Министерство образования и науки

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**НАПИСАНИЕ СПЕЦИФИКАЦИЙ И ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ**

Лабораторная работа №2 по дисциплине

«Технология разработки программного обеспечения»

учебное пособие В.Т. Калайда, В.В. Романенко

«Технология разработки программного обеспечения»

Выполнил студент

специальности 09.03.01

Бубенщиков Олег Юрьевич

20.12.2018

г. Среднеуральск 2018

Оглавление

[1 Спецификации 3](#_Toc6550543)

[1.1 Внешняя спецификация 3](#_Toc6550544)

[1.2 Внутренняя спецификация 6](#_Toc6550545)

[2 Тестирование 12](#_Toc6550546)

[2.1 Общие принципы тестирования 12](#_Toc6550547)

[2.2 Организация испытаний программных изделий 12](#_Toc6550548)

[2.3 Виды испытаний программного изделия. Стадии испытаний 12](#_Toc6550549)

[2.4 Режимы испытаний программ 12](#_Toc6550550)

[2.5 Категории испытания программного изделия 12](#_Toc6550551)

[2.6 Технология тестирования, классы эквивалентности 13](#_Toc6550552)

[2.7 Построение тестов 17](#_Toc6550553)

[3 РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА 19](#_Toc6550554)

[3.1 Общие сведения о программе 19](#_Toc6550555)

[3.2 Структура программы 21](#_Toc6550556)

[3.3 Настройка программы 21](#_Toc6550557)

[3.4. Проверка программы 21](#_Toc6550558)

[3.5 Дополнительные возможности 21](#_Toc6550559)

[3.6 Сообщения системному программисту 22](#_Toc6550560)

# 1 Спецификации

## 1.1 Внешняя спецификация

Входные данные:

* Тип АТС — выбор нужного типа обрабатываемой станции из выпадающего меню.
* Тип Номера — выбор нужного типа номера обработки из выпадающего меню.
* Папка выгрузки — поле для указания пути с указанием директории в ОС до места выгрузки обработанных данных. Тип поля — строковый.
* База данных АТС — поле для указания пути до обрабатываемого файла исходных данных АТС. Тип поля — строковый
* Шаблон категорий АОН — поле для указания пути до расположения файла с правилами преобразования категорий АОН для указанной Базы данных АТС. Тип поля — строковый
* Шаблон Услуг - поле для указания пути до расположения файла с правилами преобразования услуг номера для указанной Базы данных АТС. Тип поля — строковый
* Шаблон IMS - поле для указания пути до расположения файла с правилами конфигурации конкретного узла IMS для конечных данных абонентов. Тип поля — строковый

Выходные данные:

* Файл web\_portal.txt — содержит список сконвертированных номеров в формате, определённом для последующего импорта на узел VIMS. Тип файла — текстовый формат.
* Лог- файлы: migrate\_debug.log и migrate\_info.log — содержат информацию о процессе работы программы. Тип файлов — текстовый формат.

main\_win: procedure;

// объявить структуру, содержащую конфигурационные данные

declare 1 config; // словарь атрибутов конфигурации

2 node: string [250]; // тип АТС

2 type\_dn: string [250]; // тип обрабатываемых номеров

2 dest\_dir: string [250]; // директория выгрузки данных

2 sourse\_file\_db: string [250]; // имя файла с БД

2 mapping\_category: string [250]; // имя файла с шаблоном категорий

2 mapping\_service: string [250]; // имя файла с шаблоном услуг

2 mapping\_ims: string [250]; // имя файла с шаблоним IMS

// Запустить вывод логирования работы на экран

call show\_log;

// обработка кнопок формы

do case (кнопка)

//do выполнить конвертирование;

"кнопка RUN": call run\_migrate (config); // запустить скрипт миграции

//do получить конфигурационные данные;

"кнопка Config": call config\_window(config); // открыть графическую форму настройки конфигурации

"кнопка Quit": call Exit; // завершить работу приложения

end do;

end main\_win;

## 1.2 Внутренняя спецификация

// Вывод логирования действий программы на экран

show\_log: procedure;

begin

declare queue; // структура-очередь

// запустить чтение лог-файл в бесконечном цикле

call read\_log;

// запустить отображение данных из очереди на экран в бесконечном цикле

call pull\_display\_from\_log;

end;

end show\_log;

// Процедура чтения лог-файл в бесконечном цикле

read\_log: procedure;

// Процедура забирает из очереди данные и выводит на экран в бесконечном цикле

pull\_display\_from\_log: procedure;

// Процедура запускает скрипт миграции абонентов

run\_migrate (config): procedure

begin

// создается объект-хранилище для исходной базы номеров АТС

declare node\_repo: type (BaseRepo);

// создаётся объект-хранилище для хранения обработанной базы номеров IMS

declare main\_repo: type(ImsSubsRepo);

// выполнить обработку исходной БД

call node\_repo(config);

// выполнить конвертирование номеров по заданному алгоритму

call main\_interactor(main\_repo, node\_repo, config);

// выполнить запись результатов в файл

call write\_wp(main\_repo, config);

end;

end run\_migrate;

// Процедура выполняет запись обработанных данных о номерах в файл

write\_wp: procedure (main\_repo, config);

begin

declare wp\_list; // массив для хранения списка номеров для обработки

declare view\_wp; // массив для хранения обработанного списка номеров

// получить из репозитория список номеров по критерию

wp\_list = get(получить из main\_repo список номеров по критерию config.type\_dn)

// сформировать формат представления данных для записи в файл

view\_wp = get(call функция\_представления\_данных\_для\_записи\_в\_текстовый\_файла (wp\_list))

// записать конечные данные в файл

call write\_file (config.dest\_dir, 'имя\_файла', view\_wp);

end;

end write\_wp

// Процедура графического окна формы конфигурации

config\_window: procedure (config);

begin

declare field\_select\_node; // графический элемент-селектор

declare field\_select\_type\_number; // графический элемент-селектор

declare field\_1; // графический элемент-поле ввода

declare field\_2; // графический элемент-поле ввода

declare field\_3; // графический элемент-поле ввода

declare field\_4; // графический элемент-поле ввода

declare field\_5; // графический элемент-поле ввода

do case (кнопка)

"Кнопка OK": call return\_config(config); // возврать конфиг и закрыть окно

"Кнопка Х": call exit; // закрыть окно конфигурации

end do;

// возврать конфиг и закрыть окно

return\_config: procedure (config);

begin

if check\_fields() есть истинна; // проверка полей ввода

then

call add\_to\_config(config); // добавить атрибуты в словарь конфигурации

call Exit; // закрыть окно конфигурации

end;

end return\_config;

// Процедура проверки полей ввода

check\_fields: procedure RETURNS (BOOL);

begin

// поле директория выгрузки данных

get (field\_1)

if (field\_1 < 0 or field\_1 > 250 or field\_1 не найдена указанная директория);

then

call show\_alarm (field\_1);

return False; // неуспешная проверка

// имя файла с БД

get (field\_2)

if (field\_2 < 0 or field\_2 > 250 or field\_2 не найден указанный файл);

then

call show\_alarm (field\_2);

return False; // неуспешная проверка

// имя файла с шаблоном категорий

get (field\_3)

if (field\_3 < 0 or field\_3 > 250 or field\_3 не найден указанный файл);

then

call show\_alarm (field\_3);

return False; // неуспешная проверка

// имя файла с шаблоном услуг

get (field\_4)

if (field\_4 < 0 or field\_4 > 250 or field\_4 не найден указанный файл);

then

call show\_alarm (field\_4);

return False; // неуспешная проверка

// имя файла с шаблоним IMS

get (field\_5)

if (field\_5 < 0 or field\_5 > 250 or field\_5 не найден указанный файл);

then

call show\_alarm (field\_5);

return False; // неуспешная проверка

return True; // возвратить истину

end;

end check\_fields;

// Добавить атрибуты в словарь конфигурации

add\_to\_config: procedure;

begin

config.node = get(field\_select\_node);

config.type\_dn = get(field\_select\_type\_number);

config.dest\_dir = get(field\_1);

config.sourse\_file\_db = get(field\_2);

config.mapping\_category = get(field\_3);

config.mapping\_service = get(field\_4);

config.mapping\_ims = get(field\_5);

end;

end add\_to\_config;

// отображает графическое окно с информацией об ошибке

show\_alarm: procedure (description);

degin

показать графическое окно с информацией об ошибке

end;

end show\_alarm;

end;

end config\_window;

// Процедура завершения работы приложения

Exit: procedure;

# 2 Тестирование

## 2.1 Общие принципы тестирования

Для проведения тестирования программы «MigrateIMS» выполняется системное тестирование

## 2.2 Организация испытаний программных изделий

Организация испытаний программного изделия преследует цель установления факта наличия/отсутствия ошибок и расхождения между истинными свойствами программного изделия и его спецификациями.

## 2.3 Виды испытаний программного изделия. Стадии испытаний

Для проверки правильности работы программы выбран класс B (тестирование после разработки).

## 2.4 Режимы испытаний программ

Тестирование будет выполняться в режиме III (выполняется разработчиком).

## 2.5 Категории испытания программного изделия

Категория испытаний программного изделия – Аттестация.

## 2.6 Технология тестирования, классы эквивалентности

**2.6.1 Варианты использования**

**Тестовый сценарий использования:** Выполнить конвертирование.

Основное действующее лицо: Оператор, Конвертер

Цель: Получить исходные данные номеров в новом формате.

Уровень: Обобщенный.

Область действия: Программа MigrateIMS.

Основной сценарий:

1. Оператор запускает главный скрипт исполнения.
2. Конвертер инициирует открытие главного графического окна.
3. Оператор инициирует настройку конфигурации
4. Конвертер инициирует открытие модального окна с настройками конфигурации.
5. Оператор определяет необходимые настройки и данные и подтверждает ввод.
6. Конвертер проводит валидацию введённых данных в поля ввода, закрывает модальное окно и делает активным основное.
7. Оператор инициирует запуск процесса конвертирования.
8. Конвертер выполняет обработку данных по выбранному сценарию.
9. Конвертер формирует выходные данные.
10. Конвертер информирует оператора о завершении обработки данных.

Расширения:

5.1 Оператор выполняет отмену настройки конфигурации и закрывает окно без подтверждения кнопкой ОК.

5.1.1 Конвертер применяет текущие значения

5.1.2 Конвертер закрывает окно конфигурации и возвращается в главное окно.

5.2 Оператор применяет текущую конфигурацию без изменений.

5.2.1 Конвертер применяет текущие значения

5.2.2 Конвертер закрывает окно конфигурации и возвращается в главное окно.

6.1 Валидация входных данных не прошла

Конвертер отображает модальное окно с указанием ошибки валидации

Пользователь подтверждает прочтение

Конвертер закрывает окно с сообщением об ошибке и возвращается в

окно конфигурации.

8.1 Исходная БД не содержит указанный для обработки тип номера

8.1.1 Конвертер выводит сообщение об отсутствии номеров для запрашиваемого типа в исходной БД.

8.2 Конфигурационные файлы имеют неверный формат json

8.2.1 Конвертер сообщает об ошибке и прекращает обработку данных

8.3 Исходный файл базы номеров имеет неожидаемый формат для указаного типа АТС.

8.3.1 Конвертор сообщает об ошибке и прекращает обработку данных

9.1 Конвертер не может сохранить выходные данные в файл.

9.1.1 ошибка перехватывается встроенными обработчиками исключений и пишется в лог с указанием уровня срочности.

Классы эквивалентности для проверки форм ввода конфигурационных данных в тестовом сценарии п.п. 6, 6.1 описаны в таблице 1.1.

Таблица № 1. Классы эквивалентности формы ввода

|  |  |  |
| --- | --- | --- |
| Входные условия | Классы эквивалентности | |
| Правильные | Неправильные |
| Папка выгрузки файла  Строка длинной от 1 до 250 знаков | Нижняя граница один знак(1)  Верхняя граница 250 знаков(2) | Пустая строка(3)  Больше 250 знаков (4) |
| Папка выгрузки файла | Папка по указанному пути существует(5) | Папка по указанному пути не существует(6) |
| База данных АТС  Строка длинной от 1 до 250 знаков | Нижняя граница один знак(7)  Верхняя граница 250 знаков(8) | Пустое поле(9)  Больше 250 знаков (10) |
| База данных АТС | файл по указанному пути существует(11) | файл по указанному пути не существует(12) |
| Шаблон категорий АОН  Строка длинной от 1 до 250 знаков | Нижняя граница один знак(13)  Верхняя граница 250 знаков(14) | Пустое поле(15)  Больше 250 знаков (16) |

Окончание табл. 1.

|  |  |  |
| --- | --- | --- |
| Входные условия | Классы эквивалентности | |
| Правильные | Неправильные |
| Шаблон категорий АОН | файл по указанному пути существует(17) | файл по указанному пути не существует(18) |
| Шаблон Услуг  Строка длинной от 1 до 250 знаков | Нижняя граница 1 знак(19)  Верхняя граница 250 знаков(21) | Пустое поле(20)  Больше 250 знаков (22) |
| Шаблон Услуг | файл по указанному пути существует(23) | файл по указанному пути не существует(24) |
| Шаблон IMS  Строка длинной от 1 до 250 знаков | Нижняя граница 1 знак(25)  Верхняя граница 250 знаков(26) | Пустое поле(27)  Больше 250 знаков (28) |
| Шаблон IMS | файл по указанному пути существует(29) | файл по указанному пути не существует(30) |

## 2.7 Построение тестов

На основе тестового сценария определены следующие тесты:

1. Основной тестовый сценарий: Тип АТС: DEMO; Тип номера: SIP; положительные классы эквивалентности Таблица 1.1 подклассы: 1,5,7,11,13, 17,19,23,25,29

2. Основной тестовый сценарий: Тип АТС: DEMO; Тип номера: SIP; положительные классы эквивалентности Таблица 1.1 подклассы: 2,8,14, 21, 26

3. Альтернативный тестовый сценарий п.5.1: без изменения текущей конфигурации.

4 Альтернативный тестовый сценарий п.5.2: без изменения текущей конфигурации.

5 Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:3

6 Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:4

7 Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:6

8 Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:9

9 Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:10

10. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:12

11. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:15

12. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:16

13. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:18

14. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:20

15. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:22

16. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:24

17. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:27

18. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:28

19. Альтернативный тестовый сценарий п.6.1: отрицательные классы эквивалентности Таблица 1.1 подклассы:30

20. Альтернативный тестовый сценарий п.8.1: В исходной БД АТС отсутствуют номера указанного типа.

21. Альтернативный тестовый сценарий п.8.2: неверный формат json.

22. Альтернативный тестовый сценарий п.8.3: неверный формат БД для указанного типа АТС.

23. Альтернативный тестовый сценарий п.9.1: Недостаточно прав для записи файла в указанную директорию.

# 3 РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

## 3.1 Общие сведения о программе

Конвертер MigrateIMS предназначен для конвертирования исходной базы

номеров телефонных станций в новый формат представления, совместимый для импорта (миграции) на новый узел коммутации vIMS. Другими словами, конвертер выполняет обработку каждого номера, его атрибутов – услуги, категория АОН и т.д, которыми он пользовался, а также служебные настройки – статус отключения за неуплату, административное ограничения исходящей/входящей связи и т.д. Далее, на основании шаблонов конвертирования, все вычисленные атрибуты номера преобразуются в новые значения и структуры, которые понятны для vIMS.

На вход конвертера подаются следующие данные:

1. Некоторая база номеров АТС, выбранных для миграции.

Учитывая специфику хранения данных в разных типах АТС, исходная

база номеров может содержать разные форматы и структуры данных, в том числе это может быть не один файл и несколько. Для каждого типа АТС разрабатывается индивидуальный программный модуль, способный преобразовывать исходные «сырые» данные к нужному внутреннему формату представления. Затем к этим данным применяются правила маппинга.

2. Правила маппинга (правила преобразования исходного типа к типу,

применяемому на vIMS) описываются в шаблонах конфигурации в формате json.

Создаются следующие шаблоны:

- Шаблон для переноса услуг (ДВО – дополнительные виды обслуживания).

- Шаблон для переноса категорий АОН.

- Шаблон vIMS который содержит общие (глобальные) правила переноса и формирования конечных данных для каждого конвертируемого номера.

3. Дополнительные флаги для управления процессом конвертирования.

Дополнительные флаги позволяют оператору указать конвертеру, по каким правилам вести обработку исходных данных. Часть флагов, управляющая потоком выполнения, указывается в качестве аргументов командной строки, в случае запуска программы через консольное приложение. Другая часть флагов относится к более тонкой настройке для управления бизнес-правилами и находится в качестве атрибутов в шаблоне IMS.

В качестве результата обработки, конвертер формирует следующие данные:

1. Файл со списком абонентов в формате для последующего импорта в БД VIMS.

2. Файл со списком активных услуг абонентов в формате для последующего импорта в БД vIMS.

3. Лог-файл.

## 3.2 Структура программы

Программа MigrateIMS состоит из следующих компонентов

1. migrate.py – главный исполняемый скрипт.
2. requirements.txt – файл со списком зависимостей.
3. requirements\_doc.txt – файл со списком зависимостей для генерации документации из дистрибутива.
4. sources – пакет с кодовой базой.
5. templates – папка с дефолтными шаблонами.
6. tests – папка с тестами.
7. docs – папка с документацией.

Перед начала использования требуется установка интерпретатора Python и установка зависимостей (внешних библиотек).

## 3.3 Настройка программы

Не нуждается в настройке.

## 3.4. Проверка программы

Проверка программы производится в следующем порядке:

Формируются все необходимые исходные данные

1. В командной строке выполняется запуск главного исполняемого файла. Пример: python migrate.py

2. Результат: Программа запускается, выполняет обработку исходных данных, формирует набор результирующих файлов, завершает работу.

## 3.5 Дополнительные возможности

Программа не обладает дополнительными возможностями.

## 3.6 Сообщения системному программисту

В процессе обработки данных, конвертер выводит все диагностические сообщения в консоль (терминал), а также в лог-файл. Всем сообщения присваивается категория срочности:

1. INFO – информационное.
2. WARNING – следует обратить внимание, есть проблемы с обработкой данных.
3. ERROR - Возникшее исключение на уровне интерпретатора.