МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Програмний модуль підтримки діяльності DevOps інженера»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-52 Нечепорук Олександр Андрійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2019 р.

Науковий керівник _____ к.т.н., доц., Ващенко С.М.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В.В. Шендрик
«__» _____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Нечепорук Олександр Андрійович

1 Тема роботи *Програмний модуль підтримки діяльності DevOps інженера*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « 17 » травня 2019 р. № 0834-III

2 Строк подання студентом роботи «3» червня 2019 р.

3 Вхідні дані до роботи технічне завдання на розробку програмного модуля

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі, моделювання програмного модуля підтримки діяльності DevOps інженера, розробка програмного модуля підтримки діяльності DevOps інженера.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) опис діяльності DevOps інженера, актуальність роботи, мета та задачі, дослідження аналогів, функціональні вимоги, контекстна діаграма процесу використання програмного модуля, діаграма декомпозиції процесу використання програмного модуля, діаграма варіантів використання, інформаційна база, вибір засобів реалізації, робота програмного модуля, результат роботи, практична значимість.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	10.03.19 – 15.03.19	
2	Визначення функціоналу програмного модуля	15.03.19 – 20.03.19	
3	Вибір технічних засобів для розробки	20.03.19 – 25.03.19	
4	Дослідження існуючих методів роботи з шаблонами	25.03.19 – 05.04.19	
5	Розробка макету програмного модуля	08.04.19 – 12.04.19	
6	Практична реалізація	15.04.19 – 19.05.19	
7	Тестування програмного модуля	20.05.19 – 31.05.19	
8	Оформлення документації	20.05.19 – 31.05.19	

Студент _____ Нечепорук О.А.
(підпис)

Керівник роботи _____ к.т.н., доц. Ващенко С.М.
(підпис)

РЕФЕРАТ

Тема дипломного проекту: «Програмний модуль підтримки діяльності DevOps інженера».

Метою проекту є розробка програмного модуля, який дозволить DevOps інженеру створити інструментарій розгортання пакетів програмного забезпечення та налаштувати його в залежності від проекту виключивши вірогідність помилки та скоротивши витрати часу на даному етапі.

Актуальність даної роботи полягає в тому, що зниження витрат робочого часу та скорочення ймовірності похибки при створенні інструментарію розгортання дозволяє знизити навантаження інженерів та підвищити якість пакування програмного забезпечення.

Результатом виконання проекту є розроблений програмний модуль, що дозволяє DevOps інженеру підготувати пакет програмного забезпечення до розгортання на робочих станціях кінцевих користувачів шляхом автоматичної генерації з можливістю редагування інструментарію розгортання.

Пояснювальна записка містить 99 сторінок, 6 таблиць, 38 рисунків, список використаної літератури з 18 джерел, 5 додатків.

Ключові слова: DevOps, PSADT, програмний модуль, розробка, пакування, програмне забезпечення, MSI, PowerShell, cmd, App-V, Installer, application packaging.

ЗМІСТ

ВСТУП

Одним із етапів роботи DevOps інженера є створення та налаштування інструментарію для розгортання попередньо створених пакетів ПЗ на кінцевих робочих станціях (PowerShell App Deployment Toolkit, надалі PSADT)[1], який надає набір функцій для виконання звичайних завдань розгортання програм та взаємодії з користувачем під час розгортання. Це спрощує складні задачі сценаріїв розгортання додатків на підприємстві та забезпечує послідовний досвід розгортання та покращує показники успішної інсталяції.

На сьогодні всі етапи пакування ПЗ потребують значного впливу зі сторони DevOps інженера, а якщо врахувати різноманіття додатків та відмінності вимог кожного замовника та проекту, то стає очевидно, що цей процес потребує всебічну автоматизацію. У зв’язку з цим, актуальною є задача розробки програмного модуля для автоматизації процесу генерації та налаштування PSADT залежно від специфіки технології пакування та вимог проекту і замовника.

На основі аналізу існуючих методів було сформульовано мету роботи – розробити програмний модуль, який дозволить DevOps інженеру створити інструментарій розгортання пакетів ПЗ та налаштувати його в залежності від проекту виключивши вірогідність помилки та скоротивши витрати часу на даному етапі.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати роботу DevOps-інженера та програмне забезпечення, що використовується;
- скласти технічне завдання на виконання розробки;
- виконати моделювання системи;
- розробити шаблони та налаштуваня для тих проектів, з якиим працює DevOps-інженер;

- виконати кодування програмних модулів;
- провести тестування програмного продукту.

Результат дипломного проекту, а саме розроблений програмний модуль має практичне значення, адже зможе використовуватися DevOps інженерами відділу пакування та розробки програмного забезпечення компанії Apptimized Operations.

Основні результати роботи, зокрема програмне забезпечення, впроваджено у робочий процес відділу пакування та розробки програмного забезпечення ТОВ «Apptimized Operations», що засвідчує акт впровадження (додаток А).

Результати роботи доповідалися на щорічній конференції «Інформатика, математика, автоматизація – 2019» (м. Суми) [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис роботи DevOps інженера

DevOps – сукупність практик, призначених для збільшення взаємодії між розробниками програмного забезпечення (ПЗ) та фахівцями інформаційно-технологічного обслуговування (як системних адміністраторів, так і кінцевих користувачів), зближення їх робочих процесів одне з одним та скорочення затрат на розгортання, встановлення, оновлення, підтримку та видалення ПЗ на кінцевих робочих станціях, що пояснює актуальність впровадження DevOps у виробництво чи бізнес.[3]

Із практичної точки зору, DevOps спеціалісти визначають вимоги до інфраструктури, налаштовують ПЗ відповідно до вимог кінцевого середовища, займаються розгортанням продукту, а також подальшою підтримкою програмного забезпечення та його тестуванням для перевірки готовності робочого середовища. Основною метою роботи DevOps інженера є автоматизація вищеописаних процесів.

У зв’язку зі зростанням попиту на послуги DevOps інженерів, останні роки характеризуються значним розвитком галузі пакування додатків.[4]

На сьогоднішній день є декілька підходів при створенні інструментарію розгортання пакетів ПЗ. В залежності від проекту можуть використовуються раніше створені шаблони, які попередньо налаштовані з врахуванням вимог конкретного проекту, проте потребують додаткового налаштування вручну DevOps інженерами. Деякі проекти мають власні інструменти для генерації PSADT, проте вони не є достатньо гнучкими та можуть бути використані лише для певного проекту.

Процес пакування ПЗ можна розділити на декілька послідовних етапів, а саме: аналіз вихідних файлів програми, власне процес створення пакету додатку та налаштування інструмента для розгортання готового пакету

програми PSADT, який в свою чергу представляє собою ієрархію директорій і файлів, які містять в собі набір функцій для виконання звичайних завдань розгортання програм та основний файл-сценарій мовою PowerShell.

На етапі аналізу вхідних файлів, необхідно завантажити вихідні файли програми. Зазвичай це інсталятори, створені власне розробником даного ПЗ, або набір файлів чи набір дій, які необхідно виконати для реалізації поставленої задачі. Окрім файлів програм необхідно ретельно вивчити документацію, надану замовником, ознайомитися з інструкціями та вимогами стосовно налаштування даного ПЗ чи операційної системи.

Після вивчення всіх вхідних даних та обговорення деталей з замовником можна переходити до власне процесу пакування. В залежності від логіки розробника ПЗ та вимог замовника, обирається технологія пакування ПЗ. На сьогоднішній день можна визначити наступні основні технології:

- MSI;
- Legacy;
- App-V.

MSI – формат Windows Installer, що в свою чергу є програмним компонентом та інтерфейсом прикладного програмування (API) Microsoft Windows, що використовується для встановлення, обслуговування та видалення програмного забезпечення. Інформація про інсталяцію, файли, ключі реєстру, контроль сервісів та інші компоненти операційної системи зберігаються у вигляді реляційної бази даних всередині файлів з розширенням .msi.[5]

Інший, менш гнучкий але зручна технологія – legacy (або silent) installation. Технологія «тихого» встановлення не потребує взаємодії з користувачем та використовується зазвичай для інсталяції комплексних програмних продуктів, таких як Microsoft Office, CAD/CAM системи, пакети Adobe та інші багатокомпонентні програми. Для застосування цієї технології мають застосовуватися спеціалізовані інсталятори (InstallShield, Inno Setup,

WIX та інші) або параметри «тихої» інсталяції мають бути передбачені розробником ПЗ на етапі розробки.

App-V, скорочено від «Application Virtualization» є складовою частиною настільних і серверних операційних систем, що забезпечує можливості поділу та ізоляції для програм, що працюють в операційній системі. Налаштування програми здійснюється за допомого «App-V Sequencer», який в свою чергу є інструментом для захоплення змін в системі та їх трансформацію у контейнери. Часто можна одноразово захопити пакет і використовувати його на декількох версіях операційної системи. Після встановлення створеного App-V додатку на хостовій системі користувача всі складові частини пакету програми зберігаються в ізоляції від основної ОС, у спеціальній віртуалізованій області. Дана технологія є зручною для використання на термінальних серверах або для мінімізації впливу на хостову операційну систему (наприклад, з міркувань безпеки), проте із цього витікає основний недолік – програми, що містять драйвери, COM+ об'єкти, додатки які потребують зв’язку з будь-якими елементами хостової ОС не можуть бути віртуалізовані.[6]

Останнім етапом пакування є створення та налаштування інструментарію для подальшого розгортання вже створених пакетів програм, створення супровідної документації та тестування пакету QA інженером.

Розглянемо інструменти розгортання пакетів більш детально. В залежності від проектів та замовників, використовуються наступні технології:

- використання .cmd файлів;
- використання PSADT;
- інші специфічні інструменти.

Використання «install.cmd» та «uninstall.cmd» файлів різко зменшує гнучкість конфігурації операційної системи для встановлення чи видалення пакету ПЗ, адже за кращою практикою ці файли можуть містити в собі лише рядки власне інсталяції чи видалення програми чи її компонентів. Через

обмеженість функціональності необхідні налаштування доводиться включати в пакет, що не завжди є прийнятим рішенням. На сьогоднішній день, залишилось лише декілька замовників, що використовують дану технологію в своїх проектах.

Також можуть використовуватися специфічні інструменти в рамках певних проектів. Наприклад, для пакетів App-V або ThinApp не потрібно додаткових налаштувань, адже завдяки особливостям технології віртуалізації, дані додатки вже матимуть всі необхідні налаштування всередині власних компонентів. Частіше за все, для встановлення таких додатків необхідне додаткове програмне забезпечення, яке розпізнає певний формат віртуалізації.

Найпопулярнішим інструментом розгортання пакетів є PSADT – це інструментарій розгортання програм PowerShell що може бути використаний для будь-якого механізму розгортання. Він написаний повністю в PowerShell і має багато функцій для конфігурації цільової операційної системи на різних етапах встановлення чи видалення пакету ПЗ. Основними його перевагами є простота використання завдяки попередньо визначених функцій роботи з ОС, послідовність виконання дій, можливість конфігурації користувацького інтерфейсу під час встановлення чи видалення ПЗ, локалізація під основні мови світу (в тому числі і російську), постійні оновлення та інтеграцію з SCCM (продукт для керування IT-інфраструктурою на основі MS Windows).[7]

Необхідно зазначити, що для кожного конкретного замовника та проекту шаблонний PSADT модифіковано таким чином, щоб задовольнити всі вимоги. Для того, щоб краще зрозуміти основні принципи роботи PSADT, розглянемо його на прикладі стандартного «Apptimized Wrapper», що є модифікацією шаблонного PSADT, до якого було додано додаткові функції та модифіковано основний файл скрипт.

На Рисунку 1.1 можна побачити структуру PSADT, що складається з трьох папок і трьох файлів. Папка AppDeployToolkit містить файли

конфігурації та файли функцій. Папка Files та SupportFiles використовується для вихідних файлів. Там будуть розміщені файли, необхідні для встановлення пакета.

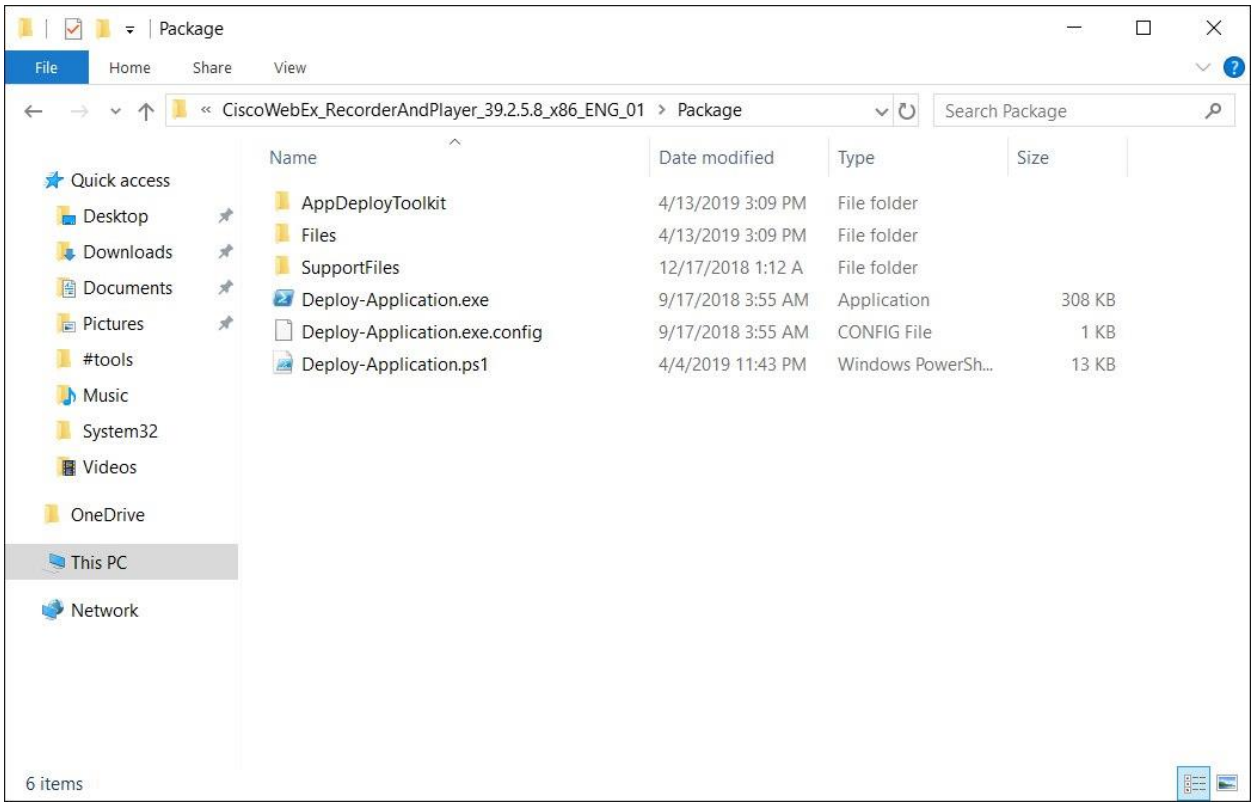


Рисунок 1.1 - Структура PSADT

Якщо перейти до папки AppDeployToolkit, ви знайдете такі файли:

- AppDeployToolkitBanner.png (банер, який бачить користувач під час інсталяції)
- AppDeployToolkitConfig.xml (конфігураційний файл із налаштуваннями мови тощо)
- AppDeployToolkitExtensions.ps1 (сценарій з розширеннями)
- AppDeployToolkitHelp.ps1 (файл допомоги, що пояснює функції)
- AppDeployToolkitLogo.ico (файл значків)
- AppDeployToolkitMain.cs (деякі додаткові функції)
- AppDeployToolkitMain.ps1 (основний сценарій з усіма функціями)

- SetACL.exe (утиліта для контролю прав доступу користувачів в межах ОС)
- changlog.txt (лог файл змін шаблонного PSADT)

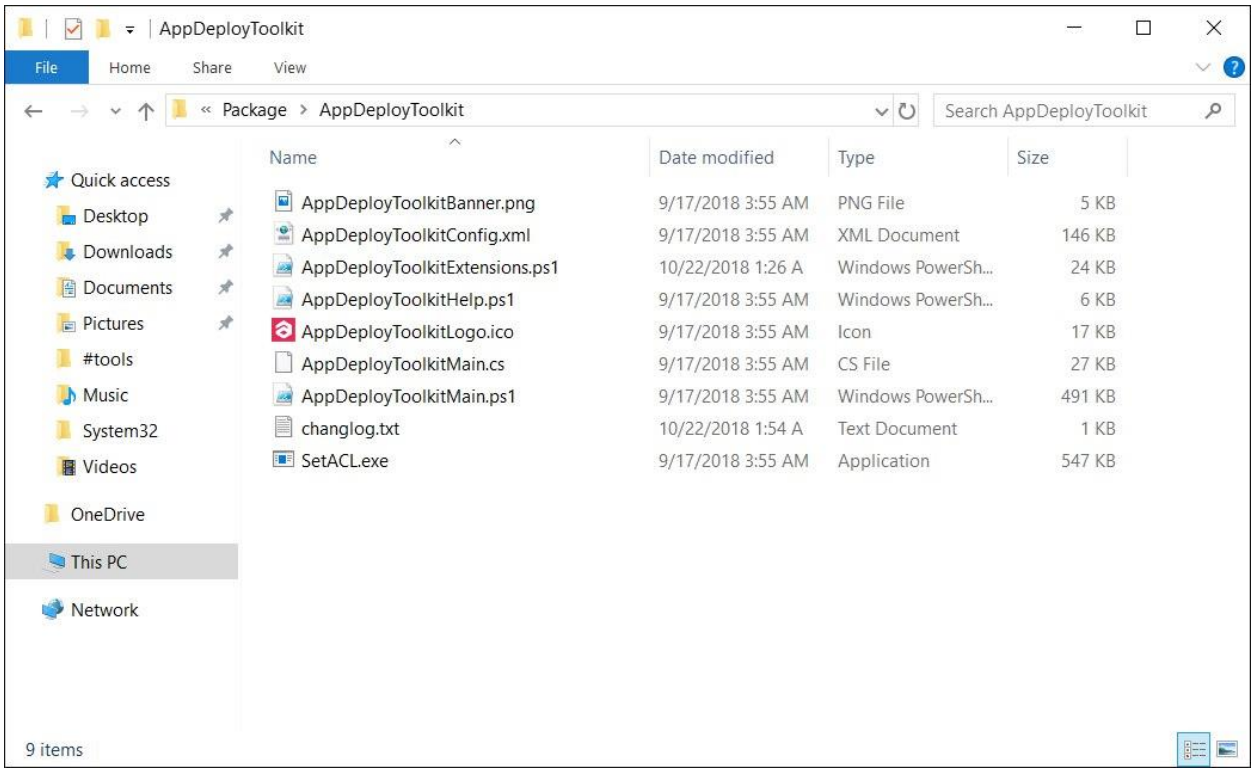


Рисунок 1.2 - Структура папки AppDeployToolkit

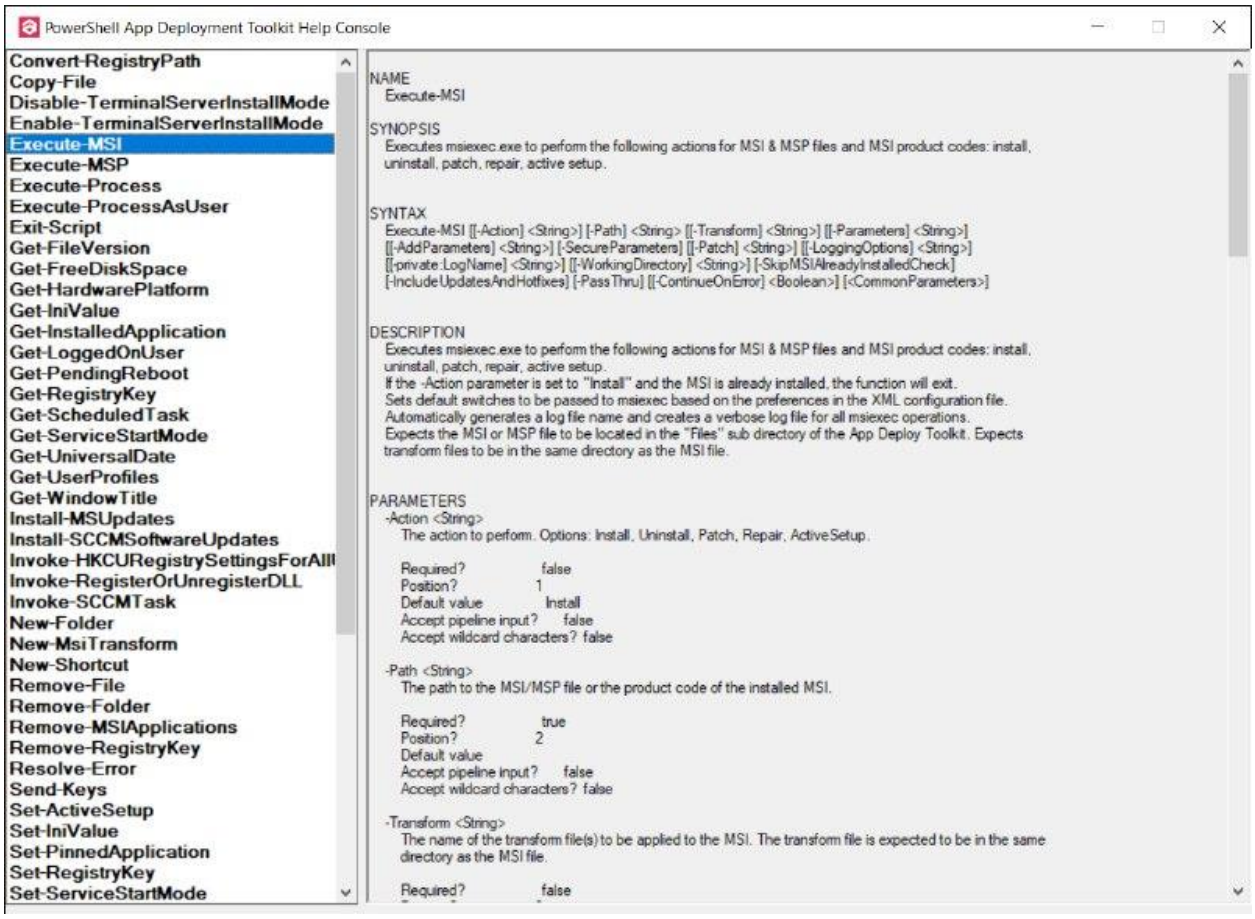


Рисунок 1.3 – Вікно допомоги

Основна взаємодія DevOps інженера з середовищем користувача відбувається в файлі Deploy-Application.ps1, в якому можна виділити три секції: оголошення змінних, секція інсталяції та секція видалення.

У розділі оголошення змінних заповнюються поля, що зберігають інформацію про пакет ПЗ, також важливою є зміна значення змінної, що визначає чи необхідне перезавантаження комп’ютера після встановлення чи видалення пакету. В залежності від її значення, після встановлення та видалення буде виведено відповідне нагадування про необхідність перезавантаження.

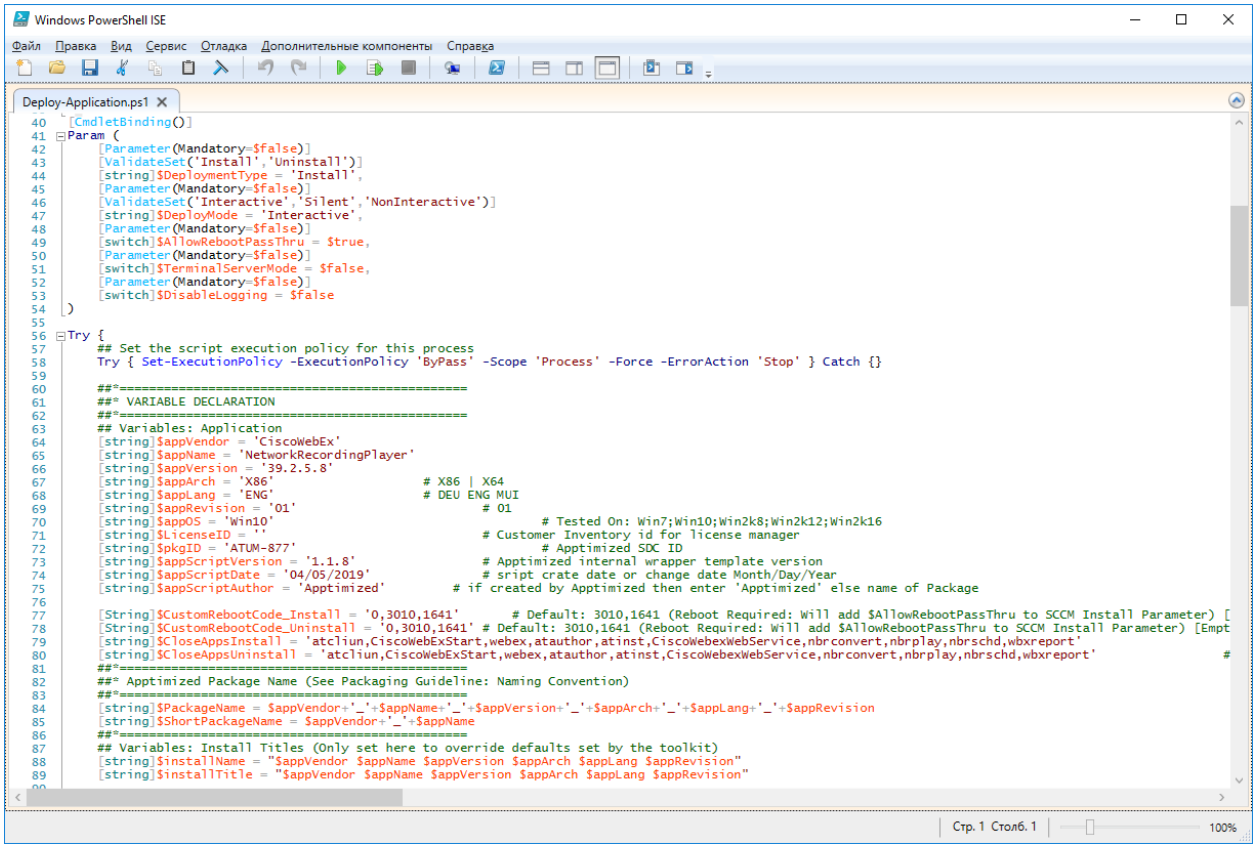


Рисунок 1.4 – Оголошення змінних у файлі Deploy-Application.ps1

Після оголошення всіх необхідних змінних DevOps інженер має можливість налаштовувати систему якщо це необхідно. Для цього виконання сценарію розділено на шість секцій: до інсталяції, власне інсталяція, після інсталяції, до видалення, власне видалення та після видалення.

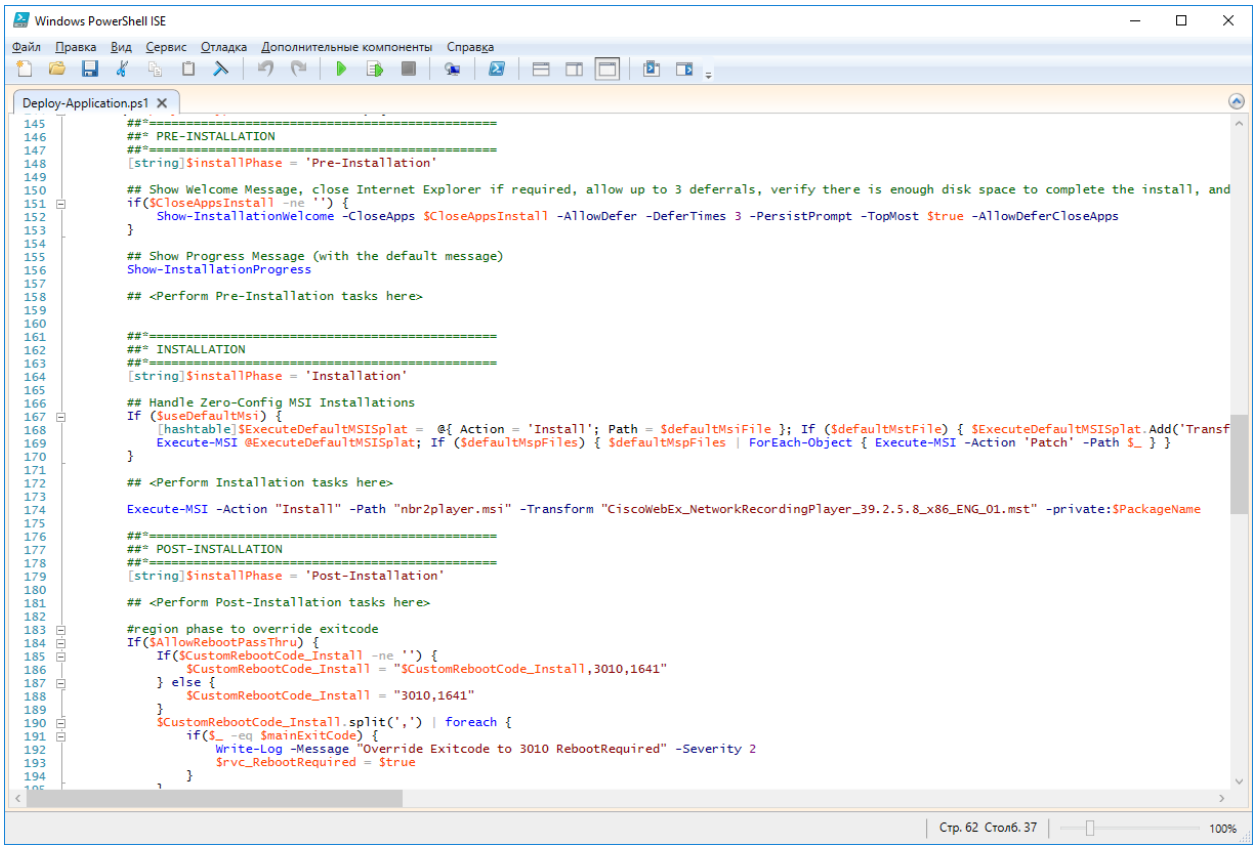


Рисунок 1.5 – Секції, що виконуються під час інсталяції пакету

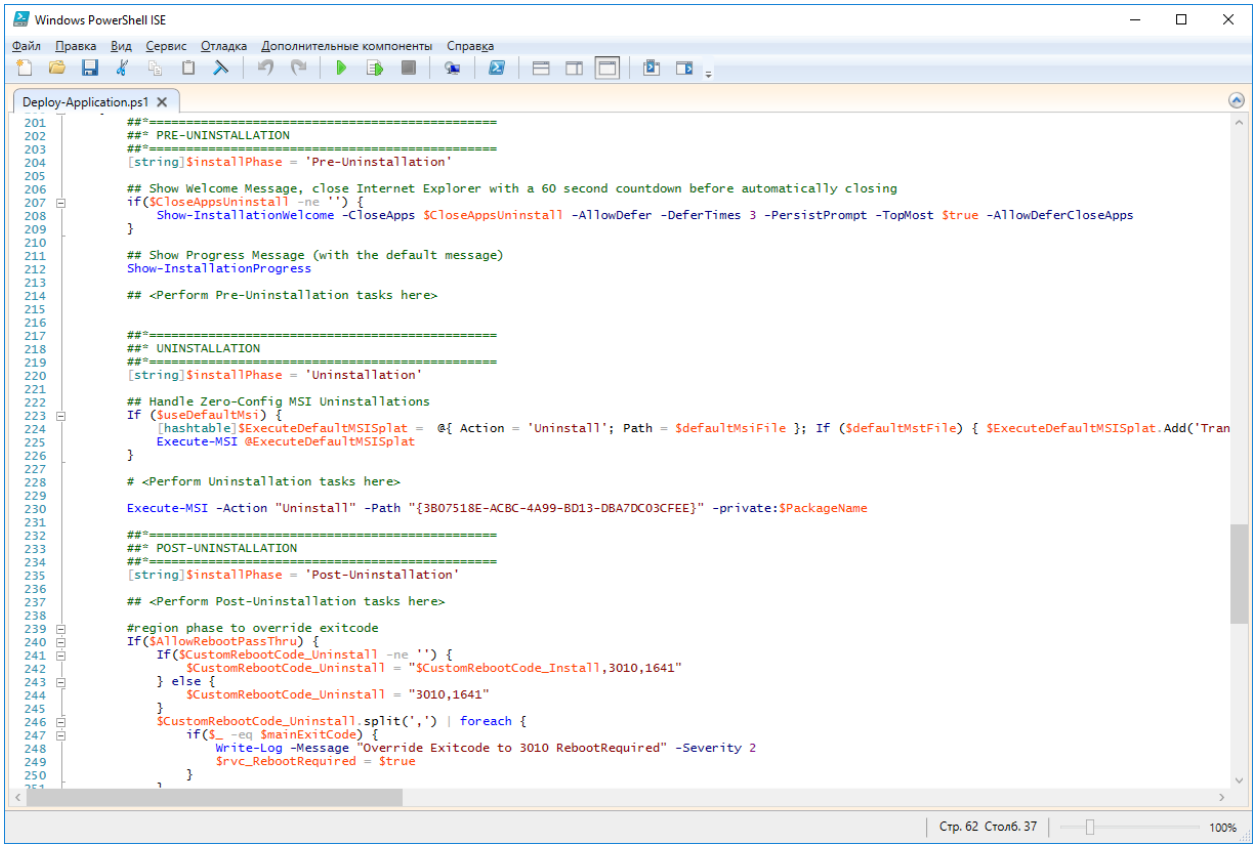


Рисунок 1.6 – Секції, що виконуються під час видалення пакету

1.2 Програмне забезпечення для автоматизації роботи

На основі проведеного аналізу предметної області можна зробити висновок, що не існує точного програмного забезпечення, яке б повністю автоматизувало та нормалізувало процес створення PSADT інструменту для розгортання пакетів ПЗ.

На деяких проектах існують власні рішення, які частково вирішують поставлену задачу генерації PSADT, проте вони не є універсальними та не мають відкритого вихідного коду, що унеможливлює модифікацію даних утиліт під налаштування інших замовників.

Враховуючи вищенаведені факти та той факт, що сьогодні даний етап пакування ПЗ DevOps-інженери виконують вручну, можна зробити висновок, що задля скорочення витрат часу та мінімізації ймовірності людської похибки необхідно розробити власний програмний модуль для генерації PSADT. Також можна константувати, що ця проблема є актуальною.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі

Метою проекту є розробка програмного модуля для автоматизації створення інструментарію розгортання пакетів програм – PSADT, його налаштування та використання зручного інтерфейсу для ручного додавання необхідних конфігурацій операційної системи із шаблонів з можливістю їх редагування на різних етапах встановлення чи видалення пакету ПЗ.

Завдяки використанню програмного модуля можна значно скоротити витрати часу та підвищити якість роботи за рахунок зменшення ймовірності помилки DevOps інженера.

Розроблений програмний модуль виконуватиме наступні функції:

- визначення структури PSADT залежно від проекту;
- копіювання вказаних користувачем файлів пакету одразу в сгенерований PSADT;
- автоматичне заповнення полів з розділу оголошення змінних з можливістю їх перевірки та модифікації користувачем;
- можливість додавати шаблонні дії (у форматі xml) для взаємодії з ОС з можливістю їх редагування користувачем;
- зберігатиме створений PSADT за вказаним шляхом на локальному чи мережевому диску.

Створений програмний модуль матиме інтуїтивно зрозумілий та зручний у користуванні інтерфейс. Оскільки інтерфейс додатку розробляється англійською мовою, то даний продукт може бути використаний не залежно від регіону чи країни.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати існуючі методи створення PSADT;
- визначити функціонал програмного модуля;

- обрати технічні засоби для розробки;
- розробити код програмного модуля;
- створити шаблони налаштувань для кожного проекту;
- провести тестування програмного модуля.

Схема роботи програмного модуля така.

1. Після запуску програмного модуля користувач має ввести унікальний ідентифікатор конкретного пакету ПЗ та вказати шлях до файлів пакету, за необхідності скоригувати технологію пакування, що заповнюється автоматично (MSI, Legacy, APP-V). В залежності від вхідних даних програма звертається до xml документу на мережевому диску, в якому зберігаються налаштування даного проекту.
2. Після цього у тимчасовій папці створюється структура PSADT з файлами із вказаної користувачем директорії. Далі на основі файлів пакету ПЗ заповнюються необхідні текстові поля, які користувач має змогу редагувати. Також є можливість додати необхідну кількість стандартних дій для конфігурації операційної системи у відповідну стадію розгортання чи видалення пакету ПЗ на кінцевих робочих станціях.
3. Після підтвердження введених даних вносяться зміни у файл сценарію та результат генерації копіюється до вказаної користувачем директорії.

Затверджене технічне завдання на виконання проекту наведено у додатку Б.

2.2 Вибір засобів реалізації

Для реалізації програмного модуля було обрано наступні засоби реалізації: об’єктно-орієнтоване програмування мовою C#, середовище розробки Microsoft Visual Studio та стандарт XML для зберігання налаштувань проектів.

C# - сучасна зручна об’єктно-орієнтована мова, зі строгою типізацією даних. Перевагою мови C# є підтримка компоненто-орієнтованого програмування, адже сьогодні все більше розробка ПЗ спирається на готові програмні компоненти у вигляді бібліотек та NuGet пакетів. Ключовим моментом для будь-якої сучасної платформи розробки є механізм, за допомогою якого розробники можуть створювати, передавати та використовувати корисний код. NuGet пакети представляють собою спеціальний архів, що містить в собі скомпільований код у форматі .dll, а також інші корисні файли, пов’язані з цим кодом та документацію. Після додавання NuGet пакету до проекту можна користуватися готовими функціями необхідного пакету. [8]

Ще одною перевагою використання C# є автоматична збірка сміття (garbage collector), яка автоматично очищує ті ділянки пам’яті, на які не існує посилань.[9] Також варто відмітити безпечну типізацію даних, що в свою чергу гарантує індексацію поза межами масивів та унеможливлює читання неініціалізованих змінних.

В якості середовища розробки було обрано Microsoft Visual Studio через зручність його використання, зокрема зручність роботи з додатками заснованих на WindowsForm. Використання Visual Studio та мови C# дозволяє використовувати NuGet пакети за їх необхідністю. Наприклад, у даному проекті використовується Atlassian.SDK для роботи з JIRA Issue tracker. Також використано пакет Costura.Fody [10], який автоматично додає всі використані бібліотеки та простори імен в Release сбірку, щоб на

отримати один результуючий виконуваний файл з вбудованими додатковими компонентами.

За зберігання даних, необхідних для роботи програмного модулю обрано стандарт XML через простоту редагування та інтеграцію з середовищем розробки. XML файли із загальною конфігурацією та окремими налаштуваннями конкретних проектів зберігаються на мережевому диску, до якого є доступ як із хостових систем, так і з віртуальних машин на підприємстві. Взаємодія з форматом xml та файловою системою в середовищі Microsoft Visual Studio здійснюється за допомогою простору імен System.Xml та System.IO відповідно.[11]

2.3 Планування робіт

Після формулювання мети розробки проекту, визначення переліку задач для її реалізації та обравши необхідні інструменти, необхідно провести планування робіт. Для цього були використані сучасні методи планування, а саме: деталізація мети розробки програмного модуля за методологією SMART[12]; розроблені діаграми OBS та WBS[13], які в свою чергу відображають ієрархічну структуру плану робіт та визначають учасників на певних етапах виконання проекту; розроблену діаграму Ганта та мережу PDM[14]; були визначені можливі ризики, що можуть виникнути при реалізації. Також було проведено розрахунок бюджету заробітної платні всіх учасників проекту.

Більш детальний опис планування робіт наведено у додатку В.

3 МОДЕЛЮВАННЯ ПРОГРАМНОГО МОДУЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ DEVOPS ІНЖЕНЕРА

Програмний модуль – завершена функціональна частина програми, збережена у окремому файлі для використання в сукупності з іншими програмами. Модулі дозволяють розбивати складні задачі на сукупність простих згідно з принципом модульності. В даному випадку, процес пакування ПЗ розбивається на створення власне файлів пакету ПЗ та створення інструментарію для розгортання та налаштування додатку та системи користувача.[15]

Створення програмного модуля має замінити ручне створення даного інструментарію в залежності від проекту та замовника.

3.1 Структурно-функціональне моделювання

Методологія IDEF0 застосовується в якості засобу аналізу та подання бізнес процесів.[16] Головними компонентами IDEF0 діаграми є блоки, які відображають процеси, завдання чи операції. Також невід’ємними елементами діаграми є:

- вхідні дані;
- вихідні дані;
- управління процесу, елементи, які необхідні під час реалізації, що впливають на кінцевий результат;
- механізми процесу.

Виконуючи побудову діаграми IDEF0 «Використання програмного модуля підтримки діяльності DevOps інженера», було визначено перелік даних:

- вхідні дані: SDC Issue ID, технологія пакування, директорія з файлами пакету;
- вихідні дані: згенерований пакет;
- управління процесу: користувач, принципи іменування SDC Issue ID, принципи роботи з файловою системою, принципи роботи з Atlassian.Jira та принципи роботи з MSI файлами;
- механізми процесу: System.IO, Atlassian.Jira, System.Xml, Microsoft.Deployment.WindowsInstaller.

Діаграма IDEF0 представлена на рис. 3.1.

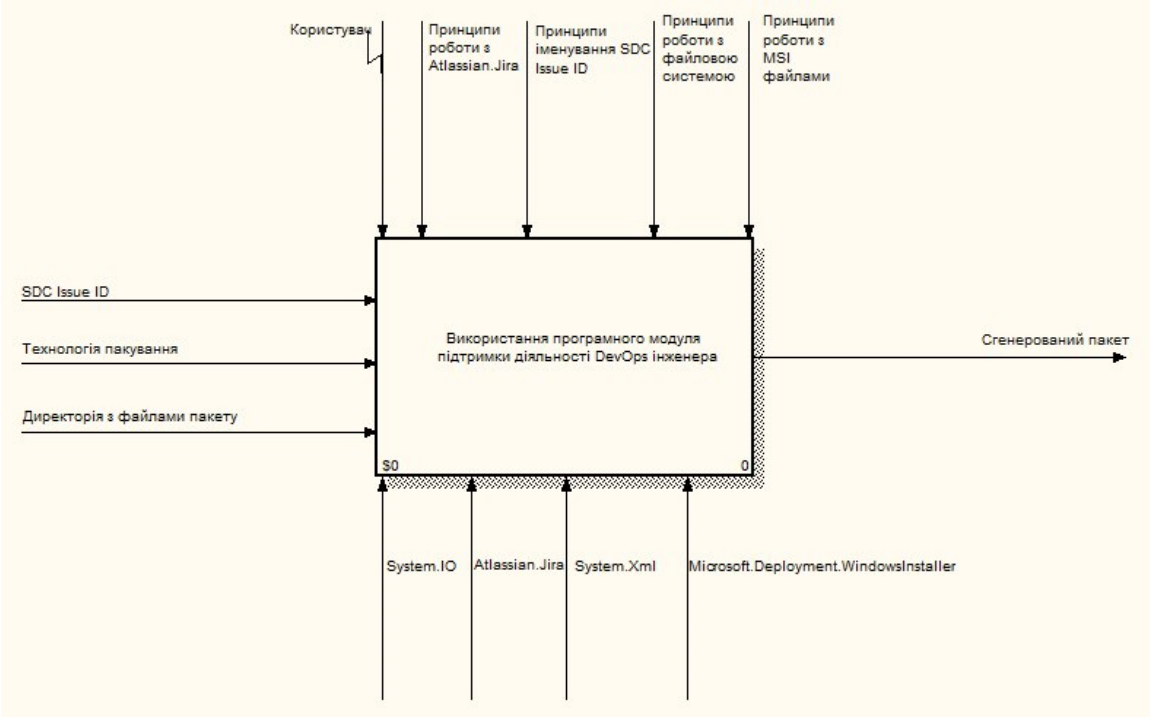


Рисунок 3.1 – Контекстна діаграма процесу використання програмного модуля у нотації IDEF0

Для більш детального ознайомлення з логікою роботи та використання продукту, необхідно виконати декомпозицію.

Діаграма була розбита на три процеси:

- введення SDC Issue ID;
- вибір директорії з файлами пакету і технології пакування;

- модифікація даних.

Для першого етапу було визначено наступний перелік даних:

- вхідні дані: SDC Issue ID;
- вихідні дані: дані конфігурації проекту;
- управління процесу: користувач, принципи іменування SDC Issue ID, принципи роботи з файловою системою, принципи роботи з Atlassian.Jira;
- механізми процесу: System.IO, Atlassian.Jira, System.Xml.

Для другого етапу було визначено наступний перелік даних:

- вхідні дані: дані конфігурації проекту, директорія з файлами пакету, технологія пакування;
- вихідні дані: масив даних;
- управління процесу: користувач, принципи роботи з файловою системою, принципи роботи з Atlassian.Jira та принципи роботи з MSI файлами;
- механізми процесу: System.IO, Atlassian.Jira, System.Xml, Microsoft.Deployment.WindowsInstaller.

Для третього етапу було визначено наступний перелік даних:

- вхідні дані: масив даних;
- вихідні дані: згенерований пакет;
- управління процесу: користувач, принципи роботи з файловою системою;
- механізми процесу: System.IO.

Діаграма декомпозиції процесу використання програмного модуля представлена на рис. 3.2.

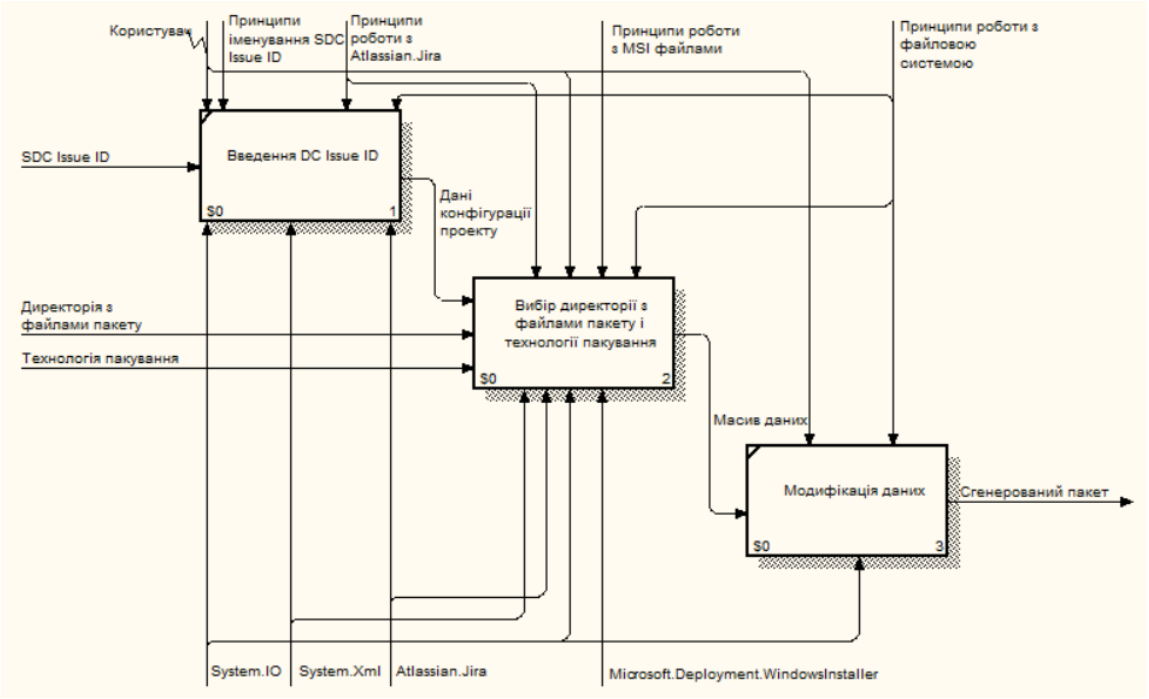


Рисунок 3.2 – Діаграма декомпозиції процесу використання програмного модуля у нотації IDEF0

Для кращого розуміння процесу вибору директорії з файлами пакету та технології пакування проведемо декомпозицію даного етапу:

- пошук MSI/MST файлів;
- вибірка даних з MSI/MST;
- генерація масиву даних.

Для процесу пошуку MSI/MST файлів визначено наступні дані:

- вхідні дані: дані конфігурації проекту, директорія з файлами пакету, технологія пакування;
- вихідні дані: MSI/MST файл;
- управління процесу: користувач, принципи роботи з файловою системою;
- механізми процесу: System.IO.

Після того як користувач затвердить необхідний MSI файл та MST файл якщо це необхідно, переходимо до процесу вибірки даних, для якого визначено наступні дані:

- вхідні дані: MSI/MST файл;
- вихідні дані: отримані дані;
- управління процесу: принципи роботи з MSI файлами;
- механізми процесу: System.Xml, Microsoft.Deployment.WindowsInstaller.

Отримавши необхідні дані необхідно згенерувати масив, який буде використовуватись в якості вхідних даних для етапу модифікації даних.

Для процесу генерації масиву даних визначено наступний перелік даних:

- вхідні дані: отримані дані;
- вихідні дані: масив даних;
- управління процесу: принципи роботи з Atlassian.Jira;
- механізми процесу: Atlassian.Jira.

Діаграма декомпозиції процесу вибору директорії з файлами пакету і технології пакування представлена на рис. 3.3.

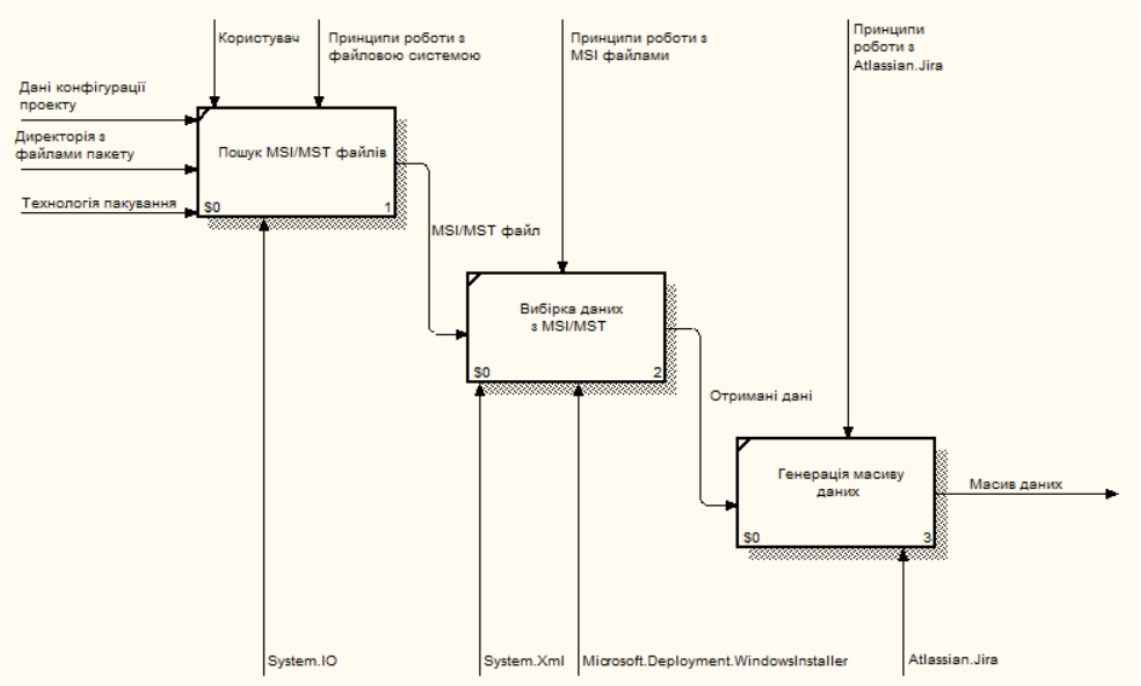


Рисунок 3.3 – Діаграма декомпозиції процесу вибору директорії з файлами пакету і технології пакування у нотації IDEF0

3.2 Моделювання варіантів використання

Необхідно розробити Use Case діаграму (діаграма варіантів використання) для відображення взаємозв’язків між акторами та варіантами використання. Під кожним варіантом використання мається на увазі набір дій, що виконує система при взаємодії з актором, при чому реалізація цих дій не описується.[17]

Отже, для побудови діаграми варіантів використання були визначені наступні актори:

- DevOps інженер – користувач програмного модуля;
- Файлова система – файлова система на комп’ютері користувача та мережевий диск, на якому зберігаються дані конфігурації програмного модуля.
- XML файли конфігурації – файли на мережевому диску з налаштуваннями проектів.
- Atlassian.Jira – система відстеження запитів на пакування ПЗ.
- Microsoft.Deployment.WindowsInstaller – бібліотека для роботи з файлами формату .msi.

Для визначених акторів можливі наступні варіанти використання:

- Введення SDC Issue ID;
- Вибір директорії з файлами пакету ПЗ;
- Вибір технології пакування;
- Модифікація даних;
- Налаштування та конфігурація операційної системи;
- Збереження готового пакету ПЗ.

Враховуючи сформовані дані про акторів та варіантів використання розроблено Use Case діаграму, рис.3.4.

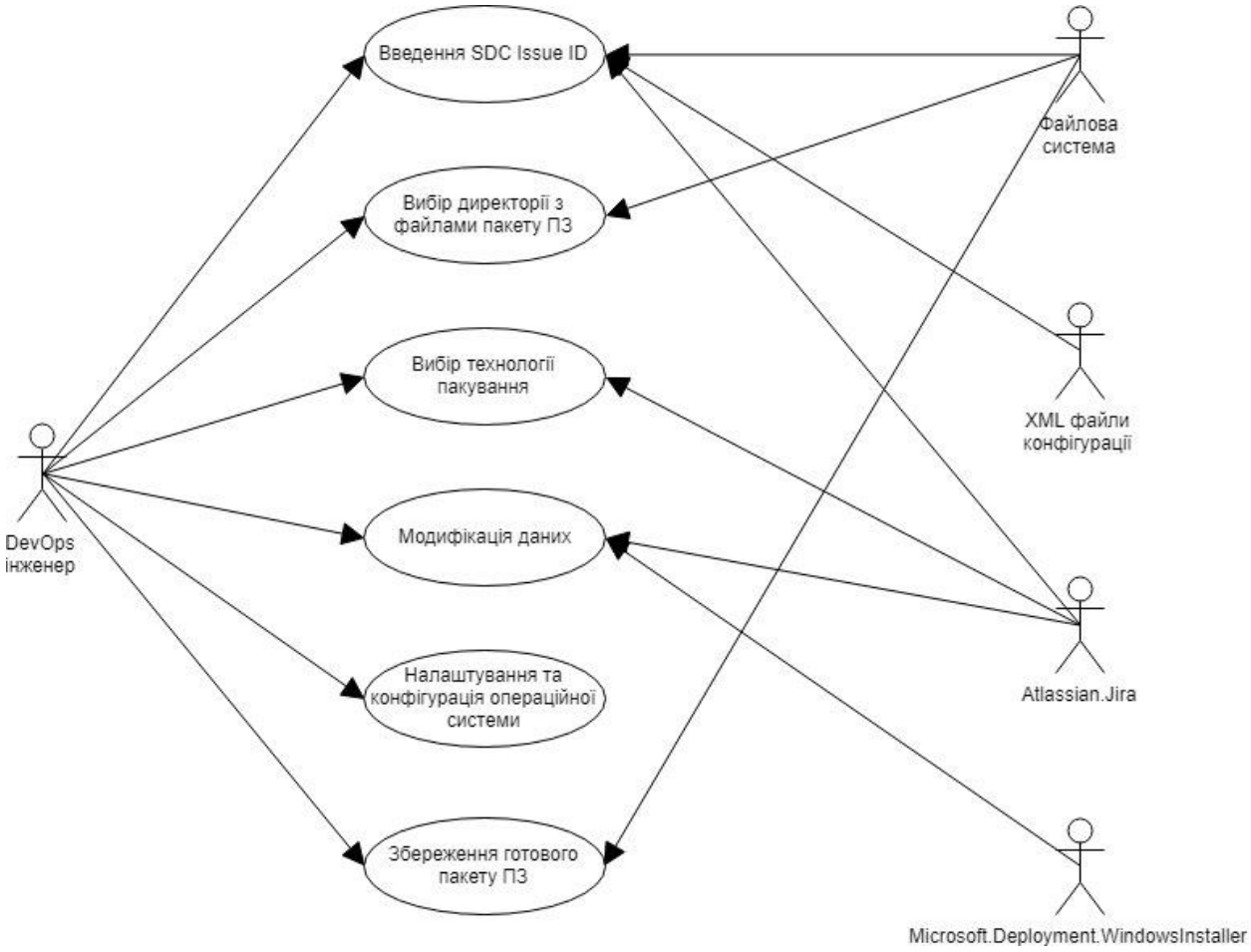


Рисунок 3.4 – Діаграма варіантів використання

3.3 Структура інформаційної моделі

Для зберігання налаштувань проектів використовується набір xml файлів, що зберігаються на мережевому диску, до якого є доступ як із хостових систем, так і з віртуальних машин на підприємстві. XML формат є зручнішим у створенні та подальшому використанні ніж використання реляційних баз даних для даної задачі, адже він дозволяє з легкістю додати чи модифікувати необхідну інформацію про існуючий чи новий проект за допомогою звичайного текстового редактора значно швидше, ніж використовуючи СУБД.

3.3.1 Структура xml файлів налаштувань

Для роботи програмного модуля необхідна наявність налаштувань. Як було описано вище, налаштування зберігаються в xml файлах, а саме використовується основний файл з налаштуваннями, який зберігає в собі теги <project>, в яких в свою чергу, міститься атрибут “name” з унікальним значенням відповідно для кожного проекту.

Всередині тегу <project> містяться теги <type>, <config> та <version>. Значення тегу <type> визначає який тип інструментарію для розгортання пакетів ПЗ притаманний для поточного проекту (можливі значення: cmd, ps1 або none). Значення тегу <config> визначає ім’я файлу з розширеними налаштуваннями даного проекту у форматі XML. Тег <version> слугує для контролю поточної версії налаштування програмного модуля.

Стосовно файлів конфігурації конкретного проекту, існує тег <Properties> всередині якого знаходяться такі теги:

- <dirname> : назва директорії з папками та файлами шаблону для даного проекту;
- <configuration> : основний тег, всередині якого зберігаються теги <config> з спеціальними атрибутами;
- <comment> : зберігає коментар, який необхідно вивести DevOps інженеру перед завершенням роботи з програмним модулем.

Необхідно розглянути тег <configuration> більш детально. Він може зберігати в собі необмежену кількість тегів <config>, проте кожен має бути з унікальним (в рамках даного файлу) значенням атрибуту “name” для його подальшої ідентифікації. Окрім обов’язкового атрибуту “name”, є опціональні атрибути “msiProperty”, “sdcProperty” та “defProperty”. Значення атрибутів “msiProperty” і “sdcProperty” зберігають в собі значення, необхідні для пошуку даної конфігурації з файлу .msi або з Atlassian.Jira відповідно. Щодо “defProperty” – задане значення буде використовуватися в програмному модулі.

Існує два типи тегів <config>: з текстом всередині та порожні. Якщо тег <config> містить в собі текст, то в графічному інтерфейсі програмного модуля будуть створені напис (з текстом що всередині тегу) та попередньо заповнене (з використанням описаних вище атрибутів) чи порожнє поле для введення, яке користувач зможе редагувати відповідно до власних потреб.

Якщо тег <config> є порожнім (проте він може містити вищевказані атрибути), то дані будуть використовуватися, проте не відображатися в користувацькому інтерфейсі, наприклад, є перелік службових тегів, які використовуються в коді програми, проте недоступні користувачеві.

Суть тегів в тому, що по атрибуту “name” проводитиметься пошук по попередньо створеному шаблонному cmd чи ps1 файлу та програмний модуль буде замінювати знайдені теги на відповідне необхідне значення.

Список значень атрибутів “name” для спеціалізованих службових тегів <config>:

- "WGReboot" – якщо даний тег присутній, користувачеві доступна пара RadioButton-ів для контролю над необхідністю перезапуску системи після встановлення пакету;
- "WG0" – використовується лише разом із "WGReboot", так як лінійно залежить від неї;
- "WGMSTransform" – необхідний для cmd проектів, якщо він присутній, програмний модуль буде використовувати його в залежності використовується чи ні MST файл в даному пакеті;
- "WGdirFiles" – ім’я директорії, в якій мають зберігатись файли пакету (є обов’язковим для всіх типів проектів);
- "WGInstall" – ім’я cmd файлу, що використовується для встановлення пакету (є обов’язковим для cmd проектів);
- "WGUninstall" – ім’я cmd файлу, що використовується для видалення пакету (є обов’язковим для cmd проектів);
- "WGInstallPS" – ім’я ps1 скрипт-файлу (є обов’язковим для ps1 проектів);

3.3.2 Структура шаблонів структури файлової системи

Окрім xml файлів налаштувань, використовуються шаблони структури файлової системи для кожного проекту.

Дана шаблонна директорія окрім структури файлової системи містить в собі спеціально модифіковані файли-шаблони (cmd чи ps1 в залежності від специфіки проекту), які в свою чергу містять всередині блоки такого ж змісту, як і значення атрибуту “name” тегів <config> у конфігураційному фалі відповідного проекту.

Використовуючи дані з тегу <dirname>, програмний модуль створює папку зі спеціальним ім’ям, що починається з префіксу WG (тобто WrapperGenerator) та містить випадкові символи в назві. Після створення тимчасової папки, до неї копіюється шаблонна директорія. Далі після обробки xml файлу конкретного проекту та модифікації даних користувачем, зміни заносяться до модифікованих файлів, після чого дана структура копіюється до вказаної користувачем цільової директорії разом із файлами готового пакету ПЗ.

4 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ DEVOPS ІНЖЕНЕРА

4.1 Розробка xml файлів конфігурації

При створенні xml файлів з налаштуваннями було використано структуру, що наведена в пункті «Структура інформаційної моделі» попереднього розділу.

Спершу створено основний файл конфігурації, який зберігає в собі список проектів, для яких працюватиме створений програмний модуль. Структуру та зміст файлу “projects.xml” можна побачити на рис. 4.1.

```
<?xml version="1.0" encoding="UTF-8"?>

<projectlist>

<project name="BIS">            <!-- Project name in SDC -->
    <type>ps1</type>            <!-- Wrapper type: "ps1" or "cmd" or "none"(package files only) -->
    <config>BIS.xml</config>    <!-- Project settings file -->
    <version>1.1.7</version>    <!-- Wrapper version -->
</project>

<project name="DEMAT">
    <type>ps1</type>
    <config>BIS.xml</config>
    <version>1.0.0</version>
</project>

<project name="GICT">
    <type>cmd</type>
    <config>GICT.xml</config>
    <version>1.0.0</version>
</project>

<project name="ISAG">
    <type>cmd</type>
    <config>ISAG.xml</config>
    <version>1.0.0</version>
</project>

</projectlist>
```

Рисунок 4.1 – Зміст файлу “projects.xml”

Наступним кроком було створення xml файлів конфігурації для кожного проекту. Їх зміст відрізняється в залежності від типу інструментарію

для розгортання пакетів та вимог конкретного проекту. Приклад файлу налаштувань для ps1 типу проектів наведено на рис. 4.2.

```
<?xml version="1.0" encoding="UTF-8"?>

<Properties>
    <dirname>BIS</dirname>   <!-- Name of directory with wrapper template -->
        <configuration>
            <config name="WGPkgName" msiProperty="APPT_PackageName" sdcProperty="Application Name">Package Name</config>    <!-- just for folder name-->
            <config name="WGVendor" msiProperty="Manufacturer" sdcProperty="Application Vendor">Manufacturer</config>
            <config name="WGName" msiProperty="ProductName" sdcProperty="Application Name">ProductName</config>
            <config name="WGVersion" msiProperty="ProductVersion" sdcProperty="Application Version">Version</config>
            <config name="WGArch" defProperty="X">Architecture (X86 | X64)</config>
            <config name="WGLang" defProperty="ENG">Language (DEU ENG MUI)</config>
            <config name="WGRevision" defProperty="01">Revision</config>
            <config name="WGAppOS" defProperty="Win10">appOS</config> <!-- check in code-->
            <config name="WGTaskToClose">Task to close</config>

            <config name="WGReboot"></config>   <!-- use code for radio button $true or $false-->
            <config name="WGInstallPS" defProperty="Deploy-Application.ps1"></config>   <!--mandatory for ps1-->
            <config name="WGSDCID"></config> <!--check in code-->
            <config name="WGDate" defProperty="MM/dd/yyyy"></config> <!-- check in code may be "MM/dd/yyyy" "dd/MM/yyyy" or smth else-->
            <config name="WGAuthor" defProperty="Apptimized"></config> <!--sdcProperty="Packager" may be used if needed-->
            <config name="WG0"></config>    <!-- check in code "0," if WGReboot is true, else set "" -->
            <config name="WGProductCode" msiProperty="ProductCode"></config>
            <config name="WGdirFiles" defProperty="Package\Files"></config>
            <config name="WGPSHelp" defProperty="Package\AppDeployToolkit\AppDeployToolkitHelp.ps1"></config>
            <config name="WGPSPreInstall"></config>
            <config name="WGPSInstall"></config>
            <config name="WGPSPostInstall"></config>
            <config name="WGPSPreUninstall"></config>
            <config name="WGPSUninstall"></config>
            <config name="WGPSPostUninstall"></config>

        </configuration>
    <comment>Don`t forget about:
    - complexity estimation in the Doc folder
    - icon in the File folder
    - main exe name and version in .ps1 file
    </comment>
</Properties>
```

Рисунок 4.2 – Зміст файлу “BIS.xml”

4.2 Розробка інтерфейсу користувача

Наступним кроком після створення файлів з налаштуваннями є створення користувацького інтерфейсу – програмного додатку WindowsForms мовою програмування C# з використанням середовища розробки Microsoft Visual Studio 2017.

За допомогою технології WindowsForms зручно створювати додатки з графічним інтерфейсом користувача. Використовуючи ресурси .NET Framework можна створити програми, які відображатимуть стандартні елементи керування на будь-якій цільовій платформі сімейства Windows.[18] Використовуючи конструктор WindowsForms розробник може створювати інтерфейс візуально, при чому необхідний програмний код автоматично зберігається у файлі коду даної форми.

На даному етапі розробки було створено інтерфейси чотирьох форм даного програмного модуля, що отримав назву «WrapperGenerator». Всі форми підтримують переміщення в межах робочої області операційної системи, незалежно від кількості моніторів.

Перша форма – «Welcome», яка доступна користувачеві одразу після запуску програмного модуля. Дана форма містить поле для введення ідентифікатора пакету та дві кнопки: виходу та переходу на наступну форму, а також GIF-анімацію для відображення процесу роботи. З форми були прибрані стандартні елементи керування вікном та рядок заголовку. Дизайн форми «Welcome» представлено на рис. 4.3.

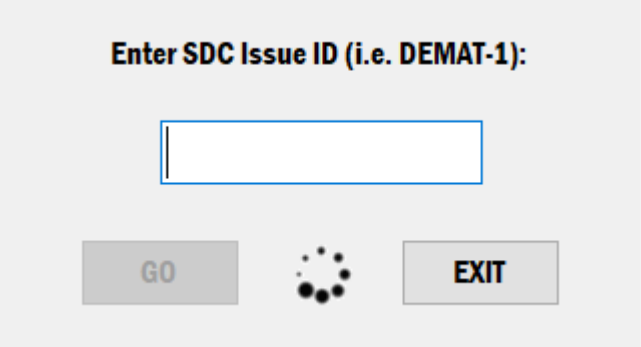


Рисунок 4.3 – Форма «Welcome»

Наступна форма – «ChoosePath». Серед елементів управління можна виділити поле для введення шляху до директорії з файлами пакету, кнопку для вибору шляху, якщо користувачу зручніше вказати розташування файлів використовуючи інтерфейс вибору директорії ОС Windows. Також на формі розміщено три об’єкти класу RadioButton для вибору технології пакування, три об’єкта класу ComboBox для вибору файлів, два ComboBox для зазначення конфігураційних файлів («MST» для технології MSI та «MSI» для Legacy), а також попередження при невірно вказаному шляху. GIF анімація присутня на даній формі, аналогічно до попередньої. Інтерфейси форми «ChoosePath» представлено на рис. 4.4-4.6.

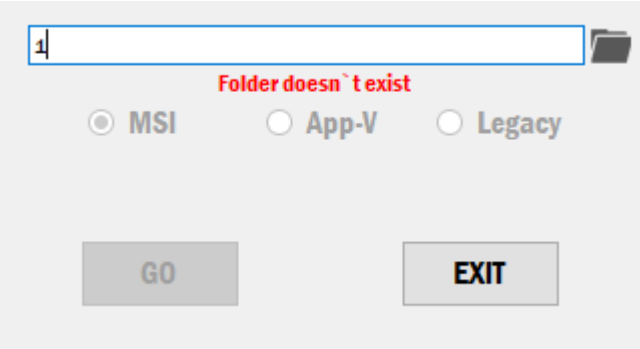


Рисунок 4.4 – Форма «Welcome». Невірно вказана директорія

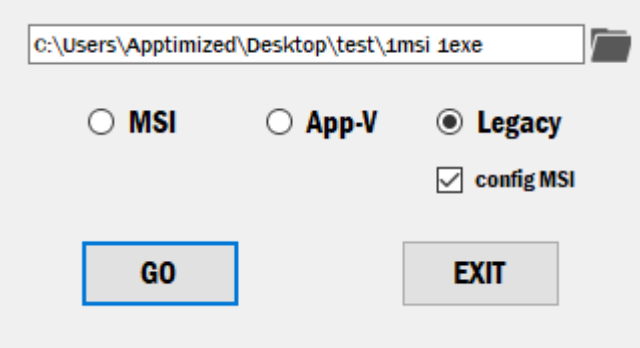


Рисунок 4.5 – Форма «Welcome». CheckBox “Config MSI”

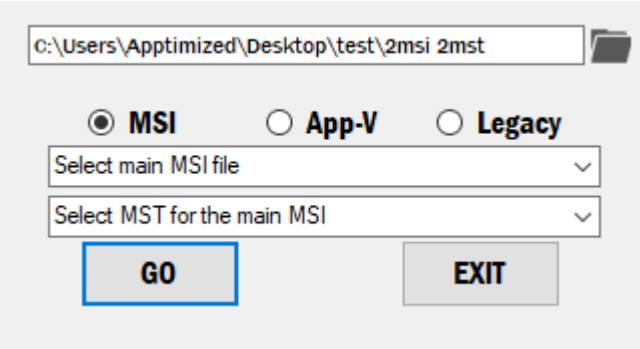


Рисунок 4.6 – Форма «Welcome». Списки для вибору файлів

На головній формі розміщено поле для введення цільової директорії для зберігання готового пакету та кнопка для вибору директорії як на попередній формі. Також створено дві групи перемикачів залежно від обраної технології та налаштувань проекту, основна панель для подальшої генерації полів для введення даних та кнопка модифікації .ps1 файлу. Повідомлення про помилку в шляху директорії та анімація як на попередній

формі. Елементи керування вікном було замінено на власноруч розроблені задля однакового вигляду незалежно від цільової операційної системи. Графічний інтерфейс форми «Main» представлено на рис. 4.7 - 4.8.

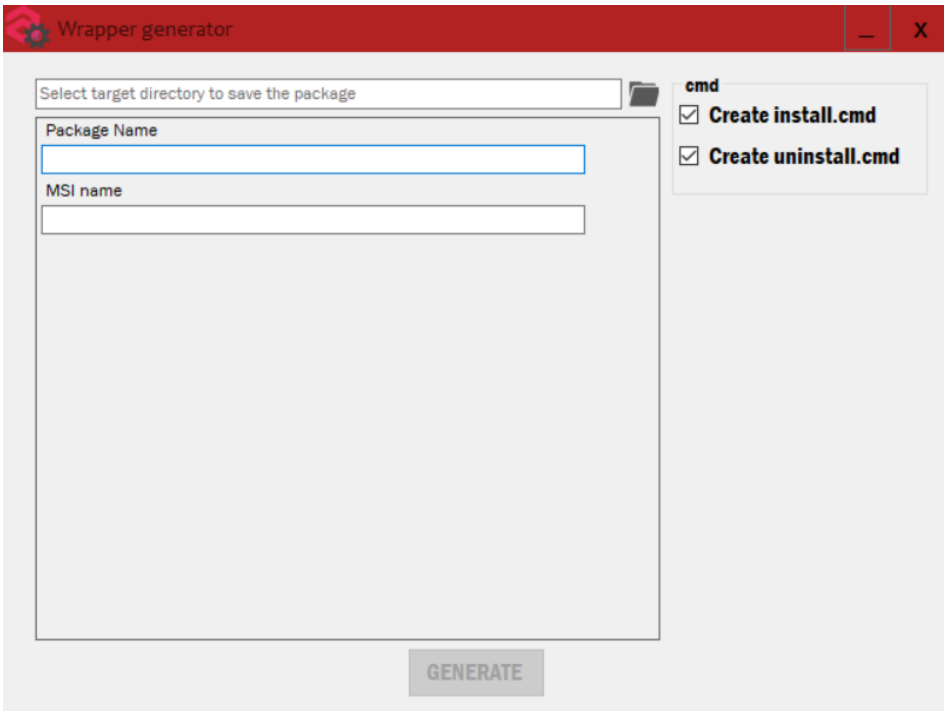


Рисунок 4.7 – Інтерфейс форми «Main» для cmd проекту

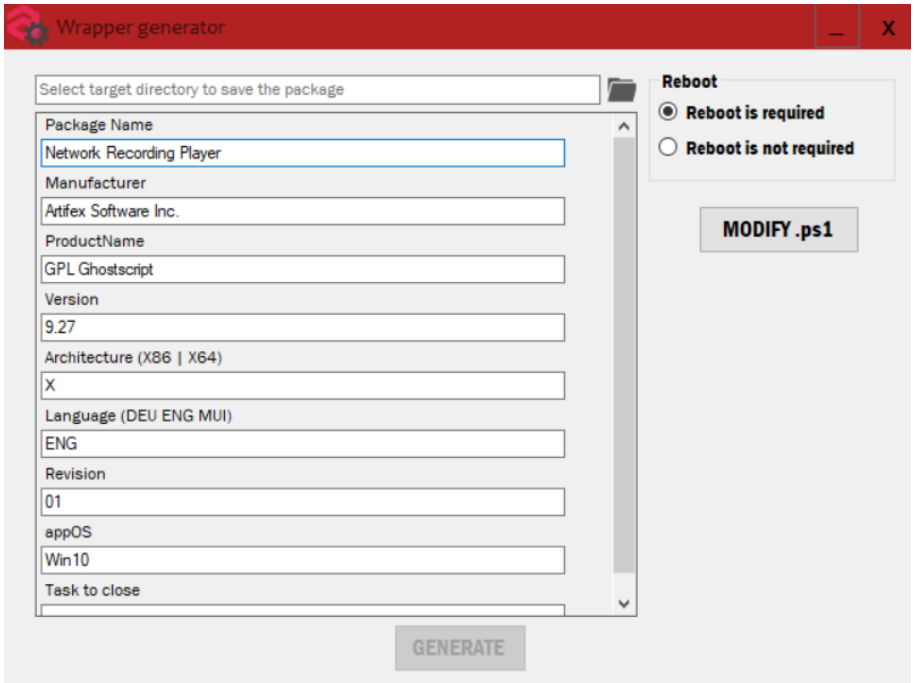


Рисунок 4.8 – Інтерфейс форми «Main» для ps1 проекту

Останньою був розроблений дизайн форми «Modifying», що використовується для модифікації файлу-скрипту PowerShell. На даній формі використано поля для введення тексту, списки ComboBox (для вибору етапу виконання скрипту та для вибору шаблонів), а також об’єкти класу Label з зображенням для кнопок видалення даних. Дизайн форми представлено на рис. 4.9.

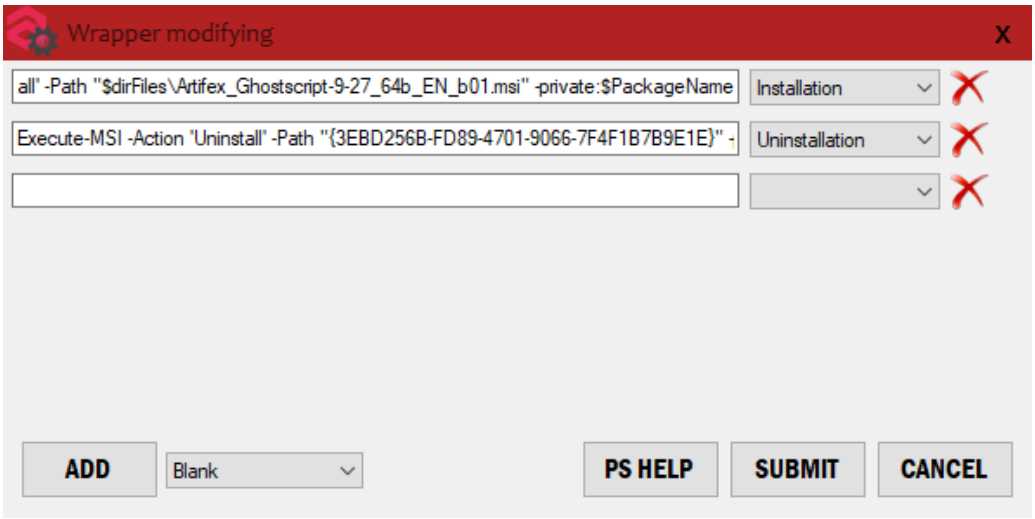


Рисунок 4.9 – Інтерфейс форми «Modifying» для ps1 проекту

4.3 Розробка логіки роботи програмного модуля

Після розробки інтерфейсів був реалізований функціонал кожної форми.

Основною функцією форми «Welcome» є ініціалізація ідентифікатора проекту та його обробка. Після введення користувачем id до текстового поля та після натискання кнопки “GO” здійснюється перевірка синтаксису, наявності даного проекту в xml файлі “projects.xml” та його валідність засобами Atlassian.Jira. Якщо ідентифікатор валідний, то наступній формі передається змінна, що зберігає в собі технологію пакування, яка в свою

чергу ініціалізується даними, отриманими з Atlassian.Jira. Програмний код перевірки валідності представлено на рис. 4.10.

```
//JIRA check
Jira jira = Jira.CreateRestClient("https://sdc.revacom.com/", "o.necheporuk", "...");
try
{
    var issues = from i in jira.Issues.Queryable
                 where i.Key == new LiteralMatch(id)
                 select i;
    issues.Count();
    string type = await GetIssueProperties(id);

    ChoosePath choose_form = new ChoosePath(id, type, screen);
    this.Hide();
    choose_form.ShowDialog();
}
catch
{
    MessageBox.Show(String.Format("Issue {0} not found", id));
}
```

Рисунок 4.10 – Перевірка валідності введеного користувачем ідентифікатора

Після перевірки введених даних ініціалізується змінна, що зберігає ім’я файлу налаштувань даного проекту. Здійснюється перехід на наступну форму.

Одразу після завантаження форми формою «ChoosePath» здійснюється зчитування даних з файлу налаштувань даного проекту, після чого, залежно від його вмісту та обраної користувачем директорії з файлами пакету автоматично виконується пошук у даній директорії файлів формату .msi, .mst, .exe та .appv для генерації CheckBox-ів та списків. Реалізація генерації масиву даних та заповнення списку для файлів формату .exe наведено на рис. 4.11.

```
foreach (string i in Directory.GetFiles(PATH_textbox.Text, "*.exe", SearchOption.AllDirectories))
{
    exes.Add(i);
    EXElist.Items.Add(Path.GetFileName(i));
    EXECount++;
}
```

Рисунок 4.11 – Пошук .exe файлів у вказаній директорії

Також на даному етапі роботи програмного модуля створюється директорія зі згенерованим псевдо-випадковим ім’ям у тимчасовій папці

користувача зі структурою інструментарію розгортання пакетів, що попередньо зберігається на мережевому диску та містить модифікований шаблонний файл-скрипт.

Після натискання кнопки “GO” заповнюється двовимірний динамічний масив з необхідними налаштуваннями, попередньо отриманими з xml файлу. Дані зберігаються попарно, ім'я тегу конфігурації – його значення. Залежно від обраної технології та наявних файлів здійснюється пошук необхідних даних у наступній послідовності: спочатку з таблиці Property .msi файлу, потім з Atlassian.Jira, далі – стандартне значення. Необхідні для пошуку атрибути зберігаються в xml файлі проекту під назвами “msiProperty ”, “sdcProperty ” та “defProperty” відповідно. Реалізація пошуку даних в таблиці Property .msi файлу продемонстровано на рис. 4.12.

```
public static string GetMsiProperty(string msiFile, string property, string mstFile)
{
    using (var database = new Database(msiFile, DatabaseOpenMode.ReadOnly))
    {
        if (mstFile != "none")
        {
            database.ApplyTransform(mstFile);
        }

        using (var view = database.OpenView(database.Tables["Property"].SqlSelectString))
        {
            view.Execute();

            foreach (var rec in view) using (rec)
                {
                    if (rec.GetString("Property")==property)
                        return rec.GetString("Value");
                }
        }
    }
    return "";
}
```

Рисунок 4.12 – Пошук даних в .msi файлі

Використовуючи дані з масиву, на формі «Main» генеруються необхідні поля для введення, що вже містять значення, отримані до цього. Здійснюється перехід до наступної, головної форми.

На формі «Main» користувач має змогу модифікувати автоматично заповнені дані, що будуть використані у файлі-скрипті. Якщо даний проект використовує технологію .cmd файлів для встановлення готових пакетів на

кінцевих робочих станціях, то користувач має змогу відключити генерацію файлів встановлення та/або видалення. Цей функціонал буде корисним у тому випадку, коли DevOps інженер створив дані файли та провів необхідну конфігурацію на етапі створення пакету додатку. В цьому випадку, файли не будуть перезаписані згенерованими файлами.

У випадку використання PSADT для розгортання пакетів є можливість модифікувати файл-скрипт шляхом додавання спеціальних команд для налаштування ОС кінцевого користувача. Для цього слугує кнопка “MODIFY .ps1”, яка відкриває форму «Modifying».

Під час виклику форми «Modifying» автоматично генеруються скриптові дії для встановлення та видалення обраних файлів .msi, .exe або .appv. На даній формі у користувача є можливість додати пустий або згенерований з шаблону рядок мовою PowerShell, його редагування та видалення. Для кожного рядка має бути обраний відповідний етап встановлення чи видалення пакету з випадаючого списку (Pre-Installation, Installation, Post-Installation, Pre-Uninstallation, Uninstallation чи Post-Uninstallation).

Також на даній формі є можливість викликати довідку, що містить функції з прикладами команд PSADT для поточного проекту. На рис. 4.13 наведено реалізацію виклику форми «Modifying», на рис. 4.14 - запуск вікна допомоги.

```
private void Btn_modifyPs_Click(object sender, EventArgs e)
{
    if (msi != "none")
    {
        int conf = list.Count;
        for (int i = 0; i < conf; i++)
        {
            if (list[i][0] == "WGProductCode")
            {
                productCode = list[i][1];
                break;
            }
        }
    }
    string strCmdText;
    strCmdText = "-noprofile -ExecutionPolicy ByPass -File \"" + tmppath + "\\" + pshelp + "\"";
    Modifying modify_form = new Modifying(strCmdText, msi, exe, mst, appv, productCode, FirstModify);
    modify_form.ShowDialog();
    FirstModify = false;
}
```

Рисунок 4.13 – Створення об’єкту форму «Modifying»

```
private void Btn_pshelp_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("C:\\windows\\system32\\windowspowershell\\v1.0\\powershell.exe ", strHelpCmd);
}
```

Рисунок 4.14 – Запуск вікна допомоги з функціями PSADT

Після натискання кнопки “SUBMIT” форма «Modifying» закривається, а дані зберігаються у відповідних рядкових змінних, які будуть використані для модифікації скрипт-файлу PSADT.

Після всіх змін та модифікацій на формі «Main» по натисканню на кнопку «GENERATE» відбувається автоматичне редагування файлу-скрипту в залежності від введених даних. Після того, як файл-скрипт готовий, виконується копіювання структури файлової системи з тимчасової папки користувача та файлів пакету, вказаних на формі «ChoosePath» до цільової директорії. Після того, як копіювання завершено, запускається процес провідника Windows, який автоматично відкриває папку з готовим пакетом та згенерованим інструментарієм розгортання пакетів. По завершенню роботи користувач має вийти з програмного модуля шляхом закриття вікна, під час чого видаляється раніше створена тимчасова папка та завершується робота програмного модуля.

Повний програмний код наведено у додатку Г.

ВИСНОВКИ

Під час виконання дипломного проекту був розроблений програмний модуль підтримки діяльності DevOps інженера під назвою “WrapperGenerator”. Продукт було розроблено для використання працівниками відділу пакування та розробки програмного забезпечення компанії Apptimized Operations.

У процесі виконання проекту було обґрунтовано актуальність роботи, проаналізовано існуючі аналоги та їх переваги і недоліки, визначено функціональні вимоги до програмного модуля та визначено засоби реалізації.

Було виконано планування робіт, за результатами якого розроблено календарний план, а також визначено перелік ризиків під час виконання проекту.

Розроблений програмний модуль відповідає функціональним вимогам, а саме, реалізовано функціонал:

- визначення структури інструментарію розгортання залежно від проекту;
- автоматичний збір даних та їх подальша модифікація користувачем;
- передбачена можливість додавати шаблонні дії для взаємодії з ОС з можливістю їх редагування користувачем;
- зберігання готового пакету ПЗ за вказаним шляхом локально чи на мережевому диску.

Програмний модуль було розроблено засобами об'єктно-орієнтованої мови C# з використанням WindowsForms у середовищі розробки Microsoft Visual Studio 2017. Під час розробки було враховано виключні ситуації та відповідно їх оброблено.

Для роботи програмного модуля було створено шаблони структури інструментарію розгортання пакетів програмного забезпечення та файли з конфігураціями проектів за стандартом XML.

Після завершення розробки було проведено тестування продукту в компанії Apptimized Operations та за його результатами отримано акт впровадження від відділу пакування та розробки програмного забезпечення, який наведено у додатку А.

Отже, виконавши всі етапи дипломного проекту, було розроблено програмний модуль підтримки діяльності DevOps інженера, що дозволить покращити показники роботи та автоматизувати процес створення інструментарію для розгортання пакетів програмного забезпечення.

Використання розробленого програмного забезпечення дозволить зекономити час на виконаня робіт по стовренню інсталяційних пакетів.

СПИСОК ЛІТЕРАТУРИ

1 11 важных вещей, которые нужно знать про DevOps — часть первая [Електронний ресурс] – режим доступу: https://habr.com/ru/company/scrumtrek/blog/166039/

2 Нечепорук О.А. Програмний модуль підтримки діяльності DevOps інженера. / Нечепорук О.А., Ващенко С.М. // Інформатика, математика, автоматика: матеріали та програма науково-технічної конференції, м. Суми, 2019 р. – Суми : СумДУ, 2019. – 82 с.

3 PSAppDeployToolkit [Електронний ресурс] – режим доступу: https://psappdeploytoolkit.com/

4 David M. Stein. Software Repackaging and Deployment for the Windows Platform / David M. Stein., 2012. – 146 с. – (Amazon Kindle).

5 Windows Installer [Електронний ресурс] – режим доступу: https://en.wikipedia.org/wiki/Windows_Installer

6 What is App-V and how does it work? [Електронний ресурс] – режим доступу: http://www.tmurgent.com/TmBlog/?p=2489

7 Why I love the PowerShell AppDeployment Toolkit [Електронний ресурс] – режим доступу: https://www.sinisasokolic.com/why-i-love-the-powershell-appdeployment-toolkit/

8 Введение в NuGet [Електронний ресурс] – режим доступу: https://docs.microsoft.com/ru-ru/nuget/what-is-nuget

9 Сборка мусора, управление памятью и указатели [Електронний ресурс] – режим доступу: https://metanit.com/sharp/tutorial/8.1.php

10 Costura is an add-in for Fody [Електронний ресурс] – режим доступу: https://github.com/Fody/Costura

11 XML технології та засоби розробки Gupta Team Developer: XML I CTD [Електронний ресурс] – режим доступу: http://easy-

code.com.ua/2012/07/xml-texnologi%D1%97-ta-zasobi-rozrobki-gupta-team-developer-xml-i-ctd-chastina-1-rizne-programuvannya-statti/

12 Технологія формулювання цілей SMART! [Електронний ресурс] – режим доступу: https://biznesua.com.ua/tehnologiya-formulyuvannya-tsiley-smart/

13 Что такое WBS проекта, и зачем она нужна [Електронний ресурс] – режим доступу: http://upravlenie-proektami.ru/chto-takoe-wbs-proekta-i-zachem-ona-nuzhna

14 Product Data Management [Електронний ресурс] – режим доступу: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:PDM_(Product_Data_Management)_-_%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8_%D0%BE%D0%B1_%D0%B8%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D0%B8

15 Системне програмування [Електронний ресурс] – режим доступу: http://khpi-iip.mipk.kharkiv.edu/library/sp/sp_book/sp01.html

16 IDEF0: функціональне моделювання ділових процесів, CASE-засоби (моделювання) [Електронний ресурс] – режим доступу: http://easy-code.com.ua/2012/08/idef0-funkcionalne-modelyuvannya-dilovix-procesiv-case-zasobi-modelyuvannya-programuvannya-statti/

17 Підходи до аналізу і проектування інформаційних систем [Електронний ресурс] – режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151203140326/204841/index.html

18 Windows Forms overview [Електронний ресурс] – режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview

ДОДАТОК А

Акт впровадження

АКТ
впровадження результатів студентської наукової роботи
«ПРОГРАМНИЙ МОДУЛЬ ПІДТРИМКИ ДІЯЛЬНОСТІ DEVOPS ІНЖЕНЕРА»
у діяльність відділу пакування та розробки програмного забезпечення компанії Apptimized у 2019 р.

Комісія фахівців компанії Apptimized в м. Суми у складі: директора компанії Смірнова В. В. та першого заступника Домбровського В.С. склала цей акт в тому, що результати студентської дипломної роботи Нечепорука О.А. на тему «Програмний модуль підтримки діяльності DevOps інженера» впроваджено у діяльність відділу пакування та розробки програмного забезпечення компанії Apptimized в м. Суми.

Було впроваджено використання програмного модуля «WrapperGenerator» для забезпечення ефективної реалізації проектів пакування програмного забезпечення, який дозволяє скоротити часові витрати процесу розроблення та налаштування інструментарію для розгортання пакетів програмного забезпечення.

Впровадження результатів студентської наукової роботи Нечепорука О.А. дозволяє робітникам компанії Apptimized покращити виробничий процес шляхом зменшення часових витрат на виконання проекту або портфелів проектів перепакування програмного забезпечення.

Голова комісії: Смірнов В. В.

Комісія: Домбровський В.С.

Рисунок А.1 – Акт впровадження

ДОДАТОК Б

Технічне завдання

Назва програмного модуля: «WrapperGenerator».

Область застосування програмного модуля: програмний модуль призначений для створення та налаштування інструменту розгортання пакетів ПЗ створених DevOps інженерами компанії Apptimized Operations.

Об'єкт, у якому використовують програму: Apptimized Operations.

1 Основи для розробки

Розробка виконується на основі завдання, виданого викладачами секції Інформаційних технологій проектування.

2 Призначення розробки

Розробка повинна надати можливість скорочення витрат часу за рахунок автоматизації процесу створення структури PSADT.

Тема проекту: «Розробка програмного модуля підтримки діяльності DevOps інженера»

3 Вимоги до програмного модуля

Програмний модуль має бути реалізований як Windows додаток, мати інтуїтивно зрозумілий інтерфейс та бути універсальним для якнайбільшої кількості проектів.

3.1. Вимоги до програмного модуля

Програмний модуль розробляється мовою C# з використанням технології ООП. Інтерфейс має бути реалізований за допомогою Windows Form. Готовий файл програми має містити в собі всі необхідні додаткові компоненти та бути портативним, без необхідності встановлення.

3.2. Вимоги до програмного забезпечення

Для коректної роботи з програмним модулем необхідне підключення до Інтернету та до мережевого диску з файлами налаштування проектів.

3.3.Вимоги до функціональних характеристик

Програмний модуль має забезпечувати виконання наступних функцій:

- визначати структуру PSADT залежно від проекту;
- автоматично заповнювати поля з розділу оголошення змінних з можливістю їх перевірки та модифікації користувачем;
- передбачити можливість додавати шаблонні дії для взаємодії з ОС з можливістю їх редагування користувачем;
- зберігати створений PSADT за вказаним шляхом.

4 Перелік програмної документації

- Опис проекту продукту;
- Макет інтерфейсу програмного модуля;
- Програмний код розробки;
- Технічне завдання;
- Інструкція користувача;

5 Порядок виконання робіт і етапи розробки

- Оформлення завдання для дипломного проекту;
- Планування роботи. Розроблення ТЗ ,побудова мережевого графіку та діаграми Ганта;
- Розроблення програмного модуля;
- Розробка шаблонів та налаштувань для проектів;
- Тестування програмного модуля;
- Розроблення інструкції користувача;
- Оформлення пояснювальної записки про виконання дипломного проекту;
- Здача пояснювальної записки до дипломного проекту та розробленого програмного модуля;
- Презентація роботи та її захист.

6 Порядок контролю та приймання

Контроль коректності функціонування та придатності програмного модуля здійснюється тестувальником на основі наданої інструкції користувача, а

також викладачем на основі пояснювальної записки до дипломного проекту та програмних файлів. Контроль ходу виконання проекту здійснюється на основі календарного плану виконання дипломного проекту:

- Перевірка завдання дипломного проекту;
- Перевірка розробленого ТЗ, мережевого графіка та діаграми Ганта.
- Перевірка програмного коду розробленого модуля;
- Перевірка шаблонів проектів;
- Перевірка коректності тестування;
- Перевірка інструкції користувача;
- Здача пояснювальної записки;
- Презентація дипломного проекту.

ДОДАТОК В

Планування робіт

В.1 Деталізація мети методом SMART

Мета проекту: розробити програмний модуль для підтримки діяльності DevOps інженера у створенні PSADT. Застосовуючи його, користувач з легкістю зможе обрати проект, для якого автоматично згенерується структура та власне сценарій PowerShell. Інженер матиме можливість змінити останній за допомогою елементів Windows форми. Також за бажанням користувач зможе додати бажану кількість дій із шаблоніві з можливістю редагування, необхідних для налаштування ПЗ в одну із секцій PSADT сценарію: pre-installation, installation, post-installation, pre-uninstallation, uninstallation та post-uninstallation. Результати деталізації методом SMART розміщенні у табл. В.1.1.

Таблиця В.1.1 – Деталізація мети методом SMART

Specific (конкретна)	Розробити програмний модуль для підтримки діяльності DevOps інженера.
Measurable (вимірювання)	Оскільки даний проект не є комерційним, то результатом його роботи є оцінка замовника.
Achievable (досяжна, узгоджена)	Ціль даного проекту вважається досяжною, оскільки розробник володіє необхідними навичками у створенні Windows-додатків засобами мови C# та має необхідні знання у сфері технології PowerShell та мови розмітки XML. Мета була узгоджена з вимогами та потребами замовника.

Продовження таблиці В.1.1

Relevant (реалістична)	Для реалізації продукту проекту є всі необхідні технічні та програмні засоби (середовище розробки Microsoft Visual Studio, універсальний інтерпретатор Notepad++, оболонка PowerShell), доступ до мережі Інтернет та до FTP-серверу. Розробник досить кваліфікований для виконання поставлених задач.
Time-framed (обмежена в часі)	Програмний модуль розроблюється з обмеженням у часі на основі сформованого календарного плану та матриці відповідальності.

В.2 Планування змісту робіт

WBS (Work Break Structure) – це графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. На верхньому першому рівні WBS фіксується продукт проекту. Він повинен відповідати продукту проекту. Наступний другий рівень відповідає діям або основним заходам для досягнення продукту проекту. Виконуємо декомпозицію робіт для проекту. Діаграма WBS представлена на рис. В.2.1.

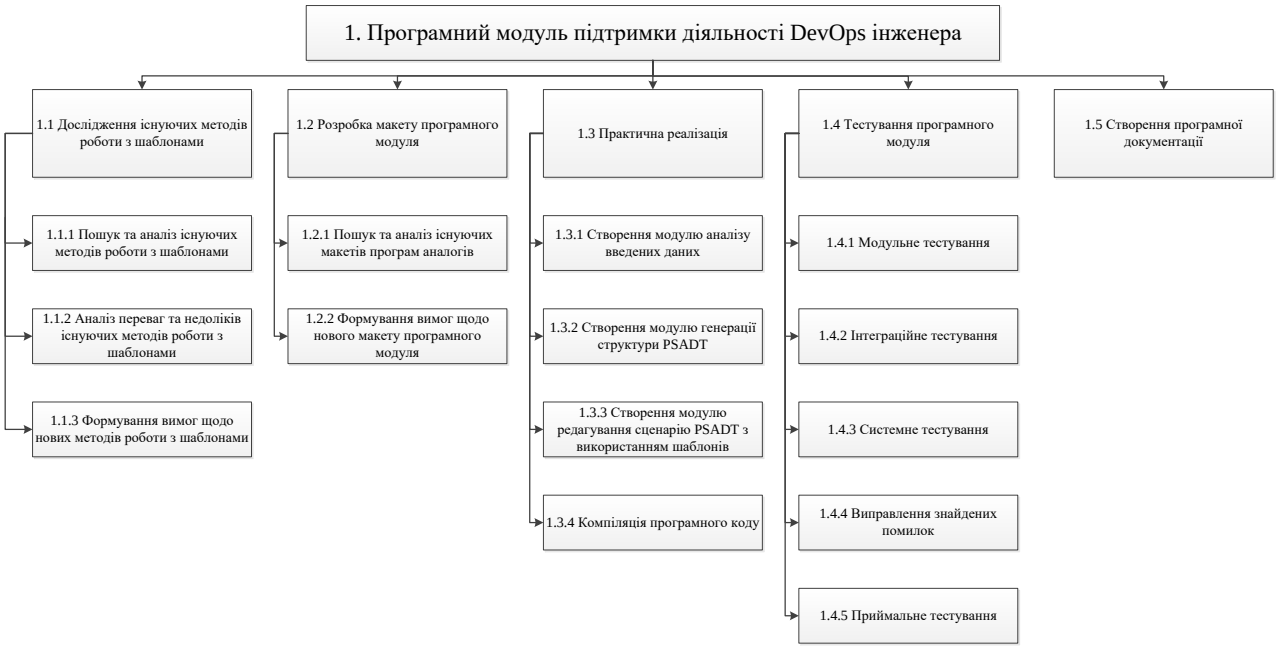


Рисунок В.2.1 – WBS структура проекту

В.3 Планування структури виконавців

OBS-структура проекту – організаційна структура виконавців (організацій) проекту. Визначається за переліком пакетів робіт нижнього рівня кожної гілки WBS-структури. Представляється відповідальними особами. Організаційна структура представляє собою графічне відображення учасників проекту та їх відповідальних осіб, які задіяні в реалізації проекту. На верхньому рівні OBS розташована команда проекту. На наступному рівні фіксуються виконавці: організації, відділи тощо. Потім, рівнем нижче, для кожного виконавця вказують прізвища конкретних осіб, які будуть відповідати за виконання елементарних робіт WBS. Організаційна структура проекту зображена на рис. В.3.1.

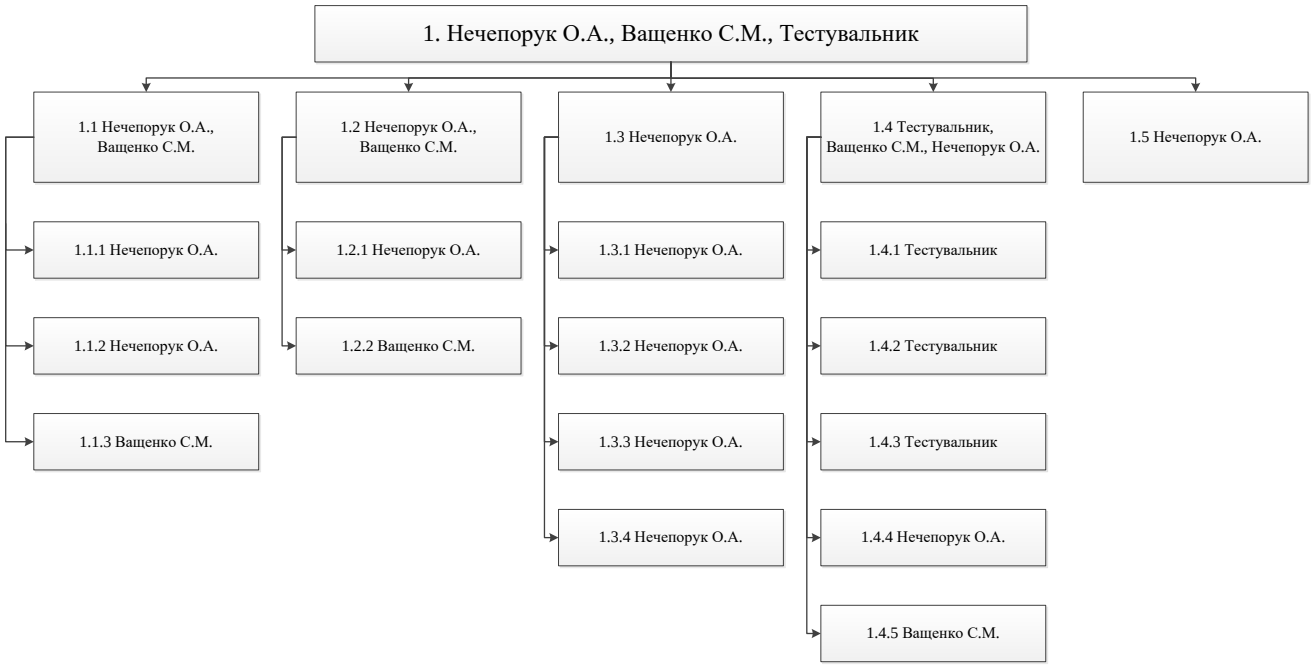


Рисунок В.3.1 – OBS структура проекту

В.4 Побудова матриці відповідальності

Матриця відповідальності надає інформацію про особу, яка є відповідальною за виконання кожної тої чи іншої елементарної роботи. На основі розроблених WBS та OBS структур проекту була побудована матриця відповідальності проекту, яка представлена в табл. В.4.1.

Таблиця В.4.1 – Матриця відповідальності

WBS\OBS	Нечепорук О.А.	Ващенко С.М.	Тестува-льник
1. Створення програмного модуля підтримки діяльності DevOps інженера			
1.1 Дослідження існуючих методів роботи з шаблонами			
1.1.1 Пошук та аналіз існуючих методів роботи з шаблонами			

Продовження таблиці В.4.1.

1.1.2 Аналіз переваг та недоліків існуючих методів роботи з шаблонами			
1.1.3 Формування вимог щодо нових методів роботи з шаблонами			
1.2 Розробка макету програмного модуля			
1.2.1 Пошук та аналіз існуючих макетів програм аналогів			
1.2.2 Формування вимог щодо нового макету програмного модуля			
1.3 Практична реалізація			
1.3.1 Створення модулю аналізу введених даних			
1.3.2 Створення модулю генерації структури PSADT			
1.3.3 Створення модулю редагування сценарію з використанням шаблонів			
1.3.4 Компіляція програмного коду			
1.4 Тестування програмного модуля			
1.4.1 Модульне тестування			
1.4.2 Інтеграційне тестування			
1.4.3 Системне тестування			
1.4.4 Виправлення знайдених помилок			
1.4.5 Приймальне тестування			
1.5 Створення програмної документації			

В.5 Розробка PDM мережі

Для кожного пакета робіт розробляють часткові мережеві моделі, в яких встановлюють логічні взаємозв'язки між всіма роботами, які необхідно виконувати для отримання запланованого продукту з пакету робіт. Мережеві моделі дозволяють визначити тривалість виконання пакету робіт. Вони встановлюють логічний взаємозв'язок між всіма роботами, які необхідно виконати для отримання конкретного продукту (кожного WBS елемента або проекту в цілому). Сьогодні найбільше використовують так звані PDM-мережі або мережі типу «вершина-робота» (рис. В.5.1-В.5.6).

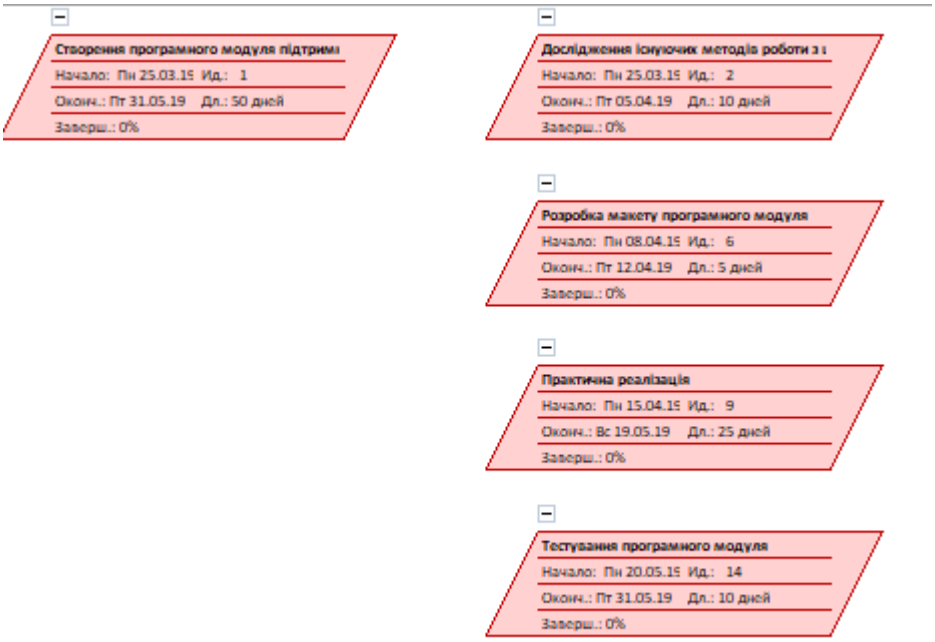


Рисунок В.5.1 - PDM-мережа частина 1

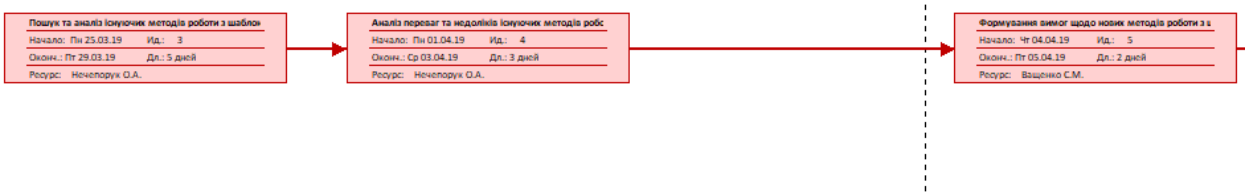


Рисунок В.5.2 - PDM-мережа частина 2

Рисунок В.5.3 - PDM-мережа частина 3

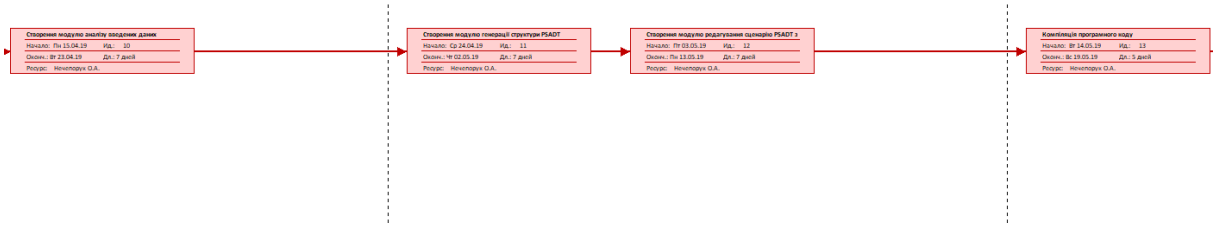


Рисунок В.5.4 - PDM-мережа частина 4

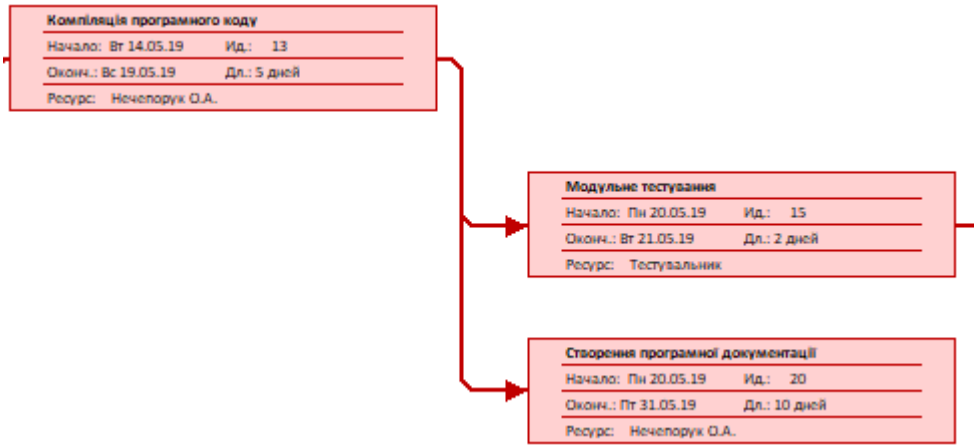


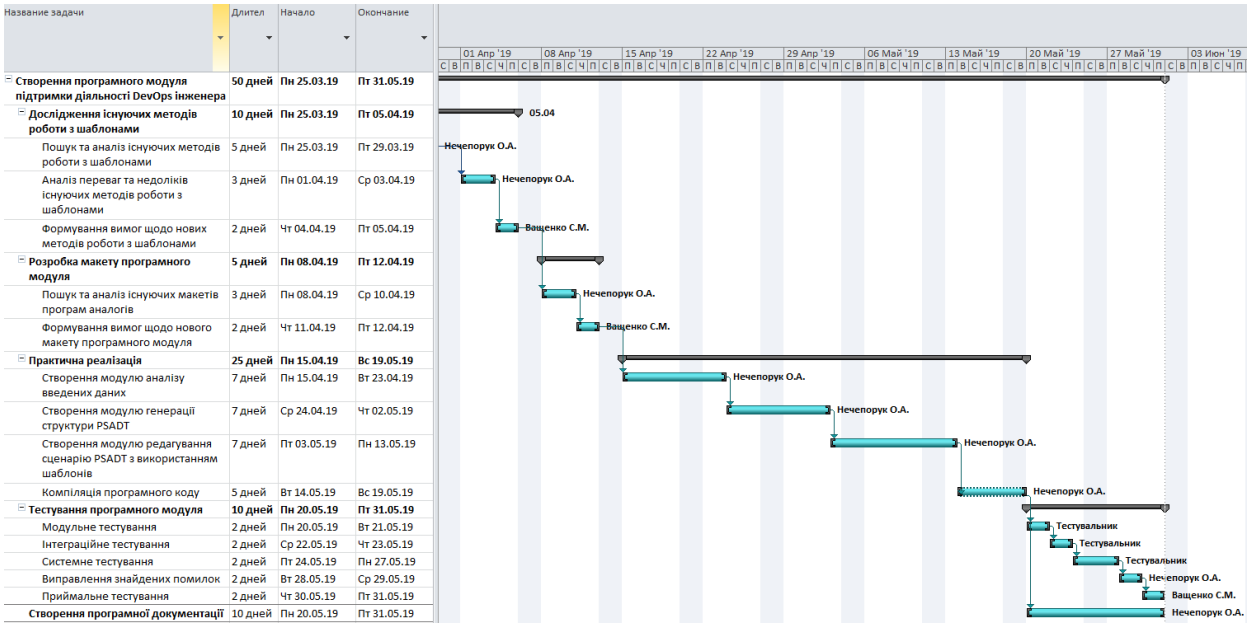
Рисунок В.5.5 - PDM-мережа частина 5

Рисунок В.5.6 - PDM-мережа частина 6

В.6 Побудова календарного графіку

Так само як PDM-мережі діаграма Ганта вважається якісним інструментом для відображення цілей та задач. Основна відмінність цих інструментів – у способі демонстрації і відображення інформації.

На відміну від PDM-мережі, що пропонує мережеву модель, управління проектами з діаграмами Ганта засноване на форматі гістограм. Це допомагає відслідковувати відсоток робіт, виконаних по кожному завданню. Керівникам проектів дуже важливо правильно розподілити завдання і бути впевненими в тому, що проект буде завершений вчасно. Основна увага діаграм Ганта зосереджено на процентному завершенні кожного завдання. Крім того, діаграми Ганта краще для проектів з невеликою кількістю взаємопов'язаних завдань. За допомогою MS Project була розроблена діаграма Ганта, яка у вигляді гістограми відображає тривалість кожного процесу, що був визначений на етапі формування WBS. Діаграма Ганта представлена на рис. В.6.1.



Название задачи	Длител	Начало	Окончание
Створення програмного модуля підтримки діяльності DevOps інженера	**50 дней**	**Пн 25.03.19**	**Пт 31.05.19**
Дослідження існуючих методів роботи з шаблонами	**10 дней**	**Пн 25.03.19**	**Пт 05.04.19**
Пошук та аналіз існуючих методів роботи з шаблонами	5 дней	Пн 25.03.19	Пт 29.03.19
Аналіз переваг та недоліків існуючих методів роботи з шаблонами	3 дней	Пн 01.04.19	Ср 03.04.19
Формування вимог щодо нових методів роботи з шаблонами	2 дней	Чт 04.04.19	Пт 05.04.19
Розробка макету програмного модуля	**5 дней**	**Пн 08.04.19**	**Пт 12.04.19**
Пошук та аналіз існуючих макетів програм аналогів	3 дней	Пн 08.04.19	Ср 10.04.19
Формування вимог щодо нового макету програмного модуля	2 дней	Чт 11.04.19	Пт 12.04.19
Практична реалізація	**25 дней**	**Пн 15.04.19**	**Вс 19.05.19**
Створення модулю аналізу введених даних	7 дней	Пн 15.04.19	Вт 23.04.19
Створення модулю генерації структури PSADT	7 дней	Ср 24.04.19	Чт 02.05.19
Створення модулю редагування сценарію PSADT з використанням шаблонів	7 дней	Пт 03.05.19	Пн 13.05.19
Компіляція програмного коду	5 дней	Вт 14.05.19	Вс 19.05.19
Тестування програмного модуля	**10 дней**	**Пн 20.05.19**	**Пт 31.05.19**
Модульне тестування	2 дней	Пн 20.05.19	Вт 21.05.19
Інтеграційне тестування	2 дней	Ср 22.05.19	Чт 23.05.19
Системне тестування	2 дней	Пт 24.05.19	Пн 27.05.19
Виправлення знайдених помилок	2 дней	Вт 28.05.19	Ср 29.05.19
Приймальне тестування	2 дней	Чт 30.05.19	Пт 31.05.19
Створення програмної документації	10 дней	Пн 20.05.19	Пт 31.05.19

Рисунок В.6.1 – Діаграма Ганта

В.7 Управління ризиками проекту

При плануванні робіт над проектом необхідно враховувати ризики, що можуть негативно вплинути на час та якість розробки продукту.

Для даного проекту були виділені наступні ризики:

- R1 – Форс-мажорні обставини;
- R2 – Людський фактор;
- R3 – Апаратно-програмні збої;
- R4 – Неправильне розподілення ресурсів;
- R5 – Зміна вимог;
- R6 - Зміна цілей в ході реалізації проекту;
- R7 - Неякісне проведене тестування.

Таблиця В.7.1 – Класифікація рисків

№	**Ризик**	**Ймовірність виникнення**	**Обсяг втрат**
1	Форс-мажорні обставини	2	3
2	Людський фактор	1	2
3	Апаратно-програмні збої	1	3
4	Неправильне розподілення ресурсів	3	4
5	Зміна вимог	2	5
6	Зміна цілей в ході реалізації проекту	2	2
7	Неякісне проведення тестування	3	5

Використовуючи дану класифікацію, була побудована матриця ризиків, що представлена в таблиці В.7.2.

Таблиця В.7.2 – Матриця ризиків

Ймовірність виникнення							
		5	10	15	20	25	Неприпустимі ризики
		4	8	12	16	20	
	4, 7	3	6	9	12	15	Виправдані ризики
	1, 5, 6	2	4	6	8	10	
	2, 3	1	2	3	4	5	Допустимі ризики
			2, 6	1, 3	4	5, 7	
	Обсяг втрат						

Визначимо рівні ризиків та ступінь їх дії.

Рівні можуть бути:

– допустимі 1<R<4;

– виравдані 5<R<10;

– недопустимі 11<R<25.

Ступінь дії ризиків:

– ті, що можна проігнорувати 1<R<4;

– незначні 5<R<8;

– помірні 9<R<10;

– істотні 11<R<16;

– критичні 17<R<25.

На основі цих даних була виконана оцінка ступенів та рівнів для кожного ризику в проекті. Результати роботи представлені в табл. В.7.3.

Таблиця В.7.3 – Визначення ступенів та рівнів ризиків

№	Ризик	Ймовір-ність виникнен-ня	Обсяг втрат	Індекс ризи-ку	Рівень ризику	Ступінь дії
1	Форс-мажорні обставини	2	3	6	Виправданий	Незначний
2	Людський фактор	1	2	2	Допустимий	Проігнорувати
3	Апаратно-програмні збої	1	3	3	Допустимий	Проігнорувати
4	Неправильне розподілення ресурсів	3	4	12	Неприпустимий	Істотний
5	Зміна вимог	2	5	10	Виправданий	Помірний
6	Зміна цілей в ході реалізації проекту	2	2	4	Допустимий	Проігнорувати
7	Неякісне проведення тестування	3	5	15	Неприпустимий	Істотний

Ризик неякісного проведення тестування можна уникнути при побудові календарного плану додавши часу на цей етап. Щодо неправильного розподілення ресурсів, його можна уникнути при глибокому аналізі задач та ретельному плануванню робіт.

В.8 Формування бюджету проекту

В управлінні проектами використовують декілька термінів, пов’язаних із фінансуванням проекту: кошторис проекту, бюджет проекту, план фінансування продукту проекту. При цьому їх можуть застосовувати стосовно:

- всього життєвого циклу проекту;
- моменту отримання продукту проекту;
- фази реалізації проекту.

Кошторис продукту проекту – це загальні майбутні витрати, які необхідні безпосередньо для створення продукту проекту. Тобто це витрати на фінансування всіх робіт, передбачених WBS-структурою проекту.

Бюджет продукту проекту – це кошторис продукту проекту, розподілений в часі на основі календарного плану реалізації робіт або за окремими WBS елементами. План фінансування – це кошторис продукту проекту в розрізі основних джерел фінансування робіт з проекту. Розрахуємо бюджет заробітної плати.

На початку реалізації даного процесу необхідно визначити учасників проектів, які безпосередньо приймали участь в етапах реалізації задачі. На основі структури OBS було виділено таких працівників/виконавців:

- розробник;
- менеджер проекту;
- тестувальник.

У табл. В.8.1 приведена заробітна плата кожного учасника проекту з урахуванням різної кількості робочих годин кожного з них.

Таблиця В.8.1 – Заробітна плата учасників проекту

Посада	**Грн. за 1 год**	**Робочих годин**	**Заробітна плата**
Розробник	150	312	46800
Менеджер проекту	145	48	6960
Тестувальник	90	48	4320

Бюджет заробітної плати складає 58080 грн.

ДОДАТОК Г

Лістинг програмного коду

Файл Welcome.cs

```
using System;
using System.Collections;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.Xml;
using System.IO;
using Atlassian.Jira;
using WrapperGenerator;
using System.Threading.Tasks;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Drawing.Text;
using System.Drawing;

namespace WrapperGen
{
   public partial class Welcome : Form
   {
      int mov, movX, movY;
      Screen screen;
      public Welcome()
      {
         InitializeComponent();
      }

      [DllImport("gdi32.dll")]
      private static extern IntPtr AddFontMemResourceEx(IntPtr pbFont, uint cbFont, IntPtr pdv, [In] ref uint pcFonts);

      public static PrivateFontCollection private_fonts = new PrivateFontCollection();

      public static void LoadFont(string FontResourceName)
      {
         // receive resource stream
         Stream fontStream = Assembly.GetExecutingAssembly().GetManifestResourceStream(FontResourceName);
         //create an unsafe memory block for the data
         System.IntPtr data = Marshal.AllocCoTaskMem((int)fontStream.Length);
         //create a buffer to read in to
         Byte[] fontData = new Byte[fontStream.Length];
         //fetch the font program from the resource
         fontStream.Read(fontData, 0, (int)fontStream.Length);
         //copy the bytes to the unsafe memory block
         Marshal.Copy(fontData, 0, data, (int)fontStream.Length);
         // We HAVE to do this to register the font to the system (Weird .NET bug !)
         uint cFonts = 0;
         AddFontMemResourceEx(data, (uint)fontData.Length, IntPtr.Zero, ref cFonts);
```

```
    //pass the font to the font collection
    private_fonts.AddMemoryFont(data, (int)fontStream.Length);
    //close the resource stream
    fontStream.Close();
    //free the unsafe memory
    Marshal.FreeCoTaskMem(data);
}

public static void LoadFont(Byte[] fontData)
{
    //create an unsafe memory block for the data
    System.IntPtr data = Marshal.AllocCoTaskMem(fontData.Length);
    //copy the bytes to the unsafe memory block
    Marshal.Copy(fontData, 0, data, fontData.Length);
    // We HAVE to do this to register the font to the system (Weird .NET bug !)
    uint cFonts = 0;
    AddFontMemResourceEx(data, (uint)fontData.Length, IntPtr.Zero, ref cFonts);
    //pass the font to the font collection
    private_fonts.AddMemoryFont(data, fontData.Length);
    //free the unsafe memory
    Marshal.FreeCoTaskMem(data);
}

private async Task<string> GetIssueProperties(string id)
{
    Jira jira = Jira.CreateRestClient("https://sdc.revacom.com/", "o.necheporuk", "password");
    var issue = await jira.Issues.GetIssueAsync(id);
    string Type = issue["Packaging technology"].ToString();
    if (Type == "MSI")
    {
        Type = "MSI";
    }
    else if (Type.Contains("App-V"))
    {
        Type = "App-V";
    }
    else
    {
        Type = "Legacy";
    }
    return Type;
}

private async void Go_ClickAsync(object sender, EventArgs e)
{
    waitingGIF.Visible = true;
    waitingGIF.Refresh();
    btn_go.Enabled = false;
    string homepath = @"\\10.100.0.4\Storage\Temp\Necheporuk\WrapperGenerator";
    if (!Directory.Exists(homepath))
    {
        MessageBox.Show("Check your VPN settings");
    }
    else
    {
        //read projects.xml
        ArrayList projects = new ArrayList();
        XmlDocument doc = new XmlDocument();
        doc.Load(homepath + @"\projects.xml");
        foreach (XmlNode node in doc.GetElementsByTagName("project"))
        {
            projects.Add(node.Attributes["name"].Value);
```

```
                }
                string id = SDC_textbox.Text;       //SDC ID
                int index = id.IndexOf("-");
                int def = id.Length - id.Replace("-", "").Length;   // "-" amount
                //Check syntax of SDC id
                if (def != 1 || index < 1 || !int.TryParse(id.Substring(index + 1), out int n))
                {
                    MessageBox.Show("Wrong issue ID");
                }
                else if (!projects.Contains(id.Remove(index)))
                {
                    MessageBox.Show(String.Format("Project {0} is not available in WrapperGenerator.",
id.Remove(index)));
                }
                else
                {
                    //JIRA check
                    Jira jira = Jira.CreateRestClient("https://sdc.revacom.com/", "o.necheporuk", "password");
                    try
                    {
                        var issues = from i in jira.Issues.Queryable
                                     where i.Key == new LiteralMatch(id)
                                     select i;
                        issues.Count();
                        string type = await GetIssueProperties(id);

                        ChoosePath choose_form = new ChoosePath(id, type, screen);
                        this.Hide();
                        choose_form.ShowDialog();
                    }
                    catch
                    {
                        MessageBox.Show(String.Format("Issue {0} not found", id));
                    }
                }
            }
            waitingGIF.Visible = false;
        }

        private void Welcome_Load(object sender, EventArgs e)
        {
            LoadFont(WrapperGenerator.Properties.Resources.FRADMCN);
            Font FRADMCNFont = new Font(private_fonts.Families[0], 12);     //main buttons

            LoadFont(WrapperGenerator.Properties.Resources.framd);
            Font TBFont = new Font(private_fonts.Families[1], 16);     //welcome textbox
            btn_go.Font = FRADMCNFont;
            btn_exit.Font = FRADMCNFont;
            label1.Font = FRADMCNFont;
            SDC_textbox.Font = TBFont;
            screen = Screen.FromPoint(Cursor.Position);
            this.Location = screen.Bounds.Location;
            this.Left = screen.Bounds.Left + screen.Bounds.Width / 2 - this.Width / 2;
            this.Top = screen.Bounds.Top + screen.Bounds.Height / 2 - this.Height / 2;
        }

        private void Btn_exit_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }

        private void SDC_textbox_TextChanged(object sender, EventArgs e)
```

```
        {
            if (SDC_textbox.Text == "")
                btn_go.Enabled = false;
            else if (SDC_textbox.Text != "")
                btn_go.Enabled = true;
        }

        private void Welcome_MouseDown(object sender, MouseEventArgs e)
        {
            mov = 1;
            movX = e.X;
            movY = e.Y;
        }

        private void Welcome_MouseMove(object sender, MouseEventArgs e)
        {
            if (mov == 1)
            {
                this.SetDesktopLocation(MousePosition.X - movX, MousePosition.Y - movY);
            }
        }

        private void Welcome_MouseUp(object sender, MouseEventArgs e)
        {
            mov = 0;
        }
    }
}
```

Файл ChoosePath.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.IO;
using System.Xml;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using WrapperGen;
using System.Collections;
using Atlassian.Jira;
using Microsoft.Deployment.WindowsInstaller;


namespace WrapperGenerator
{
    public partial class ChoosePath : Form
    {
        int mov, movX, movY;
        string id, type;
        string tmppath = "";
        string homepath = @"\\10.100.0.4\Storage\Temp\Necheporuk\WrapperGenerator";
        int MSICount = 0, MSTCount = 0, EXECount = 0, APPVCount = 0;
        int conf = 0;
        string msi = "none", mst = "none", exe = "none", appv = "none";
        public static bool reboot = false;
        public static bool pshelp = false;
        ArrayList msis = new ArrayList();
        ArrayList msts = new ArrayList();
```

```
ArrayList exes = new ArrayList();
ArrayList appvs = new ArrayList();

Screen screen;
private async Task<string> GetIssueProperties(string property, Jira jira)
{
   var issue = await jira.Issues.GetIssueAsync(id);
   string value = issue[property].ToString();
   return value;
}

public ChoosePath(string ID, string Type, Screen Screen)
{
   InitializeComponent();
   this.id = ID;
   this.type = Type;
   this.screen = Screen;
   PATH_textbox.ForeColor = SystemColors.GrayText;
   PATH_textbox.Text = "Select working directory with package files";
   this.PATH_textbox.Leave += new System.EventHandler(this.PATH_textbox_Leave);
   this.PATH_textbox.Enter += new System.EventHandler(this.PATH_textbox_Enter);
   if (type == "MSI")
      rbMSI.Checked = true;
   else if (type == "App-V")
      rbAppV.Checked = true;
   else
      rbLegacy.Checked = true;
}

private void PATH_textbox_Leave(object sender, EventArgs e)
{
   if (PATH_textbox.Text.Length == 0)
   {
      PATH_textbox.Text = "Select working directory with package files";
      PATH_textbox.ForeColor = SystemColors.GrayText;
   }
}

private void PATH_textbox_Enter(object sender, EventArgs e)
{
   if (PATH_textbox.Text == "Select working directory with package files")
   {
      PATH_textbox.Text = "";
      PATH_textbox.ForeColor = SystemColors.WindowText;
   }
}

private void MSIlist_Leave(object sender, EventArgs e)
{
   if (PATH_textbox.Text.Length == 0)
   {
      MSIlist.Text = "Select main MSI file";
      MSIlist.ForeColor = SystemColors.GrayText;
   }
}

private void MSIlist_Enter(object sender, EventArgs e)
{
   if (PATH_textbox.Text == "Select working directory with package files")
   {
      PATH_textbox.Text = "";
      PATH_textbox.ForeColor = SystemColors.WindowText;
```

```
    }
}

private async Task<string> Get_Property(XmlNode node, Jira jira)
{
    string prop ="";
    if (type == "MSI" || type == "Legacy")
    {
        try
        {
            prop = node.Attributes["msiProperty"].Value;
            prop = GetMsiProperty(msi, prop, mst);
        }
        catch
        {
            prop = "";
        }
    }
    if (prop == "")
    {
        try
        {
            var issue = await jira.Issues.GetIssueAsync(id);
            prop = issue[node.Attributes["sdcProperty"].Value].ToString();
        }
        catch
        {
            prop = "";
        }
    }
    if (prop == "")
    {
        try
        {
            prop = node.Attributes["sdcProperty"].Value;
            //JIRA check
            var issues = from i in jira.Issues.Queryable
                         where i.Key == new LiteralMatch(id)
                         select i;
            issues.Count();
            prop = await GetIssueProperties(prop, jira);
        }
        catch
        {
            prop = "";
        }
    }
    if (prop == "")
    {
        try
        {
            prop = node.Attributes["defProperty"].Value;
        }
        catch
        {
            prop = "";
        }
    }
    return prop;
}

private async void Btn_go_Click(object sender, EventArgs e)
```

```
{
    waitingGIF.Visible = true;
    waitingGIF.Refresh();
    btn_go.Enabled = false;
    //Check is path exists
    if (!Directory.Exists(PATH_textbox.Text))
    {
        MessageBox.Show("Folder doesn`t exist");
    }
    else
    {
        tmppath = Path.GetRandomFileName();
        tmppath = "WG" + tmppath.Remove(tmppath.IndexOf("."));

        XmlDocument doc = new XmlDocument();
        doc.Load(homepath + @"\projects.xml");

        string config = "";
        string WRtype = "";
        //generate list of projects
        foreach (XmlNode node in doc.GetElementsByTagName("project"))
        {
            if (node.Attributes["name"].Value == id.Remove(id.IndexOf("-")))
            {
                config = node["config"].InnerText;      //dirname section from xml
                WRtype = node["type"].InnerText;        //Wrapper type (ps1/cmd/none)
                break;
            }
        }
        Directory.CreateDirectory(Path.Combine(Path.GetTempPath(), tmppath));

        doc.Load(homepath + "\\" + config);

        string srcpath = homepath + @"\templates\" +
doc.DocumentElement.SelectSingleNode("/Properties/dirname").InnerText;

        string comment = "";
        comment = doc.DocumentElement.SelectSingleNode("/Properties/comment").InnerText;

        // Now Create all of the directories
        foreach (string dirPath in Directory.GetDirectories(srcpath, "*", SearchOption.AllDirectories))
            Directory.CreateDirectory(dirPath.Replace(srcpath,          Path.Combine(Path.GetTempPath(),
tmppath)));

        //Copy all the files & Replaces any files with the same name
        foreach (string newPath in Directory.GetFiles(srcpath, "*.*",
            SearchOption.AllDirectories))
            File.Copy(newPath, newPath.Replace(srcpath, Path.Combine(Path.GetTempPath(), tmppath)),
true);


        if (MSIlist.SelectedIndex == -1)
            MSIlist.SelectedIndex = 0;
        if (MSTlist.SelectedIndex == -1)
            MSTlist.SelectedIndex = 0;
        if (EXElist.SelectedIndex == -1)
            EXElist.SelectedIndex = 0;

        if (MSICount == 1)
            msi = msis[0].ToString();
        else if (MSICount == 0 || MSIlist.SelectedIndex == 0)
            msi = "none";
```

```
                else
                {
                    msi = msis[MSIlist.SelectedIndex - 1].ToString();
                }

                if (MSTCount == 1)
                    mst = msts[0].ToString();
                else if (MSTCount == 0 || MSTlist.SelectedIndex == 0)
                    mst = "none";
                else
                {
                    mst = msts[MSTlist.SelectedIndex - 1].ToString();
                }

                Jira jira = Jira.CreateRestClient("https://sdc.revacom.com/", "o.necheporuk", "password");
                if (type == "Legacy")
                {
                    if (exes.Count == 0 || (exes.Count > 1 && EXElist.SelectedIndex == 0))
                        exe = "none";
                    else if (exes.Count == 1)
                        exe = exes[0].ToString();
                    else
                        exe = exes[EXElist.SelectedIndex - 1].ToString();

                    if (!configMSI.Checked && MSIlist.SelectedIndex == 0)
                        msi = "none";
                    mst = "none";
                }
                else if (type == "App-V")
                {
                    msi = "none";
                    mst = "none";
                    exe = "none";
                    if (appvs.Count == 0 || (appvs.Count > 1 && EXElist.SelectedIndex == 0))
                        appv = "none";
                    else if (appvs.Count == 1)
                        appv = appvs[0].ToString();
                    else
                        appv = appvs[EXElist.SelectedIndex - 1].ToString();
                }
                Main main_form = new Main(id, type, WRtype, Path.Combine(Path.GetTempPath(), tmppath),
PATH_textbox.Text, msi, exe, mst, appv, comment);
                {
                    conf = 0;
                    int tab = 1;
                    foreach (XmlNode node in doc.GetElementsByTagName("config"))
                    {
                        if (node.InnerText != "")
                        {
                            if (msi == "none")
                            {
                                if (node.Attributes["name"].Value == "WGMSIName" ||
node.Attributes["name"].Value == "WGMSTName")
                                    continue;
                            }
                            if (mst == "none")
                            {
                                if (node.Attributes["name"].Value == "WGMSTName")
                                    continue;
                            }
                            if (exe == "none")
                            {
```

```
                    if (node.Attributes["name"].Value == "WGEXEName" || node.Attributes["name"].Value == "WGEXEParams" || node.Attributes["name"].Value == "WGUninstallEXEName" || node.Attributes["name"].Value == "WGUninstallEXEParams")
                        continue;
                }
                main_form.list.Add(new List<string>());
                main_form.list[conf].Add(node.Attributes["name"].Value);
                Label label = new Label
                {
                    Text = node.InnerText,
                    Size = new System.Drawing.Size(300, 15),
                    Font = new System.Drawing.Font("Franklin Gothic Book", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)))
                };

                string prop = "";

                if (msi != "none")
                {
                    if (node.Attributes["name"].Value == "WGMSIName")
                        prop = Path.GetFileName(msi);
                    if (node.Attributes["name"].Value == "WGMSTName")
                    {
                        if (mst != "none")
                            prop = Path.GetFileName(mst);
                        else
                            continue;
                    }
                    if (prop == "")
                    {
                        prop = await Get_Property(node, jira);
                    }
                }
                if (exe != "none")
                {
                    if (node.Attributes["name"].Value == "WGEXEName" || node.Attributes["name"].Value == "WGUninstallEXEName")
                        prop = Path.GetFileName(exe);
                    if (prop == "")
                    {
                        prop = await Get_Property(node, jira);
                    }
                }
                if (appv != "none")
                {
                    if (prop == "")
                    {
                        prop = await Get_Property(node, jira);
                    }
                }
                TextBox tb = new TextBox
                {
                    Text = prop,
                    Size = new System.Drawing.Size(370, 25),
                    TabIndex = tab,
                    Name = node.Attributes["name"].Value
                };
                main_form.list[conf].Add(prop);
                main_form.panel.Controls.Add(label);
                main_form.panel.Controls.Add(tb);
            }
            else
```

```
                {
                    //Config which not showed
                    main_form.list.Add(new List<string>());
                    main_form.list[conf].Add(node.Attributes["name"].Value);

                    string prop = "";

                    if (node.Attributes["name"].Value == "WGReboot" || node.Attributes["name"].Value ==
"WG0")
                    {
                        reboot = true;
                        prop = "true";
                        main_form.list[conf].Add(prop);
                        conf++;
                        continue;
                    }
                    else if(node.Attributes["name"].Value == "WGDate")
                    {
                        prop = DateTime.Now.ToString(node.Attributes["defProperty"].Value);
                        main_form.list[conf].Add(prop);
                        conf++;
                        continue;
                    }
                    else if (node.Attributes["name"].Value == "WGMSTransform")
                    {
                        if (mst != "none")
                            prop = "true";
                        else
                            prop = "false";
                        main_form.list[conf].Add(prop);
                        conf++;
                        continue;
                    }
                    else if (node.Attributes["name"].Value == "WGEXEString" ||
node.Attributes["name"].Value == "WGUninstallEXEString")
                    {
                        if (exe != "none")
                            prop = "true";
                        else
                            prop = "false";
                        main_form.list[conf].Add(prop);
                        conf++;
                        continue;
                    }
                    else if (node.Attributes["name"].Value == "WGPSHelp")
                    {
                        pshelp = true;
                    }
                    else if (node.Attributes["name"].Value == "WGPSPreInstall" ||
node.Attributes["name"].Value == "WGPSInstall" || node.Attributes["name"].Value == "WGPSPostInstall" ||
node.Attributes["name"].Value == "WGPSPreUninstall" || node.Attributes["name"].Value == "WGPSUninstall" ||
node.Attributes["name"].Value == "WGPSPostUninstall")
                    {
                        prop = "";
                        main_form.list[conf].Add(prop);
                        conf++;
                        continue;
                    }

                    prop = await Get_Property(node, jira);

                    main_form.list[conf].Add(prop);
```

```
                }
                conf++;
            }
        }

        this.Hide();

        main_form.ShowDialog();
    }
    waitingGIF.Visible = false;
}

public static string GetMsiProperty(string msiFile, string property, string mstFile)
{
    using (var database = new Database(msiFile, DatabaseOpenMode.ReadOnly))
    {
        if (mstFile != "none")
        {
            database.ApplyTransform(mstFile);
        }

        using (var view = database.OpenView(database.Tables["Property"].SqlSelectString))
        {
            view.Execute();

            foreach (var rec in view) using (rec)
                {
                    if (rec.GetString("Property")==property)
                        return rec.GetString("Value");
                }
        }
    }
    return "";
}

private void Btn_exit_Click(object sender, EventArgs e)
{
    Environment.Exit(0);
}

private void SelectFolder_Click(object sender, EventArgs e)
{
    string folderPath = "";
    FolderBrowserDialog folderBrowserDialog1 = new FolderBrowserDialog();
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        folderPath = folderBrowserDialog1.SelectedPath;
        PATH_textbox.Text = folderPath;
        PATH_textbox.ForeColor = SystemColors.WindowText;
    }
}

private void PATH_textbox_TextChanged(object sender, EventArgs e)
{
    if (PATH_textbox.Text == "" || PATH_textbox.Text == "Select working directory with package
files")
    {
        rbMSI.Enabled = false;
        rbAppV.Enabled = false;
        rbLegacy.Enabled = false;
        btn_go.Enabled = false;
        cbMST.Enabled = false;
```

```
            configMSI.Enabled = false;
            MSIlist.Enabled = false;
            MSTlist.Enabled = false;
            EXElist.Enabled = false;
        }
        else
        {
            if (!Directory.Exists(PATH_textbox.Text))
            {
                folderErr.Visible = true;
                btn_go.Enabled = false;

                rbMSI.Enabled = false;
                rbAppV.Enabled = false;
                rbLegacy.Enabled = false;
                cbMST.Enabled = false;
                configMSI.Enabled = false;
                MSIlist.Enabled = false;
                MSTlist.Enabled = false;
                EXElist.Enabled = false;
            }
            else
            {
                waitingGIF.Visible = true;
                MSIlist.Items.Clear();
                MSTlist.Items.Clear();
                EXElist.Items.Clear();
                MSICount = 0;
                MSTCount = 0;
                EXECount = 0;
                APPVCount = 0;

                MSIlist.Items.Add("none");
                MSTlist.Items.Add("none");
                EXElist.Items.Add("none");
                folderErr.Visible = false;

                exes.Clear();
                msis.Clear();
                msts.Clear();
                appvs.Clear();

                foreach (string i in Directory.GetFiles(PATH_textbox.Text, "*.appv",
SearchOption.AllDirectories))
                {
                    appvs.Add(i);
                    APPVCount++;
                }

                foreach (string i in Directory.GetFiles(PATH_textbox.Text, "*.exe",
SearchOption.AllDirectories))
                {
                    exes.Add(i);
                    EXElist.Items.Add(Path.GetFileName(i));
                    EXECount++;
                }

                foreach (string i in Directory.GetFiles(PATH_textbox.Text, "*.msi",
SearchOption.AllDirectories))
                {
                    msi = i;
                    msis.Add(i);
```

```
                    MSIlist.Items.Add(Path.GetFileName(i));
                    MSICount++;
                }

                foreach (string t in Directory.GetFiles(PATH_textbox.Text, "*.mst",
SearchOption.AllDirectories))
                {
                    mst = t;
                    msts.Add(t);

                    MSTlist.Items.Add(Path.GetFileName(t));
                    MSTCount++;
                }


                if (rbMSI.Checked)
                    MSIaction();
                if (rbLegacy.Checked)
                    LegacyAction();
                if (rbAppV.Checked)
                    AppVAction();

                waitingGIF.Visible = false;

                rbMSI.Enabled = true;
                rbAppV.Enabled = true;
                rbLegacy.Enabled = true;
                btn_go.Enabled = true;
                cbMST.Enabled = true;
                configMSI.Enabled = true;
                MSIlist.Enabled = true;
                MSTlist.Enabled = true;
                EXElist.Enabled = true;
            }
        }
    }

    private void MSIaction()
    {
        if (rbMSI.Checked)
        {
            MSIlist.Location = new System.Drawing.Point(24, 72);
            if (MSICount == 0)
            {
                cbMST.Visible = false;
                MSIlist.Visible = false;
                MSTlist.Visible = false;
            }
            else if (MSICount == 1)
            {
                if (MSTCount == 0)
                {
                    cbMST.Visible = false;
                    MSIlist.Visible = false;
                    MSTlist.Visible = false;
                }
                else if (MSTCount == 1)
                {
                    cbMST.Location = new System.Drawing.Point(45, 80);
                    cbMST.Visible = true;
                    cbMST.Checked = true;
                    MSIlist.Visible = false;
```

```
            MSTlist.Visible = false;
        }
        else if (MSTCount > 1)
        {
            MSTlist.Location = new System.Drawing.Point(24, 72);
            cbMST.Visible = false;
            MSIlist.Visible = false;
            MSTlist.Visible = true;
        }
    }
    else if (MSICount > 1)
    {
        if (MSTCount == 0)
        {
            mst = "none";
            cbMST.Visible = false;
            MSIlist.Text = "Select main MSI file";
            MSIlist.Visible = true;
            MSTlist.Visible = false;
        }
        else if (MSTCount == 1)
        {
            cbMST.Location = new System.Drawing.Point(45, 95);
            cbMST.Visible = true;
            cbMST.Checked = true;
            MSIlist.Visible = true;
            MSTlist.Visible = false;
        }
        else if (MSTCount > 1)
        {
            MSTlist.Location = new System.Drawing.Point(24, 97);
            cbMST.Visible = false;
            MSIlist.Visible = true;
            MSTlist.Visible = true;
        }
    }
}
else
{
    cbMST.Visible = false;
    mst = "none";
    MSIlist.Visible = false;
    MSTlist.Visible = false;
}
}

private void LegacyAction()
{
if (rbLegacy.Checked)
{
    if (EXECount == 0 || EXECount == 1)
    {
        configMSI.Location = new System.Drawing.Point(218, 80);
        EXElist.Visible = false;
    }
    else
    {
        EXElist.Text = "Select main EXE file";
        EXElist.Items.Clear();
        EXElist.Items.Add("none");
        foreach (string s in exes)
        { EXElist.Items.Add(Path.GetFileName(s)); }
```

```
            EXElist.Visible = true;
            configMSI.Location = new System.Drawing.Point(218, 95);
        }

        if (MSICount == 0)
        {
            msi = "none";
            configMSI.Visible = false;
            MSIlist.SelectedIndex = 0;
            configMSI.Checked = false;
            MSIlist.Visible = false;
        }
        else if (MSICount == 1)
        {
            configMSI.Visible = true;
            configMSI.Checked = true;
            MSIlist.Visible = false;
            MSIlist.SelectedIndex = 0;
        }
        else if (MSICount > 1)
        {
            configMSI.Visible = false;
            configMSI.Checked = false;
            MSIlist.Text = "Select config MSI if needed";
            if (EXECount == 0 || EXECount == 1)
                MSIlist.Location = new System.Drawing.Point(24, 72);
            else
                MSIlist.Location = new System.Drawing.Point(24, 97);
            MSIlist.Visible = true;
        }
    }
    else
    {
        MSIlist.Location = new System.Drawing.Point(24, 72);
        EXElist.Visible = false;
        configMSI.Checked = false;
        configMSI.Visible = false;
        MSIlist.Text = "Select main MSI file";
        MSIlist.Visible = false;
    }

}


private void AppVAction()
{
    if (rbAppV.Checked)
    {
        if (APPVCount == 0 || APPVCount == 1)
        {
            EXElist.Visible = false;
        }
        else
        {
            EXElist.Text = "Select main App-V file";
            EXElist.Items.Clear();
            EXElist.Items.Add("none");
            foreach (string s in appvs)
            { EXElist.Items.Add(Path.GetFileName(s)); }
            EXElist.Visible = true;
        }
```

```
        configMSI.Visible = false;
        cbMST.Visible = false;
        msi = "none";
        mst = "none";
        MSIlist.Visible = false;
        MSTlist.Visible = false;
    }
    else
    {
        EXElist.Text = "Select main EXE file";
        EXElist.Items.Clear();
        EXElist.Items.Add("none");
        foreach (string s in exes)
        { EXElist.Items.Add(s); }
    }
}

private void RbAppV_CheckedChanged(object sender, EventArgs e)
{
    if (rbAppV.Checked)
        type = "App-V";
    AppVAction();
}

private void RbLegacy_CheckedChanged(object sender, EventArgs e)
{
    if (rbLegacy.Checked)
        type = "Legacy";
    LegacyAction();
}

private void RbMSI_CheckedChanged(object sender, EventArgs e)
{
    if (rbMSI.Checked)
        type = "MSI";
    MSIaction();
}

private void ChoosePath_Load(object sender, EventArgs e)
{

    Font FRADMCNFont = new Font(Welcome.private_fonts.Families[0], 12);
    btn_go.Font = FRADMCNFont;
    btn_exit.Font = FRADMCNFont;
    rbMSI.Font = FRADMCNFont;
    rbAppV.Font = FRADMCNFont;
    rbLegacy.Font = FRADMCNFont;

    Font FRADMCNFontS = new Font(Welcome.private_fonts.Families[0], 9);
    configMSI.Font = FRADMCNFontS;
    cbMST.Font = FRADMCNFontS;
    folderErr.Font = FRADMCNFontS;

    Welcome.LoadFont(WrapperGenerator.Properties.Resources.FRABK);
    Font FRABKFont = new Font(Welcome.private_fonts.Families[2], 8);
    PATH_textbox.Font = FRABKFont;

    this.Location = screen.Bounds.Location;
    this.Left = screen.Bounds.Left + screen.Bounds.Width / 2 - this.Width / 2;
    this.Top = screen.Bounds.Top + screen.Bounds.Height / 2 - this.Height / 2;
}
```

```
        //Form moving
        private void ChoosePath_MouseDown(object sender, MouseEventArgs e)
        {
            mov = 1;
            movX = e.X;
            movY = e.Y;
        }
        private void ChoosePath_MouseMove(object sender, MouseEventArgs e)
        {
            if (mov == 1)
            {
                this.SetDesktopLocation(MousePosition.X - movX, MousePosition.Y - movY);
            }
        }
        private void ChoosePath_MouseUp(object sender, MouseEventArgs e)
        {
            mov = 0;
        }

    }
}
```

Файл Main.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using WrapperGenerator;

namespace WrapperGen
{
   public partial class Main : Form
   {
      int mov, movX, movY;
      string id, type, WRtype, comment;
      string tmppath, fpath, msi, exe, mst, appv, unexe, productCode;
      public List<List<string>> list = new List<List<string>>();
      string pshelp = "";

      public static List<string> PreInst = new List<string>();
      public static string PreInstall = "";
      public static string Install = "";
      public static string PostInstall = "";
      public static string PreUninstall = "";
      public static string Uninstall = "";
      public static string PostUninstall = "";
      bool FirstModify = true;

      public Main()
      {
         InitializeComponent();
      }

      public Main(string ID, string Type, string WRtype, string TempPath, string FilesPath, string MSI, string EXE, string MST, string APPV, string Comment)
      {
         InitializeComponent();
         this.id = ID;
         this.type = Type;
         this.WRtype = WRtype;
         this.tmppath = TempPath;
         this.fpath = FilesPath;
         this.msi = MSI;
         this.exe = EXE;
         this.mst = MST;
         this.appv = APPV;
         this.unexe = EXE;
         this.comment = Comment;

         PATH_textbox.ForeColor = SystemColors.GrayText;
         PATH_textbox.Text = "Select target directory to save the package";
         this.PATH_textbox.Leave += new System.EventHandler(this.PATH_textbox_Leave);
         this.PATH_textbox.Enter += new System.EventHandler(this.PATH_textbox_Enter);

         if (WRtype == "cmd")
         {
            cmdGroup.Location = new System.Drawing.Point(455, 45);
            cmdGroup.Visible = true;
            cbInstall.Checked = true;
            cbUninstall.Checked = true;
         }
```

```
    else
    {
        cmdGroup.Location = new System.Drawing.Point(800, 45);
        cbInstall.Checked = false;
        cbUninstall.Checked = false;
        cmdGroup.Visible = false;
    }
    if (WRtype == "ps1")
    {
        btn_modifyPs.Visible = true;
    }
    else
        btn_modifyPs.Visible = false;

    if (type == "App-V")
    {
        cmdGroup.Location = new System.Drawing.Point(800, 45);
        cbInstall.Checked = false;
        cbUninstall.Checked = false;
        cmdGroup.Visible = false;
    }
}

private void Main_Load(object sender, EventArgs e)
{
    if (ChoosePath.reboot)
    {
        rebootGroup.Location = new System.Drawing.Point(455, 45);
        rebootGroup.Visible = true;
        rbRebootTrue.Visible = true;
        rbRebootFalse.Visible = true;
    }
    else
    {
        rebootGroup.Visible = false;
        rbRebootTrue.Visible = false;
        rbRebootFalse.Visible = false;
    }
    if (ChoosePath.pshelp)
    {
        int conf = list.Count;
        for (int i = 0; i < conf; i++)
        {
            if (list[i][0] == "WGPSHelp")
            {
                pshelp = list[i][1];
                break;
            }
        }
    }



    Font TBFont = new Font(Welcome.private_fonts.Families[1], 10);

    Font FRADMCNFontS = new Font(Welcome.private_fonts.Families[1], 9);

    Font FRADMCNFont = new Font(Welcome.private_fonts.Families[1], 12);

    cmdGroup.Font = TBFont;
    rebootGroup.Font = TBFont;
    folderErr.Font = FRADMCNFontS;
```

```
            rbRebootTrue.Font = TBFont;
            rbRebootFalse.Font = TBFont;

            btn_gen.Font = FRADMCNFont;
            btn_modifyPs.Font = FRADMCNFont;
            cbUninstall.Font = FRADMCNFont;
            cbInstall.Font = FRADMCNFont;


            Screen screen = Screen.FromPoint(Cursor.Position);
            this.Location = screen.Bounds.Location;
            this.Left = screen.Bounds.Left + screen.Bounds.Width / 2 - this.Width / 2;
            this.Top = screen.Bounds.Top + screen.Bounds.Height / 2 - this.Height / 2;
        }
        //Form moving
        private void Header_MouseDown(object sender, MouseEventArgs e)
        {
            mov = 1;
            movX = e.X;
            movY = e.Y;
        }
        private void Header_MouseMove(object sender, MouseEventArgs e)
        {
            if (mov == 1)
            {
                this.SetDesktopLocation(MousePosition.X - movX, MousePosition.Y - movY);
            }
        }
        private void Header_MouseUp(object sender, MouseEventArgs e)
        {
            mov = 0;
        }



        private void Btn_modifyPs_Click(object sender, EventArgs e)
        {
            if (msi != "none")
            {
                int conf = list.Count;
                for (int i = 0; i < conf; i++)
                {
                    if (list[i][0] == "WGProductCode")
                    {
                        productCode = list[i][1];
                        break;
                    }
                }
            }
            string strCmdText;
            strCmdText = "-noprofile -ExecutionPolicy ByPass -File \"" + tmppath + "\\" + pshelp + "\"";
            Modifying modify_form = new Modifying(strCmdText, msi, exe, mst, appv, productCode,
FirstModify);
            modify_form.ShowDialog();
            FirstModify = false;
        }

        private void SelectFolder_Click(object sender, EventArgs e)
        {
            string folderPath = "";
            FolderBrowserDialog folderBrowserDialog1 = new FolderBrowserDialog();
            if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
```

```
        {
            folderPath = folderBrowserDialog1.SelectedPath;
            PATH_textbox.Text = folderPath;
            PATH_textbox.ForeColor = SystemColors.WindowText;
        }
    }

    private void PATH_textbox_TextChanged(object sender, EventArgs e)
    {
        if (PATH_textbox.Text == "" || PATH_textbox.Text == "Select target directory to save the
package")
        {
            btn_gen.Enabled = false;
            folderErr.Visible = false;
        }
        else if (!Directory.Exists(PATH_textbox.Text))
        {
            folderErr.Visible = true;
            btn_gen.Enabled = false;
        }
        else
        {
            folderErr.Visible = false;
            btn_gen.Enabled = true;
        }
    }

    private void PATH_textbox_Leave(object sender, EventArgs e)
    {
        if (PATH_textbox.Text.Length == 0)
        {
            PATH_textbox.Text = "Select target directory to save the package";
            PATH_textbox.ForeColor = SystemColors.GrayText;
        }
    }

    private void PATH_textbox_Enter(object sender, EventArgs e)
    {
        if (PATH_textbox.Text == "Select target directory to save the package")
        {
            PATH_textbox.Text = "";
            PATH_textbox.ForeColor = SystemColors.WindowText;
        }
    }

    private void CbInstall_CheckedChanged(object sender, EventArgs e)
    {
        if (!cbInstall.Checked)
        {
            try
            {
                panel.Controls["WGMSIName"].Enabled = false;
                panel.Controls["WGMSTName"].Enabled = false;
                panel.Controls["WGEXEName"].Enabled = false;
                panel.Controls["WGEXEParams"].Enabled = false;
                panel.Controls["WGUninstallEXEName"].Enabled = false;
                panel.Controls["WGUninstallEXEParams"].Enabled = false;
            }
            catch
            { }
        }
        else
```

```
        {
            try
            {
                panel.Controls["WGMSIName"].Enabled = true;
                panel.Controls["WGMSTName"].Enabled = true;
                panel.Controls["WGEXEName"].Enabled = true;
                panel.Controls["WGEXEParams"].Enabled = true;
                panel.Controls["WGUninstallEXEName"].Enabled = true;
                panel.Controls["WGUninstallEXEParams"].Enabled = true;
            }
            catch
            { }
        }
    }


    private async void Btn_gen_Click(object sender, EventArgs e)
    {
        btn_gen.Enabled = false;
        PATH_textbox.Enabled = false;
        SelectFolder.Enabled = false;
        btn_modifyPs.Enabled = false;

        waitingGIF.Visible = true;
        waitingGIF.Refresh();
        int conf = list.Count;      //list size
        string PkgName = "";
        string dirFiles = "";
        bool rebootReq = false;
        for (int i=0; i<conf;i++)
        {
            if (list[i][0] == "WGdirFiles")
                dirFiles = list[i][1];
            if (panel.Controls.ContainsKey(list[i][0]))
                {
                    list[i][1] = panel.Controls[list[i][0]].Text;
                }
            if (list[i][0] == "WGPkgName")
                {
                    PkgName = list[i][1];
                }

        }


        if (WRtype=="cmd")
        {
            string install = "", uninstall = "";
            for (int i = 0; i < conf; i++)
            {
                if (list[i][0] == "WGInstall")
                    install = list[i][1];
                else if (list[i][0] == "WGUninstall")
                    uninstall = list[i][1];
                if (list[i][0] == "WGMSTransform" && list[i][1] == "true")
                    if (panel.Controls.ContainsKey("WGMSTName"))
                        list[i][1] = " TRANSFORMS=\"%~dp0" + panel.Controls["WGMSTName"].Text + "\"";
                    else
                        list[i][1] = "false";

                if (list[i][0] == "WGEXEString" && list[i][1] == "true")
                    if (panel.Controls.ContainsKey("WGEXEName"))
```

```
                        list[i][1] = "\"%~dp0" + panel.Controls["WGEXEName"].Text + "\" " +
panel.Controls["WGEXEParams"].Text;
                    else
                        list[i][1] = "false";

                if (list[i][0] == "WGUninstallEXEString" && list[i][1] == "true")
                    if (panel.Controls.ContainsKey("WGUninstallEXEName"))
                        list[i][1] = "\"%~dp0" + panel.Controls["WGUninstallEXEName"].Text + "\" " +
panel.Controls["WGUninstallEXEParams"].Text;
                    else
                        list[i][1] = "false";
            }
            if (!cbInstall.Checked)
            {
                if (File.Exists(Path.Combine(tmppath, dirFiles, install)))
                    File.Delete(Path.Combine(tmppath, dirFiles, install));
            }
            else
            {
                //modify install.cmd
                if (msi == "none")
                {
                    if (exe != "none")
                    {
                        if (File.Exists(Path.Combine(tmppath, dirFiles, install)))
                            File.WriteAllText(Path.Combine(tmppath, dirFiles, install), "\"%~dp0" +
panel.Controls["WGEXEName"].Text + "\" " + panel.Controls["WGEXEParams"].Text);
                    }
                }
                else
                {
                    if (File.Exists(Path.Combine(tmppath, dirFiles, install)))
                    {
                        string text = File.ReadAllText(Path.Combine(tmppath, dirFiles, install));
                        for (int i = 0; i < conf; i++)
                        {
                            if (list[i][0] == "WGMSTransform")
                            {
                                if (list[i][1] == "false")
                                    text = text.Replace("<WGMSTransform>", "");
                                else
                                    text = text.Replace("<WGMSTransform>", list[i][1]);
                            }
                            else if (list[i][0] == "WGEXEString")
                            {
                                if (list[i][1] == "false")
                                    text = text.Replace("<WGEXEString>", "");
                                else
                                    text = text.Replace("<WGEXEString>", list[i][1]);
                            }
                            else
                                text = text.Replace("<" + list[i][0] + ">", list[i][1]);
                        }
                        File.WriteAllText(Path.Combine(tmppath, dirFiles, install), text);
                    }
                }
            }


            if (!cbUninstall.Checked)
            {
                if (File.Exists(Path.Combine(tmppath, dirFiles, uninstall)))
```

```
                    File.Delete(Path.Combine(tmppath, dirFiles, uninstall));
                }
                else
                {
                    //modify uninstall.cmd
                    if (msi == "none")
                    {
                        if (exe != "none")
                        {
                            if (File.Exists(Path.Combine(tmppath, dirFiles, uninstall)))
                                File.WriteAllText(Path.Combine(tmppath, dirFiles, uninstall), "\"%~dp0" +
panel.Controls["WGUninstallEXEName"].Text + "\" " + panel.Controls["WGUninstallEXEParams"].Text);
                        }
                    }
                    else
                    {
                        if (File.Exists(Path.Combine(tmppath, dirFiles, uninstall)))
                        {
                            string text = File.ReadAllText(Path.Combine(tmppath, dirFiles, uninstall));
                            for (int i = 0; i < conf; i++)
                            {
                                if (list[i][0] == "WGUninstallEXEString")
                                {
                                    if (list[i][1] == "false")
                                        text = text.Replace("<WGUninstallEXEString>", "");
                                    else
                                        text = text.Replace("<WGUninstallEXEString>", list[i][1]);
                                }
                                else
                                    text = text.Replace("<" + list[i][0] + ">", list[i][1]);
                            }
                            File.WriteAllText(Path.Combine(tmppath, dirFiles, uninstall), text);
                        }
                    }
                }
            }
            else if (WRtype == "none")
            {
            }
            else if (WRtype == "ps1")
            {
                string DeployPS = "";

                for (int i = 0; i < conf; i++)
                {
                    if (list[i][0] == "WGReboot")
                    {
                        if (rbRebootTrue.Checked)
                        {
                            rebootReq = true;
                            list[i][1] = "$true";
                        }
                        else
                            list[i][1] = "$false";
                    }
                    else if (list[i][0] == "WGInstallPS")
                        DeployPS = list[i][1];
                    else if (list[i][0] == "WGPSPreInstall")
                    {
                        list[i][1] = PreInstall;
                    }
                    else if (list[i][0] == "WGPSInstall")
```

```
        {
            list[i][1] = Install;
        }
        else if (list[i][0] == "WGPSPostInstall")
        {
            list[i][1] = PostInstall;
        }
        else if (list[i][0] == "WGPSPreUninstall")
        {
            list[i][1] = PreUninstall;
        }
        else if (list[i][0] == "WGPSUninstall")
        {
            list[i][1] = Uninstall;
        }
        else if (list[i][0] == "WGPSPostUninstall")
        {
            list[i][1] = PostUninstall;
        }

    }
    foreach (string i in Directory.GetFiles(tmppath, DeployPS, SearchOption.AllDirectories))
    {
        DeployPS = i;
    }
    //Headers sets
    if (File.Exists(DeployPS))
    {
        string text = File.ReadAllText(DeployPS);
        for (int i = 0; i < conf; i++)
        {
            if (list[i][0] == "WG0")
            {
                if (rebootReq)
                    text = text.Replace("<WG0>", "0,");
                else
                    text = text.Replace("<WG0>", "");
            }
            else if (list[i][0] == "WGSDCID")
            {
                if (rebootReq)
                    text = text.Replace("<WGSDCID>", id);
            }
            else
                text = text.Replace("<" + list[i][0] + ">", list[i][1]);
        }
        File.WriteAllText(DeployPS, text);
    }
    else
        MessageBox.Show("Deploy script not found.");
}

//Copy template folder with modified wrapper to the TARGET directory
string targetdir = PATH_textbox.Text;
if (PkgName!="")
{
    Directory.CreateDirectory(Path.Combine(PATH_textbox.Text, PkgName));
    targetdir = Path.Combine(PATH_textbox.Text, PkgName);
}

foreach (string dirPath in Directory.GetDirectories(tmppath, "*", SearchOption.AllDirectories))
    Directory.CreateDirectory(dirPath.Replace(tmppath, targetdir));
```

```
                foreach (string newPath in Directory.GetFiles(tmppath, "*.*", SearchOption.AllDirectories))
                    File.Copy(newPath, newPath.Replace(tmppath, targetdir), true);

                //Copy Package files to the TARGET directory
                    try
                    {
                        dirFiles = Directory.GetDirectories(targetdir, dirFiles)[0];
                        foreach (string dirPath in Directory.GetDirectories(fpath, "*", SearchOption.AllDirectories))
                            Directory.CreateDirectory(dirPath.Replace(fpath, dirFiles));

                        foreach (string newPath in Directory.GetFiles(fpath, "*.*", SearchOption.AllDirectories))
                            File.Copy(newPath, newPath.Replace(fpath, dirFiles), true);
                    }
                    catch
                    {
                        MessageBox.Show(String.Format("dirFiles not found."));
                    }


                waitingGIF.Visible = false;
                PATH_textbox.Text = "Your package is ready.";
                folderErr.Visible = false;

                if (comment != "")
                    MessageBox.Show(String.Format(comment));
                System.Diagnostics.Process.Start("explorer.exe", targetdir);
            }

            private void Min_Click(object sender, EventArgs e)
            {
                this.WindowState = FormWindowState.Minimized;
            }

            private void Cls_Click(object sender, EventArgs e)
            {
                if (MessageBox.Show("Do you really want to exit? Unsaved data will be lost.", "Warning",
MessageBoxButtons.OKCancel) == DialogResult.OK)
                {
                    //Temp folder cleanup
                    for (int i = 0; i < panel.Controls.Count; i++)
                    {
                        panel.Controls[i].Dispose();
                    }
                    panel.Dispose();
                    if (Directory.Exists(tmppath))
                    {
                        Directory.Delete(tmppath, true);
                    }
                    this.Close();
                    Application.Exit();
                }
            }
        }
    }
```

Файл Modifying.cs

```
using System;
using System.Drawing;
using System.IO;
```

```
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using WrapperGen;

namespace WrapperGenerator
{
   public partial class Modifying : Form
   {
      int mov, movX, movY;
      int mods = 0;
      int tab = 10;
      int y = 5;

      int PreInstCount = 0;
      int InstCount = 0;
      int PostInstCount = 0;
      int PreUninstCount = 0;
      int UninstCount = 0;
      int PostUninstCount = 0;

      string strHelpCmd = "";
      string msi, mst, exe, appv, productCode;
      bool isFirst;

      public Modifying(string strCmdText, string MSI, string EXE, string MST, string APPV, string PC,
bool FirstClick)
      {
         this.strHelpCmd = strCmdText;
         this.msi = MSI;
         this.exe = EXE;
         this.mst = MST;
         this.appv = APPV;
         this.productCode = PC;
         this.isFirst = FirstClick;
         InitializeComponent();
      }

      private void Header_MouseDown(object sender, MouseEventArgs e)
      {
         mov = 1;
         movX = e.X;
         movY = e.Y;
      }

      private void Btn_cancel_Click(object sender, EventArgs e)
      {
         if (MessageBox.Show("Do you really want to cancel? Unsaved data will be lost.", "Warning",
MessageBoxButtons.OKCancel) == DialogResult.OK)
         {
            PanelDispose();
            this.Close();
         }
      }

      private void Btn_cls_Click(object sender, EventArgs e)
      {
         if (MessageBox.Show("Do you really want to exit? Unsaved data will be lost.", "Warning",
MessageBoxButtons.OKCancel) == DialogResult.OK)
         {
            PanelDispose();
            this.Close();
```

```
            }
        }

        private int StringCount(string InputString)
        {
            int count = (InputString.Length - InputString.Replace("\n", "").Length);   // "\n" amount
            return count;
        }

        private void Modifying_Load(object sender, EventArgs e)
        {
            if (isFirst)
            {
                if (msi != "none")
                {
                    string installString = "";
                    string uninstallString = "";
                    if (mst == "none")
                    {
                        installString = "Execute-MSI -Action 'Install' -Path \"$dirFiles\\" + Path.GetFileName(msi)
+ "\" -private:$PackageName";
                    }
                    else
                        installString = "Execute-MSI -Action 'Install' -Path \"$dirFiles\\" + Path.GetFileName(msi)
+ "\" -Transform \"" + Path.GetFileName(mst) + "\" -private:$PackageName";
                    uninstallString = "Execute-MSI -Action 'Uninstall' -Path \"" + productCode + "\" -
private:$PackageName";
                    WrapperGen.Main.Install += installString + "\n\t\t";
                    WrapperGen.Main.Uninstall += uninstallString + "\n\t\t";
                }
                if (exe != "none")
                {
                    string installString = "Execute-Process -Path \"$dirFiles\\" + Path.GetFileName(exe) + "\" -
Parameters \"/S\"";
                    WrapperGen.Main.Install += installString + "\n\t\t";
                }
                if (appv != "none")
                {
                    string installString = "Add-AppvClientPackage -Path \"$dirFiles\\" + Path.GetFileName(appv)
+ "\"";
                    WrapperGen.Main.Install += installString + "\n\t\t";
                    installString = "Publish-AppvClientPackage -Name \"$PackageName\" -Global";
                    WrapperGen.Main.Install += installString + "\n\t\t";
                    string uninstallString = "Unpublish-AppvClientPackage -Name \"$PackageName\" -Global";
                    WrapperGen.Main.Uninstall += uninstallString + "\n\t\t";
                    uninstallString = "Remove-AppvClientPackage -Name \"$PackageName\"";
                    WrapperGen.Main.Uninstall += uninstallString + "\n\t\t";
                }
            }

            PreInstCount = StringCount(WrapperGen.Main.PreInstall);
            InstCount = StringCount(WrapperGen.Main.Install);
            PostInstCount = StringCount(WrapperGen.Main.PostInstall);
            PreUninstCount = StringCount(WrapperGen.Main.PreUninstall);
            UninstCount = StringCount(WrapperGen.Main.Uninstall);
            PostUninstCount = StringCount(WrapperGen.Main.PostUninstall);

            cb_template.SelectedIndex = 0;

            for (int i = 0; i < PreInstCount; i++)
            {
```

```
                    AddControls(0, WrapperGen.Main.PreInstall.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }
                for (int i = 0; i < InstCount; i++)
                {
                    AddControls(1, WrapperGen.Main.Install.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }
                for (int i = 0; i < PostInstCount; i++)
                {
                    AddControls(2, WrapperGen.Main.PostInstall.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }
                for (int i = 0; i < PreUninstCount; i++)
                {
                    AddControls(3, WrapperGen.Main.PreUninstall.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }
                for (int i = 0; i < UninstCount; i++)
                {
                    AddControls(4, WrapperGen.Main.Uninstall.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }
                for (int i = 0; i < PostUninstCount; i++)
                {
                    AddControls(5, WrapperGen.Main.PostUninstall.Split(new string[] { "\n\t\t" },
StringSplitOptions.None)[i]);
                }

                Font FRADMCNFont = new Font(Welcome.private_fonts.Families[1], 12);

                btn_add.Font = FRADMCNFont;
                btn_pshelp.Font = FRADMCNFont;
                btn_submit.Font = FRADMCNFont;
                btn_cancel.Font = FRADMCNFont;

                Screen screen = Screen.FromPoint(Cursor.Position);
                this.Location = screen.Bounds.Location;
                this.Left = screen.Bounds.Left + screen.Bounds.Width / 2 - this.Width / 2;
                this.Top = screen.Bounds.Top + screen.Bounds.Height / 2 - this.Height / 2;
            }

            private void AddControls(int Index, string Text)
            {
                mods++;
                TextBox tb = new TextBox
                {
                    Text = Text,
                    Size = new System.Drawing.Size(420, 25),
                    TabIndex = tab,
                    Name = "tbMod" + mods,
                    Location = new System.Drawing.Point(5, y + pnl_modify.AutoScrollPosition.Y)
                };
                tab++;
                pnl_modify.Controls.Add(tb);
                ComboBox cb = new ComboBox
                {
                    Name = "cbMod" + mods,
                    Size = new System.Drawing.Size(110, 25),
                    TabIndex = tab,
                    DropDownStyle = ComboBoxStyle.DropDownList,
                    Location = new System.Drawing.Point(431, y + pnl_modify.AutoScrollPosition.Y)
```

```
            };
            tab++;
            cb.Items.Add("Pre-Installation");
            cb.Items.Add("Installation");
            cb.Items.Add("Post-Installation");
            cb.Items.Add("Pre-Uninstallation");
            cb.Items.Add("Uninstallation");
            cb.Items.Add("Post-Uninstallation");
            cb.SelectedIndex = Index;
            pnl_modify.Controls.Add(cb);

            Label lb = new Label
            {
               Name = "lbMod" + mods,
               Text = "",
               Dock = DockStyle.None,
               Cursor = Cursors.Hand,
               BackColor = Color.Transparent,
               AutoSize = false,
               Size = new System.Drawing.Size(25, 25),
               Image = WrapperGenerator.Properties.Resources.remove25x25,
               Location = new System.Drawing.Point(545, y - 2 + pnl_modify.AutoScrollPosition.Y)
            };
            lb.Click += new EventHandler(Del_Click);
            pnl_modify.Controls.Add(lb);
            y += 30;
         }
         private void Btn_add_Click(object sender, EventArgs e)
         {
            string action_text="";
            int template = cb_template.SelectedIndex;
            cb_template.SelectedIndex = 0;
            switch (template)
            {
               case 0:
                  {
                     break;
                  }
               case 1:
                  {
                     action_text = "Copy-File -Path \"$dirFiles\\FileName.ext\" -Destination
\"$envWindir\\Filename.cfg\"";
                     break;
                  }
               case 2:
                  {
                     action_text = "Execute-MSI -Action 'Install' -Path \"$dirFiles\\MSIName.msi\" -Transform
\"$dirFiles\\MSTName.mst\" -private:$PackageName";
                     break;
                  }
               case 3:
                  {
                     action_text = "Execute-Process -Path \"$dirFiles\\Bin\\setup.exe\" -Parameters \"/S\" -
WindowStyle 'Hidden'";
                     break;
                  }
               case 4:
                  {
                     action_text = "New-Folder -Path \"$env:ProgramData\\AppName\"";
                     break;
                  }
               case 5:
```

```
                {
                    action_text = "Remove-File -Path \"$env:ProgramFiles\\AppName\\File.ext\"";
                    break;
                }
            case 6:
                {
                    action_text = "Remove-Folder -Path \"$envWinDir\\Downloaded Program Files\"";
                    break;
                }
            case 7:
                {
                    action_text = "Remove-RegistryKey -Key \"HKLM:SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\" -Name 'RunAppInstall'";
                    break;
                }
            case 8:
                {
                    action_text = "Set-RegistryKey -Key \"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Example\" -Value '(Default)'";
                    break;
                }
            case 9:
                {
                    action_text = "Start-ServiceAndDependencies -Name 'wuauserv'";
                    break;
                }
            case 10:
                {
                    action_text = "Stop-ServiceAndDependencies -Name 'wuauserv'";
                    break;
                }
            case 11:
                {
                    action_text = "Update-Desktop";
                    break;
                }
            default:
                {
                    action_text = "";
                    break;
                }
        }
        AddControls(-1, action_text);
    }

    private void Del_Click(object sender, EventArgs e)
    {
        y -= 30;
        int id = Convert.ToInt32(((Control)sender).Name.Replace("lbMod", "")); //Return ID
        string stb = "tbMod" + id;
        string scb = "cbMod" + id;
        string slb = "lbMod" + id;

        //Deleting current raw
        foreach (Control item in pnl_modify.Controls.OfType<TextBox>())
        {
            if (item.Name == stb)
            {
                pnl_modify.Controls.Remove(item);
                break;
            }
        }
```

```
    foreach (Control item in pnl_modify.Controls.OfType<ComboBox>())
    {
        if (item.Name == scb)
        {
            pnl_modify.Controls.Remove(item);
            break;
        }
    }
    foreach (Control item in pnl_modify.Controls.OfType<Label>())
    {
        if (item.Name == slb)
        {
            pnl_modify.Controls.Remove(item);
            break;
        }
    }
    //Moving up
    foreach (Control item in pnl_modify.Controls.OfType<TextBox>())
    {
        if (Convert.ToInt32(Regex.Replace(item.Name, @"[^\d]+", "")) > id)
            item.Location = new Point(5, item.Location.Y - 30);
    }
    foreach (Control item in pnl_modify.Controls.OfType<ComboBox>())
    {
        if (Convert.ToInt32(Regex.Replace(item.Name, @"[^\d]+", "")) > id)
            item.Location = new Point(431, item.Location.Y - 30);
    }
    foreach (Control item in pnl_modify.Controls.OfType<Label>())
    {
        if (Convert.ToInt32(Regex.Replace(item.Name, @"[^\d]+", "")) > id)
            item.Location = new Point(545, item.Location.Y - 30);
    }
}

private void Btn_submit_Click(object sender, EventArgs e)
{
    WrapperGen.Main.PreInstall = "";
    WrapperGen.Main.Install = "";
    WrapperGen.Main.PostInstall = "";
    WrapperGen.Main.PreUninstall = "";
    WrapperGen.Main.Uninstall = "";
    WrapperGen.Main.PostUninstall = "";
    foreach (Control item in pnl_modify.Controls.OfType<ComboBox>())
    {
        int id = Convert.ToInt32(Regex.Replace(item.Name, @"[^\d]+", ""));
        string tb = "tbMod" + id;
        string action = pnl_modify.Controls[tb].Text;
        if (item.Text== "Pre-Installation")
        {
            WrapperGen.Main.PreInstall += action + "\n\t\t";
        }
        else if (item.Text == "Installation")
        {
            WrapperGen.Main.Install += action + "\n\t\t";
        }
        else if (item.Text == "Post-Installation")
        {
            WrapperGen.Main.PostInstall += action + "\n\t\t";
        }
        else if (item.Text == "Pre-Uninstallation")
        {
            WrapperGen.Main.PreUninstall += action + "\n\t\t";
```

```
                }
                else if (item.Text == "Uninstallation")
                {
                    WrapperGen.Main.Uninstall += action + "\n\t\t";
                }
                else if (item.Text == "Post-Uninstallation")
                {
                    WrapperGen.Main.PostUninstall += action + "\n\t\t";
                }

            }
            PanelDispose();
            this.Close();
        }

        private void PanelDispose()
        {
            foreach (Control item in pnl_modify.Controls.OfType<TextBox>())
            {
                item.Dispose();
            }
            foreach (Control item in pnl_modify.Controls.OfType<ComboBox>())
            {
                item.Dispose();
            }
            foreach (Control item in pnl_modify.Controls.OfType<Label>())
            {
                item.Dispose();
            }
        }

        private void Btn_pshelp_Click(object sender, EventArgs e)
        {

System.Diagnostics.Process.Start("C:\\windows\\system32\\windowspowershell\\v1.0\\powershell.exe ",
strHelpCmd);
        }

        private void Header_MouseMove(object sender, MouseEventArgs e)
        {
            if (mov == 1)
            {
                this.SetDesktopLocation(MousePosition.X - movX, MousePosition.Y - movY);
            }
        }
        private void Header_MouseUp(object sender, MouseEventArgs e)
        {
            mov = 0;
        }
    }
}
```

ДОДАТОК Д

Інструкція користувача

Запуск програмного модуля «WrapperGenerator»

Для запуску програмного модуля:

- запустіть файл WrapperGenerator.exe з локального комп’ютера, або
- створіть ярлик на файл WrapperGenerator.exe з мережевого диску та запустіть його.

Робота з програмним модулем «WrapperGenerator»

Процес генерації інструментарію для розгортання пакетів ПЗ за допомогою програмного модуля «WrapperGenerator» складається з п’ятьох етапів:

- введення унікального ідентифікатора пакету;
- вибір файлів пакету та технології пакування;
- перевірка та редагування отриманих даних;
- модифікація скрипт-файлу за вимогами користувача;
- зберігання результатів генерації.

Введення SDC ID пакету

Після запуску модуля введіть ідентифікатор замовлення та натисніть кнопку “GO”.

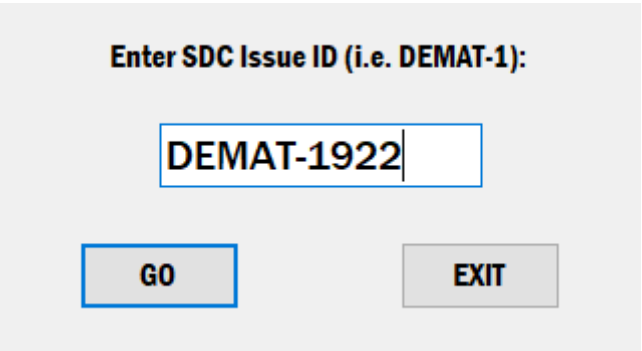


Рисунок Д.1 – Вікно введення ID

Вибір файлів пакету

Для вибору файлів пакету можна вставити розташування робочої директорії з буферу до текстового поля або натиснути на кнопку з зображенням теки для вибору розміщення файлів засобами ОС Windows.

Технологія пакування визначається автоматично на основі введеного ідентифікатора, проте її можна змінити обравши відповідний перемикач.

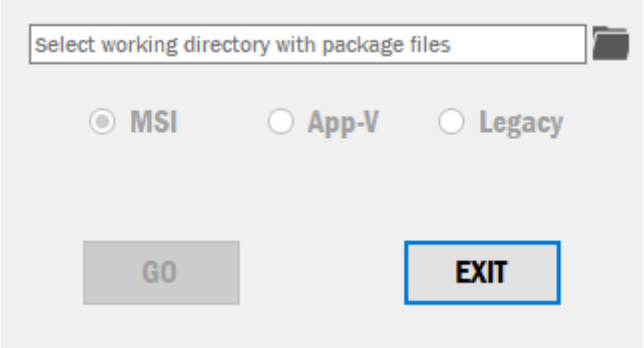


Рисунок Д.2 – Вікно вибору файлів та технології

В залежності від вказаної директорії необхідно вказати основні файли пакету для подальшої генерації скрипту. Це можна зробити за допомогою відповідних елементів списків або прапорців задежно від кількості та форматів файлів. Вказавши необхідні файли, натисніть на кнопку “GO”.

Перевірка та редагування отриманих даних

Після завантаження головного вікна програмного модуля необхідно перевірити отримані дані у текстових полях та модифікувати їх за необхідністю. За допомогою відповідних елементів керування можна налаштувати необхідність перезапуску комп’ютера для ps1 проектів та створення install/uninstall файлів для cmd проектів.

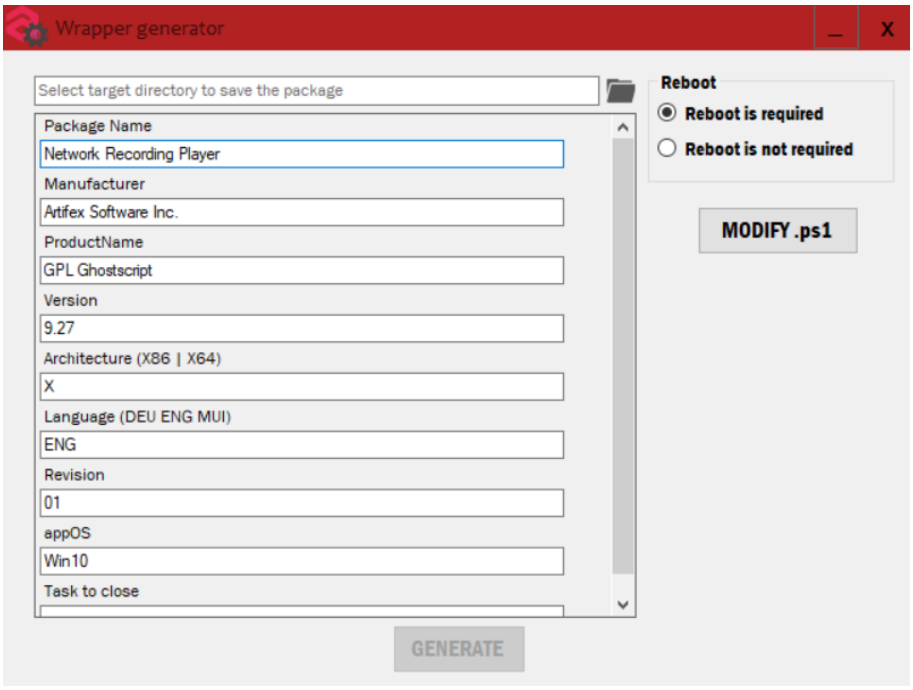


Рисунок Д.3 – Голове вікно програми

Модифікація скрипт-файлу

Для ps1 проектів є можливість налаштувати операційну систему кінцевого користувача натиснувши кнопку “MODIFY .ps1”. У вікні модифікації рядками відображаються скриптові дії мовою PowerShell, секція до якої вони відносяться та кнопка видалення радка. Для додавання порожнього рядка необхідно натиснути на кнопку “ADD”, якщо необхідно додати шаблон скрипту, попередньо оберіть відповідну дію зі списку поруч із кнопкою “ADD”.

Для видалення непотрібного рядку натисніть на кнопку з червоним крестом у відповідному рядку.

По натисканню кнопки “PS HELP” запускається довідка з функціями PowerShell для поточного проекту.

Для збереження даних нтисніть кнопку “SUBMIT”, для виходу – кнопку “CANCEL” або кнопку закриття вікна, після чого необхідно підтвердити вихід без збереження.

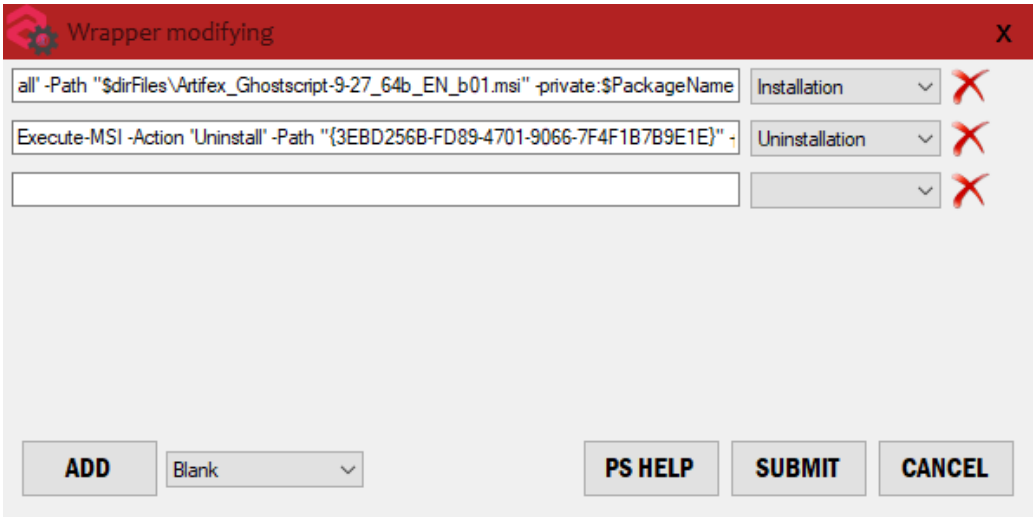


Рисунок Д.4 – Вікно модифікації скрипту

Зберігання результатів генерації

Для збереження готового пакету з усіма файлами та налаштуваннями, на головній формі необхідно зазначити цільову директорію таким же чином, як і під час вибору файлів пакету та натиснути на кнопку “GENERATE”, після чого по завершенню генерації автоматично відкриється вікно провідника з готовим пакетом.

Для виходу з програмного модуля натисніть кнопку закриття вікна та підтвердіть вихід.

Видалення програмного модуля «WrapperGenerator»

Для видалення програмного модуля необхідно:

- видалити файл WrapperGenerator.exe з локального комп’ютера, або
- видалити ярлик, якщо файл WrapperGenerator.exe виконувався з мережевого диску.