**Тема 3. Моделі розробки проектів на основі Agile**

**1. Основні моделі розробки ПЗ**

**2. Scrum для проектів**

**1. Основні моделі розробки ПЗ**

Модель розробки програмного забезпечення Agile була призначена головним чином для того, щоб допомогти розробникам створити проект, який може швидко адаптуватися до перетворення запитів. І підходить для проектів з високим ступенем невизначеності.

Agile – продукт будується інкрементально, тобто замість початкового жорсткого планування створення і випуску всього продукту, одразу невеликі цінні інкременти продукту плануються та випускаються поступово. Інкремент – операція в певному циклі, який може бути придатним до використання клієнтом-замовником. Інкрементальну модель можна розглядати як тривимірне представлення моделі водоспаду (рис.1.)



Рис.1. Інкримінаційна модель

Agile-проекти ітераційні, тобто операції можуть багаторазово повторюватись, а ефективність досягається за рахунок жорсткого зворотного зв’язку. У ітеративній моделі ітераційний процес почи­нається з простої реалізації невеликого набору про­грамних вимог і ітераційно посилює нові версії, по­ки не буде реалізована і готова до розгортання повна система (рис.2).



Рис.2. Ітераційна модель

Використання ітераційної моделі знижує ризики глобального провалу і розтрати всього бюджету, отримання несинхронізованих очікувань і помилкового розуміння процесів як клієнтом, так і кожним учасником команди розробки. Воно також дає можливість завершення розробки в кінці будь-якої ітерації (в каскадної моделі ви повинні перш за завершити всі етапи).

Сукупність практик створюють методологію Agile, наприклад:

1. **Екстремальне програмування XP**
2. **Scrum**
3. **Lean**
4. **Kanban**

**1. Екстремальне програмування XP** – використовує практики, що базуються на максимальному задоволення будь-якого клієнта, в основі яких покладено п’ять аспектів розвитку продукту та його постійного покращення:

1. Комунікації
2. Простота
3. Зворотній зв'язок
4. Повага
5. Мужність.

Роль усіх часників проекту є цінною та рівнозначною. Дана практика співвідноситься із практикою Scrum і полягає у постійній неперервній взаємозалежній командній співпраці над продуктом, а також на тестовій розробці кожної функції. Всі помилки та успіхи – це колективна відповідальність. Всі розробники в команді є взаємозамінними і мають набір компетентностей. Команда розробників є невеликою.

Планування є простим та адаптивним, на невеликі проміжки часу. Якщо ці проміжки часу є дуже малими, то процес вважається екстремальним, оскільки часу на виконання обмаль, а команда працює понаднормово із залученням до процесу замовника.

<https://coursera.org/share/d68b0c7dd7f3d2a9955b50936703de96>

<https://coursera.org/share/ca1d164ba61cf9d0b25e3f8358946073>

**2. Scrum**

Основні підходи практики:

Прозорість – усі можуть слідкувати за всіма частинами проекту, як команда так і стейкхолдери;

Інспекція – часта перевірка робочих продуктів та прогресу для виявлення відхилень;

Адаптація – адаптація до змін та усунення недоліків.

Всі процеси розробляються «власником продукту» і всі зміни відбуваються через нього. Команда розробників – самоорганізуюча група фахівців, тобто самостійно визначають способи досягнення цілей продукту, вона є крос-функціональною, тобто складається із фахівців різного профілю, які не мають ролі, статуси і посади, усі розробники. Команди невеликі.

Скрам-майстер – слідкує за дотримання основних підходів, методик та практик Scrum у команді, сприяє ефективній співпраці до дотримання продуктивності, проводить комунікаційні заходи всередині команди

Спринт – фаза розробки робочого прототипу продукту (приблизно 1-2 тижні)

Готовим робочим прототипом можна назвати продукт, який має закодовану функцію та протестовану та задокументовану.

<https://coursera.org/share/4fa587c3e51587ffe042a1a381984573>

**3.Lean**

Технологія зниження ризиків проекту та швидкого створення високоякісного продукту. Основні принципи:

1. Усунення відходів або ліквідація відходів, тобто те, що не додає цінності продукту вважається відходами і видаляється з процесу розробки (непостійні люди на постійній основі, зайві переговори, зустрічі, заві документи і т.д.).

2. Посилення навчання – вивчення альтернатив однієї ідеї, а потім дії, тобто вибір найкращого підходу до створення функцій відповідно до потреб клієнта. Також передбачає інтеграцію та тестування кожного випуску продукту в межах коротких ітерацій та демонстрації їх клієнту.

3. Швидке забезпечення при пізньому рішенні – отримання максимально більше інформації про продукт на основі розробки постійних оновлень та альтернатив у погодженні із замовником, не погоджуючись на першій версії продукту, тобто прийняття рішення тільки після розгляду усіх можливих альтернатив, навіть якщо клієнт погодив перший варіант продукту. Цей процес забезпечується у короткі ітерації.

4. Розширення можливостей команди – надання можливості професійній команді самостійно представити рішення клієнту та обирати способи його досягнення.

5. Побудова якості. – побудова тестів коду за умови надобності, а зосередження здійснюється вирішенні фактичної проблеми, яку повинен вирішити код. Запровадження принципу парного програмування, тобто робота в команді над кодом, створення якісної документації, автоматизоване тестування, зменшення часу на виправлення помилок.

6. Стратегічне бачення, бачення цілого. Чітке розуміння кінцевого продукту і його бачення з точки зору клієнта. Розробники мають бачити картину в цілому.

### Принципи Lean

### <https://coursera.org/share/05515fdccb047211e7253fd6ecbbd350>

**4.Kanban**

Це методологія, яка застосовується на принципах ощадливого виробництва. Це система поставки, при якій постачання відбувається тільки відповідно по потребі, тобто першому закінченні продукту в партії, замовляється наступна партія.



Кожна фаза може мати різний цикл створення.

**Сутність Кабан**

<https://coursera.org/share/b6d095805dcb37f9d7a333de13fee13b>

**2. Scrum для проектів**

[Scrum](https://tim.com.ua/2009/06/scrum-na-prostom-yazyke/) — це практика управління проєктами, дуже популярна серед програмістів. Вона виникла, коли стало зрозуміло, що традиційний метод [Waterfall (каскадна модель)](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) не підходить для середовища, наповненого невизначеністю, яка виникає під час розробки програмного забезпечення.

Потрібен був процес, який дозволив би швидше випустити програмне забезпечення, щоб його можна було тестувати та збирати дані про те, що потрібно змінити під час наступної ітерації.

### Як працює Scrum

Ви [розділяєте свій проєкт](https://java.lviv.ua/chomu-same-scrum-za-i-proty) на часові інтервали, так звані спринти. Багато компаній використовують для спринтів двотижневий проміжок часу, хоча ви можете встановити інший: однотижневий або тритижневий. Але вони не мають бути надто короткими або довгими.



Scrum та його застосування в Jira

<https://coursera.org/share/73038b5143c7a9a5300a2f7f8baf88d9>

Ролі та комунікаційні події у Scrum

<https://coursera.org/share/727f7ad07957da106df5b95227126c6b>

**Scrum-дошка Jira** - це інструмент, що дозволяє об'єднати команди для досягнення спільної мети і забезпечити ітеративну Інкрементальний розробку.

За допомогою Scrum-дощок Jira можна ділити роботу на етапи і отримувати звіти про швидкість і burndown-звіти. Завдяки цьому діяльність команди стає більш прозорою.

Scrum-дошки Jira ідеально підходять для високотехнологічних команд, що використовують методологію agile для управління проектами. Учасники команди Scrum проводять збори, використовують спеціальні інструменти і приймають на себе особливі ролі, щоб організувати роботу і керувати нею.

Згідно Scrum, команда не знає всього на початку проекту, але буде розвиватися, отримуючи уроки з досвіду. У структурі Scrum закладена та свобода, з якою команди пристосовуються до мінливих умов і вимог користувачів. Робочий процес передбачає зміну пріоритетів і короткі цикли релізу, що сприяє постійному навчанню та вдосконаленню команди.

Спочатку визначимо три артефакту Scrum. **Артефакт** - це те, що ми створюємо, наприклад інструмент для вирішення проблеми. У Scrum існує три артефакту: беклог продукту, беклог спринту і інкремент з вашими критеріями готовності

**Беклог продукту**  - це головний список робіт, які необхідно виконати. Його веде власник продукту або менеджер продукту. Це постійно мінливий перелік функцій, вимог, поліпшень і виправлень, з якого складаються завдання для беклога спринту. У загальному і цілому, це список завдань команди. Власник продукту постійно звертається до беклогу продукту, змінює в ньому пріоритети і підтримує його актуальність, бо може з'явитися нова інформація або можуть статися зміни на ринку, через що не буде сенсу виконувати задачі або з'являться нові способи вирішення проблем.



**Беклог спринту**  - це список робочих завдань, [призначених для користувача історій](https://www.atlassian.com/ru/agile/project-management/user-stories) або виправлень багів, відібраних командою розробників для реалізації в поточному циклі спринту. Перед кожним спринтом проводяться збори з планування спринту, на якому команда вибирає, які завдання з беклога продукту вона виконає в рамках спринту. Беклог спринту може не бути фіксованим і може змінюватися по ходу спринту. Однак ніщо не повинно заважати досягненню основної мети спринту - того, чого команда хоче добитися за поточний спринт.

**Інкремент** (або мета спринту) - це готовий до використання кінцевий продукт виконання спринту. У компанії Atlassian прийнято представляти інкремент на демонстрації в кінці спринту, на якій команда показує, що вона зробила за спринт. Слово «інкремент» не так уже й широко зустрічається в звичайному житті, але його часто визначають як прийняті в команді критерії готовності продукту, контрольну точку, мета спринту або навіть повну версію або [поставлений епік](https://www.atlassian.com/ru/agile/project-management/epics). Все залежить від того, якими критеріями готовності керується ваша команда і як вибираються цілі спринту. Наприклад, деякі команди воліють випускати що-небудь для своїх клієнтів в кінці кожного спринту. Для них «готовий» одно «поставлений». Однак для інших команд це може бути непрактично. Уявіть, що ви працюєте над серверним продуктом, який можна постачати клієнтам лише раз в три місяці. Ви як і раніше можете розбивати роботу на двотижневі спринти, але для вас продукт буде «готовий», коли ви завершите роботу над частиною більшої версії, яку ви плануєте поставити цілком. Природно, що чим більше часу йде на випуск ПО, тим менше шансів у цього ПО здобути успіх.

Перелічимо основні збори, в яких може взяти участь команда Scrum.

Ролі та комунікаційні події у Scrum

<https://coursera.org/share/727f7ad07957da106df5b95227126c6b>

**Організація беклога** . За цей захід, також відоме як ведення беклога, несе відповідальність власник продукту. У число його основних обов'язків входять приведення продукту у відповідність з його концепцією і постійне відстежування настроїв на ринку і потреб клієнта. Для цього власник продукту і веде цей список, змінюючи в ньому пріоритети і підтримуючи його в актуальному вигляді на підставі інформації від користувачів і команди розробників, щоб в будь-який час можна було приступити до роботи над внесенням в нього завданнями. Детальніше про те, [як правильно вести беклог, можна прочитати тут](https://www.atlassian.com/ru/agile/scrum/backlogs) .

<https://www.atlassian.com/ru/agile/scrum/backlogs>

**Планування спринту** . Роботу, яку необхідно виконати (обсяг спринту) протягом поточного [спринту](https://confluence.atlassian.com/agile/glossary/sprint) , планують на цих зборах всією командою розробників під керівництвом scrum-майстра. На ньому приймається рішення про мету спринту. Потім в спринт додаються конкретні призначені для користувача історії з беклога продукту. Ці історії завжди співвідносяться з метою. При цьому команда Scrum погоджує такі історії, які можна буде реалізувати на практиці в ході спринту.

В кінці зборів з планування кожен член команди Scrum повинен мати чітке уявлення про те, що можна виконати за спринт і як поставити інкремент.

**Спринт** . Спринт - це фактичний проміжок часу, протягом якого команда Scrum спільно працює над створенням готового инкремента. Як правило, спринт триває два тижні, хоча деяким командам простіше спланувати обсяг спринту на один тиждень або поставити інкремент, що володіє достатньою цінністю, за місяць. Дейв Уест з Scrum.org рекомендує вибирати спринт тим коротше, чим складніше робота і чим більше в ній невідомих. Але останнє слово завжди за командою, і вам не варто боятися змінювати тривалість спринту, якщо здасться, що вона вам не підходить. Протягом цього періоду власник продукту і команда розробників можуть переглянути обсяг спринту, якщо це необхідно. Це і є ключ до розуміння емпіричної суті Scrum.

Всі заходи, від планування до ретроспективи, проводяться протягом спринту. Після того як часовий проміжок для спринту визначено, він повинен залишатися незмінним, поки ведеться розробка. Так команда буде отримувати цінні уроки з минулого досвіду і застосовувати зроблені висновки до майбутніх спринті.

**Щоденна scrum-нарада, або стендап** . Це щоденні короткі збори, яке для простоти проводиться в один і той же час (зазвичай вранці) і в одному і тому ж місці. Багато команд намагаються вкластися в 15 хвилин, але це не більше ніж орієнтир. Таке зібрання ще називається «щоденним стендапом», що підкреслює його стислість. Щоденна scrum-нарада проводиться, щоб кожен учасник команди був в курсі того, що відбувається, не відхилявся від шляху до мети і отримував план роботи на найближчі 24 години.

**Стендап - саме час повідомити про все, що заважає вам досягти мети спринту.**

Найчастіше в рамках стендапа кожному учаснику команди пропонується відповісти на наступні три питання, пов'язані з досягненням мети спринту:

• «Що мені вдалося зробити вчора?»

• «Що я планую зробити сьогодні?»

• «Чи може мені щось завадити?»

Втім, нам доводилося спостерігати, як такі збори перетворювалося в зачитування людьми записів з щоденника. У теорії, стендап потрібен, щоб вся балаканина залишалася на щоденному зборах, а в решту часу доби команда могла зосередитися на роботі. Тому якщо хтось починає просто зачитувати свій щоденник, не соромтеся внести зміни в збори; проявіть кмітливість.

**Огляд підсумків спринту** . В кінці спринту команда збирається для перегляду демонстрації інкремента (або для його вивчення) в неформальній обстановці. Команда розробників представляє робочі завдання з беклога, які на той момент вважаються завершеними, на суд зацікавлених осіб і колег. Власник продукту вирішує, чи варто випускати інкремент чи ні, хоча в більшості випадків інкремент випускається.

На зборах з огляду підсумків власник продукту також переробляє беклог продукту на підставі результатів щойно закінчився спринту. Цей процес може перейти в планування наступного спринту. Якщо спринт триває один місяць, відводите під збори для огляду підсумків не більше чотирьох годин.

**Ретроспектива спринту** . [Ретроспектива](https://www.atlassian.com/ru/agile/scrum/retrospectives) проводиться, щоб команда задокументувала і обговорила всі успіхи і невдачі спринту, проекту, людських відносин, інструментів або навіть певних зборів. Мета ретроспективи - створити умови, щоб команда могла приділити увагу всьому, що вдалося і що потрібно поліпшити в наступний раз, і не зациклювалася на те, що не вдалося.

Три найважливіші ролі, від яких залежить успіх застосування Scrum

Склад scrum-команди передбачає три окремі ролі: **власник продукту, scrum-майстер і команда розробників.** Оскільки scrum-команди поєднують в собі безліч функцій, в команду розробників також входять тестувальники, дизайнери, UX-фахівці та інженери за операціями.

**Власник продукту Scrum**

Власники продукту ратують за свій продукт. Їх завдання - розуміти вимоги бізнесу, клієнта і ринку. На основі цього розуміння вони розставляють пріоритети між робочими завданнями, які технічна команда буде виконувати у відповідному порядку. Ось що відрізняє успішних власників продукту:

Вони складають беклог продукту і керують їм.

Вони тісно співпрацюють з керівництвом компанії і командою, доносячи до кожного значення всіх робочих завдань в беклоге продукту.

Вони дають команді зрозумілі вказівки, щоб вона знала, які можливості поставити наступними.

Вони вирішують, коли поставити продукт, прагнучи робити це якомога частіше.

Роль власника продукту не завжди поєднана з роллю менеджера продукту. Власники продукту прагнуть створити всі умови, щоб команда розробників створювала максимальну цінність для бізнесу. Важливо, щоб власником продукту був один чоловік. Навряд чи команда розробників захоче отримувати різні вказівки від різних власників продукту одночасно.

**Scrum-майстер**

Scrum-майстри стежать за застосуванням принципів Scrum в своїх командах. Вони навчають команди, власників продуктів і решту компанію тонкощам scrum-процесу і шукають способи оптимізувати застосування цієї практики.

Успіх scrum-майстра залежить від того, наскільки добре він розбирається в роботі, яку виконує команда, і може допомогти команді підвищити прозорість роботи і оптимізувати процес поставки продукту. Це головний координатор, який складає перелік необхідних ресурсів (кадрових і матеріально-технічних) для зборів з планування спринту і огляду його підсумків, стендапа і ретроспективи спринту.

**Команда розробників Scrum**

На команди Scrum лягає вся основна робота. Вони фахівці за принципами збалансованої розробки. Найуспішніші команди згуртовані, знаходяться в одному місці і зазвичай складаються з 5-7 учасників. Щоб визначити розмір команди, можна звернутися до відомого «правилом двох піц», яке сформулював глава Amazon Джефф Безос (в команді має бути стільки учасників, щоб їм вистачало двох піц).

Кожен учасник команди має свій набір навичок. Учасники навчають один одного виконання різних робіт, щоб жоден з них не став перешкодою на шляху до мети. Успішні scrum-команди здатні до самоорганізації, і їх підхід до проектів пронизаний командним духом. Всі учасники команди допомагають один одному, щоб успішно завершити спринт.

Scrum-команди складають план на кожен спринт. Вони прогнозують обсяг роботи, який здатні виконати за ітерацію, використовуючи в якості орієнтира показники своєї швидкості у минулих спринтах. Завдяки фіксованою тривалості ітерації команда розробників може проаналізувати правильність оцінки складності і процесу постачання продукту, що, в свою чергу, значно підвищує точність її прогнозів з часом.