**УДК 005.330:004.942**
**DOI: 10.15587/2312-8372.2019.180459**

# АДАПТАЦІЯ МОДЕЛЕЙ ГНУЧКОГО УПРАВЛІННЯ ПРОГРАМНИМ ПРОЕКТОМ НА ОСНОВІ ТЕХНОЛОГІЙ SCRUM ТА KANBAN

**Попова О. О.**

## 1. Вступ

На сьогодні дуже важливо при розробці програмного забезпечення (ПЗ) притримуватись певної методології для керування розробкою продукту, яка забезпечує, зокрема, планування, контроль та аналіз виконання поточних завдань для досягнення мети проекту. Не використовуючи таку методологію складно провести розробку програмного продукту вчасно та якісно. У програмному проекті важливо розподілити всі необхідні роботи між етапами, розпланувати кожен із них і визначити скільки потрібно часу на виконання кожного завдання. Також протягом всієї розробки потрібно контролювати визначені параметри. На сьогодні розроблено цілий ряд методологій розробки ПЗ, до яких належать:

– класичні методології, наприклад, Waterfall, Iterative;

– гнучкі методології розробки, такі як екстремальне програмування (XP), Lean (Lean software development), FDD (Feature Driven Development), DSDM (Dynamics System Development Method), Crystal, Scrum та Kanban.

Але кожну з названих методологій потрібно адаптувати до виконання проекту конкретного програмного проекту з врахуванням особливостей виконуючої проект організації. Тому адаптація та використання певної методології для розробки програмного забезпечення, а також створення комбінованої методології, яка б краще адаптувалася до сучасних умов виконання програмних проектів є важливою та актуальною задачею.

Є декілька різних підходів до організації розробки програмного продукту. Одним з найкращих вважається методологія Agile, яка є гнучкою методологією розробки, а її представниками для невеликих команд розробників є Scrum і Kanban. Вони використовуються при роботі над програмними проектами та дозволяють виконувати поставлені завдання максимально швидко, ефективно та якісно.

У даній роботі представлене рішення, у якому поєднані ці дві гнучкі методології та запропонований результат, який суміщає найкращі риси з кожної методології та дозволяє зменшити недоліки. А також підвищити корисні якості Scrum та Kanban у їх комбінованому варіанті. Зазначені якості досліджувались у експерименті, який дозволив на побудованих графових моделях порівняти характеристики та запропонувати комбінований варіант методу, що має кращі показники.

## 2. Об’єкт дослідження та його технологічний аудит

*Об’єктом дослідження* є методології розробки Scrum і Kanban.

Технічна користь даного дослідження полягає у розробці нового методу, який відповідав би потребам команди, яка складається з кількох невеликих та малих груп, при розробці програмного проекту. Підставою для поєднання Scrum та Kanban у одній методології є корисні властивості Scrum для невеликих та середніх команд, які є надто обтяжливими для малих груп. Натомість спрощені і оптимізовані організаційні процедури Kanban не завжди доцільно використовувати у невеликій команді. Доволі часто виникає ситуація на проекті, коли потрібно сумістити корисні властивості розглянутих методологій та узгодити між собою в єдиній організаційній технології.

**3. Мета та задачі дослідження**

*Метою дослідження* є проведення експерименту для створення комбінованої технології розробки ПЗ на основі методологій Scrum і Kanban.

Для досягнення мети вирішуються 3 задачі:

1. Проаналізувати та зменшити недоліки, які викликають необхідність адаптації до різних типів команд. Як у Scrum, так і у Kanban, є свої недоліки, які можна врахувати при розробці нової технології.

2. Додати нові правила для комбінованої технології, які б дозволили узгодити позитивні якості базових методологій та використовувати їх разом.

3. Забезпечити більш гнучкий процес організації програмного проекту.

**4. Дослідження існуючих рішень проблеми**

Перш за все, як підґрунтя для пропонованої комбінованої технології, проведемо загальний огляд гнучких методологій розробки Agile. Методологія XP використовувалась для невеликих команд, які мали виконати проект за короткий час і з загальним розумінням задач всіма членами команди. Головна мета даної методології – справлятися з вимогами до програмного продукту, які постійно змінюються і підвищувати якість розробки [1]. Вона характеризується частими невеликими релізами, в яких реалізовані прості гарно структуровані рішення, при розробці використовується підхід TDD – розробка через тестування, колективні володіння кодом чи шаблоном проектування, гра у планування. Недоліками XP є досить складна адаптація до проекту в разі збільшення команди та розширення границь проекту. Однією з гнучких методологій є Lean (Lean software development). Вона виникла з технології ощадливого виробництва і спрямована на усунення всіх видів непотрібних витрат. Це непотрібні функції, непотрібні переробки, недовиконані роботи, час на пошук та усунення дефектів, який можна скоротити за рахунок раннього аналізу та виявлення помилок на ранніх стадіях, та ін. FDD (Feature Driven Development) – функціонально-орієнтована розробка, яка робить акцент на розбиття проекту на невеликі функції, які корисні з точки зору клієнта і які реалізуються не більше, ніж за два тижні. Ця методологія являє собою комбінацію п’яти елементів: розробка загальної моделі, створення списку

функцій, планування, дизайн та розробка і реалізація [2]. Найцінніші риси двох вищеназваних методологій знайшли свій розвиток у сучасних гнучких підходах.

Методологія Scrum передбачає вільний комплекс впорядкованих заходів, що базуються на найкращих практиках та постійно вдосконалюються [3]. Важливими характеристиками Scrum є її гнучкість і орієнтованість на клієнта, так як вона передбачає його (клієнта) безпосередню участь в процесі роботи [4].

Методологія Kanban розроблена для малих і дуже малих команд і дозволяє виокремити з RUP найнеобхідніших процесів виконання програмного проекту у відповідності з принципом «точно у зазначений час». Дві останні методології найбільш повно пристосовані до сучасних організаційних і технічних особливостей виконання програмних проектів і можуть бути використанні в якості основи для комбінованої технології, яка пропонується.

Scrum – це авторська гнучка методологія розробки з нестандартним розподілом ролей в команді і унікальною організацією ітерацій. У скрам-командах ключові позиції займають scrum-master і product owner, процес розробки складається з певної кількості ітерацій, ітерація починається плануванням, далі відбувається проектування і розробка частини продукту, що завершується демо з ретроспективою [5]. Основні події Scrum – це:

1. Спринт – у рамках Scrum всі дії, необхідні для впровадження записів із блоку продукту Scrum, виконуються в спринтах (також називаються «ітерації»). Спринти завжди короткі: зазвичай це приблизно 2–4 тижні [6].

2. Sprint backlog – список завдань, які мають бути зроблені в даному спринті.

3. Планування спринта – зустріч, на якій відбувається планування спринта, тобто завдань, які будуть до нього включені.

4. Scrum-дошка – це інструмент, який допомагає командам зробити видимими елементи Sprint backlog [7].

5. Щоденний Scrum-мітинг – зустріч, як правило, проводиться в одному місці та в один і той же час кожного дня. Ці зустрічі суворо розраховані на 15 хвилин [8].

6. Спринт-огляд (іншими словами, демо) – демонстрація завершених (готових) завдань з метою сприяння обговорення між командою Scrum та іншими учасниками огляду спринту [9].

7. Ретроспектива спринта – можливість для команди переглянути свою роботу та створити план для покращення якихось елементів в процесі розробки для майбутнього спринта [10].

Kanban – це метод управління розробкою програмного забезпечення з дотриманням принципу «точно у зазначений час» та унікальними перевантаженнями членів команди. У цьому методі процес від опису задачі до доставки результатів її виконання користувачу наочно показується учасникам процесу, члени команди можуть брати роботу з черги [11].

Для Kanban важливо виконувати наступне:

1. Kanban дошка – використання дошки, що зображає завдання, які будуть виконуватися, які у процесі виконання та завершені завдання. Крім того, лімітує кількість задач, які знаходяться в колонці «у процесі виконання» [12].

2. Kanban-мітинг – також важливий для підтримування синхронізації команди, щоб знати, хто якими задачами займається і які у нього успіхи з їх виконанням. В ідеалі мітинг має тривати 10–15 хв [13].

Кожна з цим двох методологій розробки ПЗ має як свої переваги, так і недоліки. Крім того, у цих двох методологій дуже багато спільного, але є і відмінні речі. Для наочності зобразимо переваги і недоліки кожної технології за допомогою порівняльних характеристик в табл. 1. У даній таблиці зображено порівняння методологій на основі декількох критеріїв. Показано, що головне у розробці, планування завдань, важливість спринта, довжина спринта, планування спринта, зміни та доповнення у спринті, ролі, дошка із завданнями та її організація. Також зустрічі команди, вимірювання продуктивності та оцінка задач.

**Таблиця 1**

Порівняльна характеристика Scum та Kanban

| Scrum | Kanban |
|---|---|
| 1 | 2 |
| Головне у розробці | |
| Спринт | Задача |
| Планування завдань | |
| Створюється backlog з чіткими завданнями, які мають бути зроблені в спринті | Є workflow, де відображені всі завдання, які мають бути виконані і вони просто переміщаються із одного статусу в інший |
| Важливість спринта | |
| В Scrum все виконується в спринтах, які тривають від 2 до 4 тижнів відповідно. В кінці спринта є sprint review (демо) | В Kanban відбувається реліз, як тільки готовий продукт. Це може бути декілька разів на день або раз на тиждень |
| Довжина спринта | |
| Завжди однакова (2–4 тижні) | Немає чітких рамок |
| Планування спринта | |
| На початку спринта проводиться sprint planning | Чіткого планування немає |
| Зміни/доповнення в спринті | |
| Не можуть бути. На початку спринта вирішується скільки і які завдання мають бути зроблені (sprint scope) | Можливі. Коли завдання завершене, може бути створене нове завдання; дуже гнучка методологія |
| Ролі | |
| Обов’язково повинні бути (Product owner, Scrum master, Scrum team members) | Необов’язкові (можуть бути SDM і SRM), фокус йде не на розподіленні ролей, а на те, щоб доставити продукт [14] |

**Продовження таблиці 1**

| 1 | 2 |
|---|---|
| Дошка із завданнями | |
| Нова для кожного нового спринта | Одна й та ж сама. Це перевага, якщо команда невелика, тому що можна бачити на одній дошці всі завдання, їх зв'язки між собою та прогрес виконання |
| Організація дошки із завданнями | |
| Зазвичай складається із колонок «To do», «In progress», «In testing» та «Done» | Зазвичай складається із колонок «To do», «In progress» та «Done» |
| Зустрічі | |
| Щоденні і обов'язкові. Можуть бути таких типів: sprint planning, daily scrum, sprint review, retrospective. Зазвичай щоденні зустрічі займають не більше, ніж 15 хв. | Необов'язкові, але рекомендуються. Можуть бути таких типів: daily meeting, replenishment, delivery planning meeting, service delivery meeting, operations review, risk review, strategy review [15] |
| Вимірювання продуктивності | |
| Вимірюється в story points, а точніше в швидкості виконання завдань за спринт | Вимірюється в швидкості проходження завдання від статусу «To do» в «Done» |
| Оцінка задач | |
| Завжди є | Не завжди задачі оцінюються |

З табл. 1 видно, що в кожній з даних методологій є свої переваги та недоліки. При виборі методології потрібно задавати собі питання, що потрібно – постійна розробка чи ітерації, структуровані ролі чи команда без ролей, адаптація до змін чи стабільність.

**5. Методи досліджень**

Для проведення даного дослідження буде використано моделювання процесу, середовища та експеримент, в результаті якого буде виявлено, які сторони Scrum та Kanban варто використати, як їх поєднати та в яких умовах.

Для моделювання було обрано метод моделювання за допомогою мереж Петрі. Мережі Петрі – це математичний апарат для моделювання динамічних дискретних систем [16]. Мережі Петрі являють собою двохдольний орієнтований граф, елементи якого (вузли) застосовуються для відображення певних подій [17]. У процесі цього моделювання бачимо, як змінюється статус елементів моделі та як це впливає на модель у загальному. За допомогою переходів у мережах Петрі можна з'ясувати, як система себе поводитиме в тому чи іншому середовищі. Мережі Петрі показують перехід в мережі, при якому мітки із вхідних позицій переходу переходять у вихідні позиції.

Мережі Петрі складаються із чотирьох елементів:

– множина позицій *P*;

– множина переходів *T*;

– вхідна функція *I*;

– вихідна функція *O* [18].

Безперечна перевага мереж Петрі – математично точний опис моделі. Це дає змогу провадити науковий аналіз за допомогою сучасної обчислювальної техніки (також із паралельною архітектурою) [19].

Використовуючи мережі Петрі можна змоделювати поставлену задачу – визначити, як краще всього поєднати Scrum та Kanban. Перш за все, зобразимо модель Scrum методології, виходячи з вище згаданої порівняльної табл. 1.

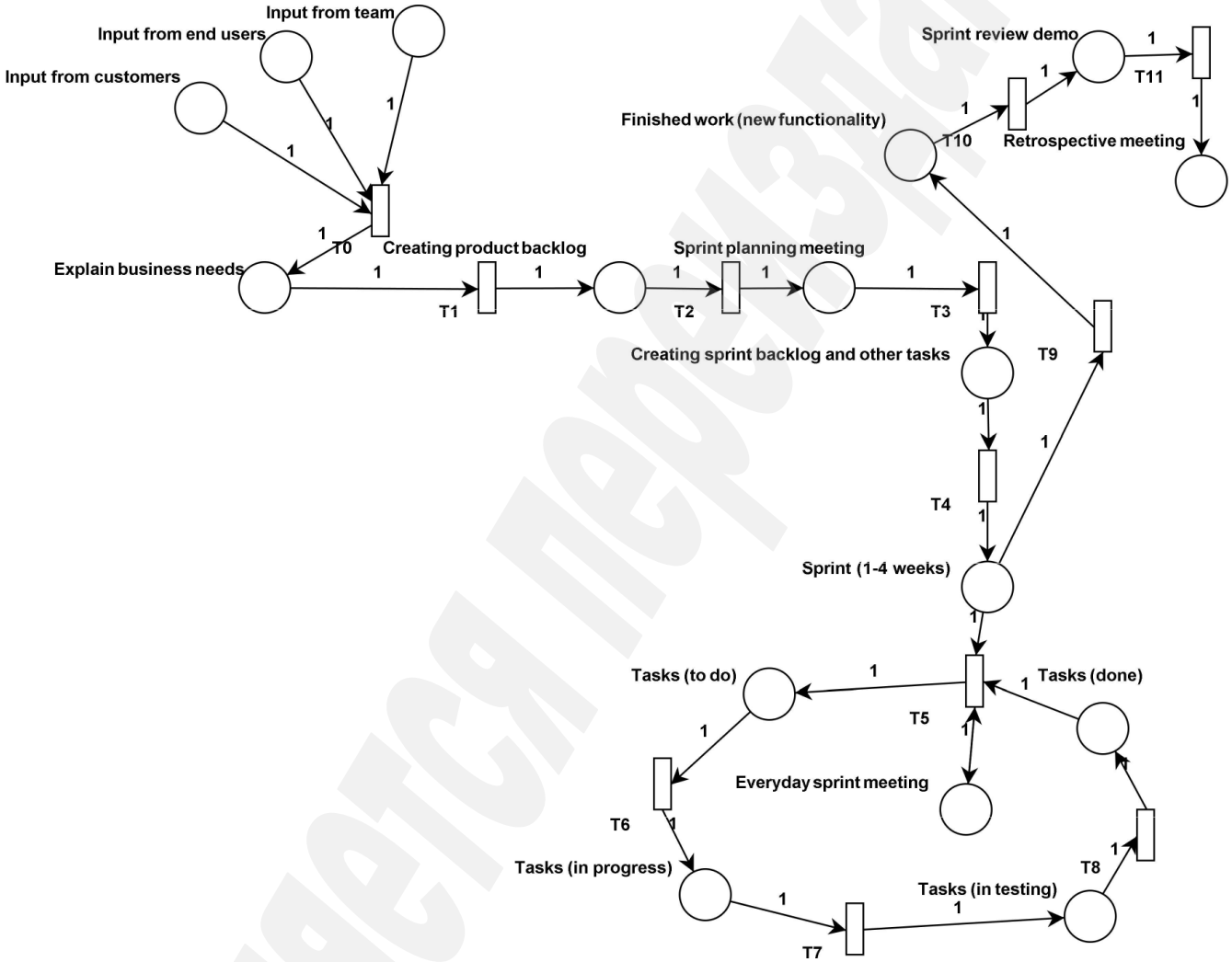


**Рис. 1**. Модель методології Scrum

На рис. 1 зображено весь процес Scrum: ідея, яка формується на основі того, що хочуть кінцеві користувачі, клієнти та команда, створення загального списку завдань на весь час розробки продукту (user stories; завдання, які пріоритетизовані по business value), створення подібного списку на конкретний спринт на мітингу, проведення спринту, який триває 1–4 тижні. Кожного дня відбувається зустріч, де обговорюються три питання в команді – що було зроблено, які є перепони та що планується зробити до наступного мітинга. Завдання мають 4 статуси – To do, In progress, In testing і Done, які відображаються на дошці із завданнями. Після завершення спринта відбувається

зустріч, де показують і обговорюють результати нового функціоналу. Також в кінці проводиться ретроспектива. Весь цей процес побудований на тому, що методологія є доволі гнучкою і спрямована на різні команди.

Так само зобразимо модель для методології розробки Kanban (рис. 2). Спершу відбувається аналіз того, що треба зробити. Після цього слідує проектування, а далі розробка, тестування, тестування UAT і SIT та деплоймент. Варто згадати, що немає обмежень на завдання в спринті, тому також можуть додаватися change requests (зміни для вже виконаних завдань), bugs (не функціонуючі елементи програм) або maintenance tasks (завдання підтримки).

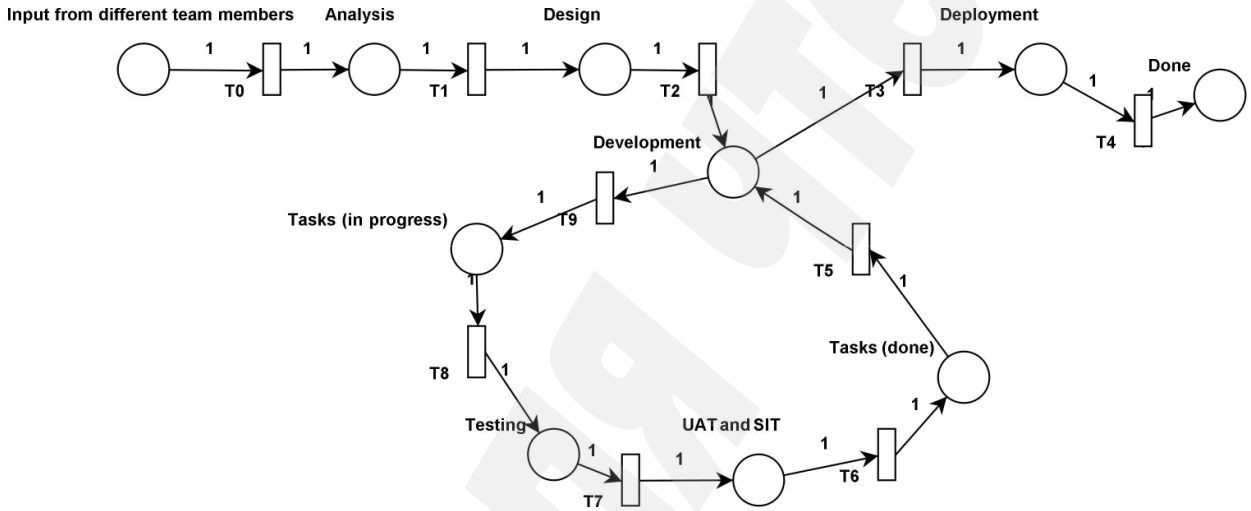


**Рис. 2.** Модель методології Kanban

На основі моделей Scrum та Kanban буде створена нова методологія розробки.

**6. Результати досліджень**

При поєднанні даних двох методологій було вирішено взяти найкращі сторони кожної з них. Саме на цьому будується гіпотеза для проведення експерименту. Для створення нової методології на основі Scrum та Kanban виберемо наступні пункти по вище згаданій табл. 1:

1. Головне у розробці – спринт. Дана гіпотеза була взята з методології Scrum. Найкраще розробляти продукт поступово, використовуючи ітераційний метод.

2. Планування завдань. Було б гарною ідеєю створити головні задачі, які мають бути обов'язково виконані і по мірі їх виконання можна брати задачі з беклогу, так само як баги та завдання типу change request. Таким чином, з одного боку це частина методології Kanban, а з іншого – нова вимога.

3. Важливість спринта. Пропонується взяти правило зі Scrum методології та все виконувати в спринтах, які б тривали 2 тижня та в кінці було б демо (sprint review) для клієнтів.

4. Довжина спринта. Для довжини спринта пропонується ввести 2 тижні, але з наступними поправками: якщо завдання виконані раніше – брати наступні

завдання з беклогу, change requests або баги. Якщо ж команда не встигає – переносити завдання на наступний спринт.

5. Планування спринта. Планування має проводитись для структурованості розробки продукту. Має бути планування на спринт, але також завдання, які не потребують планування і які можна виконувати в другорядному пріоритеті (як в методології Kanban).

6. Зміни/доповнення в спринті. Зміни можливі, як в методології Kanban.

Якщо всі завдання виконані, можна брати наступні завдання, які не входили до спринта при його плануванні. Також може бути заміна якихось завдань під час виконання спринта, якщо це потрібно.

1. Ролі. Для даної методології пропонується, як і в методології Scrum, розподілити людей на два типи – Product Owner, Scrum master (його роль виконуватиме Project Manager) та команда розробників. У команді в кожної людини буде чітка роль, але тестуванням зможе займатись кожен член команди для кращої ефективності. Дана методологія має бути дуже гнучкою для команд і врховувати інтереси кожної людини.

2. Дошка із завданнями. Дошка має змінюватись кожен спринт, щоб не було великої кількості завдань і було чітко видно, що виконується саме зараз в спринті. Але при цьому на дошці можуть бути зображені і прикріплені ключові завдання зверху кожної колонки статусу.

3. Організація дошки із завданнями. При цьому як і в методології Scrum має бути чотири колонки статусу – To do, In progress, In testing та Done.

4. Зустрічі. Scrum має хорошу практику частих зустрічей, які допомагають діяти структуровано та знати, як просуваються справи в команді. Крім того, дуже корисні зустрічі для планування завдань, демо (sprint review) та ретроспективи, щоб обговорити спринт. Але ж, якщо вважається, що зустріч не потрібна (наприклад, ретроспектива), її можна не проводити, як це використовується в методології Kanban.

5. Вимірювання продуктивності. Було б добре використовувати для вимірювання продуктивності кількість завдань, виконаних за спринт. Адже не можна брати за одиницю продуктивності перехід завдання зі статусу to do в done через те, що завдання можуть бути різного об’єму та складності.

6. Оцінка задач. Для того, щоб знати, скільки задач можна буде виконати в спринті, в ідеалі їх завжди потрібно оцінювати та додавати ще якусь частину часу. Цей принцип взято із методології Scrum.

Таким чином, було визначено основні принципи, які повинна мати нова методологія. Як видно із попередніх пунктів, за основу була взята методологія Scrum, яка була доповнена деякими правилами із методології Kanban, а також власними правилами. Далі, використовуючи мережі Петрі, буде сформовано модель.

В результаті проведення дослідження була спроектована модель нової методології. Вона була спроектована на основі двох інших методологій – Scrum та Kanban. Сформована модель показана на рис. 3.

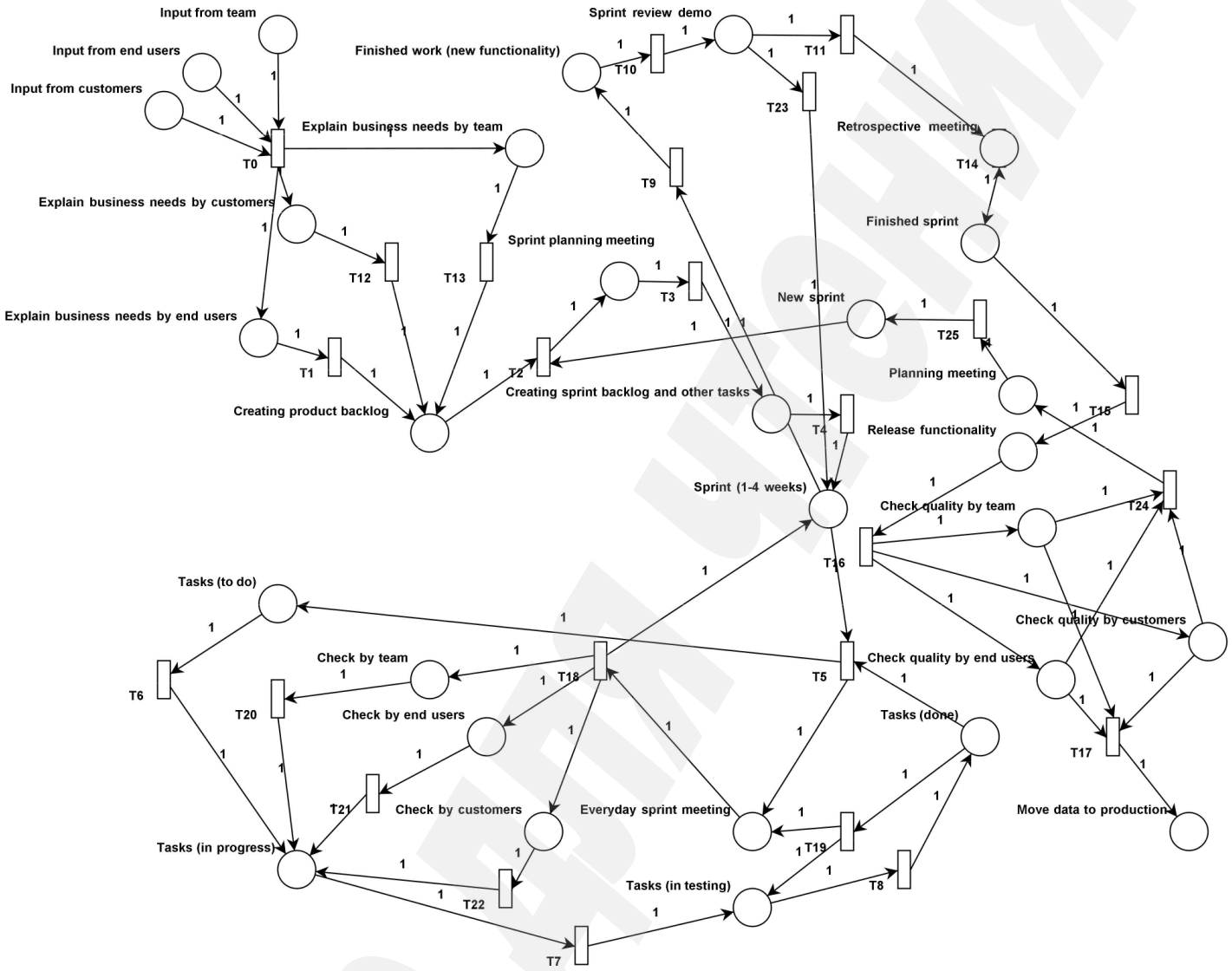


**Рис. 3**. Модель нової методології на основі Scrum і Kanban

Нижче показано основні властивості елементів даної моделі:

Спершу відбувається пояснення бізнес-цілей від бізнес-користувачів, клієнтів, команди та інших людей. Кожен з них вносить свій внесок у розробку програмного продукту ще на початковій стадії.

Далі після визначення головних бізнес-цілей відбувається створення беклогу із пріоретизованими вимогами. Завдяки тому, що бізнес-цілі були створені завдяки всім учасникам, можна помітити дуже важливу роль кожного учасника команди.

Після цього відбувається планування спринта, де складається список завдань на конкретний спринт. У цьому процесі також бере участь вся команда.

Спринт має тривати 1–4 тижня. Кожного дня відбувається зустріч, на якій команда дізнається, що робив, що робить та що буде робити кожен її учасник, а також чи є якісь перепони. Якщо ж необхідності у мітингу немає – його може не бути.

Далі відбувається виконання всіх завдань. Вони переходять зі статусу To do в In progress, In Testing та Done. Якщо заплановані завдання виконані, то можна робити інші завдання із беклогу, а також change requests та bugs.

Після того, як був розроблений новий функціонал, відбувається sprint review (демо) і якщо після цього потрібно щось доробити – спринт продовжується. Якщо ж ні, то проходить ретроспектива і завершення спринта. Після цього починається новий спринт з усіма відповідними стадіями планування, виконання та перевірки.

Коли ж всі спринти були виконані, відбувається реліз програмного продукту. На цьому етапі відбувається перевірка виконаної роботи і якщо потрібно щось ще доробити, відбувається зустріч з плануванням та продовження спринта. Якщо ж дороблювати не потрібно, то відбувається реліз програмного продукту.

З даної моделі видно, що вона повністю відповідає вимогам нової методології. Вона є дуже гнучкою та адаптивною. Дана методологія підходить для невеликих команд і враховує думку кожного учасника. В результаті, дана методологія дійсно може бути використана для розробки програмного продукту.

**7. SWOT-аналіз результатів досліджень**

*Strengths.* Нова створена методологія значно покращує існуючі методології розробки, має свій набір правил і дозволяє командам легко її використовувати, адже є гнучкою методологією.

*Weaknesses.* При всій адаптованості та гнучкості методології, вона має застосовуватись правильно, в залежності від типу програмного продукту, мети проекту та команди розробників.

*Opportunities.* Серед нових переваг є те, що дану методологію можна використовувати для зручного та ефективного управління програмами, для яких з одного боку підходить Scrum, а з іншого – Kanban. За основу взято методологію Scrum.

*Threats.* З об’єктивної точки зору, ризики можливі, якщо не дотримуватися деяких правил даної методології. Наприклад, не провести правильне планування, не відвідувати мітинги, не аналізувати зроблену роботу.

**8. Висновки**

1. В результаті аналізу, підбору методів для виконання дослідження та проведення дослідження було прибрано та зменшено недоліки методологій Scrum та Kanban. Крім цього, нова методологія підходить до різних типів команд і покращує їх роботу. За допомогою моделей вже існуючих методологій Scrum та Kanban було запропоновано гіпотези для створення нової методології і в результаті проведення експерименту ці гіпотези були підтверджені.

2. У новій методології розробки сформовано набір правил, оскільки не завжди методології Scrum та Kanban підходять для розробки проекту. Тому було вирішено додати нові правила. Ці правила сформовані, враховуючи плюси та мінуси методологій Scrum та Kanban.

3. Забезпечено більш гнучкий процес організації програмного за рахунок встановлення параметрів комбінованої технології. У даній технології розробки ПЗ пропонується підвищити гнучкість шляхом структурно-параметричного аналізу уже існуючих методологій Scrum і Kanban.

**Література**


1. Гранько, О. (23 июня 2017). Экстремальное программирование (XP). *Блог. PM Решения.* Available at: https://worksection.com/blog/extreme-programming.html
2. Кратко о методологиях разработки ПО: Waterfall, Lean и Feature Driven Development. (30 ноября 2017). *Блог. ИТ Гильдия.* Available at: https://habr.com/ru/company/it-guild/blog/341932/
3. «Scrum. Революционный метод управления проектами». Книга за 15 минут. (15 декабря 2015). *Блог. MakeRight.* Available at: https://habr.com/ru/company/makeright/blog/297250/
4. Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time.* New York: Random House, 256.
5. Гранько, О. (8 червня 2017). Scrum чи не Scrum – який підхід обрати. *Блог. PM Рішення.* Available at: https://worksection.com/ua/blog/scrum.html
6. *What is a sprint*. Available at: https://www.scrum-institute.org/What_is_a_Sprint.php. Last accessed: 22.07.2016.
7. *What is Scrum.* Available at: https://www.scrum.org/resources/what-is-scrum. Last accessed: 01.08.2016.
8. *Daily Scrum Meeting.* Available at: https://www.mountaingoatsoftware.com/agile/scrum/meetings/daily-scrum. Last accessed: 03.08.2016.
9. *Sprint demo.* Available at: https://innolution.com/resources/glossary/sprint-demo. Last accessed: 10.08.2016.
10. *What is a sprint retrospective.* Available at: https://www.scrum.org/resources/what-is-a-sprint-retrospective. Last accessed: 11.08.2016.
11. Канбан (розробка). *Вікіпедія.* Available at: https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B1%D0%B0%D0%BD_(%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0). Last accessed: 12.08.2016.
12. *Kanban Encyclopedia: Concepts and Terms.* Available at: https://kanbanize.com/kanban-resources/getting-started/kanban-encyclopedia/. Last accessed: 12.08.2016.
13. Разбираемся в Scrum и Kanban. (16.12.2016). *Нетология.* Available at: https://netology.ru/blog/scrum-kanban
14. *What is Kanban board.* Available at: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board. Last accessed: 22.08.2016.
15. *Kanban Explained for Beginners.* Available at: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban. Last accessed: 26.08.2016.

16. Мережі Петрі – інструмент для моделювання динамічних систем в економіці. (15.03.2017). *Наукова спільнота.* Available at: http://www.spilnota.net.ua/ru/article/id-1759/

17. *Прості мережі Петрі*. Available at: https://studopedia.su/6_41767_prosti-merezhi-petri.html. Last accessed: 02.09.2016.

18. *Сети Петри. Структура и правила выполнения сетей Петри.* Available at: https://itmodeling.fandom.com/ru/wiki/%D0%A1%D0%B5%D1%82%D0%B8_%D0%9F%D0%B5%D1%82%D1%80%D0%B8._%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B8_%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D0%B0_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%B5%D1%82%D0%B5%D0%B9_%D0%9F%D0%B5%D1%82%D1%80%D0%B8. Last accessed: 07.09.2016.

19. *Мережі Петрі.* Available at: https://stud.com.ua/98834/informatika/merezhi_petri. Last accessed: 10.09.2016.