Електронне наукове фахове видання **"Ефективна економіка"** включено до переліку наукових фахових видань України з питань економіки (Наказ Міністерства освіти і науки України від 11.07.2019 № 975)
www. economy.nayka.com. ua | № 1, 2020 | 30.01.2020 р.

DOI: 10.32702/2307-2105-2020.1.8

УДК 656:004.75

В. М. Марченко,
д. е. н., професор кафедри економіки і підприємництва, НТУУ «КПІ ім. Сікорського»
ORCID: 0000-0002-4756-3703
О. О. Мезенцева,
к. е. н., доцент кафедри МЗЕДП, Національний авіаційний університет
ORCID: 0000-0002-8430-4022

ОПТИМІЗАЦІЯ ЗАСТОСУВАННЯ ГНУЧКИХ МЕТОДИК МЕНЕДЖМЕНТУ В IT-ПРОЕКТАХ

V. Marchenko
Doctor of Economic Sciences, Professor of the Department of Economics and Economics , NTUU "KPI im. Sikorsky "
O. Mezentseva
PhD in Economics, associate professor of the Department of MZEDP, National Aviation University

OPTIMIZATION OF APPLICATION OF FLEXIBLE MANAGEMENT TECHNIQUES IN IT PROJECTS

У статті розглядаються особливості оптимізації управління сучасними IT-проектами, ризики застосування гнучких методик для управління IT-проектами і їх причини, питання використання сучасних методологій і програмних інструментів моделювання оптимізації процесів для забезпечення пришвидшення та зниження ризиків реалізації IT-проектів. Досліджуються перспективи застосування Canban моделювання для вироблення ефективної управлінської культури постійного удосконалення. Проведено аналіз Agile-світогляду, методології та мислення для команди проекту. В роботі досліджено як концентрація на потоці та теорії масового обслуговування допомагає команді IT-проекту втілити в життя важливі програмні продукти.
Обґрунтовано доцільність переходу на управління Scrum-моделюванням використання комбінованої методології моделювання на базі процесного і агентно-динамічного підходів у розрізі фаз життєвого циклу і агентів команди IT-проекту для зниження інтегральних ризиків.

Features include optimizing the management of the most common IT projects, securing the best practices for managing the IT projects and the other reasons, providing the most advanced methodologies and optimizing the technology. Prospect of the future Canban model for the development of effective management culture of a constant udosonyalny. Agile analysis was conducted, methodology and knowledge for the project team. In the robot, there is a concentration on the potential and the theory of mass servicing of the additional team of the IT project to bring important software products to life.

12 principles of flexible software development are considered in the paper and the corresponding ways of their optimization are analyzed. It is investigated how KANBAN practices help to stabilize and improve the software development system. Agile methodologies are popular because many teams that have switched to them report excellent results: they create high-quality software, work more successfully together, satisfy the needs of their users and achieve all this in a relaxed working environment. Some agile teams have even advanced in solving problems that have bothered programmers for decades.
It is a change in the way of thinking that turns a group of employees, adding several agile methods into their work, into a real team that really improves the way of creating software.
The purpose of the study is to analyze the features and risks of methods for optimizing the use of flexible IT project methodologies, systematizing methodologies and tools for optimizing the management of IT projects. Analyzed that Agile will help to work with demanding users who are constantly changing their decision. Any change that the customer requires means additional work for your team and leads to a spaghetti code, from which sticky tape and paper clips stick out. Therefore, programs are becoming increasingly vulnerable and difficult to support.
The reason was the transition to control over Scrum-models, the victorious combined methodology, on the basis of the process and agent-dynamic steps in the separate phases of the live cycle and the agent's team for the project.

Ключові слова: *Scrum, Canban, IT-проект, гнучкі методології, Agile-світогляд.*

Keywords: *Scrum, Canban, IT project, agile methodologies, Agile outlook.*

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями. Рух гнучких методик Agile революційний. Команди, які використовують цю технологію, систематично відзначають поліпшення (іноді стрибкоподібні) в умінні створювати краще програмне забезпечення. Ті, хто успішно впровадив Agile, створюють високоякісні продукти і роблять це швидше, ніж раніше.

Agile з аутсайдера перетворився на працюючий інститут. В даний час практично не залишилося сумнівів, що agile-методології - це ефективний спосіб створення програмного забезпечення. У 2008 році було проведено дослідження [1], яке показало, що більше половини всіх опитаних команд, що займаються розробкою програмних продуктів, використовують agile-методології, практики або принципи.

Традиційно при виконанні проектів по розробці програмних продуктів компанії використовували Waterfall (водоспадний) підхід, згідно з яким команда спочатку визначає вимоги до продукту, планує проект в цілому, розробляє програмне рішення, а потім створює код і тестує продукт. Значна частина програмного забезпечення - як грандіозного, так і зовсім малорозмірного - роками створювалася саме таким чином. Однак, протягом десятиліть різні команди у всіляких компаніях стикалися з одними і тими ж проблемами. І деякі з них запідозрили, що головна причина невдач - сам Waterfall підхід. Використання Canban на сьогоднішній день це найбільш поширений метод (і ефективний для багатьох agile-практиків) впровадження бережливого мислення в організацію.

Аналіз останніх досліджень і публікацій. Питання управління проектами теоретично розглянуті в роботах Мазура І.І., Шапіро В.Д., Андрєєва А.А., Буркова В.М., Морозова В.В., Бушуєва С.Д., наборі принципів, процесів і областей знань (PMBOK - The Project Management Book of Knowledge) [2,4,6].

Формулювання цілей статті. Метою дослідження є аналіз особливостей та ризиків методів оптимізації застосування гнучних методологій IT-проектів, систематизація методологій та інструментальних засобів щодо оптимізації управління IT-проектами.

Виклад основного матеріалу дослідження. При використанні гнучких методик керуються 12 принципами гнучкої розробки програмного забезпечення

1. Наш найвищий пріоритет - це задоволення замовника за допомогою частих і безперервних постачань цінного для нього програмного забезпечення.
2. Зміни у вимогах приймаються навіть на пізніх етапах реалізації проекту. Agile-процеси дозволяють використовувати зміни для підвищення конкурентоспроможності продукту.
3. Прагнення поставляти повністю робоче програмне забезпечення кожні кілька тижнів, в крайньому випадку - кожні кілька місяців. Чим частіше, тим краще.
4. Найбільш ефективний і дієвий спосіб передачі інформації - це зустріч членів команди розробки ПЗ.
5. Представники бізнесу і команда розробки повинні працювати над проектом разом.

6. Проекти будуються навколо мотивованих людей. Слід створити для них відповідне навколишнє середовище, забезпечити всім необхідним і довірити зробити свою роботу.

7. Робоче програмне забезпечення - це головна міра прогресу проекту.

8. Гнучкі процеси сприяють безперервному розвитку. Спонсори, розробники і користувачі повинні мати можливість підтримувати постійний темп роботи протягом невизначеного терміну.

9. Постійна увага до технічної досконалості і якісної архітектури сприяє гнучкості.

10. Простота представля для команди мистецтво не робити зайвої роботи.

11. Краща архітектура, вимоги та дизайн створюються в самоорганізованих командах.

12. Команда постійно шукає способи стати більш ефективною шляхом настройки і корекції своїх дій [3].

Перший принцип включає в себе три важливі умови: ранній випуск програмного продукту, безперервна реалізація цінності і задоволення клієнта. Щоб зрозуміти суть цього принципу, потрібно знати, як ці три ідеї працюють разом. Проектні групи існують в реальному світі, в якому не буває нічого досконалого. Навіть блискуче працюючій команді при зборі і записі вимог до проекту буде чогось не вистачати, тому що неможливо повністю і точно відобразити вимоги для кожної системи. Це не означає, що команди не повинні намагатися, але agile-методології грунтуються на безпрецедентних методах комунікації і фіксації вимог. Проте, поки клієнти не отримають в руки працююче ПЗ, їм важко уявити, як саме воно буде функціонувати.

В такому випадку, якщо якщо ви хочете отримувати від клієнта реальний, обґрунтований зворотній зв'язок після перегляду робочого програмного продукту, то краще за все робити це через ранню поставку: «відвантажити» замовнику першої робочої версії програмного забезпечення якомога раніше. Навіть якщо реалізувано лише одну робочу функцію, яку клієнт може використовувати, - це вже досягнення певного успіху для команди. Тому що тепер споживач здатний надати зворотний зв'язок, що допоможе рухати проект в потрібному напрямку[4].

Це також успіх для замовників, тому що, маючи цю робочу програму, вони можуть реалізувати те, що раніше тільки планували. Можна сказати, що команда створила реальну цінність, оскільки ПЗ працює і клієнти можуть його використовувати. Нехай це лише невелика частина того, чого потребує споживач, але все-таки це краще аніж очікування без ПЗ, особливо якщо альтернатива цьому - довге очікування, перш ніж отримає програмний продукт. Недолік ранньої поставки в тому, що програмне забезпечення далеко від досконалості. Це являє складність для деяких клієнтів і зацікавлених сторін: якщо одні користувачі звикли до думки, що побачать програмне забезпечення рано, то іншим звикнутися з цією думкою важче. Багато працівників дуже переживають, коли їх софт не ідеальний. У компаніях (особливо великих, де витрачаються роки на роботу з командами, які розробляють програмне забезпечення) ці команди повинні дуже ретельно домовлятися про умови поставки ПЗ зацікавленим сторонам. При відсутності партнерських відносин між замовником ПЗ і командою-розробником поставляється неповний програмний продукт може бути оцінений строго і навіть привести користувача в жах, якщо дійсно не відповідає очікуванням. Основні agile-цінності вирішують подібні питання: співпраця з замовником важливіше узгодження умов контракту. Команда, жорстко прив'язана до специфікації, не може вносити зміни в програмне забезпечення в процесі його створення. Працюючи в таких умовах, потрібно запустити новий процес тотального управління змінами, який вимагатиме нових переговорів за контрактом із замовником [4].

Зате команда, дійсно співпрацює з клієнтами, має можливість внести всі необхідні зміни в ході робіт. Ось що означає безперервна поставка. Саме тому гнучкі методології, як правило, ітеративна. Agile-команди планують ітерації проекту шляхом вибору показників і вимог, які забезпечують максимальну віддачу. Єдиний спосіб, за допомогою якого команда може з'ясувати, які показники реалізують цю цінність, - співпраця з клієнтом і вбудовування зворотного зв'язку від попередньої ітерації. Це дозволяє команді задовольнити запити споживача в короткостроковій перспективі, демонструючи цінність продукту на ранньому етапі співробітництва, а в довгостроковій перспективі надати йому готовий продукт, що володіє всіма можливими цінними якостями.

Команда використовує ітерації, щоб розбити проект на частини з регулярними термінами здачі. Під час ітерацій команда поставляє робоче ПЗ. Після закінчення кожної ітерації команда проводить демонстрацію, показуючи клієнту створений продукт, а також попередні варіанти, щоб подивитися, які уроки можна витягти з даної ситуації. Потім починають сеанс планування, щоб з'ясувати, що вони будуть створювати в наступній ітерації. Передбачуваний графік і постійні точки контролю допомагають команді відстежити ці зміни на ранніх стадіях і створюють атмосферу, в якій не прийнято шукати винуватого, коли кожен може обговорити зміни і узгодити стратегію, щоб включити її в проект [5].

У цей момент Agile стає привабливою для традиційного командноадміністративного менеджера проекту. Такий менеджер хоче контролювати термін. Установка обмежень на тривалість ітерацій дає йому цю можливість. Крім того, вирішується також одна з головних задач менеджера - робота зі змінами, які виникають в самому кінці проекту.

Один з найбільш складних моментів у роботі традиційного менеджера - моніторинг змін. Щоденні звіти і ретроспективи ітерацій дозволяють керівнику проекту заручитися підтримкою всіх членів команди, адже вони тепер очі і вуха керівника, що допомагають виявити необхідність змін, перш ніж вони стануть причиною більш серйозних проблем в проекті. Роль менеджера проекту в тому, щоб ініціювати заміну командно-адміністративної системи управління, при якій йому необхідно щодня давати завдання членам команди і

постійно коригувати план робіт, щоб направити їх в потрібне русло. Тепер менеджер взаємодіє з командою лише для того, щоб переконатися: кожен фахівець бачить єдину картину і трудиться над загальними цілями. Найпростіше це виконати, коли команда працює в режимі коротких ітерацій, що дозволяють постачати робоче програмне забезпечення.

Це ставить перед кожним учасником конкретні цілі і дає йому вичерпне уявлення про те, над чим працює команда, а також почуття відповідальності не тільки за свою роботу, а й за загальний результат після закінчення ітерації.

Канбан - це метод поліпшення процесів, використовуваних гнучкими командами. Команди, які застосовують його, починають розуміти, як вони створюють програмне забезпечення, і поступово покращують його. Канбан, так само як Scrum і XP, зачіпає образ мислення людини. Зокрема, він вимагає бережливого мислення. В свою чергу, Lean - це образ мислення, цінності та принципи. Команди, що використовують Канбан, почали з застосування світогляду Lean. Це забезпечує міцний фундамент, який в поєднанні з Канбаном дає можливість поліпшити процеси. Коли команди використовують Канбан для удосконалення, вони зосереджені на усуненні втрат з процесів. Канбан - це технологічний термін, адаптований Девідом Андерсоном для розробки програмного забезпечення [6].

Ось як він описує взаємовідносини з Lean в своїй книзі «Канбан. Альтернативний шлях в Agile » [7]:« Канбан-метод являє собою складну адаптивну систему, призначену для активації lean-рішень в рамках організації». Є команди, які застосовують Lean і бережливе мислення для розробки програм без використання канбан, але на сьогоднішній день це найбільш поширений метод (і ефективний для багатьох agile-практиків) впровадження бережливого мислення в організацію. Канбан відрізняється від гнучких методологій, таких як Scrum і XP.

Scrum переважно орієнтований на управління проектами: обсяг роботи, який повинен бути проведений, щоб ця робота була виконана, і результат, необхідний користувачам і стейкхолдерам. XP орієнтована на розробку програмного забезпечення, її цінності і практики будуються навколо створення сприятливих умов для розвитку і формування звичок, які допомагають розробнику писати простий і легко змінюваний код. Канбан допомагає команді поліпшити способи розробки програмного забезпечення. Команда, яка використовує цей метод, має чітке уявлення про те, які дії робить при створенні програмного продукту, як взаємодіє з іншою компанією, як утворюються втрати, викликані неефективністю і нерівномірністю, і яким чином з часом виправити ситуацію, усунувши основну причину цих втрат. Коли команда покращує спосіб створення ПО, це традиційно називається процесом удосконалення. Канбан - хороший приклад застосування гнучких ідей (таких як останній момент) для створення методу удосконалення процесу, який команди можуть легко прийняти [].

Практики КАНБАН дозволяють стабілізувати і поліпшити систему розробки програмного забезпечення. Актуальний список основних практик можна знайти в групі Kanban Yahoo! По-перше, слід дотримуватися основоположних принципів:

• Починати з того, що робите зараз, зберігати наявні ролі, обов'язки та посадові інструкції.

• Домовитися про еволюційний розвиток.

• Заохочувати лідерство на всіх рівнях.

Потім слід застосовувати основні практики: візуалізацію; обмеження числа завдань в роботі (WIP); управління потоком; зробити правила явними; ввести петлі зворотнього зв'язку; розвиватися разом і експериментувати (використовуючи моделювання або науковий підхід).

На початковому етапі не потрібно впроваджувати всі шість практик. Часткова реалізація визнається поверхневої і передбачає поступове поглиблення, оскільки більшість практик приймаються і виконуються з великим потенціалом.

Канбан - це не методологія розробки програмного забезпечення і не система управління проектами. Одна з найпоширеніших помилок на початку вивчення КАНБАН - це спроба використовувати його як методологію для розробки програмного забезпечення.

Канбан - це спосіб поліпшення процесів. Він допоможе зрозуміти методику, що використовується, і знайти способи вдосконалити її. Канбан оперує не процесами, а робочими елементами. Так називається самостійна одиниця роботи, яку потрібно відстежити через всю систему. Як правило, вимоги, призначені для користувача історії або подібний фрагмент загального пулу робіт. Існує різниця між дошкою завдань і канбан-дошкою: в той час як завдання переміщаються по дошці завдань, робочі елементи не є завданнями. Завдання - це те, що роблять люди, щоб перемістити робочі елементи в рамках системи, тобто це «гвинтики» механізму, який штовхає робочі елементи.

Саме тому можена використовувати системне мислення для розуміння робочого процесу при створенні програмного забезпечення, не принижуючи людську гідність розробників думкою, ніби вони частини якоїсь машини. Завдання - це «гвинтики», а люди - унікальні особистості з власним характером, бажаннями і мотивацією. Стовпці на канбан-дошці можуть здатися схожими на етапи в потоці цінності.

Однак, багато канбан-експерти розділяють поняття «систематизація потоку цінності» і «канбан-дошка». Вони відображають на дошці стан робочих елементів в робочому процесі незалежно від потоку цінності і називають це картою життєвого циклу. Різниця в тому, що потік створення цінності - інструмент бережливого мислення, допомагає зрозуміти роботу системи, а карта життєвого циклу - це те, як канбан-метод визначає конкретні етапи, які проходить кожен робочий елемент.

Scrum допомагає командам в самоорганізації та виконання колективних зобов'язань. До моменту початку використання дошки завдань команда вже вибрала елементи беклогу (робочі елементи), які включені в спринт, і розбила їх на завдання. Так як поточні завдання переміщаються по дошці завдань, робочі елементи починають переміщатися з колонки «зробити» в колонку «зроблено». Команда сприймає це як прогрес[7].

Канбан - це метод, який спрямований на поліпшення процесу, базується на цінностях Lean і бережливого мислення. Він був розроблений Девідом Андерсоном, який першим почав експериментувати з ідеями Lean під час роботи в Microsoft і Corbis. Так само як Lean, назва «канбан» пов'язане з ідеями розробки на автомобільних виробництвах в Японії. Але що робить Канбан гнучким? Чим він відрізняється від традиційного процесу поліпшення?

Команди розробки програмного забезпечення займаються удосконаленням процесів з тих пір, як люди почали створювати ПЗ. В ідеалі вдосконалення процесів працює дуже добре: команда отримує підтримку з боку керівництва, проводить виміри, визначає проблеми, покращує знаряддя праці, а потім знову починає виявляти те, що можна вдосконалити. Зрештою поліпшення всієї організації в першу чергу дозволяє створювати повторювані процеси, потім управляти ними і нарешті взяти під статистичний контроль. Вже багато компаній заявили про великі успіхи в цій галузі. Якщо ви розробник, вже пережив типову спробу удосконалення процесів, то, швидше за все, поспішіть відкласти книгу в сторону, прочитавши цей опис.

Термін «удосконалення процесу» асоціюється з нескінченними нарадами і процесами, повними технологічної документації, тому що типове поліпшення процесу сильно відрізняється від ідеального. У разі типового процесу вдосконалення велика компанія вирішує, що їх програмісти виробляють програмне забезпечення недостатньо ефективно (або що вони потребують сертифікації процесу для укладання контрактів або для маркетингових цілей), тому наймають консалтингові компанії, витрачають багато часу (і грошей) на створення блок- схем існуючих і бажаних процесів розвитку і навчають команди використовувати нові процеси.

Потім команди витрачають близько 10 хвилин, щоб випробувати один з нових процесів, з'ясовують, що він їм не підходить, і відмовляються від нього. Але оскільки топ-менеджери проспонсорували весь процес удосконалення зусиль, команда змушена робити вигляд, що використовує нововведення, тому старанно заповнює будь-які документи за новими вимогами (наприклад, область застосування документа і технічне завдання) і створює обов'язкові артефакти (протоколи зборів для перегляду коду та звіти про тестування, які, як правило, порожні і шаблонні).

Для кожного успішного процесу вдосконалення зусиль (їх кілька) є багато нічого не дають або зовсім невдалих спроб докладання зусиль, побічний продукт яких - глибока неприязнь до терміну «покращення процесу». Існує одне велике відмінність між Канбаном і традиційним процесом поліпшення. В останньому випадку рішення, як правило, спонсоруються вищим менеджментом, готуються комітетом (наприклад, групами процесів розробки програмного забезпечення) і доводяться до відома команд через їх безпосередніх керівників. У Канбані поліпшення залишається в руках команди. Саме тому agileкоманди домоглися успіху в застосуванні цього методу. Члени команди самостійно знаходять проблеми в робочому процесі, пропонують свої варіанти поліпшення, оцінюють результати і беруть на себе відповідальність за власні стандарти.

Перший етап в поліпшенні процесу - це розуміння того, як в даний час працює команда, і практика візуалізації в Канбані дозволяє це зробити. Проте все відбувається набагато складніше, ніж здається, тому що багато традиційних процесів вдосконалення йдуть неправильно.

Візуалізація робочого процесу дозволяє команді розпізнати перевантаження. Це перший крок до усунення проблеми. Нерівномірність і перевантаження проявляються на канбан-дошці, коли стікери збираються в одному стовпці. На щастя, теорія масового обслуговування не тільки попереджає нас про проблему, а й пропонує спосіб її виправити.

Після виявлення неравномерностей в робочому процесі ми можемо управляти об'ємом робіт у всій системі, поставивши жорстке обмеження на виконання незавершеної завдання. Така канбан-практика називається обмеження на виконання незавершених робіт (limit work in progress, WIP-ліміт). Обмеження на виконання незавершених робіт означає встановлення обмеження на кількість робочих елементів, які можуть перебувати на різних стадіях реалізації проекту.

Як правило, більшість співробітників зосереджені на тому, щоб якомога швидше пересунути робочі елементи всередині процесу. Для одного елемента цей робочий процес - лінійний: якщо ви програміст і закінчили написання коду для функціоналу, а ваш робочий процес вимагає, щоб ви ще й протестували його, то легко почати непотрібну поспіх [8].

Візуалізація робочого процесу за допомогою канбан-дошки дозволяє команді побачити петлі зворотного зв'язку і експериментувати з WIP-лімітами, щоб знайти оптимальну довжину зворотного зв'язку. Це забезпечує регулярну зворотний зв'язок, причому команда встигає відповісти на зауваження перш, ніж прийде наступна партія. Потрібно уникати при цьому багаторазової відправки одних і тих же елементів через цикл зворотного зв'язку, тому що це засмічує систему. Але Канбан допомагає помітити і це. Наприклад, коли менеджери дають зворотний зв'язок команді і просять переробити, робота відправляється назад в беклог. Члену команди доводиться переміщати стікер з завданням в колонку з більш раннім етапом.

Команда може відстежувати стікери, які були зміщені назад, якщо поставить на них крапку або будь-який інший знак - явний показник того, що петля зворотного зв'язку для цієї функції буде повторюватися. Коли функція повертається через робочий процес, вона в підсумку знову опиниться в стовпці «Приймання

керівництвом» і заповнить собою одне місце серед WIP-лімітів. Це призводить до ефекту засмічення петлі зворотного зв'язку.

Команди продовжують постачати функціонал, вони виявляють проблеми робочого процесу і коригують WIP-ліміт так, щоб петлі зворотного зв'язку забезпечували достатній обсяг інформації, не викликаючи пробуксовки. Потік - це швидкість, з якою робочі елементи переміщаються по системі. Коли команда знаходить оптимальний темп для поставки в поєднанні зі зручною зворотним зв'язком, вона максимізує потік. Скорочення нерівномірної роботи і перевантажень, можливість закінчувати одне завдання і тільки потім переходити до наступного збільшує потік. Коли через нерівномірне роботи накопичуються завдання, це призводить до переривання робочого процесу і скорочення потоку.

Канбан-команда використовує практику управління потоком шляхом його вимірювання і робить активні кроки щодо його поліпшення. Ви вже знаєте, як це надихає, коли робота просувається. Ви відчуваєте, що встигаєте багато виконати, не витрачаєте час і не чекаєте, коли хтось інший завершить свою частину роботи і передасть її вам. Коли ви працюєте в команді з великим потоком, ви постійно відчуваєте, що виконуєте щось цінне. До цього прагнуть усі. Ви також знаєте, як це, коли робота буксує. Таке відчуття, ніби ви загрузли в багнюці і ледь можете рухатися. Здається, що ви вічно когось чекаєте, щоб закінчити створення необхідного, або приймаєте рішення, що впливає на роботу, погоджуєте картку, або знаходиться ще що-небудь, що перешкоджає роботі, - навіть якщо ви знаєте, що ніхто навмисно не заважає. Ви відчуваєте неузгодженість і непослідовність і витрачаєте багато часу на пояснення, чому ви чекаєте. Це не означає, що команда недостатньо завантажена, можливо, люди викладаються на 100%, а може навіть і перевантажені. Але в той час, як план проекту говорить про те, що робота виконана на 90%, вам здається, що, навпаки, ще потрібно зробити 90% роботи.

Ефективний інструмент вимірювання потоку - кумулятивна діаграма потоку (cumulative flow diagram, CFD). CFD схожа на WIP-діаграму, але має одна важлива відмінність: замість того щоб зникати, виконані робочі елементи накопичуються в нижній смузі діаграми. І якщо на WIP-діаграмі смуги відповідають станам в карті потоку створення цінності, то CFD (і WIP-діаграма) в цьому розділі має смуги, відповідні стовпцях на канбан-дошці.

Команда може експериментувати і використовувати зворотний зв'язок, щоб визначити пріоритети WIP-ліміту, які працюють для їх системи, і якщо їм вдасться отримати їх, то в кінці кінців встановиться баланс між швидкістю надходження і виконання. Тенденція довгострокового списку придбає вигляд рівної лінії, лінія швидкості надходження - теж. І як тільки це станеться, система стане стабільною. Коли система стабільна, існує простий зв'язок між цими величинами, так званий закон Літтла (час, що витрачається на виконання корисної роботи). Ця теорема - частина теорії масового обслуговування, названа на честь Джона Літтла, який вперше запропонував її в 1950-х роках і вважається «батьком маркетингу»:

$$L = W \times \lambda$$

Де L – кількість елементів у списку черги
W - час, необхідний для виконання нового замовлення
λ - швидкість надходження елементів черги

У перекладі на звичайну мову це означає, що якщо у вас є стабільний робочий процес, то середній список завжди дорівнює середній швидкості надходження, помноженої на середній час, необхідний для виконання нового замовлення. Це математичний закон: якщо доведено і якщо система стабільна, це завжди вірно. Вірно також і зворотнє:

$$L = W \div \lambda$$

Де L – кількість елементів у списку черги
W - час, необхідний для виконання нового замовлення
λ - швидкість надходження елементів черги

Якщо відома середня кількість і середню швидкість надходження, то можна розрахувати середній час виконання нового замовлення. Обчислити середню кількість і швидкість надходження досить просто: кожен день записувати на канбан-дошку загальне кількість робочих елементів, а також число, яке було додано в першу колонку в той же день.

Якщо система працює стабільно, то через деякий час можна побачити, що середній щоденний список і щоденна швидкість надходження виглядають як товсті прямі лінії. Розділіть середній список на середню швидкість надходження, і ви отримаєте час на виконання нового замовлення.

Таким чином, багато команд, що працюють над питанням медичних черг експериментальним шляхом приходять до цікавого висновку: вони виявлють, що після введення WIP-лімітів з'явилася можливість контролювати час очікування пацієнтом лікарського прийому. Якщо на кожну годину планувалося багато візитів, то чотирьох-п'яти пацієнтам довелося б довго чекати. А якщо запис скорочувалася, то перед кабінетом лікаря сиділи б тільки два-три пацієнта, і чекати їм довелося б менше.

Співробітники клініки виявили існування в стабільній системі зв'язку між списком прийому (кількістю), часом обслуговування і швидкістю надходження. Наприклад, якщо персонал щогодини записує 11 пацієнтів (так як швидкість надходження λ дорівнює 1 годині) і середня кількість протягом дня перевищує сім пацієнтів (так як L дорівнює 7), то закон Літтла говорить нам, що середній час очікування пацієнтами свого прийому складе :

$$W = L \div \lambda = 7 \text{ пацієнтів} \div (11 \text{ годин}) = 0{,}63 \text{ години} = 37 \text{ хвилин}$$

Але раптом після ряду експериментів виявляється, що планування 10 пацієнтів, які прибувають кожну годину, знижує список до чотирьох пацієнтів. У годину пік всі процедурні переповнені, але більшу частину часу один пацієнт знаходиться в приймальному покої, один чекає на прийом лікаря в процедурній кімнаті та двоє - на прийомі у лікаря:

W = 4 пацієнта ÷ (10 пацієнтів кожну годину) = 0,4 години = 24 хвилини. За допомогою канбан-дошки, CFD і експериментування з WIP-лімітом співробітники клініки виявили, що могли б зменшити час очікування пацієнта майже на 15 хвилин тільки за рахунок скорочення запису на одну людину в годину. Це працює, тому що закон Літтла говорить нам, що в стабільній системі на час обслуговування пацієнта впливає дві речі: черга і швидкість прибуття. А WIP-ліміти дозволяють контролювати одну з них. При додаванні WIP-ліміту на канбан-дошку ви можете зменшити нерівномірність, яка призводить до нагромадження в черзі.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі. Одна з важливих цілей, які ставить перед собою канбан-команда, - максимізація потоку або швидкості, з якою робочі елементи переміщаються по системі. Канбан-практика вимірювання і управління потоком на увазі вимір потоку і коригування процесу для його максимізації. Кумулятивна діаграма потоку (CFD) показує ліміт на виконання незавершених робіт (WIP-ліміт), число робочих елементів, що додаються щодня (швидкість надходження), загальна кількість робочих елементів в робочому процесі (список) і середній час перебування робітників елементів в системі (час на виконання замовлення). Коли швидкість надходження і список з часом не змінюються, система стабільна, канбан-команди встановлюють WIP-ліміт для її стабілізації. Коли система стабільна, до неї можна застосовувати закон Літтла, який означає, що середній час на виконання замовлення завжди одно довгостроковій швидкості надходження, поділеної на довгостроковий список. Якщо команда може стабілізувати робочий процес за допомогою WIP-лімітів, то команда зуміє зменшити час виконання замовлення для користувачів за умови, що вони не будуть додавати нові робочі елементи, що скорочують швидкість надходження. Що і потимізує загаьний процес використання грнучких методологій.

Список літератури.

1. Бабаєв В.М. Управління проектами: Навчальний посібник для студентів спеціальності «Управління проектами» / Бабаєв В.М. – Харків: ХНАМГ, 2006. – 244 с.
2. Батенко Л. П. Управління проектами: Навч. посібник / Батенко Л. П., Загородніх О. А., Ліщинська В. В— К.: КНЕУ, 2003. — 231 с.
3. Ефимов В. В. Улучшение качества проектов и процессов: Учебное пособие /В. В. Ефимов. – Ульяновск: УлГТУ, 2004. - 185 с.
4. Крайнік О.М.. Планування проектних дій: навчально-методичний посібник для студентів ЗДІА спеціальності 8.18010013“Управління проектами» денної форми навчання / О.М. Крайнік, Н.І. Тахтаджиєва – Запоріжжя, ЗДІА, 2015. – 80 с.
5. Мазур И.И. Управление проектами: Учебное пособие / Мазур И.И., Шапиро В.Д., Ольдерогге Н.Г.; Под общ. ред. И.И. Мазура. – 2-е изд. – М.: Омега-Л., 2004. – 664с.
6. Ноздріна Л.В. Управління проектами: підручник / Ноздріна Л.В., Ящук В.І., Полотай О.І./ За заг.ред.Л.В.Ноздріної. – К.: Центр учбової літератури, 2010. – 432с.
7. Руководство к своду знаний по управлению проектами, 5-е издание/ Project Management Institute (PMI). – Project Management Institute, Inc., 2012. – 614 с.
8. Управление проектами: учебник для бакалавров / А. И. Балашов, Е. М. Рогова, М. В. Тихонова, Е. А. Ткаченко; под ред. Е. М. Роговой. —М. : Издательство Юрай, 2013. — 383 с.

References.

1. Babaev, V.M. (2006), Upravlinnia proektamy [Project Management], KHNAMG, Kharkiv, Ukraine.
2. Batenko, L.P. Zagorodnikh, O.A. and Lischinska V.V. (2003), Upravlinnia proektamy [Project Management], KNEU, Kyiv, Ukraine.
3. Efimo, V.V. (2004), Uluchshenie kachestva proektov i processov [Improving the quality of projects and processes:], UlSTU, Ulyanovsk, RF.
4. Kraynik, O.M. and Takhtadzhieva, N.I. (2015), Planuvannia proektnykh dij [Planning project activities], ZDIA, Zaporizhzhia, Ukraine.
5. Mazur, I.I. Shapiro, V.D. and Olderogge, N.G .(2004), Upravlenie proektami [Project Management, Omega-L., Moscow, RF.
6. Nozdrina, L.V. Yashchuk, V.I. and Polotay, O.I.(2010), Upravlinnia proektamy [Project management], Center for Educational Literature, Kyiv, Ukraine.

7. Project Management Institute (PMI) (2012), Rukovodstvo k svodu znanij po upravleniju proektami, 5-e izdanie [Guide to the Project Management Body of Knowledge, 5th Edition], Project Management Institute, Inc., Moscow, RF.

8. Balashov, A. I. Rogova, E. M. Tikhonova, M. V. and Tkachenko, E. A. (2013), Upravlenie proektami [Project management], Publishing house Yuray, Moscow, RF.

Стаття надійшла до редакції 19.01.2020 р.