1. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

1.1.1 Аналіз предметної області

1.1.1.1 Чат бот

Чат бот – це програмне забезпечення або програмний продукт що дозволяє здійснювати комунікацію між людьми за допомогою засобів які використовують аудіо або частіше з використанням тексту та текстових повідомлень. Такі програми часто спроектовані для симуляції того як спілкуються люди між собою. На момент 2019 року розвиток чат ботів досягнув такого рівня що вони в подальшому зможуть пройти тест Тюрінга завдяки використанню нейромереж та машинного навчання. Крім того чат боти використовуються в випадках де є спілкування з клієнтами задля автоматизації та спрощення процесу надання допомоги клієнтам або надання ним певної інформації тощо. В процесі спілкування з людьми чат боти з складнішою будою навчаються розуміти людську мову з певними аспектами та діалектами але зважаючи на складність таких систем найчастіше використовують підхід де програма знаходить ключові слова в тексті та дає на основі цього відповідь.

Схожими за принципом роботи до чат ботів є інтернет боти які являють собою скрипт або програмне забеспечення яке виконує певне завдання в мережі інтернет. Боти виконують завдання які за своєю структурою є повторюваними та простими набагато швидше ніж це робила б людина. Найчастіше використовуються пошукові боти, це боти які мають певний скрипт та алгоритми знаходження, пошуку та аналізу знайденої інформації та файлів з веб-сайтів та серверів роблячи це набагато ефективніше та швидше за людей. Більше половини трафіку в мережі інтернет створюється такими ботами.

Ставлення щодо ботів в власників хостингів різняться. Сервери можуть містити певні файли які вказують ботам що вони можуть робити на даному ресурсі. В теорії боти які порушують ці правила повинні бути видалені або заблоковані на цьому ресурсі проте насправді все залежить від власника бота який вирішує чи дотримуватись цих правил чи ні.

1.1.1.2 Месенджер Telegram

Telegram – це сервіс миттєвого обміну повідомленнями між користувачами з використанням хмарних технологій. Клієнти телеграму доступні на всіх актуальних платформах на сьогоднішній день. Користувачі можуть обмінюватись різними типами медіа матеріалів, наліпками, файлами будь-якого типу.

Клієнтська частина є в відкритому доступі проте вихідний код останніх версій виходить з затримкою після виходу самого клієнта в той час серверна частина має закритий код та є пропрієтарним продуктом компанії. Телеграм надає засоби для розробників для всіх охочих. В 2018 році було досягнуто 200 мільйонів активних користувачів.

Звичайні повідомлення зберігаються зашифрованими на сервері і можуть бути прочитаними тільки тими хто має доступ до ключів шифрування. Крім того є наскрізне шифрування коли ці ключі мають лише відправник та отримувач. Взаємодія клієнта та сервера відбувається з шифруванням, дзівнки в мережі також мають наскрізне шифрування.

Система безпеки телеграму отримала критику з сторони експертів криптографії. Вони розкритикували те що дані користувачів і повідомлення та ключі шифрування зберігаються разом на сервері хоча можна було застосувати наскрізне шифрування за замовчуванням проте таке рішення не було прийняте в зв’язку з ризиком того що ключі шифрування можуть бути скомпрометовані користувачем.

Аккаунти в телеграмі прив’язані до номеру телефона і верифікуються за допомогою смс повідомлень та за вибору двох факторною аутентифікацією що додає безпеки користувачу. Користувачі можуть додавати декілька аккаунтів до клієнту і використовувати їх паралельно, також з’явилась можливість змінювати номер до якого прив'язаний аккаунт. Двох факторна аутентифікація являє собою пароль який створює користувач в налаштуваннях. Ввімкнення двох факторної аутентифікації дозволяє використовувати телеграм паспорт - сервіс для зберігання особистих документів таких як паспорт, посвідчення водія та інші. Такі міри застосовуються в зв'язку з тим що смс або сім карта можуть бути перехоплені або скомпрометовані.

Повідомлення в телеграмі можуть бути прочитані на будь якому пристрої користувача. Користувачі можуть ділитись фото, відео, аудіо повідомленнями та навіть файлами об’ємом до 1.5 гігабайт для 1 файла. Повідомлення можуть бути надіслані напряму до користувачів або у групи які можуть налічувати до 100 тисяч користувачів. Всі повідомлення та файли які зберігаються на серверах залишаються там доки останній користувач який має до них доступ не видалить їх у себе з повідомлень.

Починаючи з 2015 року телеграм дозволяє розробникам створювати ботів. Завдяки інструментам для розробників стало можливо створити ботів які будуть мати найрізноманітніший функціонал від відправки текстових повідомлень до опрацювання платежів та навіть роботи з групами користувачів.

У червні 2015 року Telegram запустив платформу для сторонніх розробників для створення ботів.Боти - це рахунки телеграм, якими керуються програми. Вони можуть відповідати на повідомлення або згадки, можуть бути запрошені в групи і можуть бути інтегровані в інші програми. Він також приймає онлайн-платежі за допомогою кредитних карток та Apple Pay. Запрошений бот може потенційно читати всі групові повідомлення, коли контролер бота мовчки змінює налаштування доступу в наступний момент

часу. Телеграм зазначив, що розглядає можливість використання функції, яка оголосить про таку зміну статусу у відповідній групі. Також є вбудовані боти, які можна використовувати з будь-якого екрана чату. Для активації вбудованого бота користувачеві необхідно ввести в поле повідомлення ім’я користувача та запит бота. Тоді бот запропонує свій вміст. Користувач може вибрати цей вміст і надіслати його в чаті.

У вересні 2015 року Telegram додав канали. Канали - це форма одностороннього обміну повідомленнями, де адміністратори можуть розміщувати повідомлення, але інші користувачі - ні. Будь-який користувач може створити та передплатити канали. Канали можуть бути створені для трансляції повідомлень необмеженій кількості передплатників. Канали можуть бути загальнодоступними з псевдонімом та постійною URL-адресою, щоб кожен приєднався. Користувачі, які приєднуються до каналу, можуть бачити всю історію повідомлень. Користувачі можуть в будь-який час приєднуватися та залишати канали. Залежно від налаштувань каналу, повідомлення можуть бути підписані з ім'ям каналу або з ім'ям користувача адміністратора, який їх розмістив. Користувачі, які не є адміністратором, не можуть бачити інших користувачів, які підписалися на канал. Крім того, користувачі можуть ігнорувати канал, тобто користувач все одно отримуватиме повідомлення, але не отримуватиме повідомлення. Адміністратори можуть дати дозвіл на публікацію коментарів на телеканалі Telegram за допомогою ботів. Адміністратор каналу може отримати загальні дані про канал. Кожне повідомлення має власний лічильник перегляду, який показує, скільки користувачів бачили це повідомлення. Станом на травень 2019 року творець каналу може додати групу обговорень, окрему групу, де повідомлення в каналі автоматично розміщуються для підписників для спілкування.

Наклейки - це зображення, що базуються на хмарі, із високою чіткістю, які мають надати більш виразні смайли. Під час введення емоджи користувачеві пропонується надіслати відповідну наклейку замість цього.

Наліпки поставляються в колекціях, що називаються "набори", і для однієї смайлики можна запропонувати кілька наклейок. Telegram постачається з одним набором наклейок за замовчуванням, але користувачі можуть встановлювати додаткові набори наклейок, надані сторонніми учасниками. Набори наклейок, встановлені від одного клієнта, стають автоматично доступними для всіх інших клієнтів. Зображення наклейок використовують формат файлу WebP, який краще оптимізувати для передачі по інтернету.

Чернетки - це незавершені повідомлення, синхронізовані на пристроях користувача. Можна почати вводити повідомлення на одному пристрої та продовжувати на іншому. Чернетка зберігатиметься в області редагування на будь-якому пристрої, поки її не буде надіслано або видалено.

Повідомлення також можуть надсилатися із шифруванням від клієнта до клієнта у так званих таємних чатах. Ці повідомлення шифруються протоколом MTProto служби. На відміну від хмарних повідомлень Telegram, до повідомлень, надісланих у таємному чаті, можна отримати доступ лише на пристрої, на якому було розпочато секретне чат, і на пристрої, на якому було прийнято секретний чат; до них неможливо отримати доступ до інших пристроїв. Повідомлення, надіслані в таємних чатах, в принципі можуть бути видалені в будь-який час і, за бажанням, можуть бути самознищені.Секретні чати повинні бути ініційовані та прийняті запрошенням, після чого обмінюються ключі шифрування сеансу. Користувачі в таємному чаті можуть переконатися, що жодної посередньої атаки не відбулося, порівнюючи фотографії, які візуалізують відбитки їхніх відкритих ключів. За даними Telegram, таємні чати підтримують ідеальну таємницю прямої передачі з грудня 2014 року. Ключі шифрування періодично змінюються після того, як ключ використовувався більше 100 разів або використовується більше тижня. Старі ключі шифрування знищені. Користувачі Windows та Linux досі не в змозі використовувати таємні чати за допомогою офіційного додатку Telegram Desktop, тоді як офіційний клієнт, що підтримує лише macOS, підтримує їх. Секретні чати недоступні для груп чи каналів.Місцева

база даних повідомлень Telegram за замовчуванням не шифрується. Деякі клієнти Telegram дозволяють користувачам шифрувати локальну базу повідомлень, встановлюючи парольну фразу.

Наприкінці березня 2017 року Telegram представила власні голосові дзвінки. Виклики побудовані на основі шифрування таємних чатів. З'єднання встановлюється як рівномірний, коли це можливо, інакше використовується найближчий сервер до клієнта. Як повідомляє Telegram, існує нейронна мережа, яка працює для вивчення різних технічних параметрів щодо виклику, щоб забезпечити кращу якість послуги для подальшого використання. Після короткого початкового випробування у Західній Європі голосові дзвінки тепер доступні для використання у більшості країн.

Починаючи з версії 4.0, випущеної в травні 2017 року, Telegram пропонує спеціалізовану платформу для відеохостингу під назвою Telescope. Круглі відеозаписи можуть тривати до однієї хвилини та автоматично відтворюватись. Коли вони розміщені на загальнодоступному каналі Telegram, відеозаписи також завантажуються та переглядаються без акаунта на telesco.pe. Однак відео-повідомлення Telegram та відеотелефони "Телескоп", надіслані в межах непублічних чатів або груп, не публікуються.

Протягом 15 хвилин, однієї години або восьми годин користувачі Telegram можуть поділитися своїм прямим місцезнаходженням у чаті з моменту версії 4.4, випущеної у жовтні 2017 року. Якщо декілька користувачів діляться своїм поточним місцезнаходженням у групі, вони відображаються на інтерактивній карті. Спільний доступ до "прямого місцезнаходження" можна зупинити будь-коли.

У лютому 2018, Телеграма запустила свій соціальний Логін функцію для своїх користувачів, названий як Телеграма Логін. У ньому є віджет веб-сайту, який можна вбудувати на веб-сайти, дозволяючи користувачам входити на веб-сайт третьої сторони з їх обліковим записом Телеграма. Шлюз відправляє користувачів Телеграма ім'я, ім'я користувача і зображення

профілю до власника веб-сайту, в той час як номер телефону користувачів залишається прихованим. Шлюз інтегрований з бота, який пов'язаний з конкретним веб-сайтом розробника.

У липні 2018, Телеграма представила свою онлайн-авторизацію і систему управління ідентифікацією, паспорта Телеграма, для платформ, які вимагають реального життя ідентифікації. Він просить користувачів завантажити свої офіційні документи, такі як паспорт, посвідчення особи, водійські права і т. д. Коли онлайнова служба потребує таких ідентифікаційних документів і перевірок, він передає інформацію на платформу з дозволом користувача. Телеграма заявила, що він не має доступу до даних, в той час як платформа буде ділитися тільки інформацією з авторизованим одержувачем. Однак, служба була критикувала за те, що вразливі для онлайн атак грубої сили.

1.1.1.3 Людинно-машинна взаємодія

Люди взаємодіють з комп'ютерами багато в чому; інтерфейс між людьми і комп'ютерами має вирішальне значення для полегшення цієї взаємодії. Настільні програми, Інтернет-браузери, кишенькові комп'ютери, ERP та комп'ютерні кіоски використовують поширені графічні інтерфейси користувача (GUI) сучасності .Голосові інтерфейси (VUI) використовуються для систем розпізнавання та синтезування мовлення, а нові багатомодальні та графічні інтерфейси (GUI) дозволяють людям взаємодіяти з втілено характер агентів таким чином, що не може бути досягнуто з іншими парадигм інтерфейс. Зростання людини-комп'ютерна взаємодія області була в якості взаємодії, і в різних розгалужень в його історії. Замість того, щоб проектування регулярних інтерфейсів, різні наукові гілки мали різні зосередити увагу на концепціях мультімодальності, а не уніальність, інтелектуальні адаптивні інтерфейси, а не командування/дії на основі них, і, нарешті, активні, а не пасивні інтерфейси.

Асоціація обчислювальної техніки (ACM) визначає людино-комп'ютерну взаємодію як "дисципліну, пов'язане з дизайном, оцінкою та впровадженням інтерактивних обчислювальних систем для використання людиною і з вивченням основних явищ, що їх оточують". Важливим аспектом HCI є задоволеність користувачів (або просто задоволеність кінцевого користувача). "Тому що людина-комп'ютерна взаємодія вивчає людину і машину в спілкуванні, він звертає від підтримки знань як на машині

і в людській стороні. На боці машини актуальні методики в комп'ютерній графіці, операційних системах, мовах програмування та середовищах розробки.

З боку людини, теорія комунікацій, графічних і промислових дисциплін дизайну, лінгвістики, соціальних наук, когнітивна психологія, соціальна психологія, і людські фактори, такі як задоволеність користувачів комп'ютера є актуальними. І, звичайно, актуальні методи проектування та дизайну. " Завдяки міждисциплінарній природі HCI, люди з різним досвідом сприяють її успіху. HCI також іноді називають людина-машина взаємодії (HMI), людино-машинної взаємодії (MMI) або комп'ютер-людська взаємодія (CHI). Погано продумані людино-машинної інтерфейси можуть призвести до багатьох несподіваних проблем.

Класичним прикладом є три милі острова аварії, аварії ядерної кризи, де розслідування висновку, що дизайн людини-машина інтерфейс, принаймні частково відповідальність за катастрофу. Ааналогічним чином, нещасні випадки в авіації призвели до рішень виробників, щоб використовувати нестандартний інструмент польоту або макети дросельної квадранта: навіть якщо нові проекти були запропоновані вище в основних людино-машинної взаємодії, пілоти вже вкоренилася "Стандартний" Макет і, таким чином, концептуально гарною ідеєю дійсно були небажані результати.

Людина-комп'ютерна взаємодія дослідження шляхів, в яких люди роблять-або не роблять-використання обчислювальних артефактів, систем і інфраструктури. Значна частина досліджень в області прагне до поліпшення людини-комп'ютерна взаємодія шляхом поліпшення юзабіліті комп'ютерних інтерфейсів. Як зручність слід розуміти, як це відноситься до інших соціальних і культурних цінностей, і коли вона є, і коли воно не може бути бажаним властивістю комп'ютерних інтерфейсів, все частіше обговорюється.

Значна частина досліджень в області прав людини – комп'ютерна взаємодія бере на себе інтерес до: Методи розробки нових комп'ютерних інтерфейсів, тим самим оптимізуючи конструкцію для бажаної властивості, наприклад, здатність до навчання, вміння, ефективність використання.

Методи реалізації інтерфейсів, наприклад, за допомогою програмних бібліотек. Методи оцінки та порівняння інтерфейсів щодо їх юзабіліті та інших бажаних властивостей.

Методи вивчення людського використання комп'ютера та її соціокультурні наслідки більш широко. Методи визначення, чи є користувач людиною або комп'ютером. Моделі та теорії використання комп'ютерних інтерфейсів, а також концептуальних фреймворків для проектування комп'ютерного інтерфейсу, таких як моделі, теорії діяльності або рахунки використання людиною комп'ютера.

Перспективи, які критично відображають цінності, які лежать в основі обчислювальної конструкції, комп'ютерного використання і HCI дослідницької практики. Бачення того, що дослідники в області прагнуть досягти різні. При досягненні когнітивної точки зору, дослідники HCI

можуть прагнути до узгодження комп'ютерних інтерфейсів з ментальна модель, що люди мають свою діяльність. Коли ви переслідуєте перспективу постпізнання, дослідники HCI можуть прагнути узгодити комп'ютерні інтерфейси з існуючими соціальними практиками або існуючими соціокультурними цінностями.

Дослідники в HCI зацікавлені в розробці методології проектування, Експериментуючи з пристроями, Прототипування програмних і апаратних систем, вивчення парадигм взаємодії, а так і розробку моделей і теорій взаємодії.

1.1.1.4 Логістика

Логістика – це, як правило, детальна організація та впровадження комплексної операції. У загальному розумінні бізнесу, логістика-це управління потоком речей між точкою відліку і точкою споживання відповідно до вимог замовників або корпорацій. Ресурси, що керуються в логістиці, можуть включати матеріальні вантажі, такі як матеріали, обладнання, витратні матеріали, а також харчові та інші витратні предмети. Логістика фізичних предметів зазвичай включає в себе інтеграцію інформаційного потоку, обробку матеріалів, виробництво, упаковку, інвентар, транспортування, складування, і часто безпеку. У військовій науці, логістика пов'язана з підтримкою армії лінії постачання, порушуючи тих з ворога, оскільки збройні сили без ресурсів і транспорту беззахисних. Військова логістика вже практикувалося в стародавньому світі і оскільки сучасні військові мають значну потребу в логістичних рішеннях, розроблені передові реалізації.

У військовій логістиці, співробітники логістики керують тим, як і коли переміщати ресурси в місця, які вони потребують. Управління логістикою є частиною управління ланцюжком поставок, який планує, реалізує, і контролює ефективне, ефективне вперед, і зворотний потік і зберігання товарів, послуг і пов'язаної інформації між точкою походження та точки споживання для задоволення вимог замовника. Складність логістики може бути моделюється, проаналізована, візуалізується і оптимізована за допомогою виділеного імітаційного програмного забезпечення. Мінімізація використання ресурсів є загальною мотивацією у всіх логістичних полях. Професійна робота в галузі управління логістикою називається логіком.

Автоматизація логістики – застосування комп'ютерного програмного забезпечення або автоматизованої техніки для підвищення ефективності логістичних операцій. Зазвичай це стосується операцій у складі або дистриб'юторському центрі з більш широким завданням, здійснювані системами управління Ланцюжками поставок і системами корпоративного

планування ресурсів. Промислова техніка зазвичай може ідентифікувати продукти через штрих-код або RFID-технології.

Інформація в традиційних штрих-кодів зберігається як послідовність чорно-білих смуг різної ширини, яка при прочитанні лазером перекладається в цифрову послідовність, яка за фіксованими правилами може бути перетворена в десяткове число або інші дані. Іноді інформація в штрих-коді може передаватися через радіочастотну частоту, більш типово радіопередачу використовується в RFID-тегах. Тег RFID-це картка, яка містить чіп пам'яті та антену, яка передає сигнали читачеві. RFID можна знайти на товари, тварини, транспортні засоби і люди, а також.

Логіком є професійним практикуючим логістики. Професійні логісти часто сертифіковані професійними асоціаціями. Можна працювати в чистому логістичному підприємстві, такій як судноплавна лінія, аеропорт або експедитор, або в логістичному відділі компанії. Однак, як уже згадувалося вище, логістика є широким полем, що охоплює закупівлі, виробництво, розподіл та утилізація діяльності. Отже, перспективи кар'єри також широкі. Нова тенденція в галузі-4PL, або четверта логістика, фірми, консалтингові компанії, що пропонують логістичні послуги. Деякі університети та академічні установи навчаються студентам як Логістів, пропонуючи Бакалаврат та післядипломну підготовку. Університет з головним акцентом на логістиці є університетом логістики Кюне в Гамбурзі, Німеччина. Це некомерційна та підтримка Кюне-заснування логістичного підприємця Клаус Михайла Кюне.

Чартерний інститут логістики і транспорту (сіт), створений у Сполученому Королівстві 1919, отримав королівський Статут у 1926. Дипломований Інститут є одним з професійних органів або установ для логістики і транспортної галузі, яка пропонує професійні кваліфікації або ступені в логістичному управлінні. CIT програми можуть бути вивчені в центрах навколо Великобританії, деякі з яких також пропонують варіанти дистанційного навчання. [28] інститут також має закордонні філії, а саме чартерний інститут логістики & транспорту Австралії (CITTA) [29] в Австралії і чартерний інститут логістики і транспорту в Гонконзі (КІТБК) [30] у Гонконгу.

У Великобританії програми управління логістикою проводяться багатьма університетами та професійними органами, такими як ЦИТ. Ці програми, як правило, пропонуються на післядипломного рівня. Глобальний інститут логістики [31] заснований в Нью-Йорку в 2003 році є аналітичного танка для професії і в першу чергу пов'язане з Інтерконтиненталь морської логістики. Він особливо стурбований контейнерним логістикою і роллю адміністрації морського порту в морській логістичній мережі. Інститут розробив співтовариство понад 8 500 Логістів, які виступають глобальної мережі знань, скоєних на підтримку місії інституту сприяння у вирішенні попередніх проблем в глобальній логістиці. Проблеми, пов'язані з традиційним підходом управління єдиним транспортним режимами, Модальні системи як самостійні операції.

Ключем до подолання цих попередніх проблем є окремі групи зацікавлених сторін у логістичному ланцюжку, які активно взаємодіють один з одним. Просування цієї програми є роботою інституту. Міжнародна асоціація Логістів охорони здоров'я (IAPHL) [32] є професійною мережею, яка сприяє професійному розвитку менеджерів мережі поставок та інших, що працюють у сфері логістики громадської охорони здоров'я та товарної безпеки, з Особлива увага зосереджена на країнах, що розвиваються. Асоціація підтримує Логістів у всьому світі шляхом надання спільноти практики, де члени можуть мережі, обмінюватися ідеями і вдосконалювати свої професійні навички.

Вантажі, тобто перевезення товарів, можуть бути переміщені через різні транспортні засоби і організовані в різних категоріях відвантаження. Пристрій навантаження, як правило, зібрані в більш високі стандартизовані одиниці, такі як: Контейнери ISO, своп органів або напівпричепів. Особливо на дуже великі відстані, перевезення продукції, швидше за все, виграє від використання різних транспортних засобів: мультимодальних перевезень, інтермодальний транспорт (без обслуговування) і комбінованих перевезень (мінімальний Автомобільний транспорт). При переміщенні вантажу, типові обмеження є максимальною вагою і обсягом. Оператори, що займаються перевезеннями, включають: всі поїзди, дорожні машини, човни, літаки-компанії, Кур'єри, експедитори та мультимодальні транспортні оператори. Товари, що перевозяться на міжнародному рівні, зазвичай підлягають нормам Інкотермс, емітовані міжнародною торговою Палатою.

1.1.2 Постановка задачі

Завданням магістерської роботи є розробка автоматизованого веб-додатку для вирішення логістичних задач використовуючи мову Python. Результатом розробки повинен бути веб-додаток створений на мові Python який виконуватиме роль сервера.

Архітектура веб-додатку повинна бути нескладною та інтуїтивно зрозумілим для будь-якого користувача. Основні функції додатку, а саме – вирішення логістичних задач повинні бути чітко виокремлені, так як вони являються ключовими у системі. Також візуальна частина застосунку та її компоненти повинні бути виконані згідно сучасних вимог та стандартів. Навігація по додатку повинна бути продуманою та вивіреною. Застосунок

повинен працювати плавно та без жодних перебоїв. Усе повинно бути виконано з врахуванням дизайну досвіду користувача.

Після аналізу предметної області було визначено основний функціонал системи, що повинен бути реалізований в процесі розробки програмної системи. Отже, ключовими модулями застосунку є: модуль обробки фалів, модуль обробки команд, модуль збережених даних та модуль взаємодії з логістичним сервісом.

Взаємодія з користувачем з використанням машинного навчання. Інструмент, який буде використовуватися для роботи з машинним навчання, буде визначено на одному з наступних етапів проектування програмної системи.

Під підтримкою мови Python розуміється, що після успішного розпізнавання тексту, отриманий результат повинен бути відправлений для користувача на клієнт.

Порядок задач та кроків, які необхідно вирішити:

- Визначення акторів та варіантів використання системи
- Обрати модель розробки застосунку
- Обрати архітектурний шаблон додатку
- Побудувати діаграму класів
- Обрати середовище розробки та мову програмування
- Обрати бібліотеку для роботи з машинним навчання
- Обрати спосіб збереження даних
- Реалізувати систему
- Повести тестування системи

1.1.3 Пошук акторів та варіантів використання

Ознайомившись з поставленою задачею та вимогами системи було визначено, що в даній системі буде лише один актор – Користувач. Оскільки

технічним завданням до системи не передбачено користувачів системи з різним рівнем доступу до системи та різним функціоналом.

Користувач	Введення номеру накладної	Користувач вводить номер накладної який він отримав і по якому він хоче дізнатись інформацію або відстежити
Користувач	Дізнатись стан накладної	Користувач повинен мати змогу отримати актуальну інформацію щодо надання послуг логістики а саме стан обробки замовлення
Користувач	Дізнатись місце знаходження	Користувач обирає попередьо введену накладну або додає нову після чого в меню обирає потрібний пункт а саме відстеження посилки за накладною після чого обирає пункт знайти на мапі
Користувач	Дізнатись вартість доставки замовлення	Користувач, після того як система виконала успішне додавання та обробку накладної має змогу відстежити посилання та якщо воно виконцється має можливість отримати вартість послуг включно з доставкою
Користувач	Видалити накладну	Користувач, після того як система виконала успішне додавання має можливість видалити номер накладної якщо введення було помилкове або її не потрібно відстежувати
Користувач	Здійснити оплату	Користувач, після того як система виконала успішно видачу інформації щодо вартості доставки має можливість за допомогою посилання здійснити оплату товару та послуг зазначених раніше

Користувач	Зберегти результат роботи	Користувач, після того як система виконала успішно всі дії через будь-який час може продовжити роботу з даними в тому ж стані що і до цього часу

Користувач – повинен мати змогу вибрати мову, вводити номери накладних, переглядати список накладних, відстежувати стан замовлення за накладною, дізнаватись повну інформацію щодо посилки.

Після визначення основного актора системи можна приступити до визначення основних варіантів використання системи. Результат виявлення варіантів використання представлено в таблиці 1.1.

Таблиця 1.1 Виявлення варіантів використання

Актор	Назва	Опис
Користувач	Вибір мови спілкування та мови інтерфейсу	Користувач повинен мати змогу вибрати мову спілкування з ботом з певного списку мов. В залежності від обраної мови елементи керування та відповіді будуть відрізнятись.

Продовження таблиці 1.1

Користувач	Переглянути збережені накладні	Користувач повинен мати змогу переглянути історію додавання накладних які він додав
Користувач	Редагування накладної	Користувач повинен мати змогу здійснити редагування накладної якщо було виявлено помилку з боку користувача або компанії перевізника
Користувач	Зміна місця доставки	Користувач повинен мати змогу здійснити зміну місця доставки якщо попередня адреса була

		введена помилково або адреса була обрана не правильно при замовленні
Користувач	Переглянути список відділень логістичної компанії поруч	Користувач повинен мати змогу отримати список відділень логістичної компанії відправивши попередньо своє місце знаходження та його основі отримати адресу найближчих відділень
Користувач	Обрати тип доставки	Користувач повинен мати змогу здійснити вибір послуг комапнії перевізника а саме тип доставки
Користувач	Отримати інформацію щодо карти лояльності	Користувач повинен мати змогу отримати номер та інформацію щодо програми лояльності логістичної компанії для подальшого отримання бонусів або знижок на послуги
Користувач	Реєстрація користувача	Користувач повинен мати змогу здійснити реєстрацію для отримання послуг карти лояльності та задля ідентифікації якщо відсутній профіль
Користувач	Отримати вартість послуг та список самих послуг	Користувач повинен мати змогу побачити список послуг які надає логістична компанія та їх вартість

Продовження таблиці 1.1

Користувач	Повернення посилки за накладною	Користувач повинен мати змогу повернути або відмовитись від послуг перевізника або логістичної компанії за певних обставин
Користувач	Дошка оголошень	Користувач повинен мати змогу дізнатись про новини логістичної компанії або будь-які зміни в роботі компанії та вартості послуг

Після успішного виявлення варіантів використання системи було побудовано діаграму варіантів використання системи (рисунок 1.1). На даній діаграмі відображено основні варіанти використанні описані вище.

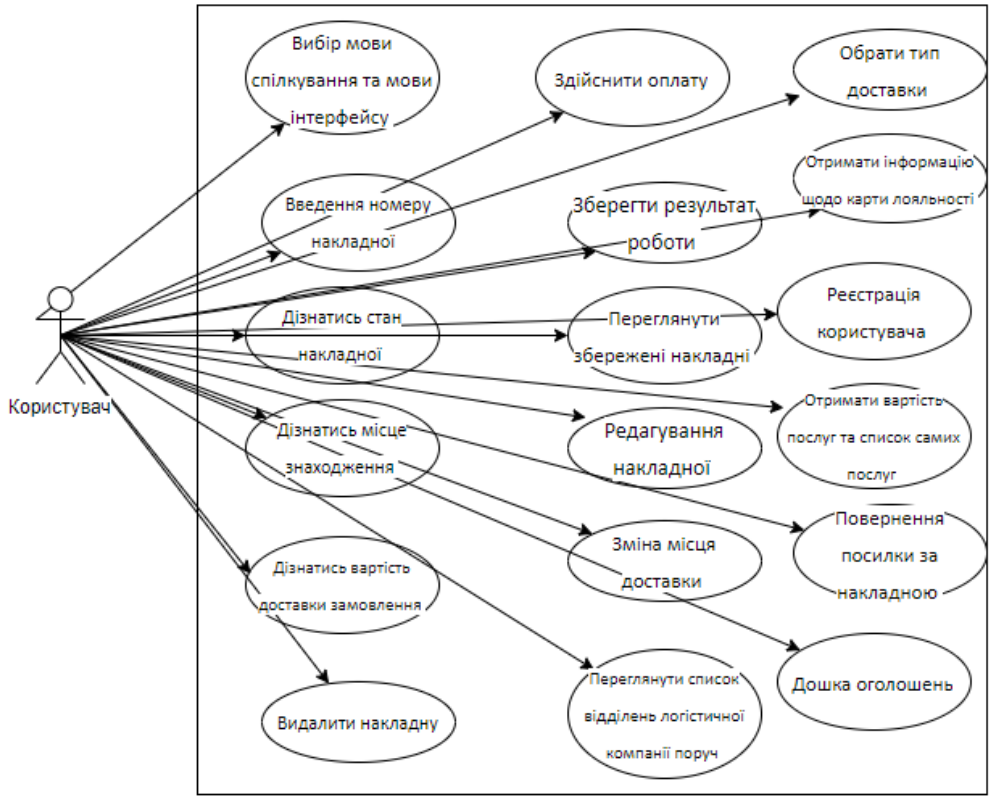

Рисунок 1.1 – Діаграма варіантів використання системи

1.1.4 Опис ключових варіантів використання

Після визначення усіх варіантів використання програмної системи було визначено ключові варіанти використання, а саме: додавання накладної, дізнатись місце знаходження, редагувати накладну, здійснити оплату, дізнатись вартість доставки, переглянути збережені накладні.

Варіант використання «Додати існуючу накладну» описаний у таблиці 1.2

Таблиця 1.2 Опис варіанту використання «Додавання накладної»

Дійові особи	Користувач, Система
Цілі	Додади існуючу накладну
Передумова	Користувач повинен мати номер накладної.
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач натискає клавішу «Меню». 3. Система видає користувачу список функції. 4. Користувач обирає дану опцію. 5. Система просить дозвіл у користувача ввести номер накладної. 6. Користувач вводить номер. 7. Система відкриває екран, який відображає, інформацію про накладну. 8. Система перевіряє валідність. 9. Система обробляє список накладних. 10. Система відображає результат на екрані.	
Результат	Система успішно додала накладну яку надав користувач
Альтернативні сценарії	
1а	Пристрій користувача не містить месенджер телеграм. Результат: неможливо скористатись функціями бота.

Продовження таблиці 1.2

4а	Користувач не підтвердив дозвіл на використання програмною системою камери його телефону. Результат: система не має можливості використовувати камеру. Наступні кроки сценарію не буде виконано
7а	Система виявила, що накладної не існує. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію додавання ще раз.
7б	Номер накладної вже існує. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію введення.

Варіант використання «Дізнатись місце знаходження» описаний у таблиці 1.3

Таблиця 1.3 Опис варіанту використання «Дізнатись місце знаходження»

Дійові особи	Користувач, Система
Цілі	Дізнатись місце знаходження
Передумова	Пристрій користувача повинен мати підключення до інтернету та мати навігацію.
Успішний сценарій: 1. Користувач відкриває месенджер. 2. Користувач натискає клавішу «Показати список накладних». 3. Система видає користувачу опцію «Відстежити місце». 4. Користувач обирає дану опцію. 5. Система просить дозвіл у користувача на використання цього місця. 6. Користувач підтверджує, що він дає дозвіл на користування місцем. 7. Система відкриває екран, який відображає, те що показує сервер. 8. Система перевіряє валідність. 9. Система обробляє дані. 10. Система відображає результат на екрані користувачу.	

Продовження таблиці 1.3

Результат	Система успішно показала місце
Альтернативні сценарії	
2а	Користувач немає доданих накладних. Результат: Система повідомляє користувача про неможливість виконання наступних операції.
6а	Користувач не підтвердив дозвіл на використання програмною системою місця його телефону. Результат: система не має можливості використовувати місце. Наступні кроки сценарію не буде виконано
8а	Система виявила, що місце виконане користувачем – невірне. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання ще раз.
10а	Результат відображення був проведений не успішно. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання ще раз.

Варіант використання «Редагувати накладну» описаний у таблиці 1.4.

Таблиця 1.4 Опис варіанту використання «Редагувати накладну»

Дійові особи	Користувач, Система
Цілі	Редагувати накладну

Передумова	Користувач повинен мати додані накладні та бути авторизованим
Успішний сценарій: 1. Користувач відкриває месенджер. 2. Користувач використовує функцію «Список накладних». 3. Система видає користувачу результат на екрані. 4. Користувач вибирає накладну. 5. Система просить дозвіл у користувача на редагування. 6. Користувач підтверджує, що він дає дозвіл на редагування. 7. Система перевіряє та що ввів користувач.	

Продовження таблиці 1.4

Результат	Система успішно відредагувала накладну
Альтернативні сценарії	
1а	Користувач не авторизований. Результат: система не має можливості редагувати накладні на сервері.
2а	Система не змогла успішно показати список. Результат: система повертає користувача на попередній екран.
5а	Користувач не підтвердив дозвіл на редагування. Результат: система не має можливості редагувати. Наступні кроки сценарію не буде виконано
7а	Система виявила, що введено текст невірний для його збереження. Результат: система повідомляє про це користувача та надає можливість повторити операцію(пункт 2), ще раз.

Варіант використання «Здійснити оплату» описаний у таблиці 1.5.

Таблиця 1.5 Опис варіанту використання «Здійснити оплату»

Дійові особи	Користувач, Система
Цілі	Здійснити оплату
Передумова	Користувач повинен авторизуватись
Успішний сценарій: 1. Користувач відкриває месенджер. 2. Користувач використовує функцію «Список накладних». 3. Система видає користувачу результат розпізнавання на екрані. 4. Користувач натискає на екран. 5. Система просить дозвіл у користувача на використання грошей його телефону. 6. Користувач підтверджує, що він дає дозвіл на користування грошей.	

Продовження таблиці 1.5

7. Система перевіряє та озвучує інформацію про предмет, який був оплачений системою.	
Результат	Система успішно вивела інформацію про оплачений предмет для користувача
Альтернативні сценарії	
2а	Користувач не увімкнув опцію авторизуватись на своєму пристрої. Результат: система не має можливості роботи з накладними для користувача.
3а	Система не змогла успішно розпізнати. Результат: система повертає користувача на попередній екран.
5а	Користувач не підтвердив дозвіл на використання програмною системою грошей його телефону. Результат: система не має можливості використовувати гроші. Наступні кроки сценарію не буде виконано
7а	Система виявила, що інформація про розпізнаний предмет невірна. Результат: система повідомляє про це користувача та надає можливість повторити операцію(пункт 2), ще раз.

Варіант використання «Дізнатись вартість доставки» описаний у таблиці 1.6.

Таблиця 1.6 Опис варіанту використання «Дізнатись вартість доставки»

Дійові особи	Користувач, Система
Цілі	Переглянути усі збережені користувачем предмети, що раніше було додано та показати вартість
Передумова	У базі даних повинні бути наявні накладні
Успішний сценарій: 1. Користувач відкриває месенджер. 2. Користувач натискає клавішу «Список накладних». 3. Система відкриває новий екран «Список накладних». 4. Система перевіряє доступ до бази даних. 5. Система показує індикатор завантаження та дістає усі наявні в базі даних збережені предмети.	

Продовження таблиці 1.6

6. Система перевіряє валідність стягнутих даних. 7. Система відображає дані у списку. 8. Користувач переглядає отримані дані.	
Результат	Система успішно продемонструвала користувачу список вартості.
Альтернативні сценарії	
4а	Система не отримала доступ до бази даних. Результат: система показує відповідне повідомлення та закриває екран.
5а	Система не вдалось завантажити дані з бази даних. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
6а	Система виявила, що стягнуті дані невірні. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
7а	Системі не вдалося правильно відобразити дані у списку. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.

Варіант використання «Переглянути збережені накладні» описаний у таблиці 1.7.

Таблиця 1.7 Опис варіанту використання «Переглянути збережені накладні»

Дійові особи	Користувач, Система
Цілі	Переглянути усі збережені користувачем накладні, що раніше було розпізнані системою та збережені ним до бази даних
Передумова	У базі даних повинні бути наявні збережені накладні
Успішний сценарій: 1. Користувач відкриває месенджер. 2. Користувач натискає клавішу «Список накладних». 3. Система відкриває новий екран «Збережені накладні». 4. Система перевіряє доступ до бази даних.	

Продовження таблиці 1.7

5. Система показує індикатор завантаження та дістає усі наявні в базі даних збережені тексти. 6. Система перевіряє валідність стягнутих даних. 7. Система відображає дані у списку. 8. Користувач переглядає отримані дані.	
Результат	Система успішно продемонструвала користувачу список збережених накладних.
Альтернативні сценарії	
4а	Система не отримала доступ до бази даних. Результат: система показує відповідне повідомлення та закриває екран.
5а	Система не вдалось завантажити дані з бази даних. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
6а	Система виявила, що стягнуті дані невірні. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
7а	Системі не вдалося правильно відобразити дані у списку. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.

1.2 Проектування програмної системи

1.2.1 Вибір моделі розробки

Спіральна модель є моделлю процесу розробки програмного забезпечення на основі ризику. На основі унікальних схем ризику даного проекту, спіральна модель направляє команду для прийняття елементів одного або декількох технологічних моделей, таких як інкрементний, водоспад або еволюційний Прототипування. Ця модель була вперше описана Баррі Boehm у своєму 1986 папері, "спіральна модель розробки програмного забезпечення і підвищення". у 1988 Boehm опублікував аналогічну паперову для широкої аудиторії. Ці документи ввести схему, яка була відтворені у багатьох подальших публікаціях обговорювали спіральну модель. Ці ранні документи використовують термін "модель процесу", щоб послатися на спіральну модель, а також на інкрементний, водоспад, Прототипування та інші підходи. Тим не менш, характерною моделлю, що керується ризиком на основі ризику інших функцій технологічних моделей, вже присутня: СК-Driven

підналаштування кроків спіральної моделі дозволяє моделі для розміщення будь-якої відповідної суміші орієнтованої специфікації, орієнтованого на прототип, орієнтованих на моделювання, автоматичної трансформації-орієнтованої, або інший підхід до програмного забезпечення Розвитку. 3 У пізніших публікаціях Boehm описує спіральну модель як «генератор моделі процесу», де вибір на основі ризиків проекту генерує відповідну модель процесу для проекту. Таким чином, додаткові, водоспад, Прототипування та інші моделі процесу є спеціальними випадками спіральної моделі, що відповідають моделям ризику певних проектів. Boehm також визначає ряд невірних помилок, що виникають в результаті oversimplifications в оригінальній схемі спіральної моделі. Він каже, що найнебезпечніші ці хибні уявлення: що спіраль є просто послідовністю приростів водоспаду; , що всі заходи по проекту слідують один

Спіральний послідовності; І що кожна вправа на схемі повинна виконуватися, і в показаному порядку. Хоча ці помилки можуть відповідати шаблонами ризику декількох проектів, вони не є істинними для більшості проектів. У звіті національної дослідницької ради Ця модель була розширена, щоб включити ризики, пов'язані з людськими користувачами. Щоб краще відрізнити їх від "небезпечних спіральних Look-Alikes", Boehm перераховує шість характеристик загальною для всіх автентичних застосувань спіральної моделі. [Автентичні застосування спіральної моделі рухає циклами, які завжди відображають шість характеристик. Boehm ілюструє кожен з прикладом "небезпечних спіральних Look-так", що порушує інваріантні. 1 Визначення артефактів одночасно. Послідовно визначаючи основні артефакти проекту, часто знижує можливість розробки системи, яка відповідає зацікавленні "Win умови" (цілі та обмеження).

Ця інваріантна виключає "небезпечні спірально схожі" процеси, які використовують послідовність інкрементного водоспаду переходить в настройках, де основні припущення водоспаду моделі не застосовуються. Boehm перераховує ці припущення наступним чином: Вимоги відомі заздалегідь реалізації. Вимоги не мають невирішені, високі ризики, такі як ризики, пов'язані з вартістю, графіком, продуктивність, безпека, інтерфейси користувача, організаційні наслідки і т. д. Характер вимог не зміниться дуже багато під час розвитку або еволюції. Вимоги сумісні з усіма очікуванням зацікавлених сторін системи, включаючи користувачів, клієнтів, розробників, супроводжуючих і інвесторів. Правильну архітектуру для виконання вимог добре розуміють. Існує достатньо часу календаря, щоб продовжити послідовно. У ситуаціях, коли ці припущення застосовуються, це ризик проекту не вказувати вимоги і діяти послідовно. Модель водоспаду, таким чином, стає ризик-Driven спеціальним випадком спіральної моделі. Виконувати чотири основні дії в кожному циклі

Ця інваріантна визначає чотири види діяльності, які повинні відбуватися в кожному циклі спіральної моделі: Розглянемо умови перемоги всіх критично важливих зацікавлених сторін. Визначте і оцініть альтернативні підходи для задоволення умов виграшу. Виявлення та

усунення ризиків, які випливають із обраного підходу (ES). Отримати схвалення від усіх успіхів-критичних зацікавлених сторін, а також прихильність до продовження наступного циклу. Проект цикли, які опускати або змінити будь-який з цих заходів ризик втрати зусиль, переслідуючи параметри, які є неприйнятними для ключових зацікавлених сторін, або занадто ризиковано. Деякі "небезпечні спірально-схожі" процеси порушують цей Інваріантний шляхом виключення ключових зацікавлених сторін з певних фаз або циклів. Наприклад, супроводжуючі системи та адміністратори можуть бути не запрошені для участі в визначенні та розвитку системи.

В результаті, система ризикує не в змозі задовольнити свої умови виграшу. Ризик визначає рівень зусиль. Для будь-якої діяльності проекту (наприклад, аналіз вимог, проектування, макетування, тестування), команда проекту повинна вирішити, скільки зусиль достатньо. У процесі проведення автентичних спіральних циклів ці рішення робляться шляхом мінімізації загального ризику. Наприклад, інвестування додаткового часу тестування програмного продукту часто знижує ризик у зв'язку з ринку відмову паскудної продукції. Однак, додатковий час тестування може збільшити ризик у зв'язку з достроковим записом конкурента. З точки зору спіральної моделі, тестування повинно бути виконано до мінімуму загальний ризик, і не далі.

"Небезпечні Спіральний погляд-Alikes", які порушують це інваріантні включають еволюційні процеси, які ігнорують ризик через масштабованість питань, а також інкрементні процеси, які вкладають значні кошти в технічну архітектуру, яка повинна бути перероблена або замінена для розміщення майбутні збільшення продукту. Ризик визначає ступінь деталі Для будь-якого артефакту проекту (наприклад, специфікація вимог, дизайн-документа, плану тестування), команда проекту повинна вирішити, наскільки докладно достатньо. У процесі проведення автентичних спіральних циклів ці рішення робляться шляхом мінімізації загального ризику. Враховуючи специфікаціях вимог як приклад, проект повинен точно вказати ті функції, в яких ризик зменшується через точну специфікацію (наприклад, інтерфейси між обладнанням та програмним забезпеченням, інтерфейси між простими та підрядними підрядниками). І навпаки, проект не повинен точно вказувати ті функції, де точна специфікація підвищує ризик (наприклад, графічні макети екрану, поведінку готових компонентів). Використання етапів прив'язки точок Оригінальна характеристика Boehm спіральної моделі не включає в себе будь-які етапи процесу. У пізніших уточнень, він вводить три віхи опорних точок, які служать як показники прогресу і точки прихильності. Ці віхи опорної точки можуть характеризуватися ключовими питаннями. Цілі життєвого циклу. Чи є достатній визначення технічного та управлінського підходу до задоволення всіх умов, що виграють? Якщо зацікавлені сторони згодні з тим, що відповідь "так", то проект очистив цей етап LCO. В іншому випадку, проект може бути покинутий, або зацікавлені сторони можуть зробити ще один цикл, щоб спробувати дістатися до "так"

Архітектура життєвого циклу. Чи є достатня визначення бажаний підхід до задовольняють всі умови перемоги, і всі суттєві ризики усунені або пом'якшені? Якщо зацікавлені сторони погоджуються з тим, що відповідь "так", то проект очистив цей етап LCA. В іншому випадку, проект може бути покинутий, або зацікавлені сторони можуть зробити ще один цикл, щоб спробувати дістатися до "так". Початкова оперативна спроможність. Чи достатньо підготовки програмного забезпечення, сайту, користувачів, операторів та супроводжуючих задовольнити всі умови перемоги, запустивши систему? Якщо зацікавлені сторони погоджуються з тим, що відповідь "так", то проект очистив віху МОК і запущений. В іншому випадку, проект може бути покинутий, або зацікавлені сторони можуть зробити ще один цикл, щоб спробувати дістатися до "так". "Небезпечні спірально Look-Alikes", які порушують це інваріантні включають еволюційні та інкрементні процеси, які здійснюють значні ресурси для реалізації рішення з погано визначена архітектура.

Три опорні точки світу легко вписуються в Раціональний уніфікований процес (НРП), що позначаючи межу між етапами створення та розробки НРП, що позначає межу між етапами розробки та конструювання, а МОК позначає межу між будівництвом і перехідними фазами. Орієнтація на систему та її життєвий цикл Ця інваріантна підкреслює важливість загальної системи та довгострокових проблем, що охоплюють весь життєвий цикл. Вона виключає "небезпечні спірально Look-Alikes", які зосереджені занадто багато на початковий розвиток програмного коду. Ці процеси можуть призвести до наступних опублікованих підходів до об'єктно-орієнтованого або структурованого програмного аналізу та проектування, при цьому нехтуючи іншими аспектами потреб процесу проекту.

Скрам – це гнучкий процес роботи з управління складними знаннями, з початковим акцентом на розробці програмного забезпечення, хоча він використовувався в інших галузях і поступово починає вивчатися для інших складних робіт, наукових досліджень та передових технологій. Вона

призначена для команд з десяти або менше членів, які порушують свою роботу в цілі, які можуть бути завершені протягом часкоробкової ітерації, званих спринт, не більше одного місяця, і найчастіше два тижні, а потім відслідковувати прогрес і повторно планувати в 15-хвилинну час-коробкової стоячи нарад, званих щоденних scrums Скрам інколи можна побачити в усіх столиць, як у СКРАЗІ.Слово не є акронім, тому ця Стилізація не є правильною; Це, ймовірно, виникло через ранні папери Кен Швабера, які капіталізуються у своїй назві.

Хоча товарний знак на термін "скрам" було дозволено до закінчення, він вважається власником більш широкої спільноти, а не фізичною особою, тому в цій статті зберігається провідний капітал для Скраму. Багато з термінів, що використовуються у Скразі, зазвичай написані провідними капітелями (наприклад, майстер Скрам, щоденний Скрам). Однак для підтримки енциклопедичного тону у цій статті використовується нормальна пропозиція для цих термінів (наприклад, майстер скрам, щоденне скрам), якщо вони не є визнаними знаками (наприклад, сертифікованими майстром Скрам).

Ключові ідеї Скрам-легка, Ітераційна і інкрементна основа для управління складною роботою. Рамкові виклики припущення про традиційний, послідовний підхід до розробки продукту, а також дозволяє групам самостійно організовувати за рахунок заохочення фізичного спільного розташування або закриття онлайнового співробітництва всіх членів команди, а так само щоденно спілкування між усіма членами команди та дисциплінами. Ключовим принципом Скраму є подвійне визнання того, що клієнти змінять їхню думку про те, що вони хочуть або потребують (часто називають волатильність вимог), і що там будуть непередбачуваними проблемами, для яких прогнозоване або заплановане підхід не підходить. Таким чином, Скрам приймає доказову емпіричний підхід, беручи до уваги, що проблема не може бути повністю зрозуміла або визначена спереду, а натомість зосередження на тому, як максимізувати здатність команди швидко доставити, реагувати на виникаючі вимоги і адаптуватися до розвиваються технології та зміни ринкових умов.

Історії Хіротака Такеучі і Ikujiro nonaka представив термін скрам в контексті розробки продукту в їх 1986 Гарвардська стаття бізнес-огляд, "Нова нова гра розробки продукту". Такеуті і Несака пізніше стверджували у створенні компанії "знання" , що вона є формою "створення організаційних знань, особливо добре в залученні про інновації безперервно, поступово і спірально ". Автори описали новий підхід до розробки комерційного продукту, що збільшить швидкість і гнучкість на основі тематичних досліджень з виробничих фірм в автомобільній, копіювальний та друкарському галузях. Вони назвали цей цілісний або регбі підхід, тому що весь процес виконується одним крос-функціональна команда через кілька перекриття фаз, в якому команда "намагається піти відстані як одиницю, передаючи м'яч вперед і назад". (у регбі, у футболі, скрам використовується

для перезапуску гри, як нападники кожної команди блокування з їх головами вниз і намагатися отримати володіння м'ячем.)

Основу Скраму було засноване на дослідженнях Швабера з Тунде Бабатунде на дослідницькій станції Дюпон і університету Делавера. Tunde повідомив, що спроби розробити комплексний продукт, наприклад, програмне забезпечення, яке не було засноване в Емпіризм, були приречені на більш високі ризики і темпи провалу як початкові умови і припущення змін. Емпіризм, використовуючи часті інспекції та адаптацію, з гнучкістю і прозорістю є найбільш підходящим підходом. На початку 1990-х, Кен Швабера використовував те, що стало б Скрам у своїй компанії, передові методи розвитку; Хоча Джефф Сазерленд, Джон Scumniotales і Джефф McKenna розробили аналогічний підхід у станкової корпорації, посилаючись на нього за допомогою одного слова скрам.

Кен і Джефф працювали разом, щоб інтегрувати свої ідеї в єдину основу, Скрам. Вони протестували Скрам і постійно вдосконалили його, що призвело до їх 1995 папери, внески в Agile маніфест в 2001, і по всьому світу розповсюдження і використання Скрам з 2002. У 1995, Сазерленд і Швабера спільно представили документ, що описує рамки Скраму на проектування та впровадження бізнес-об'єктів, що проводився в рамках об'єктно-орієнтованого програмування, систем, мов & додатків ' 95 (OOPSLA ' 95) в Остіні, штат Техас. протягом наступних років Швабера і Сазерленд співпрацювали для того, щоб поєднати цей матеріал — з їхнім досвідом та розвитком хорошої практики — розвинути те, що стало відомо як Скрам. 17 У 2001, Швабера працював з Майком Beedle для опису методу в книзі, Agile розробки програмного забезпечення зі Скрам. підхід скраму до планування та управління розробкою продукту передбачає приведення повноважень прийняття рішень до рівня властивостей операції та визначеності. 2 У 2002, Швабера з іншими, заснував Альянс «Скрам» і створив сертифікований серіал «Скрам». Швабера покинув Альянс в кінці 2009 і заснував Scrum.org, яка контролює паралельне професійне Акредитаційний серіал "Скрам".З 2009 року був опублікований та оновлений Швабера та Сазерленд. Він був переглянутий 5 разів, з поточною версією в листопаді 2017

Цей розділ, можливо, містить оригінальне дослідження. Будь ласка, Покращуйте його, підтвердивши претензії, зроблені та додаючи вкладені цитати. Заяви, що складаються лише з оригінальних досліджень, повинні бути вилучені. (2019 квітня) (Дізнайтеся, як і коли видалити це повідомлення шаблону) У рамках програми «Скрам» є три ролі. Це ідеальне спільне розташування для забезпечення оптимального спілкування між членами команди. Разом ці три ролі формують команду "скрам". Хоча багато організацій мають інші ролі, пов'язані з визначенням і поставляючи продукт, Скрам визначає лише ці три. Власник продукту Власник продукту, що представляє зацікавлених сторін продукту та голос клієнта (або може представляти бажання комітету), відповідає за досягнення хороших результатів бізнесу. Отже, власник продукту підзвітна за накопичилися

продукту і для максимального значення, що команда доставляє. власник продукту визначає продукт в умовах, орієнтованих на клієнта (зазвичай, історії користувачів), додає їх до відставання продукту, а пріоритет їх залежить від важливості та залежностей.

Команда «скрам» повинна мати лише одного власника продукту (Хоча власник продукту міг би підтримати більше однієї команди) Ця роль не повинна поєднуватися з цим майстром скраму. Власник продукту повинен зосередитися на діловій стороні розробки продукту і витратити більшість свого часу на підтримання взаємодії з зацікавленими сторонами та командою. Власник продукту не повинен диктувати, як команда досягає технічного рішення, а скоріше буде прагнути консенсусу серед членів команди. Ця роль має вирішальне значення і вимагає глибокого розуміння обох сторін: бізнес і інженерів (розробників) у команді скрам. Тому хороший власник продукту повинен бути в змозі донести те, що бізнес потребує, запитайте, чому їм це потрібно (тому що можуть бути кращими способами для досягнення цього), і передати повідомлення всім зацікавленим сторонам, включаючи команду доставки з використанням технічної мови, як вимагається . Власник продукту використовує емпіричні інструменти Скрам для управління досить складною роботою, при цьому контролює ризик і досягнення цінності.

Комунікація є основною відповідальністю власника продукту. Здатність передавати пріоритети і співпереживати з членами команди і зацікавленими сторонами є життєво важливим, щоб направити розробку продукту в правильному напрямку. Роль господаря продуктів-це розрив зв'язку між командою та її зацікавленими сторонами, який виступає в якості проксі-сервера для зацікавлених сторін команди і як представника команди до загальної спільноти зацікавлених сторін. Як особа групи зацікавлених сторін, Нижче наведені деякі комунікаційні завдання власника продукту для зацікавлених сторін: Визначте та Оголосіть випуски. Спілкуйтеся з доставкою та статусом команди. Поділіться прогресом під час нарад на управління. Частка значних рідів (ризиків, перешкод, залежностей і припущень) з зацікавленими сторонами. Домовляєтеся про пріоритети, сферу, фінансування і розклад. Переконайтеся, що відставання продукту є видимим, прозорим і ясним. Емпатія є ключовим атрибутом для власника продукту мати-здатність ставити своє себе в взуття іншого.

Власник продукту розмовляє з різними зацікавленими сторонами, які мають різні фони, ролі робочих місць і цілі. Власник продукту повинен мати можливість бачити з цих різних точок зору. Щоб бути ефективним, це розумно для власника продукту, щоб знати рівень деталізації аудиторії потреб. Команда розробників потребує ретельного зворотного зв'язку та специфікацій, щоб вони могли побудувати продукт до очікування, в той час як виконавчий спонсор може просто потрібно резюме прогресу. Надання більш детальну інформацію, ніж це необхідно, може втратити інтерес до зацікавлених сторін і витрачати час.

Прямим засобом зв'язку є найбільш переважним досвідчені Agile власників продуктів. Здатність власника продукту ефективно спілкуватися також посилюється, будучи досвідченими в техніках, які визначають потреби зацікавлених сторін, ведуть переговори про пріоритети між інтересами зацікавлених сторін і співпрацюють з розробниками для забезпечення ефективної реалізації Вимоги. Команда розробників Команда розробників має від трьох до дев'яти членів, які виконують всі завдання, необхідні для створення приросту цінного випуску кожного спринту. Хоча члени команди називаються розробниками в якійсь літературі, цей термін стосується тих, хто грає роль у розробці та підтримці системи або продукту, і може включати дослідників, архітекторів, дизайнерів, фахівців з даних, статистики, аналітики, інженери, програмісти, тестувальники тощо. Однак, через плутанину, яка може виникнути, коли деякі люди не відчувають поняття "розробник" відноситься до них, вони часто називають так само, як члени команди.

Команда є самостійною організацією. Хоча жодна робота не повинна прийти до команди, крім як через власника продукту, а майстер скраму, як очікується, захистить команду від занадто великої кількості відволікання, команда все одно повинна заохочувати безпосередньо взаємодіяти з клієнтами та/або зацікавленими сторонами для здобуття максимального розуміння і безпосередності зворотного зв'язку. Майстер зі скраму Скрам сприяє майстер скраму, який підзвітний за усунення перешкод у спроможності команди доставити цілі та кінцеві результати продукту. Майстер "скрам" не є традиційним керівником команди або менеджером проектів, але діє як буфер між командою і будь-яким відволікаючим впливам. Майстер "скрам" гарантує, що у рамках програми "скрам" дотримуються рамки. Майстер "скрам" допомагає забезпечити команді узгоджені процеси в рамках Скраму, часто полегшує ключові сесії, заохочує команду покращувати. Роль також згадується як координатор команди або слуга-лідера, щоб зміцнити ці подвійні перспективи. Основні обов'язки майстра скраму включають (але не обмежуючись ними): Допомагаючи власнику продукту підтримувати відставання продукту таким чином, що забезпечує необхідну роботу добре розуміли, так що команда може постійно робити вперед прогресу Допомагаючи команді визначити визначення, що зроблено для продукту, з введенням ключових зацікавлених сторін Коучинг команди, в рамках принципів Скраму, з метою доставки високоякісних функцій для свого продукту. Сприяння самоорганізації в команді Допомагати команді "скрам", щоб уникнути або усунути перешкоди до її прогресу, чи то внутрішня чи зовнішня команда Полегшення командних заходів для забезпечення регулярного прогресу Навчання ключових зацікавлених сторін на принципах Agile та Скраму Коучинг команда розробників в самоорганізації та крос-функціональність Майстер скрам допомагає людям і організаціям приймати Емпіричне та бережливе мислення, залишаючи за

собою сподівання на впевненість і передбачуваність. Один зі способів, як головну роль скраму відрізняється від керівника проекту, полягає в тому, що останні можуть мати обов'язки управління людьми і майстер скраму. Майстер зі скрам надає обмежену кількість напрямків, оскільки команда, як очікується, буде уповноважена і самоорганізації."скрам" не визнає роль керівника проекту, оскільки традиційні тенденції командування і контролю призведуть до труднощів.

Не слід плутати з хакатон § код спринт. Процес Скраму Sprint (також відомий як ітерація або timebox) є базовою одиницею розвитку у Скразі. Sprint — це часкоробковий зусилля; тобто, довжина узгоджується і фіксується заздалегідь для кожного Sprint і, як правило, між один тиждень і один місяць, з двома тижнем є найбільш поширеним. Кожен Sprint починається з Sprint планування події, яка встановлює Sprint мети і необхідні елементи накопичилися продукту. Команда приймає те, що вони згодні готовий і перевести це в спринті відставання, з розбивкою роботи, необхідної і передбачуваний прогноз на спринті мети. Кожен спринт закінчується з Sprint огляд і спринт ретроспективний, що огляди прогресу, щоб показати зацікавленимсторонам і визначити уроки і поліпшення для наступної спрінт.

Скрам підкреслює цінний, корисний вихід в кінці спринту, що насправді робиться. У разі програмного забезпечення, це, швидше за все, включає в себе, що програмне забезпечення було повністю інтегрований, перевірений і документально, і потенційно випустився. Спрінт-планування На початку Sprint команда "скрам" проводить спринтерські події планування: Взаємно обговорювати та погодять сферу роботи, яка має бути виконана під час цього спринту Вибрати продукт накопичилися елементи, які можна завершити в один Sprint Підготовка Sprint відставання, який включає в себе роботу, необхідну для завершення вибраного продукту накопичилися пунктів Погодьтеся, Sprint мети, короткий опис того, що вони прогнозування доставити в кінці спринті. Рекомендована тривалість-чотири години на два тижні Sprint (Pro-Рата для інших Sprint тривалості) У першій половині, вся команда скрам (команда розробників, майстер скрам і власник продукту) вибирає продукт накопичилися елементи, які вони вважають, може бути завершена в цьому спринті У другій половині, команда розробників визначає детальну роботу (завдання), необхідні для завершення цих продуктів відставання пунктів; в результаті чого підтверджено Sprint відставання Як детальна робота розроблена, деякі продукти накопичилися елементи можуть бути розділені або покласти назад в продукт відставання, якщо команда більше не вірить, що вони можуть завершити необхідну роботу в одному спринті Після того, як команда розробників підготувала свій спринт відставання, вони прогнозують (як правило, голосування), які завдання будуть доставлені в рамках Sprint.

Щоденний скрам Щоденний скрам в обчислювальному залі. Це централізоване розташування допомагає команді почати час. Кожен день під

час спринту, команда проводить щоденний скрам (або стоячи) з конкретними рекомендаціями: Всі члени команди розробників приходять підготовлені. Щоденний скрам: починається саме вчасно, навіть якщо деякі члени групи розробників відсутні має відбутися в той же час і місце кожен день обмежена (з часом) до п'ятнадцяти хвилин Будь-яка людина вітається, хоча тільки члени команди розробників повинні внести свій вклад. Під час щоденного скраму кожен член команди, як правило, відповідає на три запитання: Що я завершити вчора, що сприяло команді засідання нашої Sprint мети?

Що я планую завершити сьогодні, щоб зробити свій внесок у команду, яка відповідає нашій меті Sprint? Я бачу будь-які перешкоди, які можуть перешкодити мені або команді зустрітися з нашою метою Sprint? Будь-які перешкоди (наприклад, блокування, ризик, проблема, затримка залежність, припущення виявилися необгрунтованими) у щоденних скрам повинні бути захоплені майстром скраму і відображатися на дошці "скраб" або на дошці спільних ризиків, з узгодженою особою призначений для роботи в напрямку резолюції (за межами щоденного скраму). У той час як валюта статусу роботи-це відповідальність всієї команди, майстер скрам часто оновлює діаграму Sprint, що берндаун. де команда не бачить значення в цих подіях, це відповідальність майстра скраму, щоб з'ясувати, чому. це є частиною обов'язку виховання команди та зацікавлених сторін щодо принципів Скраму. Під час щоденного скраму не повинно відбуватися детальних обговорень. Після завершення зустрічі окремі учасники можуть зібратися разом, щоб детально обговорити питання; така зустріч іноді називають "секційних сесій" або "після вечірки".

Sprint огляд В кінці спринті, команда проводить дві події: Sprint огляд і Ретроспектива спринт. На спрінт огляд, команда: рецензії на роботу, яка була завершена і запланованих робіт, які не були завершені представляє заповнену роботу зацікавленим сторонам (ака демо) співпрацює з зацікавленими сторонами про те, що працювати на наступному Керівні принципи для Sprint Відгуки: Не вдалося продемонструвати незавершену роботу. Рекомендована тривалість-дві години на два тижні Sprint (пропорційна для інших Sprint-тривалості). Ретроспектива

На "спринт ретроспективний", команда: Відображає минуле спринт Визначає та погоджується на дії з удосконалення процесів Керівні принципи для Sprint ретроспективи: Три основні питання задають в спринт ретроспективний: що йшло добре під час спринті? Що не вийшло добре? Що може бути покращено для кращої продуктивності в наступному спринті?

Рекомендована тривалість-один з половиною години на два тижні Sprint (пропорційна для інших Sprint тривалість (и)) Цей захід сприяє майстер скраму Уточнення відставання Накопичилися уточнення (раніше званий стрижка) є триваючого процесу розгляду продукту накопичилися пунктів і перевірки, що вони належним підготовлені і наказав таким чином,

що робить їх чіткими і виконуваних для команд, як тільки вони потрапляють спрінт через спринт планувальної діяльності.

Продукт накопичилися елементи можуть бути розбиті на кілька менших. Можна уточнити критерії приймання. Залежності можуть бути ідентифіковані та досліджені. Хоча спочатку не є основною практикою Скраму, уточнення накопичилися було додано до посібника "Скрам" і прийнятий як спосіб управління якістю продукції, що накопичилися, що входять до спринту, з рекомендованою інвестицією до 10% потужності Sprint команди. Відставання може також включати технічний борг (також відомий як дизайн боргу або кодекс заборгованості). Це концепція в розробці програмного забезпечення, що відображає неявні витрати на додаткові переробки викликані вибором простого рішення зараз замість використання кращого підходу, який займе більше часу. Скасування операції Sprint Власник продукту може скасувати Sprint, якщо це необхідно. власник виробу може зробити це за допомогою команди, майстра або управління скрам. Наприклад, керівництво може побажати власнику продукту скасувати спринт, якщо зовнішні обставини заперечують значення мети Sprint. Якщо Sprint є аномально припинено, наступний крок полягає в проведенні нового Sprint планування, де причина припинення розглядається.

У цьому розділі потрібні додаткові бібліографічні посилання для перевірки. Будь ласка, допоможіть поліпшити цю статтю, додавши посилання на Надійні джерела. Матеріал без джерел може бути оскаржений і вилучений. Пошук джерел: розробка програмного забезпечення "Скрам" – Новини · Доставка преси книги вченого ЖОСТОР (березень 2013) (Дізнайтеся, як і коли видалити це повідомлення шаблону) Відставання продукту Відставання продукту є розбивка роботи, яка буде виконана і містить впорядкований список вимог до продукції, що команда скрам веде до продукту. Загальні формати включають в себе історії користувачів і використання інцидентів. вимоги визначають особливості, виправлення помилок, нефункціональні вимоги і т. д. — все, що потрібно зробити, щоб доставити життєздатний продукт. Власник продукту віддає перевагу продукту накопичилися елементи (PBIs) на основі таких міркувань, як ризик, бізнес-значення, залежності, розмір і дата необхідності. Відставання продукту є те, що буде доставлено, замовлений в послідовності, в якій вона повинна бути доставлена.

Це видно для всіх, але може бути змінений тільки за згодою власника продукту, який в кінцевому рахунку відповідальність за замовлення продукції відставання пунктів для команди розробників вибрати. Накопичилися продукту містить оцінку власника продукту, вартість бізнесу і оцінка розвитку команди розвитку зусиль, які часто, але не завжди, заявив в сюжеті точок з використанням округлої шкалою Фібоначчі. Ці оцінки допомагають власнику продукту оцінити часову шкалу і можуть впливати на порядок накопичилися товарів; Наприклад, якщо дві функції мають однаковий бізнес-значення, власник продукту може запланувати попередню

доставку однієї з нижчих зусиль розвитку (оскільки рентабельність інвестицій вища) або з вищою розробкою зусиль (оскільки вона складніший або більш ризикованими, і вони хочуть, щоб вийти з цього ризику раніше).

Відставання продукту та бізнес-значення кожного елемента накопичилися продукт відповідальність власника продукту. Зусилля для доставки кожного пункту оцінюється командою розробників у точках історії, або час. Оцінюючи в сюжеті пунктів, команда зменшує залежність в індивідуальних розробників; Це корисно, особливо в динамічних групах, де розробники часто призначаються для інших проектів після доставки Sprint. Наприклад, якщо історія користувача оцінюється як 5 в зусиллях (з використанням послідовності Фібоначчі), він залишається 5 незалежно від того, скільки розробників працюють на ньому Історія точки визначити зусилля в полі часу, так що вони не змінюються з часом. Наприклад, за одну годину людина може ходити, бігати або лізти, але витрачені зусилля явно різні. Розрив розриву між термінами в послідовності Фібоначчі заохочує команду доставити ретельно розглянуті оцінки. Оцінки 1, 2 або 3 передбачають аналогічні зусилля (1 бути тривіальним), але якщо команда оцінює 8 або 13 (або вище), вплив на обидві поставки і бюджет може бути значним. Значення використання пунктів історії є те, що команда може використовувати їх, порівнюючи аналогічні роботи з попередніх спрінт, але слід визнати, що оцінки по відношенню до команди. Наприклад, оцінка 5 для однієї команди може бути 2 для іншого, що мають старших розробників і вищих навичок.

Кожна команда повинна мати власника продукту, хоча в багатьох випадках власник продукту може працювати з більш ніж однією командою. власник продукту відповідає за максимальне значення продукту. Власник продукту збирає введення і бере зворотний зв'язок з, і лобіює, багато людей, але в остаточному підсумку робить виклик на те, що буде побудовано. Накопичилися продукту: Захоплює запити на змінення продукту — включаючи нові функції, заміну старих функцій, видалення функцій і вирішення проблем Гарантує, що команда розробників має роботу, яка максимізує бізнес-переваги для власника продукту Як правило, власник продукту та команда "скрам" працюють разом, щоб розвивати розбивку роботи; Це стає продуктом відставання, який розвивається як нові інформаційні поверхні про продукт і про своїх клієнтів, і тому пізніше спрінт може звернутися до нової роботи.

Управління Відставання продукту в найпростішому вигляді – це лише список елементів, які потрібно працювати. Маючи налагоджені правила про те, як додана робота, вилучена та впорядкована, допомагає усій команді приймати більш зважені рішення щодо зміни продукту. Власник продукту пріоритет зує продукт накопичилися пунктів, на основі яких необхідно швидше. Потім команда вибирає, які елементи вони можуть завершити в майбутній спринт. На дошці "скрам" команда переміщує елементи з

відставання продукту в спринт відставання, який є список елементів, які вони будуть будувати. Концептуально, це ідеально підходить для команди, щоб вибрати тільки те, що вони думають, що вони можуть виконати з верхньої частини списку, але це не є незвичайним, щоб побачити на практиці, що команди можуть приймати елементи нижнього пріоритету зі списку разом з вибраними. Це зазвичай відбувається тому, що є час, що залишився в спринті, щоб вмістити більше роботи.

Пунктів у верхній частині відставання, елементи для роботи на першому, повинні бути розбиті на розповіді, які підходять для розвитку команди для роботи. Далі в відставання йде, тим менш вишукані предмети повинні бути. Як Швабера і Beedle поклав його "нижня пріоритет, тим менше деталей, поки ви навряд чи може зробити з накопичилися пункту". 2 Як команда працює через відставання, слід припустити, що зміни відбуваються за межами свого оточення-команда може дізнатися про нові можливості ринку, щоб скористатися, конкурентів, що виникають, і зворотний зв'язок від клієнтів, які можуть змінити спосіб продукт мав на меті працювати. Всі ці нові ідеї, як правило, викликають команду адаптувати відставання для включення нових знань. Це є частиною фундаментального мислення Agile Team. Світ змінюється, відставання ніколи не закінчили. Sprint відставання Дошка завдань зі скраму Sprint відставання список робіт команда розробників повинна звернутися під час наступного Sprint. цей список походить від команди "скрам" поступово вибираючи елементи відставання в порядку пріоритету з верхньої частини відставання продукту, поки вони не відчувають, що вони мають достатньо роботи, щоб заповнити спринт. Команда розробників повинна мати на увазі його минулі результати оцінки його потенціалу для нового Sprint, і використовувати це як орієнтир, скільки "зусилля" вони можуть завершити. Продукт накопичилися елементи можуть бути розбиті на завдання командою розробників. завдання на Спринт відставання ніколи не призначаються (або штовхнув) для членів команди кимось іншим; скоріше члени команди Підпишіться на (або тягнути) завдання, які необхідні відповідно до пріоритету відставання і їх власні навички та здібності. Це сприяє самоорганізації розвитку команди і розробника Бай-in. Sprint відставання є власністю команди розробників, і всі включені оцінки надаються командою розробників. Часто Супутня дошка завдань використовується, щоб побачити і змінити стан завдань поточного Sprint, як це зробити, в прогрес і Готово.

Як тільки Sprint відставання прихильний, ніякі додаткові роботи можуть бути додані до Sprint відставання за винятком команди. Після того, як спринт був доставлений, накопичилися продукт аналізується і Пріоритетна при необхідності, і наступний набір функціональності вибирається для наступного Sprint. Крок Приріст є потенційно випугоздатний вихід Sprint, який відповідає мета Sprint. Він формується з усіх завершених Sprint накопичилися пунктів, інтегрований з роботою всіх попередніх спрінт. Приріст повинен бути повним, відповідно до визначення

команди скрам (МО), повністю функціонує, і в працездатний стан незалежно від того, власник продукту вирішує фактично розгорнути і використовувати його. Розширення Можна використовувати такі артефакти та методики, щоб допомогти людям використовувати Скрам. Спрінт берндаун чарт Зразок берндаун графік завершена спринт, показуючи залишилися зусилля в кінці кожного дня. Основна стаття: спалити діаграму Sprint берндаун графік є публічно відображається діаграма показує, залишковий обсяг роботи в спринт відставання. оновлюється кожен день, він дає простий погляд на Спринт прогресу. Він також забезпечує швидкі візуалізації для довідки. Горизонтальна вісь Sprint бернлдаун діаграма показує дні в спринті, в той час як вертикальна вісь показує кількість робіт, що залишилися кожен день (зазвичай представляють оцінку годин роботи, що залишилися). Під час планування Sprint, буде побудований ідеальний діаграма берндаун. Потім, під час спринті, кожен член підбирає завдання з Sprint відставання і працює на них.

Наприкінці дня вони оновлюють залишкові години для завершення завдань. Таким чином, фактичні берндаун графік оновлюється з кожним днем. Не слід плутати з діаграмою освоєного значення. Реліз згортого графіка Зразок Burn-діаграми для випуску, відображення області завершено кожного Sprint Випуск згортого діаграми – це спосіб, у який команда надає видимість і відстежує перебіг випуску. Оновлюється в кінці кожного Sprint, він показує прогрес у досягненні прогнозу масштабу. Горизонтальна вісь випуску Burn-діаграма показує спрінт у релізі, в той час як вертикальна вісь показує обсяг роботи, завершеною в кінці кожного Sprint (зазвичай представляють кумулятивні точки історії виконаних робіт). Прогрес буде відображено як рядок, що росте, щоб відповідати горизонтальній лінії, яка представляє прогнозний масштаб; часто відображається з прогнозом, виходячи з прогресу на сьогоднішній день, що вказує, скільки обсяг може бути завершений з даної дати релізу або скільки спрінт буде потрібно для завершення даної області. Реліз згорткового діаграми дозволяє легко побачити, скільки роботи було завершено, скільки роботи було додано або видалено (якщо лінія горизонтальної області рухається), і скільки роботи залишилося зробити. Визначення готовності (DoR)

Критерії початку, щоб визначити, чи специфікації та входи встановлено достатньо, щоб почати робочий елемент, тобто історія користувача. Визначення виконаної роботи (DoD) Exit-критерії, щоб визначити, чи продукт накопичилися елемент завершено. У багатьох випадках, МО вимагає, щоб всі регресійні тести будуть успішними. Визначення «Готово» може відрізнятися від однієї команди скраму до іншої, але повинна бути послідовною в межах однієї команди. Загальне зусилля команда здатна в спринті. Число походить від оцінки роботи (зазвичай в точках історії користувачів) завершені в останньому спринті. Колекція даних про історичні швидкості є орієнтиром для надання допомоги команді в розумінні того, скільки роботи вони можуть, ймовірно, домогтися в

майбутньому спринт. Історія створення Час-коробковий період, який використовується для дослідження концепції або створення простого прототипу. Шипи можуть бути або планується проходити між спринт або, для великих команд, сплеск може бути прийнята в якості одного з багатьох цілей Sprint доставки.

Шипи часто вводиться до доставки великих або складних продуктів накопичилися елементи з метою забезпечення бюджету, розширити знання, або виробляти докази концепції. Тривалість і мета (и) скарбнички узгоджується між власником продукту і командою розробників до початку. На відміну від Sprint зобов'язань, шипи можуть або не можуть доставити відчутні, Shippable, цінні функціональні можливості. Наприклад, мета сплеску може успішно досягти рішення про хід дії. Сплеск закінчився, коли час до, не обов'язково, коли мета була доставлена. Куля трасуючими Також називається безпілотник, куля трасуючими є сплеск з поточною архітектурою, поточний набір технологій, поточний набір передового досвіду, що призводить до виробництва код якості. Це може бути просто дуже вузький здійснення функціональності, але не відметані код. Вона має якість виробництва, а решта ітерації може спиратися на цей код. Назва має військове походження як боєприпаси, що робить шлях до кулі видимим, що дозволяє виправлення. Часто ці реалізації є "швидким пострілом" через всі шари програми, такі, як підключення поля введення однієї форми в задній кінець, щоб довести, що шари підключаються, як очікувалося.

Переваги Скраму можуть бути важче досягти, коли: Команди, члени яких географічно розкидані або неповний робочий час: у Скразі розробники повинні мати тісні та постійні взаємодії, ідеально працюючи разом в одному просторі, більшу частину часу. Хоча останні поліпшення в технології скоротили вплив цих бар'єрів (наприклад, будучи в змозі співпрацювати на цифровій дошці), Agile маніфест стверджує, що найкраща комунікація лицем до лиця. Команди, члени яких мають дуже спеціалізовані навички: у Скрам розробники повинні мати т-подібну кваліфікацію, дозволяючи їм працювати над завданнями за межами їхньої спеціалізації. Це може заохочуватися добрим керівництвом Скраму.

Хоча члени команди з дуже конкретними навичками можуть і робити свій внесок добре, вони повинні бути заохочені, щоб дізнатися більше про і співпрацювати з іншими дисциплінами. Продукти з багатьма зовнішніми залежностями: у Скразі, розділивши розробку продукту на короткі спринти, потрібне ретельне планування; зовнішні залежності, такі як прийняття користувачем тестування або координація роботи з іншими командами, можуть призвести до затримок і провалу індивідуальних сспринтів. Продукти, які є зрілими або застарілими або з регульованим контролем якості: у Скразі, приріст продукту повинен бути повністю розроблений і випробуваний в одному спринті; продукти, які потребують великої кількості регресійного тестування або тестування на безпеку (наприклад, медичні пристрої або управління автомобілем) для кожного релізу менш підходить

для коротких спрінт, ніж більше водоспаду-релізів. З точки зору бізнесу, Скрам має багато чеснот, однією з яких є те, що вона покликана принести кращі бізнес-рішення. Тим не менш, ефективність, за допомогою якої вона робить це в будь-якій організації може сильно відрізнятися і значною мірою залежить від здатності організації дотримуватися керівних принципів. Кожна компанія має свою власну організаційну структуру, культуру і набір ділової практики, а деякі з них більш природним піддається цієї методології, ніж інші. Інструменти для реалізації Головна стаття: порівняння програм Скрам Як і інші Agile методи, ефективне прийняття Скраму можна підтримувати за допомогою широкого спектру інструментів. Багато компаній використовують універсальні інструменти, такі як електронні таблиці для створення і підтримки артефактів, таких як спринт відставання. Також існують відкриті програмні пакети для Скрам, які присвячені розробці продукту за допомогою рамок Скрам або підтримки багатьох підходів до розробки продуктів, включаючи Скрам. Інші організації здійснюють програму «Скрам» без програмних засобів та зберігають артефакти в копіювальні форми, такі як паперові, дошки та наліпки.

«Скрам» — це емпіричний підхід, який є, як і всі емпіричні процеси контролю, що підтримується трьома стовпами прозорості, інспекції та адаптації. Усі роботи в рамках програми «Скрам» повинні бути видимими для тих, хто відповідає за результат: процес, робочий процес, прогрес тощо. Для того, щоб зробити ці речі видимими, команди скрам потрібно часто перевіряти продукт, який розробляється, і наскільки добре працює команда. При частій інспекції, команда може визначити, коли їхня робота відхиляється за межі прийнятних меж і адаптувати свій процес або продукт в стадії розробки. Ці три кити вимагають довіри і відкритості в колективі, що передбачає наступні п'ять значень Скрам: Прихильність: члени команди індивідуально зобов'язуються досягти своїх командних цілей, кожен спринт. Мужність: члени команди знають, що вони мають мужність працювати через конфлікти і проблеми разом, щоб вони могли робити правильні речі. Фокус: члени команди зосередитися виключно на своїх командних цілей і Sprint відставання; там не повинно бути ніякої роботи, ніж через їх відставання. Відкритість: члени команди та їх зацікавлені сторони погоджуються бути прозорими щодо їхньої роботи та будь-яких проблем, з якими вони стикаються. Повага: члени команди поважають один одного, щоб бути технічно здатними і працювати з хорошим наміром. Адаптація [edit] Спільна гібридизація Скраму з іншими методиками розробки програмного забезпечення є спільною, оскільки Скрам не охоплює весь життєвий цикл розробки продукту; тому організації знаходять необхідність додавання в додаткові процеси для створення більш комплексної реалізації. Наприклад, на початку розробки продукту організації зазвичай додають керівництво процесом у бізнес-справі, збір і пріоритетність вимог, початковий дизайн високого рівня та прогнозування бюджету та розкладу. Різні автори та спільноти людей, які використовують Скрам, також запропонували більш

докладні методи, як застосовувати або адаптувати Скрам до конкретних проблем або організацій. Багато хто посилається на ці методологічні методи як "Візерунки"-за аналогією з шаблонами дизайну в архітектурі і програмному забезпеченні. такі моделі розширили Скрам поза межами домену розробки програмного забезпечення до виробництва, Фінанси та людські ресурси.

Головна стаття: Scrumban Scrumban — модель виробництва програмного забезпечення, заснована на Скразі та Канбан. Scrumban особливо підходить для обслуговування продуктів з частими та несподіваними предметами роботи, такими як дефекти виробництва або помилки програмування. У таких випадках обмежені в часі сштс рамки Скраму можуть сприйматися як менша вигода, хоча щоденні події Скрам та інші практики все ще можуть застосовуватися, залежно від команди та ситуації під рукою. Візуалізація етапів роботи і обмежень для одночасної незавершеної роботи і дефектів знайомі з Канбану моделі. Використовуючи ці методи, робочий процес команди спрямований таким чином, що дозволяє мінімальний час завершення для кожного робочого елемента або помилки програмування, а з іншого боку гарантує, що кожен член команди постійно працює. Для ілюстрації кожного етапу роботи, команди, що працюють в одному просторі, часто використовують поштові нотатки або велику дошку. у випадку децентралізованих команд, стадія-ілюстрація програмного забезпечення, такого як складання, JIRA або Agilo.

Головними відмінностями між Скрам і Канбану є те, що у Скразі робота ділиться на спрінт, що останні фіксована кількість часу, тоді як в Канбан потік роботи безперервний. Це видно в таблицях робочих етапів, які у Скразі спорожніли після кожного спринту, тоді як в Канбан всі завдання позначені на одній таблиці. Скрам зосереджується на групах з багатогранним ноу-хау, а Канбан робить можливим спеціалізовані, функціональні команди. Скрам зі створення Скрам scrums-техніка для роботи зі Скрам за шкалою, для декількох команд, що працюють над тим самим продуктом, що дозволяє їм обговорювати прогрес на взаємозалежностях, зосереджуючись на тому, як координувати доставку програмного забезпечення , особливо на ділянках перекриття та Інтеграції. В залежності від каденції (часу) скраму scrums, відповідний щоденний скрам для кожної команди скраму закінчується, призначивши одного з членів як посла для участі у скразі з послів інших команд. Залежно від контексту, посли можуть бути технічними учасниками або майстром скрам кожної команди. Замість того, щоб просто оновлення прогресу, скрам scrums повинен зосередитися на тому, як команди колективно працюють, щоб вирішити, пом'якшити або прийняти будь-які ризики, перешкоди, залежності і припущення (Рідас), які були виявлені. Скрам scrums відстежує ці ріди через відставання своїх власних, таких як дошка ризику (іноді відомий як Рада БРОДЯТЬ після ініціалів вирішені, належать, прийняті, і пом'якшені), що, як правило, призводить до більшої координації та співпраці між командами.

Це має працювати подібно до щоденного скраму, де кожен Посол відповідає на такі чотири запитання: Які ризики, перешкоди, залежності або припущення, ваша команда вирішена, оскільки ми зустрілися? Які ризики, перешкоди, залежності або припущення допоможуть вашій команді вирішити, перш ніж ми зустрінемося знову? Чи є нові ризики, перешкоди, залежності або припущення уповільнення вашої команди або отримати на їхньому шляху? Ви збираєтеся ввести новий ризик, перешкода, залежність, або припущення, що буде отримати на шляху іншої команди? Як прокоментував Джефф Сазерленд, Так як я спочатку визначив Скрам Scrums (Кен Швабера був на IDX працювати зі мною), я можу остаточно сказати, Скрам Scrums не є "мета скраб". Скрам Scrums, як я використав це відповідає за доставку робочого програмного забезпечення всіх команд до визначення зроблено в кінці Sprint, або для релізів під час Sprint.

Патинохранитель доставлений у виробництво чотири рази на спринті. Ancestry.com доставляє до виробництва 220 разів на два тижні спринт. Хюбспот забезпечує живе програмне забезпечення 100-300 разів на день. За прийняття цієї роботи відповідальний за роботу майстра з Scrums. Отже, Скрам Scrums є експлуатаційним механізмом доставки. Широкомасштабний Скрам Широкомасштабний Скрам (LeSS) – це основа розробки продуктів, яка розширює свої правила та керівні принципи без втрати початкових цілей Скраму. Є два рівні в рамках: перший менший рівень призначений для до восьми команд; Другий рівень, відомий як "менш величезний", вводить додаткові елементи масштабування для розробки з до сотень розробників. «Масштабування Скраму починається з розуміння та можливості прийняти стандартну справжню команду «Скрам». Широкомасштабний Скрам вимагає вивчення мети однокомандних елементів Скраму і з'ясування того, як досягти тієї ж мети, залишаючись в межах обмежень стандартних правил Скраму ". Bas Водде і Крейг Ларман розвивалися менш рамки від свого досвіду роботи з великомасштабні розробки продуктів, особливо в телекомунікаційній і фінансовій промисловості. Він еволюціонував, прийнявши Скрам і намагаючись багато різних експериментів, щоб виявити, що працює. У 2013, експерименти були затверділих в менш рамкові правила. намір менше є "демасштабною" організацією складності, розчинення непотрібних складних організаційних рішень, а також рішення їх у простіших відносинах. Менше ролей, менше управління, менш організаційних структур.

1.2.2 Вибір мови програмування

Серверна розробка є однією з найбільш затребуваних навичок сьогодні. Майже будь-який новий бізнес/стартап потрібен веб-сайт і мобільний

додаток-обидва з яких підключаються до сервера в бекенда. Тому, бекенда розробники користуються високим попитом на ринку і компанії готові кинути великі суми грошей для розробників, які можуть керувати бекенда добре. На ринку з'являються багато мов програмування, які використовуються для розвитку бекенда:

Python: Python – один з найпопулярніших варіантів програмування на внутрішньому сервері. Вона відносно нова і має величезну бібліотечну підтримку.

PHP: PHP був на ринку протягом тривалого часу, і він широко використовується навіть сьогодні. Facebook, наприклад, має свій істотний бекенда розроблена в PHP в початкові дні. JavaScript: з потужними веб-фреймворками, такими як NodeJS, JavaScript захопив величезний шматок ринку і виник як один з найпопулярніших серверної мови програмування.

Ruby: Рубін на рейки є одним з найбільш популярних веб-розробки фреймворки і в даний час тонни стартапів використовувати його.

Точка NET: .NET Framework має свій власний вентилятор наступним. Спочатку DOT NET було більше власної рамки, але наприкінці Microsoft робить велику роботу по відношенню до відкритого джерела. Точка net використовується в основному на підприємствах з-за твердої підтримки Microsoft.

Для нових розробників, це завжди питання плутанини-який сервер основи, щоб дізнатися? з такою кількістю нових мов програмування, які пропонують так багато функцій, бібліотек та фреймворків, як насправді вирішує, на якій веб-фреймворку вчитися? У цьому блозі, ми постараємося порівняти 2 з найбільш популярних серверної мови програмування-Python і PHP. Ми постараємося охопити різні аспекти і подивитися, який з них виграє. Перш ніж ми зробимо порівняння, давайте спочатку Перелічіть точки порівняння, які суттєво вплинуть на наш вибір веб-фреймворку: Простота навчання: це, мабуть, один з найважливіших параметрів для того, щоб вирішити, який веб-каркас буде використовуватися. Якщо мова програмування важко дізнатися, немає сенсу витрачати час на нього.

Сьогодні час розробника важливіше, ніж час виконання для всіх практичних цілей. Підтримка спільноти: давайте подивимося правді в очі-ми всі боремося з помилками, ми всі стикаємося з проблемами під час написання програм, і ми всі дивимося на підтримку в Інтернеті за адресою і інші

форуми. Якщо конкретна мова програмування не добре відома, і є мало підтримки спільноти доступні, краще триматися подалі від нього. Документація: так само, як підтримка спільноти, важливо, щоб мова програмування/Framework має достатню документацію, доступну для розробників, щоб дізнатися і зрозуміти нюанси. Ціноутворення: деякі інструменти/фреймворки не є вільними. Це може суттєво вплинути на вибір організації, яка є низькою за бюджетом.

Взагалі кажучи, велика кількість технологічних компаній воліють використовувати засоби з відкритим вихідним кодом і фреймворки, а не використовувати платні системи. Однак, такі підприємства, як банки, страхові компанії і т. д. воліють використовувати платні системи. Бібліотека підтримки: Якщо мова програмування широко використовується, буде більше розробників, які будуть розробляти бібліотеки для конкретної мови. Як наслідок, розвиток стає ще простішим. Швидкість: Серверні програми можуть вимагати високої терпимості, а також низької затримки. Тому важливо, щоб побачити, яка мова швидше з точки зору виконання часу. Вибір веб-фреймворків: важливо, що мова програмування забезпечує добре продумані фреймворки веб-розробки, які є легкими у використанні і розробкою потужних додатків. Зневадження: вибір мови програмування також має залежати від доступних засобів налагодження, доступних для мови. Відсутність хороших інструментів зневадження означає, що розробники збираються витратити більше часу на налагодження, який по суті не є найбільш продуктивним використання часу.

PHP vs Python ефективності порівняння Тепер давайте перейдемо до кожного з перерахованих вище факторів і подивимося, як Python і PHP порівнюють один з одним. Простота навчання І, без сумніву, Python набагато легше освоїти. Python — це мова програмування загального призначення, яка може бути підхоплена дуже швидко. Насправді, Python так просто підібрати, що більшість курсів програмування для початківців тепер використовувати мову програмування Python, щоб навчити основам програмування. Python програми набагато коротше і легко писати в порівнянні з іншими мовами програмування і, як наслідок, вона стала кращим вибором для багатьох додатків. Синтаксис набагато простіший і код дуже читається у порівнянні з тим самим кодом, написуваному іншими мовами програмування. PHP, з іншого боку, не означало бути мовою загального призначення. Він був розроблений спеціально для веб-додатків, які, безумовно, більш складні, ніж прості, автономні програми. В результаті, навчання PHP було видно зайняти більше часу в порівнянні з навчанням Python. Час, щоб вивчити мову програмування має бути одним з найважливіших факторів у виборі мови для вибору.

Для новачків Python набагато простіше. PHP, з іншого боку, може бути трохи жорстким для початківців програмістів. PHP був розроблений для створення простих особистих сторінок, але з кінця він виріс за складністю. Спільнота розробників PHP намагається забезпечити багато підтримки для нових програмістів. Однак, як уже згадувалося вище, Python виграє тут

значну націнки завдяки притаманній простоті мови. Синтакси та конструкції на Python дивно прості, щоб зрозуміти, Підтримка спільноти Python і PHP, обидва мають відмінну підтримку спільноти. PHP був на ринку досить довгий час, особливо для розробки веб-додатків. В результаті, існує величезна спільнота розробників PHP, який готовий надати підтримку. Python матчі тісно тут з PHP. Є безліч розробників Python на ринку, які безперервно розробляють програми Python.

Як наслідок, підтримка спільноти є видатною. Python і PHP обидва близькі тут ніхто з них не є чітким переможцем. Python став популярним, коли Google почав використовувати його для деяких популярних служб Google, таких як YouTube. Багато потужних стартапів, таких як Instagram, Pinterest та веб-додатки для Python. Сказавши це, слід зазначити, що найбільша соціальна мережа у світі-Facebook була написана з використанням PHP в якості первинного бекенда. Документації Велика документація доступна як для мов програмування. Є незліченна сайти, форуми, дошки обговорень, які надають відмінні підручники про те, як розробляти додатки з використанням Python або PHP. Конкуренція є жорсткою тут і так само, як спільнота підтримки, насправді не є чітким переможцем. Обидві мови однаково гарні з точки зору доступності документації.

Підтримка для комунічності Python і PHP, обидва мають відмінну підтримку спільноти. PHP був на ринку досить довгий час, особливо для розробки веб-додатків. В результаті, існує величезна спільнота розробників PHP, який готовий надати підтримку. Python матчі тісно тут з PHP. Є безліч розробників Python на ринку, які безперервно розробляють програми Python. Як наслідок, підтримка спільноти є видатною. Python і PHP обидва близькі тут ніхто з них не є чітким переможцем. Python став популярним, коли Google почав використовувати його для деяких популярних служб Google, таких як YouTube. Багато потужних стартапів, таких як Instagram, Pinterest та веб-додатки для Python. Сказавши це, слід зазначити, що найбільша соціальна мережа у світі-Facebook була написана з використанням PHP в якості первинного бекенда. Документації Велика документація доступна як для мов програмування. Є незліченна сайти, форуми, дошки обговорень, які надають відмінні підручники про те, як розробляти додатки з використанням Python або PHP.

Конкуренція є жорсткою тут і так само, як спільнота підтримки, насправді не є чітким переможцем. Обидві мови однаково гарні з точки зору доступності документації. Ціноутворення Python і PHP є абсолютно вільними і відкритими вихідними. І виграти тут також. Насправді, як Python і PHP тут виграти значно більше інших платних веб-фреймворків. Підтримка бібліотеки Тепер, це одна точка, де Python значно б'є PHP. Python має винятково розвинену бібліотеку підтримки майже всіх типів програм. PHP відстає в цьому аспекті WRT на Python, але пакувіст (репозиторій PHP пакунків) є сильним опорним холдингами PHP. У ці дні, наприклад, багато стартапи і навіть великі організації розробляють Машинне навчання підтримки веб-додатків. Python надає деякі відмінні бібліотеки машинного

навчання, як TensorFlow, Keras, Theano, Scikit вчитися і т. д. Ці бібліотеки є швидкими, легкими у використанні і, найголовніше, вони інтегруються блискуче з веб-фреймворку. В результаті, розробка такого типу додатків, що використовують Python, набагато простіша порівняно практично з будь-якою іншою мовою програмування.

Вибір веб-фреймворків Найбільш часто використовувані веб-фреймворки на основі Python є Django, колб, пілони, пірамідою тощо. З іншого боку, найбільш ужиті PHP основі веб-фреймворки є Козапальник, Zend, Лававель, Симфнія і т. д. Django, як відомо, дуже швидкий, масштабований, безпечний і простий у використанні. Вона досить надійна і потужна і широко використовується у великій кількості застосувань. Аналогічним чином, копідзапальник і Лавель дуже широко використовуються на ринку і майже всі додатки PHP сьогодні використовують одну з вищевказаних рамок 2. Python і PHP є досить близькі на цій точці дається, що обидва забезпечують однаково хороший вибір. Нові розробники, однак, користуються використанням Django враховуючи, що час розробки в Django досить низьким і легко налаштувати.

Налагодження Python забезпечує потужний відладчик під назвою PDB (відладчик для Python). PDB добре документовані і простий у використанні, навіть для початківців. PHP, з іншого боку, забезпечує відлагодження пакет для налагодження. Обидва PDB і відлагодження забезпечують найбільш часто використовувані функції налагодження-точки зупинки, стеки, відображення шляху і т. д. Обидва Python і PHP схожі тут, і ніхто не є чітким переможцем. Підводячи підсумок, для більшості точок, як Python і PHP схожі один на одного. Для інших, Python краще, ніж PHP. Python, здається, переможець над PHP. Ось що ми рекомендували б: Якщо ви досвідчений програміст PHP, дотримуватися PHP, так як ви вже знаєте його і виходити. Якщо ви є проміжним програмістом, ви можете вивчати Python і перейти до нього для поліпшення можливостей роботи. Якщо ви початківець програміст, який хоче навчитися бекенда розвитку, почати вивчати Python і в остаточному підсумку перейти на Python основі фреймворків

1.2.3 Побудова діаграми класів