва та іншої промисловості. Як і інші методи Lean і Agile, Scrum оптимізує обмежені ресурси і підвищує ефективність. Scrum також дає змогу командам самоорганізуватися і працювати високому темпі, вільному від зовнішнього втручання. Це допомагає командам розкривати свій повний потенціал і звільняє керівництво від фокусу на повсякденне управління. Багато додаткових шарів Agile практик і моделей були розроблені за останні роки, які дозволяють Scrum вирішувати неминучі складності реального життя. Scrum ділить складну роботу на прості задачі, великі організації в невеликі команди та далекосяжні проекти в серію коротких. Коли складна робота ділиться на прості задачі, то простіше точно визначити, що потрібно зробити. За допомогою чіткої дорожньої карти команда може розпочати роботу негайно, знати, які предмети потрібно спільно опрацювати, і зрозуміти, коли така робота має бути завершена. Невеликі частини роботи, як і їжа, легше жувати, ковтати і переварити. Розділяючи великі організації на невеликі групи, Scrum дозволяє організаціям функціонувати як маленькі. Малі команди допомагають підтримувати фокус, тому що менше часу витрачається на спілкування про деталі. Координація набагато легше для команди з п'яти людей, ніж це для команди з 20. Принцип розподілу та підкорення також працює при розбиранні великих проектів на серію спринтів адже легше планувати на короткий проміжок часую. Написання розділу в той час легше, ніж писати цілий роман відразу. Цей ітераційний процес також дозволяє автору щоб застосувати те, що він навчився писати перший розділ, щоб написати наступний. Коли робота ділиться на прості задачі, її можна закінчити за коротший період часу. Прискорюючи процес розробки і отримуючи функціонуючий продукт, команда може зібрати відгуки від людей, які його використовуватимуть, швидше ніж це було раніше. Зворотній зв'язок допомагає команді поліпшити продукт, не тільки так, як вони запланували під час проектування, а й також по відгукам від людей, які взаємодіють із ним. Це як після написання першого розділу дати книгу редакторам і отримати зворотній зв'язок. Після цього автор може внести корективи. Без такого відгуку автор може попрямувати у неправильному. Якщо не виявити помилку у першому розділі то доведеться переписати цілу книгу, а не просто першу главу. З своєчасним зворотним зв'язком автор здатний внести незначні зміни до начерків книги, уникаючи масових обсягів переробки. Прозорість Scrum робить всю роботу прозорою. Немає секретів що потрібно робити, хто це робить, і як це відбувається. Коли вся команда озброєна цією інформацією, вона може вирішувати проблеми коли вони виникають і коректувати курс у найкоротші терміни. Зацікавлені сторони та керівництво зможе приймати більш обгрунтовані стратегічні рішення, коли воно має чесне і чітке уявлення про те, як проходить розробка проекту. Для досягнення прозорості Scrum використовує безліч інструментів для відстеження активності і довжини спринту. Це дозволяє легко відповідати на різні питання. Чи команда за встановлені терміни завершить всю роботу? Де є проблеми і чому?

Scrum також використовує ряд зустрічей, відомих як мітинги, для збільшення прозорості. Ці мітинги відбуваються в регулярній каденції і призначені для полегшення командної комунікації. Спільні інструменти роблять статус будь-якого конкретного завдання якісну інформацію про те, як задача або діяльність виконується. Короткий огляд мітингу:

· Спринт-планування - це коли команда вирішує, що потрібно зробити в

майбутньому спринті і скільки це може зайняти часу.

· Щоденний скрам-мітинг - коли команда торкається бази задач, щоб уникнути

будь-яких перешкод які впливають на спринт. Це також допомагає команді залишатися правильно орієнтованою.

· Огляд спринтів, який має місце наприкінці спринту щоб зібрати дієвий відгук

про те, що команда завершила цей спринт і надати можливість перевірити загальний план продукту[14].

· Ретроспектива спринту допомагає команді перевірити та адаптувати її процес,

запитуючи, що виходить добре під час спринтів і що може бути краще.У Scrum використовуються два загальні інструменти, три ролі та чотири зустрічі у звичайному режимі за період часу який називається спринтом. Ця структура сприяє прозорості та насиченість зв'язку, яка разом дозволяє групі регулярно перевіряти себе і адаптуватися до змін.

Сприт традиційно коливається від одного до чотирьох тижнів, і є просто ітерацією часу, достатнього для того, щоб команда мала суттєвий приріст продукту, але досить короткий, щоб тримати його в центрі уваги. Найважливішою загальною проблемою Scrum є відставання продукту. Простіше кажучи, невідповідність продукту - це список всього, що може бути зроблено за час життя проекту. Невідповідність продукту поділяється і визначається на елементи, які можуть бути встановлені незалежно один від одного. Щоб знову використати приклад книжки, кожен розділ повинен мати можливість писатися незалежно від наступного розділу. Можливість створити проект з незалежних шматочків дає можливість перевіряти та адаптувати план розробки продукту власника. Елементи з високою ділової цінністю розміщені вгорі, елементи з низьким значенням бізнесу виявляються внизу. Як загальне правило 80% вартості товару кінцевому користувачеві складає 20% його функцій. Замовлення розробки продукту відповідно до вимог бізнесу гарантує найбільш необхідні функції побудовані спочатку. Після того як перший набір функцій був випущений, кінцевий користувач дає відгук власнику продукту про те яку додаткову інформацію, функції включити в наступну ітерацію продукту. Елементи у верхній частині списку розробляють спочатку, тому вони повинні бути чітко визначені. У Scrum будь-який елемент продукту відповідає визначенню готовності, якщо команда ретельно обговорила це питання і має чітке уявлення про те, що буде потрібно.

Елементи, що знаходяться нижче в списку, менш визначені, оскільки розуміння цих предметів буде розвиватися разом з проектом або команда дізнається про них пізніше. Обмеження допомагає команді підтримувати фокус, обмежуючи переривання та зміни у середині спринтів. Задачі останнього спринта переміщується на дошку scrum. Традиційна дошка має три колонки: зробити, в роботі і виконано. Кожна задача спочатку поміщається в колонку "Зробити ". Розробник бере задачу з стовпеця "Зробити" і перемістить його у другий стовпець "В роботі ". Коли завдання завершено, елемент перенесено до третього та останнього стовпчика "Виконано ". Дошки Scrum можуть бути фізичними або цифровими, але важливо, щоб вони були видимими всій команді, щоб статус спринтів можна було побачити в режимі реального часу. Це дозволяє команді змінить курс, якщо задача у спринті відставання знаходиться під загрозою не бути завершеною.

Дошка Scrum дозволяє команді не тільки контролювати статус спринтів, але і відстежувати швидкість команди. У Scrum ми вимірюємо швидкість команди в кожному спринті використовуючи метрику, відому як швидкість. Швидкість вимірюється шляхом присвоєння кожному елементу відносної точки значення. В кінці спринту, бали за всі елементи у стовпці "Готово" додаються разом. Це загальна швидкість команди для спринту. Встановлення чисельного значення суми роботи команди робиться кожен спринт і перевіряється і адаптується кількома способами. По-перше, швидкість створює базовий рівень продуктивності команди. З цієї базової лінії команда здатна емпірично визначити і розраховувати скільки роботи вона може розраховувати зробити у майбутніх спринтах. Наприклад, якщо команда заробила 50 балів за роботи в останньому сприті, то швидше за все може заробити принаймні 50 балів у наступному спринті. Базова лінія допомагає команді уникнути надмірної роботи в спринті та / або форсування. По-друге, швидкість дозволяє команді бачити, чи процес зміни допоміг чи нашкодив. Наприклад, якщо команда не змогла завершити певну задачу декілька спринтерів поспіль, вони можуть вирішити, що на нього потрібно планувати двоє люде. Якщо процес пройде успішно, команда завершить свою діяльність і його швидкість підніметься. Якщо менше стає зроблено, тому що до однієї задачі прив'язано дві людини, то швидкість знизиться. Цей цикл перевірки та адаптації базується на реальних даних які допомагають команді постійно вдосконалюватися, зберігаючи лише ті зміни, які мали позитивний ефект на швидкість[15]. Нарешті, швидкість дозволяє команді прогнозувати дату завершення проекту. Якщо команда знає, скільки роботи вона може зробити кожен спринт, вона може виміряти для загальної ситуації точну дату завершення розробки продукту. Отже, якщо команда має швидкість 50 балів за спринт, а відставання продукту - 500 балів робота, команда може розумно розраховувати завершити проект у 10 спринтах. Якщо спринт триває два тижні, команда може прогнозувати завершення проекту у 20 тижні.

Команда - це всі, хто працюють над завершенням роботи продукту. Вона в ідеалі складається від трьох до дев'яти чоловік, включаючи майстра спринта, продукт-менеджера і основну команду розробників. Дуже важливо, щоб команда включала всі навички необхідні для завершення продукту. У Scrum це називається крос-функціональними командами. Крім того, у команді може бути щонайменше дві людини про запас. Це допомагає усунути залежність на певну кваліфікацію члена команди, якщо цей співробітник хворіє, продовжує відпустку або переходить на іншу роботу.

У Scrum немає менеджера команди. Замість цього команда організовує та керує сама по собою з дуже високим ступенем автономії і підзвітності. Команда, з вказівки скрам-мастер, вирішує, що може досягти в спринті і як це можна зробити. Скрам-мастер є тренером, а не босом. Він або вона не має повноважень над командою. Швидше за все, скрам-мастер слідкує, щоб команда працювала краще, допомагаючи виявити та усунути перешкоди. Перешкодою є щось таке що уповільнює або перешкоджає команді завершити спринт. Скрам-мастер також переконається, що команда добре спілкується і використовує різні методики для досягнення найбільшого ефекту. Це приводить нас до власника продукту. Власником продукту є член команди, відповідальний за комерційний успіх продукту. Власник продукту повинен знати, що хоче клієнт, і відносну ділову цінність цих бажань. Він або вона переводить ці бажання у список задач продукту відповідно до вартості бізнесу (з найціннішими позиціями зверху). Для успіх продукту, власник продукту повинен контролювати порядок списку задач продукту. Тим не менш, команда несе відповідальність за спринт. З цієї причини власник продукту повинен витратити близько половини свого часу, працюючи з командою, переконатися, що робота в спринті проходить без проблем. Іншу половину часу власник продукту повинен проводити за консультацією з клієнтами та зацікавленими особами, щоб переконатись, що задачі в спринтах відображають їхні потреби. Scrum починається з бачення продукту. Після того, як завдання продукту було встановлено, команда може почати спринт. Щоб почати спринт, команда спочатку повинна провести спринт-планування. Планування спринтів повинно обмежуватися не більше ніж двома годинами на тиждень. Ідея планування спринтерів полягає в тому, щоб провести одну всеосяжну зустріч, яка покаже що потрібно зробити до кінця спринта. Мітинг починається з того команда говорить стільки предметів з верхньої частини списку задач продукту вона може закінчити в спринті.

Команда повинна переконатися в тому, що всі задачі які вносяться у список задач спринта під час планування спринту чітко визначені і готові до роботи. Це фундаментальна концепція Scrum відома як визначення задачі. Елемент в списку задач має бути незалежним, дієвим, мати точне значення і чітке визначення критеріїв. Це, в свою чергу, називається визначенням задачі, що гарантує, що кожен точно знає що очікується від пункту завдання.

Команда створила спринтерську мету після того, як буде сформовано список задач у спринті. Ця мета повинна сформулювати цілі високого рівня задач, що знаходяться у спринті. Якщо простіше, ціль забезпечує контекст для чого команда працює спринті. Мета може бути настільки простою, як досягнення певного рівня функціональності. Наприклад до кінця цього сприту команда продемонструє, як програма може автоматично зберігати електронні адреси.

Після планування спринту команда приходить на роботу і зустрічається кожен день для щоденного скрам-мітингу. Всі члени команди, що працюють над задачами у спринті, повинні бути на щоденному мітингу і учасники повинні стояти, щоб зустріч була недовгою(не більше ніж 15 хвилин). Під час щоденного скарму кожен член команди відповідає на три запитання:

· Що я зробив вчора, щоб допомогти команді досягти цілі спрінта?

· Що я буду робити сьогодні, щоб допомогти команді досягти мети спрінта?

· Чи бачу я будь-яку перешкоду, яка перешкоджає мені чи команді завершити спрінт?

Щоденний скрам не є звітом про статус спринта. Ця процедура вирівнює команду і допомагає підтримувати відкрите спілкування. Це можливість інформувати команду про те, що кожен робить. Наприклад, якщо у когось є ідея або пропозиція зробити задачу швидше, воін можуть поділитися ним з командою на щоденному мітингу. Якщо перешкода виникає під час щоденного скарму, яка є занадто великою щоб вирішити її під час зустрічі, команда повинна координуватися поза засіданням. Скрам-майстер - це член команди, відповідальна за видалення перепон у роботі. На додаток, до щоденного мітингу команда повинна витратити 5-10% свого часу і привести в порядок задачі на дошці. Це називається уточнення задач. Це не офіційна частина скрам-мітингу, але це хороша практика[16]. Наприкінці кожного спринта команда запрошує зацікавлених сторін та клієнтів до демонстрація того, що вона закінчила. Ця церемонія, яка називається спринтерський огляд, є розроблена для того, щоб викликати зворотний зв'язок із зацікавленими сторонами та клієнтами, які можуть вказати на зауваження які команда може потім включити в список задач продукту. Демонстрація виробляє розмову між командою та зацікавленими сторонами про те, як зробити продукт кращим. Після огляду спринта команда збирається на заключну церемонію спринтів. Ретроспектива - це можливість оглядати та адаптувати команду і її процеси. Настав час подумати про те, як покращити роботу процесу і як можна збільшити швидкість. Скрип-майстер і команда повинна бути присутньою. Власнику продукту теж бажано бути присутнім, але не обов'язковий. Є багато способів вести ретроспективи, але незалежно від форми добра ретроспектива включатиме тверді істини, чесність та дебати. Поширений шлях структура ретроспективи спринтів полягає у тому, щоб кожен член команди відповідав наступним чином на питання:

· Що було добре?

· Що було б краще?

· Які речі ми можемо поліпшити у майбутньому спринті?

Обговорення того, що могло бути краще, часто призводить до аналізу. Консенсус щодо того, яке поліпшення зробити в наступному спринті зазвичай починає з'являтися. Зміна одного процесу — це додане до наступного спринта покращення. На наступному рецензуванні сприту питання знову переглядається, щоб побачити, яка різниця, якщо така є то знову відбуваються зміни в процесі. Якщо продуктивність команди покращиться, зміна має бути збережена. Якщо виконання не відбулося, команда повинна розробити іншу гіпотезу, що може бути першопричиною проблеми. Важливо просто внести одну зміну. Якщо команда намагається внести кілька змін, буде складно з'ясувати, який процес зміни вирішив реальне питання і призвів до підвищення продуктивності. Якщо команда здатна послідовно мати сильну ретроспективу, яка виробляє відповідне вдосконалення процесу, яке побудує спринт для спринтів, воно буде постійно поліпшувати. Постійне вдосконалення є кінцевою метою Scrum.

**3. Висновки**

Отже, Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — *Scrum Master*, той хто опікується процесами, веде їх і працює як керівник проєкту, *Власник Продукту*, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та *Команду*, яка включає розробників.

Протягом кожного *спринту*, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву *product backlog* ([документація](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D1%96%D1%8F) запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (*backlog items*), що визначені протягом *наради з планування спринту* (*sprint planning meeting*), переміщуються в етап 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 (*requirements*) протягом спринту.

**4.Список використаної літератури**

1. Система керування проектами за методологією Scrum на платформі SharePoint: матеріали науково-практичної конференції "Сучасні проблеми наукового забезпечення енергетики", (Київ, 24 – 27 квітня 2018 р.)/Київ. держ. ун-т ім І. Сікорського, 2018. — 139 с.

2. Introduction to Recommender Systems in 2018 [Електронний ресурс]. — 2018. — Режим доступу до ресурсу:<https://tryolabs.com/blog/introduction-to-recommender-systems/>.

3. Рекомендательные системы: что это? [Електронний ресурс]. — 2015. — Режим доступу до ресурсу:<https://lpgenerator.ru/blog/2015/12/25/rekomendatelnye-sistemy-chto-eto/>.

4. A Machine Learning Approach to Twitter User Classification [Електронний ресурс//Yahoo! Labs. — 2011. — Режим доступу до ресурсу: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2886/3262.

5. Effective Classifier for User's Behavioral Profile Classification [Електронний ресурс]//Institute of Engineering Education & Research, University of Pune, Maharashtra, India. — 2014. — Режим доступу до ресурсу: https://pdfs.semanticscholar.org/64bc/0dc483758cc553a00ec96c169174304632d2.pdf

6. Performance Metrics for Classification problems in Machine Learning [Електронний ресурс] // Medium. — 2017. — Режим доступу до ресурсу: https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b.

7. Баженов Д. Оценка классификатора (точность, полнота, F-мера) [Електронний ресурс] / Д. Баженов. — 2012. — Режим доступу до ресурсу: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html.

8. Bing L. Integrating Classification and Association Rule Mining [Електронний ресурс] / L. Bing, H. Wynne, M. Yiming // AAAI. — 1998. — Режим доступу до ресурсу: http://www.aaai.org/Papers/KDD/1998/KDD98-012.pd**f.**