МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК

КУРСОВА РОБОТА

з дисципліни «Проектування інформаційних систем»

Тема «Система реєстрації та посадки на борт пасажирів аеропорту»

Оцінка _____ балів / _____

Члени комісії:

Виконала:
студентка 3 курсу
групи КС-33
Рузудженк С. Р.
Керівник:
доцент Гамзаєв Р.О.

Харків – 2019

ЗМІСТ

Перелік позначень та скорочень 3
Вступ 4
1 Постановка прикладної задачі 5
1.1 Загальна характеристика основних положень сучасної програмної інженерії 5
1.2 Постановка задачі предметної області 9
2 Реалізація етапа Rup / Inception 10
2.1 Розробка повної специфікації системних вимог 10
2.2 Розробка варіанта СА для вибраної предметної області 12
2.2.1 Діаграма прецедентів 12
2.2.2 Діаграма послідовності 14
2.2.3 Вибір цільового варіанту архітектури ПЗ 16
3 РЕАЛІЗАЦІЯ ЕТАПУ RUP / ELABORATION 18
3.1 Проектування компонентних програмних рішень для цільової СА з використанням патернів проектування 18
3.2 Розробка статичних UML - діаграм для логічного проектування КПР 18
3.3 Розробка динамічних UML – діаграм для логічного проектування КПР 20
3.4 Розробка динамічних UML – діаграм для фізичного проектування КПР 22
4 Формування повного комплекту проектної документації для програмної реалізації цільової системи 24
4.1 Метрики якості UML – діаграм 24
4.2 Визначення специфікації необхідних ресурсів, апаратно-програмної конфігурації (операційного середовища) і програмного інструментарію для реалізації проекту на подальших RUP-етапах (Construction and Transition) 25
Висновок 26
Список джерел інформації 27

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

КР – курсова робота;

UML (англ. Unified Modeling Language) – уніфікована мова моделювання;

КПР – компонентні програмні рішення;

RUP – Rational Unified Process;

ЖЦ – життєвий цикл;

ПЗ – програмне забезпечення;

ПрО – предметна область;

СА – системна архітектура.

ВСТУП

У першому розділі даної курсової роботи будуть розглянуті основні положення програмної інженерії, а саме : стандарти ISO / IEC 12207, ISO/IEC 9126, основи методології та технології RUP, ядра знань SWEBO, практики застосування GoF. Також буде сформована задача курсової роботи.

У другому розділі буде приведена реалізація етапу RUP INCEPTION (фаза початку), що включає визначення мети, доцільності та технічної можливості виконання проекту розробки ПС. Сформовано перелік бізнес правил для обраної ПрО. Приведено загальну характеристику UML, опис діаграм класів та послідовності та їх реалізація, опис вибраного архітектурного патерну.

У третьому розділі буде наведено реалізацію етапу RUP ELABORATION (фаза розвитку), що включає формування детального проекту ПС. Буде наведено опис обраних патернів для ПрО та їх реалізація, приведена реалізація діаграм класів, кінцевого автомату, діяльності та розгортання.

У четвертому розділі буде приведена характеристика метрик UML та визначення специфікації потрібних ресурсів, апаратно-програмного середовища і програмних інструментіів для реалізації проекту на подальших RUP-етапах.

1 ПОСТАНОВКА ПРИКЛАДНОЇ ЗАДАЧІ

1.1 Загальна характеристика основних положень сучасної програмної інженерії

У даному розділі будуть розглянуті основні принципи сучасної програмної інженерії, що використовуються у даній курсовій роботі:

1) стандарти ISO / IEC 12207, ISO/IEC 9126

2) основи методології та технології RUP;

3) ядра знань SWEBOK;

4) практик застосування еталонних системних архітектур (СА) і патернів проектування (ПП).

Стандарт ISO / IEC 12207 – стандарт, що описує процеси життєвого циклу програмного забезпечення. Стандарт визначає життєвий цикл програмних систем як структуру декомпозиції робіт.

Організація послідовності робіт – модель життєвого циклу. Сукупність моделей, процесів, технік і організації проектної групи задаються методологією.

Основні процеси життєвого циклу:

1) процес замовлення – визначає роботи замовника, тобто організації, яка набуває систему, програмний продукт або програмну послугу;

2) процес поставки – визначає роботи постачальника, тобто організації, яка постачає систему, програмний продукт або програмну послугу замовнику;

3) процес розробки – визначає роботи розробника, тобто організації, яка проектує і розробляє програмний продукт;

4) процес експлуатації – визначає роботи оператора, тобто організації, яка забезпечує експлуатаційне обслуговування обчислювальної системи в заданих умовах в інтересах користувачів;

5) процес супроводу – визначає роботи персоналу супроводу, тобто організації, яка надає послуги з супроводу програмного продукту, що складаються в контрольованому зміні програмного продукту з метою збереження його початкового стану і функціональних можливостей. Даний процес охоплює перенос і зняття з експлуатації програмного продукту.

ISO/IEC 9126 – це міжнародний стандарт, який визначає оціночні характеристики якості програмного забезпечення (далі ПЗ). Стандарт поділяється на 4 частини, що описують такі питання як:

1) модель якості;

2) зовнішні метрики якості; внутрішні метрики якості;

3) метрики якості у використанні.

Модель якості, встановлена в першій частині стандарту ISO 9126-1, класифікує якість ПО в 6-ти структурних наборах характеристик, які в свою чергу розширені підкатегоріями характеристик:

а) функціональність – набір атрибутів характеризує, відповідність функціональних можливостей ПО набору необхідної користувачем функціональності. Деталізується наступними під-характеристиками:

1) придатністю для застосування;

2) коректністю (правильністю, точністю);

3) здатністю до взаємодії (зокрема мережному);

4) захищеністю.

б) надійність – набір атрибутів, що відносяться до здатності ПО зберігати свій рівень якості функціонування в встановлених умовах за певний період часу. Деталізується наступними під-характеристиками:

1) рівнем завершеності (відсутності помилок);

2) стійкістю до дефектів;

3) відновлюваністю;

4) доступністю;

5) готовністю.

в) практичність – набір атрибутів, які відносяться до обсягу робіт, необхідних для виконання і індивідуальної оцінки такого виконання певних або передбачуваним колом користувачів. Деталізується наступними під-характеристиками:

1) зрозумілістю;

2) простотою використання.

г) ефективність – набір атрибутів, що відносяться до співвідношення між рівнем якості функціонування ПЗ і обсягом використовуваних ресурсів при встановлених умовах;

д) супровід – набір атрибутів, які відносяться до обсягу робіт, необхідних для проведення конкретних змін (модифікацій). Деталізується наступними під-характеристиками:

1) зручністю для аналізу;

2) змінністю;

3) стабільністю.

е) мобільність – набір атрибутів, що відносяться до здатності ПО бути перенесеним з одного оточення в інше. Деталізується наступними під-характеристиками:

1) простотою установки (інсталяції);

2) співіснуванням (відповідністю);

3) замінністю.

Ядро знань SWEBOK є основоположним науково-технічним документом, що відображає думки багатьох закордонних і вітчизняних фахівців в області програмної інженерії та узгоджується із сучасними регламентованими процесами життєвого циклу програмного забезпечення стандарту ISO/IEC 12207. У цьому ядрі знань містяться опис 10 областей, кожна з яких представлена згідно з прийнятим усіма учасниками створенням цього ядра загальної схеми опису, що включає визначення понятійного апарата, методів і засобів, а також інструментів підтримки інженерної діяльності. У кожній області описується визначений запас знань, що повинен бути практично використаний у відповідних процесах ЖЦ.

Rational Unified Process (RUP) — це методологія розробки програмного забезпечення, що спрямована на підтримку колективної розробки. Основним принципом RUP є принцип ітеративної розробки (iterative development), в рамках якої розробка ведеться у вигляді короткочасних мініпроектів фіксованої тривалості (наприклад, по 3-4 тижні), які називаються ітераціями (iteration). Кожна ітерація складається із власних фаз аналізу вимог, проектування, реалізації, тестування, інтеграції та створенням робочої версії програмної системи (ПС). Такий ітеративний цикл базується на постійному розширенні і

доповненні проекту ПС в процесі декількох ітерацій з періодичним зворотним зв’язком і адаптацією додаткових компонентів до існуючого ядра ПС, що розробляється.

У RUP визначено 9 технологічних процесів, які діляться на дві категорії — основні процеси та процеси підтримки.

До основних відносяться:

1) бізнес-аналіз;

2) управління вимогами;

3) аналіз і проектування;

4) реалізація;

5) тестування;

6) розгортання.

Допоміжні процеси включають:

1) управління проектом;

2) управління конфігурацією;

3) управління середовищем.

Патерни проектування – загальні принципи розробки програмного забезпечення, що складаються з найліпших підходів, використання яких поліпшує архітектуру програмного забезпечення та зменшує кількість можливих помилок. Усі принципи працюють правильно та призводять до очікуваного результату тільки за доречного їх використання. У інших випадках приводять до складності та неефективності програмного коду.

Найбільш низькорівневі та прості патерни — ідіоми. Вони не дуже універсальні, позаяк мають сенс лише в рамках однієї мови програмування.

Найбільш універсальні — архітектурні патерни, які можна реалізувати практично будь-якою мовою. Вони потрібні для проектування всієї програми, а не окремих її елементів.

Крім цього, патерни відрізняються і за призначенням. Серед них можна виокремити три основні групи:

1) Породжувальні – відповідають за зручне та безпечне створення нових об'єктів або навіть цілих сімейств об'єктів.

2) Структурні – відповідають за побудову зручних в підтримці ієрархій класів.
3) Поведінкові – вирішують завдання ефективної та безпечної взаємодії між об'єктами програми.

1.2 Постановка задачі предметної області

Необхідно розробити архітектурну структуру для вибраної предметної області, а саме «Система реєстрації та посадки на борт пасажирів аеропорту».

Розрізняється 3 типи реєстрації:

1) Звичайна реєстрація: з багажем, на звичайному реєстраційному пункті.
2) 2. Експрес-реєстрація: без багажу, на спеціальному пункті реєстрації.
3) Автоматична реєстрація: без багажу у автоматі, виконується пасажиром самостійно.

Саме тому для забезпечення необхідної функціональності ПС має працювати у 2 режимах:

1. Веб-додаток, у якому необхідно організувати адаптивну роботу програми у браузерах, розробити графічний інтерфейс для користувача та налаштувати коректну взаємодію з БД.
2. Десктопний-додаток, який необхідно синхронізувати з веб-застосунком для коректного зберігання інформації та опрацювання запитів користувачів.

2 РЕАЛІЗАЦІЯ ЕТАПА RUP / INCEPTION

2.1 Розробка повної специфікації системних вимог

1) Предметна область розробки ПЗ[1,2]

«Система реєстрації та посадки на борт пасажирів аеропорту».

2) Перелік основних прецедентів використання цієї ПС

2.1 Звичайна реєстрація пасажира, яку виконує представник реєстраційної служби аеропорту.

2.2 Експрес-реєстрація (без багажу), яка проводиться представником реєстраційної служби на окремому пункті реєстрації.

2.3 Автоматична реєстрація, що виконується пасажиром самостійно у автоматі.

3) Розробка розгорнутого сценарію для прецеденту.

(3.1) Зацікавлені особи прецеденту та їх вимоги:

- Пасажир: хоче швидко зареєструватися на посадку, зареєструвати багаж та перейти до наступного етапу.
- Представник реєстраційної служби: повинен швидко і безпомилково обробити персональні дані кожного пасажира, а також зареєструвати багаж, вага якого не перевищує допустимої норми.

(3.2) Користувач відповідного ПЗ, тобто основний актор цього прецеденту – це представник реєстраційної служби аеропорту, який проводить реєстрацію пасажирів за допомогою ПС, що має бути розроблена.

(3.3) Передумови прецеденту (pre - conditions):

- Представник реєстрації має пройти процедуру аутентифікації в системі.
- ПС має бути активною, а також повинен бути забезпечений доступ до бази даних.

(3.4) Основний успішний сценарій:

1. Представник реєстрації розпочинає реєстрацію за запитом нового пасажира.
2. Представник реєстрації обробляє особисті дані (ПІБ, дата народження, номер паспорту і т. п.) пасажира.
3. Представник реєстрації проводить процедуру реєстрації багажу.

4. Пасажир успішно зареєстрований на літак та переходить до наступного етапу

(3.5) Розширення основного сценарію або альтернативні потоки: проблеми з документами пасажира (закінчився термін дії):

1. ПС повідомляє представника реєстрації про помилку та припиняє реєстрацію даного пасажира.

2. Представник реєстрації переглядає інформацію про помилку.

3. Представник реєстрації повідомляє пасажира про проблему з документами.

4. Пасажир не зареєстрований на даний літак.

5. Пасажир покидає стійку реєстрації.

(3.6) Пост-умови прецеденту (post - conditions):

- Особисті дані (ПІБ, дата народження, номер паспорту і т. п.) пасажира збережені у системі.
- Особисті дані пасажира (ПІБ, дата народження, номер паспорту і т. п.) зафіксовані у БД системи.
- Пасажир отримав посадочний талон для посадки на літак.
- У ПС зафіксовано дані про багаж пасажира.
- Багаж відправлений на завантаження до літака.

(3.7) Спеціальні вимоги прецеденту:

- Необхідно забезпечити надійність та стійкість ПС, оскільки нею користуватимуться одночасно декілька представників реєстрації.
- Необхідно забезпечити можливість локалізації інтерфейсу ПС.

(3.8) Список необхідних технологій та додаткових пристроїв:

- Необхідно забезпечити кросплатформність ПС.
- Необхідно забезпечити наявність монітору для користувача ПС.
- Необхідно забезпечити закритий доступ до ПС (вхід до системи лише за допомогою відповідного особистого ключа користувача).
- Необхідно забезпечити безперервний доступ до мережі Internet.

Розробка розгорнутого сценарію для прецеденту (2.2).

(3.1) Зацікавлені особи прецеденту та їх вимоги:

- Пасажир: хоче швидко зареєструватися на посадку без багажу та перейти до наступного етапу.
- Представник реєстраційної служби: повинен швидко і безпомилково обробити персональні дані (ПІБ, дата народження і т. п.) кожного пасажира.

(3.2) Користувач відповідного ПЗ, тобто основний актор цього прецеденту – це представник реєстраційної служби аеропорту, який проводить реєстрацію пасажирів за допомогою ПС, що має бути розроблена.

(3.3) Передумови прецеденту (pre - conditions):

- Представник реєстрації має пройти процедуру аутентифікації в системі.
- ПС має бути активною, а також повинен бути забезпечений доступ до бази даних.

(3.4) Основний успішний сценарій:

1. Представник реєстрації розпочинає реєстрацію за запитом нового пасажира.
2. Представник реєстрації обробляє особисті дані (ПІБ, дата народження, номер паспорту і т. п.) пасажира.
3. Пасажир успішно зареєстрований на літак та переходить до наступного етапу.

(3.5) Розширення основного сценарію або альтернативні потоки: Аналогічно прецеденту 2.1.

(3.6) Пост-умови прецеденту (post - conditions):

Особисті дані (ПІБ, дата народження, номер паспорту і т. п.) пасажира збережені у системі.

- Особисті дані пасажира (ПІБ, дата народження, номер паспорту і т. п.) зафіксовані у БД системи.
- Пасажир отримав посадочний талон для посадки на літак.

(3.7) Спеціальні вимоги прецеденту: Аналогічно прецеденту 2.1.

(3.8) Список необхідних технологій та додаткових пристроїв: Аналогічно прецеденту 2.1.

Розробка розгорнутого сценарію для прецеденту (2.3).

(3.1) Зацікавлені особи прецеденту та їх вимоги:

Пасажир: хоче швидко зареєструватися на посадку та перейти до наступного етапу.

(3.2) Користувач відповідного ПЗ, тобто основний актор цього прецеденту – це пасажир, який самостійно проводить реєстрацію за допомогою ПС, що має бути розроблена.

(3.3) Передумови прецеденту (pre - conditions):

- ПС має бути активною, а також повинен бути забезпечений доступ до бази даних.

(3.4) Основний успішний сценарій:

1. Пасажир розпочинає реєстрацію.
2. Пасажир заносить особисті дані (ПІБ, дата народження, номер паспорту і т. п.) до системи.
3. Пасажир успішно зареєстрований на літак та переходить до наступного етапу.

(3.5) Розширення основного сценарію або альтернативні потоки.

У пасажира виникли проблеми з реєстрацією:

1. Пасажир розпочинає реєстрацію.
2. Пасажир заносить особисті дані (ПІБ, дата народження, номер паспорту і т. п.) до системи.
3. Виникають проблеми з реєстрацією.
4. Пасажир звертається до представника реєстрації на окремій стійці реєстрації для пасажирів без багажу.
5. Представник реєстрації розпочинає реєстрацію даного пасажира на літак.
6. Представник реєстрації заносить особисті дані (ПІБ, дата народження, номер паспорту і т. п.) до системи (*точка повернення в основний сценарій).*
7. Пасажир зареєстрований на даний літак та переходить до наступного етапу.

(3.6) Пост-умови прецеденту (post - conditions):

- Особисті дані (ПІБ, дата народження, номер паспорту і т. п.) пасажира збережені у системі.

- Особисті дані пасажира (ПІБ, дата народження, номер паспорту і т. п.) зафіксовані у БД системи.
- Пасажир отримав посадочний талон для посадки на літак.

(3.7) Спеціальні вимоги прецеденту: Аналогічно прецеденту 2.1.

(3.8) Список необхідних технологій та додаткових пристроїв: Аналогічно прецеденту 2.1.

2.2 Розробка варіанта СА для вибраної предметної області

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю[3,4]. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але на основі UML-моделей можлива генерація коду.

2.2.1 Діаграма прецедентів

Діаграма прецедентів (англ. Use case diagram) – є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами[3]. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

1) асоціації (англ. association relationship);

2) включення (англ. include relationship);

3) розширення (англ. extend relationship);

4) узагальнення (англ. generalization relationship).

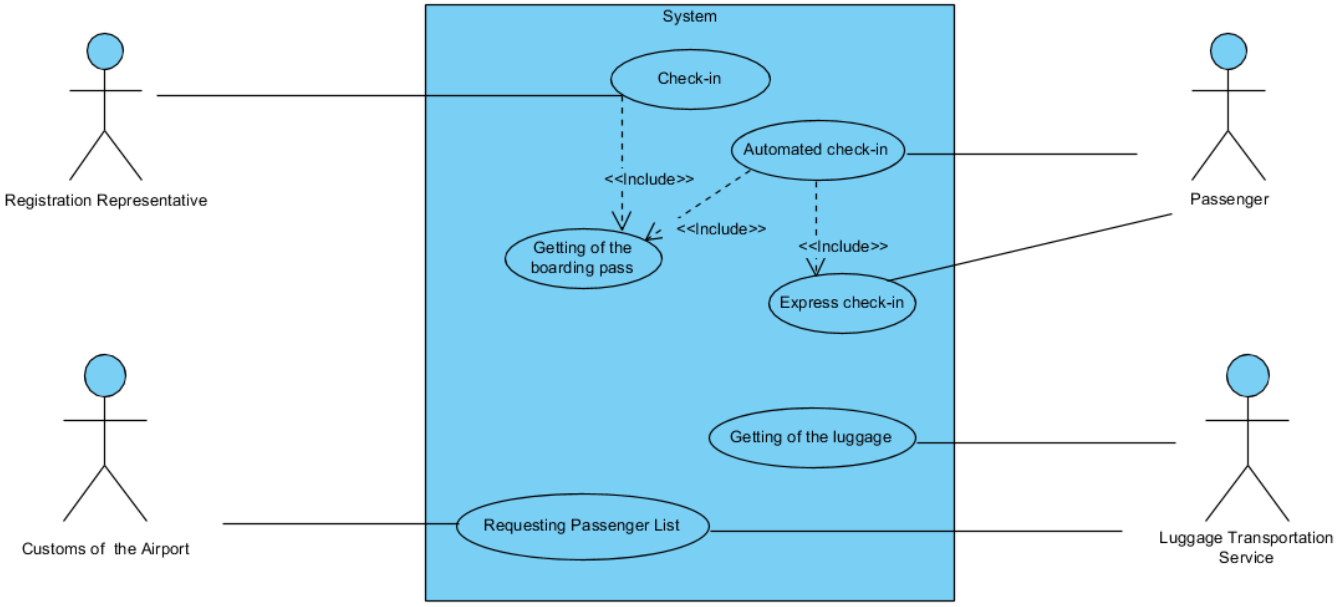

Рисунок 1 – Діаграма прецедентів для вибраної предметної області

На рисунку 1 зображені актори (Registration Representative, Passenger, Customs of the Airport, Luggage Transportation Service) та перелік дій, які вони можуть виконувати. На діаграмі бачимо, що пасажир обирає вид реєстрації, в свою чергу представник реєстрації аеропорту реєструє/пасажир реєструється самостійно, залежно від його вибору. Якщо пасажира реєструє представник, він видає посадочний квиток. У свою чергу, після реєстрації пасажира, сервіс транспортування багажу та митниця аеропорту виконують необхідні дії (отримання у системі інформацію про багаж та списку пасажирів відповідно).

2.2.2 Діаграма послідовності

Діаграма послідовності – діаграма, на якій для деякого набору об'єктів на єдиній тимчасової осі показаний життєвий цикл будь-якого певного об'єкта (створення-діяльність-знищення якоїсь сутності) і взаємодія акторів (дійових осіб) ІС в рамках певного прецеденту.

Для користувача сценарій має вигляд такої послідовності кроків:

1)пошук відповідного квитка;

2) додання нової інформації про пасажира;

3) підтвердження інформації;

4)передання інформації відповідним підрозділам.

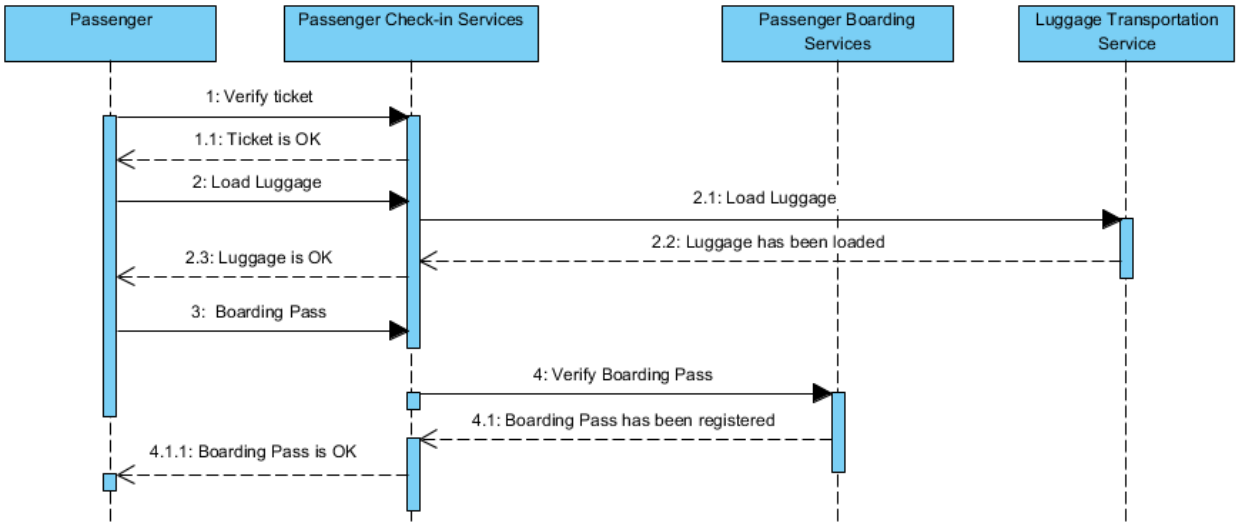

Рисунок 2 – Діаграма послідовності

На рисунку 2 зображена діаграма, що демонструє один з сценаріїв взаємодії об’єктів (реєстрація пасажира на літак), впорядкованих за часом їх появи.

2.2.3 Вибір цільового варіанту архітектури ПЗ

У якості архітектури ПЗ для вибраної предметної області був обраний патерн Layered pattern (багаторівневий шаблон). Даний шаблон дозволить забезпечити добру структурованість програми, оскільки її можна розкласти на групи деяких задач, що знаходяться на певних рівнях абстракції. Кожен рівень обслуговує наступний, більш високий. Найчастіше зустрічаються 4 основні пласти: пласт представлення, пласт сервісу, пласт бізнес-логіки та пласт зберігання даних. До переваг даного шаблону можна віднести: одним нижнім пластом можуть бути використані різні пласти більш високого рівня; пласти значно спрощують стандартизацію; зміни, внесені всередині певного пласту, не несуть за собою змін інших пластів. З недоліків можна виділити неуніверсальність даного шаблону та можливість пропуску деяких пластів у деяких випадках.

3 РЕАЛІЗАЦІЯ ЕТАПУ RUP / ELABORATION

3.1 Проектування компонентних програмних рішень для цільової СА з використанням патернів проектування

Під час проектування компонентних програмних рішень для цільової СА було обрано два патерна проектування:

1) Посередник (англ. Mediator) – поведінковий патерн проектування, що дає змогу зменшити зв’язаність великої кількості класів між собою, завдяки переміщенню цих зв’язків до одного класу-посередника. Посередник визначає інтерфейс для обміну інформацією з компонентами. Зазвичай достатньо одного методу, щоби повідомляти посередника про події, що відбулися в компонентах. У параметрах цього методу можна передавати деталі події: посилання на компонент, в якому вона відбулася, та будь-які інші дані.

2) Будівник (англ. Builder) – твірний патерн проектування, що дозволяє полегшити процес створення об’єкту класу, шляхом використання внутрішнього класу, що відповідає за створення та бере на себе всю відповідальність. Це дозволяє в результаті одного та того ж конструювання отримувати різні представлення.

Обрані патерни для вибраної предметної області дозволять забезпечити коректну роботу, більш зрозумілу структуру ПС, а також, за несправності певного компоненту системи, локально усунути проблему. Реалізація патернів представлена на діаграмі класів, що наведена на рисунку 4.

3.2 Розробка статичних UML - діаграм для логічного проектування КПР

Діаграма класів (англ. Class Diagram) – структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними. Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування.

Для вибраної предметної області була розроблена діаграма класів, що описує основні сутності, відносини та операції над ними. У нас існує головна сутність, що

породжує інші сутності в вигляді Main класу, що відповідає за створення графічного інтерфейсу та підключення до бази даних.

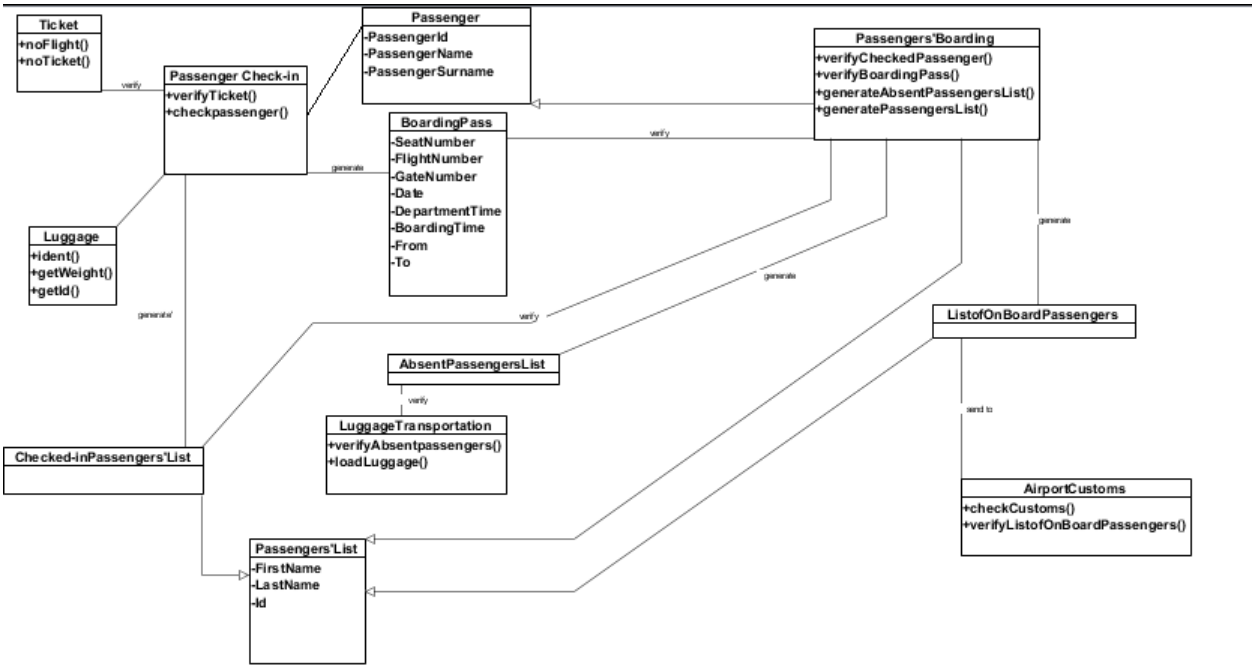

Рисунок 4 – Діаграма класів

Патерн будівельник був реалізований для клієнта в вигляді методу Build, що створює необхідний об’єкт клієнта в визначений час. Патерн Посередник реалізований через клас Passengers’Boarding, що являю собою проміжну сутність для взаємодії усіх компонентів системи.

3.3 Розробка динамічних UML – діаграм для логічного проектування КПР

Діаграми кінцевого автомату (англ. State Machine Diagrams) – використовуються для подання стану системи або частини системи при кінцевих моментах часу. Це поведінкова діаграма, і вона являє поведінку з використанням кінцевих переходів станів. Діаграми станів також називаються машинами станів і діаграмами станів. Головне призначення цієї діаграми – описати можливі послідовності станів і переходів, які в сукупності

характеризують поведінку певного елементу UML-моделі ПС протягом його життєвого циклу.

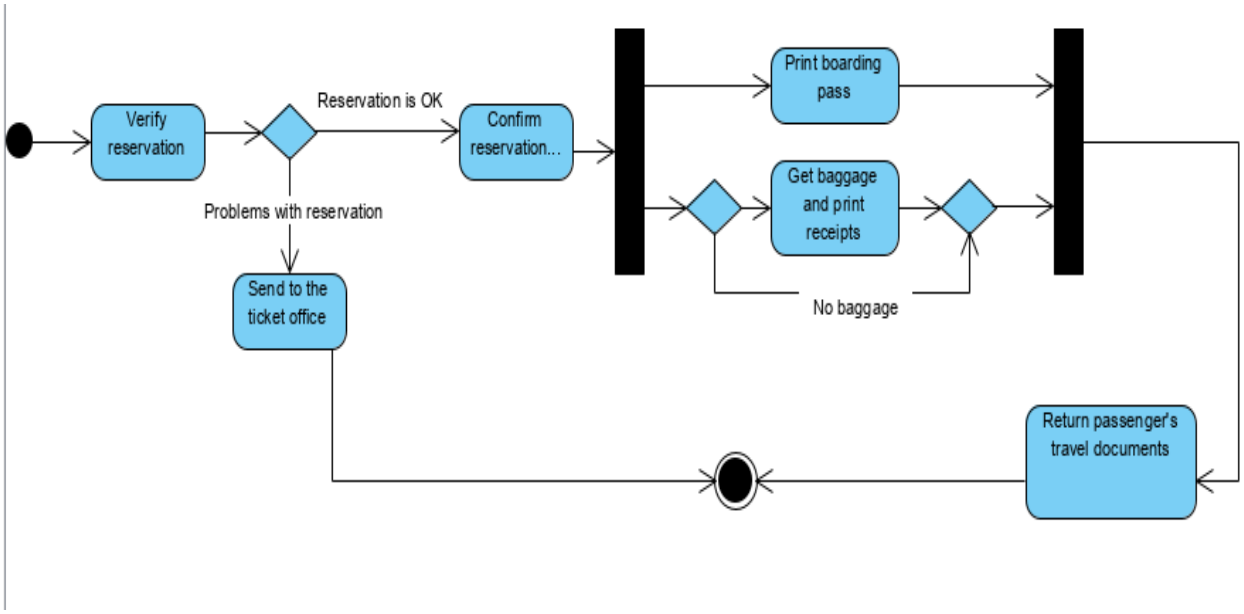

Рисунок 5 – Діаграма кінцевого автомату

Діаграма для вибраної предметної області наведена на рисунку 5, де існують декілька станів ПС, зокрема «Підтвердження резервації», «Ідентифікація пасажира», «Запит на видачу талону та його обробка».

Діаграма діяльності (англ. Activity Diagrams) – візуальне представлення графу діяльності. Кожен стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує лише при завершенні цієї операції в попередньому стані.

Діаграми активності будуються з обмеженої кількості фігур, з’єднаних стрілочками. Найважливіші типи фігур:

1) скруглені прямокутники позначають дії;

2) ромби позначають рішення;

3) риски позначають початок (розподіл) чи кінець (об’єднання) паралельних активностей;

4) чорний кружок позначає старт (початковий стан) процесу;

5) чорний кружок в колі позначає кінець (кінцевий стан).

Стрілки ведуть від старту до кінця і позначають порядок в якому відбуваються активності. Діаграма діяльності наведена на рисунку 6.

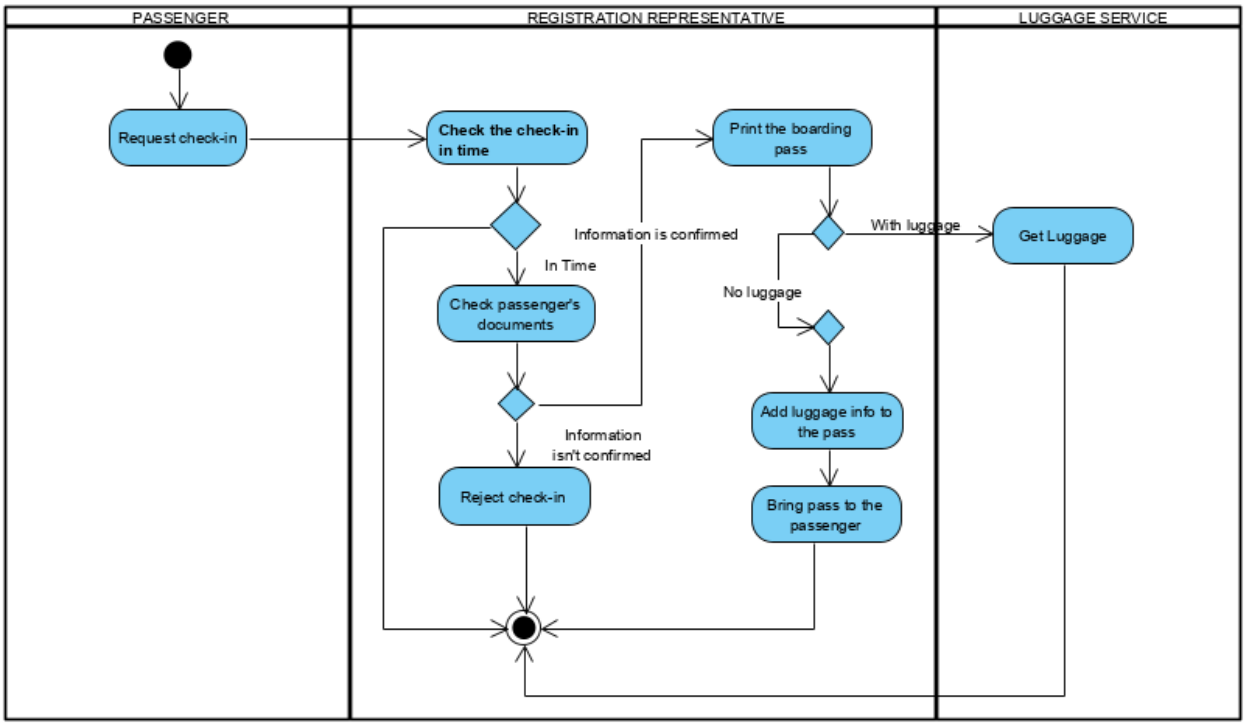

Рисунок 6 – Діаграма діяльності

3.4 Розробка динамічних UML – діаграм для фізичного проектування КПР

Діаграма розгортання (англ. deployment diagram) — діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що

виконуються на цих вузлах. Діаграму розгортання системи можна описати наступним чином. Працівник взаємодіє із системою шляхом використання програми управління магазином. Програма відправляє запити та отримує відповіді від web-сервера. Сервер надає користувацький інтерфейс, а також – використовує інтерфейс бази даних для доступу до даних, їх зміни, видалення, модифікації. Діаграма для предметної області наведена на рисунку 7. Діаграма розгортання містить графічні зображення процесорів, пристроїв, процесів і зв'язків між ними. На відміну від діаграм логічного уявлення, діаграма розгортання є єдиною для системи в цілому, оскільки повинна цілком відображати особливості її реалізації.

Розробка діаграми розгортання, як правило, є останнім етапом специфікації моделі програмної системи.

При розробці діаграми розгортання переслідують наступні цілі:

- визначити розподіл компонентів системи по її фізичних вузлах;

- показати фізичні зв'язки між всіма вузлами реалізації системи на етапі її виконання;

- виявити вузькі місця системи і реконфигурировать її топологію для досягнення необхідної продуктивності.

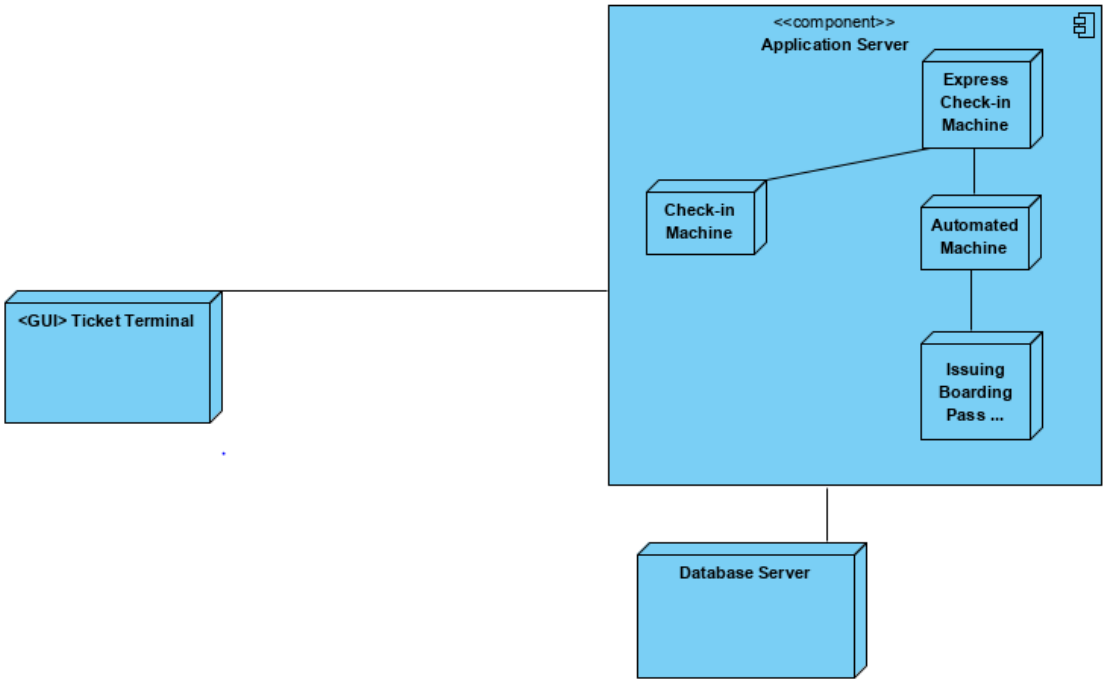

Рисунок 7 – Діаграма розгортання

4 ФОРМУВАННЯ ПОВНОГО КОМПЛЕКТУ ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ ДЛЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЦІЛЬОВОЇ СИСТЕМИ

4.1 Метрики якості UML – діаграм

Створення віртуальних моделей зазвичай займає значну частину ресурсів при розробці об'єктно – орієнтованих програмних систем. Саме тому необхідно оцінювати якість цих моделей, зіставивши якості числову оцінку. Рішення даної задачі потребує введення спеціального метричного апарату. Такий апарат розвиває ідеї класичного оцінювання складних програмних систем, заснованого на метриках складності, зв'язаності та зчеплення. Разом з тим він враховує специфічні особливості об'єктно – орієнтованих рішень.

Об'єктно – орієнтовані метрики вводять з ціллю:

1) покращити розуміння якості продукту;

2) оцінити ефективність процесу створення;

3) покращити якість роботи на етапі проектування.

Для будь-якого інженерного продукту метрики повинні орієнтуватися на його унікальні характеристики. З точки зору метрик виділяють п'ять характеристик об'єктно – орієнтованих систем: локалізацію, інкапсуляцію, інформативну закритість, успадкування та способи абстрагування об'єктів. Наведені характеристики здійснюють максимальний вплив на об'єктно – орієнтовані метрики.

4.2 Визначення специфікації необхідних ресурсів, апаратно-програмної конфігурації (операційного середовища) і програмного інструментарію для реалізації проекту на подальших RUP-етапах (Construction and Transition)

Програмна система має бути адаптивною до усіх браузерів, також має запускатися під найвідомішими ОС, такими як LINUX, Windows, iOS.

Для розробки програми е вигляді веб-додатку, з боку клієнту використовувати JS, HTML, а також фреймворк Angular. Сервер має бути реалізованих на Java. Для більшої економічності застосунку необхідно реалізувати кросплатформність даної ПС.

Необхідні ресурси:

1) сервер, що має швидко реагувати на запити та забезпечувати стабільну роботу.

2) база даних, у якій має бути доречно організований спосіб збереження даних.

ВИСНОВОК

У першому розділі даної курсової роботи було розглянуто основні положення програмної інженерії, а саме : стандарти ISO / IEC 12207, ISO/IEC 9126, основи методології та технології RUP, ядра знань SWEBO, практики застосування GoF. Також буде сформована задача курсової роботи.

У другому розділі було приведено реалізацію етапу RUP INCEPTION (фаза початку), що включає визначення мети, доцільності та технічної можливості виконання проекту розробки ПС. Сформовано перелік бізнес правил для обраної ПрО. Приведено загальну характеристику UML, опис діаграм класів та послідовності та їх реалізація, опис вибраного архітектурного патерну.

У третьому розділі було наведено реалізацію етапу RUP ELABORATION (фаза розвитку), що включає формування детального проекту ПС. Було наведено опис обраних патернів для ПрО та їх реалізація, приведена реалізація діаграм класів, кінцевого автомату, діяльності та розгортання.

У четвертому розділі була приведена характеристика метрик UML та визначення специфікації потрібних ресурсів, апаратно-програмного середовища і програмних інструментіів для реалізації проекту на подальших RUP-етапах.

У ході виконання курсової роботи було розглянуто необхідну інформацію для реалізації поставленої задачі, розроблено бізнес вимоги, реалізовано перші два етапи RUP, з всіма необхідними діаграмами та розроблено додаткові вимоги для подальшої розробки, використано два патерни проектування та архітектурний патерн.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Кулямин В. В. Технологии программирования. Компонентный подход // Спец. курс на фак-те вычисл. техн. и кибернетики МГУ им. М.В. Ломоносова – М., 2006 . Електр. ресурс

http://panda.ispras.ru/~RedVerst/RedVerst/Lectures%20and%20training%20courses/Software%20Development%20Technologies/All.pdf

2. Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ. - М.: Издательский дом "Вильямс", 2002. - 432 с..

3. Вигерс, К. Разработка требований к программному обеспечению/Пер. с англ. — М.:Издательско-торговый дом «Русская Редакция», 2004. — 576с.: ил.

4. Fowler M. UML Distilled. A Brief Guide to the Standard Object Modeling Language / Martin Fowler. // Addison-Wesely. – 2004. – №3. – С. 175.

5. Стандарт UML 2.0 // http://www.omg.org/spec/UML/2.0/

6. Ларман, К. Применение UML 2.0 и шаблонов проектирования: практическое руководство - : Пер. с англ. – М. ООО «И.Д. Вильямс», 2009.