Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Петрозаводский государственный университет
Математический факультет
Кафедра информатики и математического обеспечения

Отчет по дисциплине «Верификация ПО»

Тестирование администраторского клиента для интеллектуального зала ПетрГУ

Выполнил:
студент 6 курса группы 22609 А. С. Вдовенко

_____
подпись

Научный руководитель:
к.ф.-м.н., доцент Д. Ж. Корзун

Лектор:
к.ф.-м.н., доцент К. А. Кулаков
Итоговая оценка: _____

_____
подпись

Петрозаводск

2015

Содержание

1 Объект исследования

AdminClient – это приложение, входящее в состав системы Petrsu SmartRoom и являющееся агентом в интеллектуальных пространствах (ИП). На Рис. 1 представлена высокоуровневая архитектура системы. Данный агент участвует в процессе проведения конференции и предназначен для создания программы мероприятия, управления ходом конференции. Работа агента подразумевает взаимодействие с другими компонентами SmartRoom (*Conference-service, Presentation-service*).

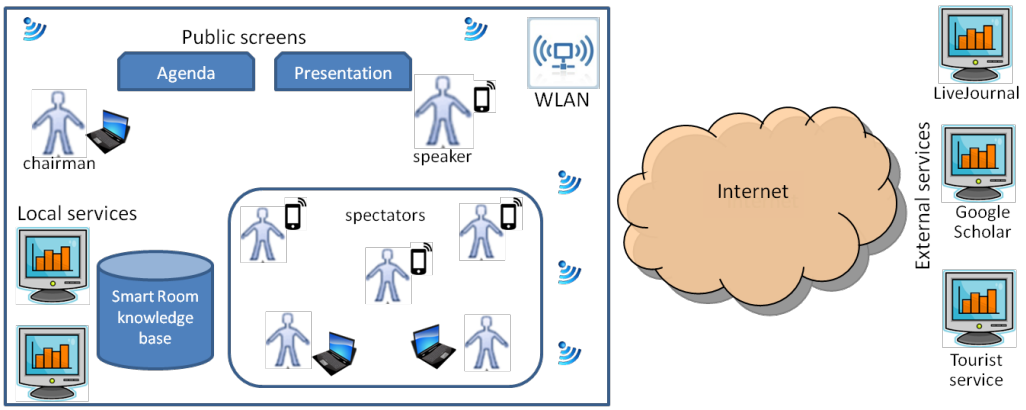

Рис. 1: Высокоуровневая архитектура системы PetrSU SmartRoom

Можно выделить две группы возможностей для данного агента и рассмотреть подробнее из чего они состоят.

1. **Создание программы мероприятия**

2. **Управление ходом мероприятия**

Подробнее возможности будут рассмотрены при аттестационном тестировании.

На Рис. 2 представлена схема взаимодействия между модулями клиента и компонентами SmartRoom.

2 Структура объекта тестирования

Приложение *AdminClient* состоит из следующих классов:

- Timeslot (представление сущности Timeslot)
  - **Свойства:**
    - string Duration — длительность выступления, в секундах

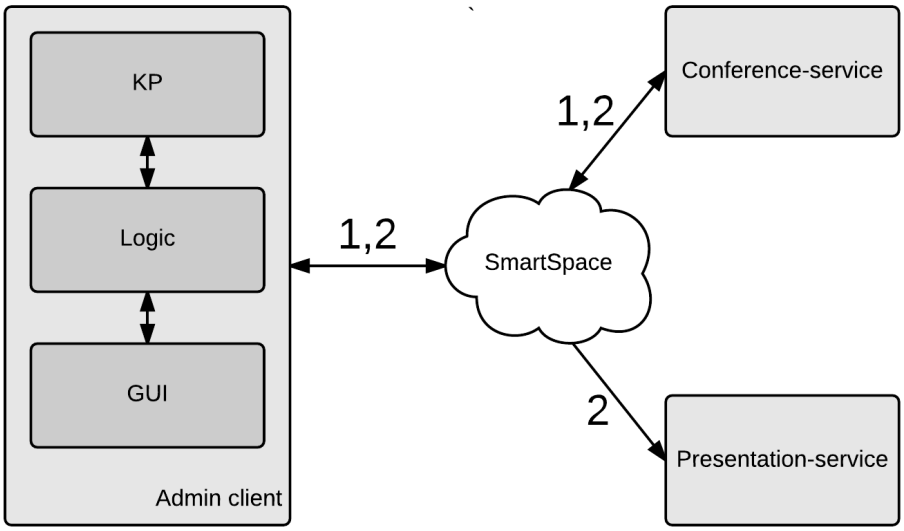

Рис. 2: Взаимодействие *AdminClient* с компонентами системы PetrSU SmartRoom

  * string PersonName — имя выступающего
  * string PersonUuid — идентификатор выступающего
  * string PresentationTitle — название выступления
  * string PresentationUuid — идентификатор презентации
  * string Uuid — идентификатор индивида таймслота
  * bool isEnabled — флаг сигнализирующий, выступил человек или нет

- Section (представление сущности Section)
  - **Свойства:**
    * string Date — дата выступления, формат DD.MM.YYYY
    * bool isAnyChanges — флаг наличия изменений, true/false
    * string StartTime — время начала, формат HH:MI
    * ObservableCollection<Timeslot> Timeslots — коллекция таймслотов для данной секции
    * string Title — название секции
    * string Uuid — идентификатор индивида секции
  - **Методы:**
    * Section ReadXML (string filename) — функция считывает объект секции из xml файла по указанному пути
    * void WriteXML(Section section, string filename) — записывает указанный объект в файл по выбранному пути

KP (процессор знаний, осуществление взаимодействия со SmartSpaces)

- **Свойства:**
  - Node node — внутренняя перемен
  - bool isJoined — флаг наличия соединения с ИП
  - Subscription subscriptionAgendaNotification — объект подписки для класса AgendaNotification
  - Subscription subscriptionCurrentSection — объект подписки на свойство CurrentSection
  - Subscription subscriptionCurrentTimeslot — объект подписки на свойство CurrentTimeslot
  - List<Subscription> subscriptionList — список ссылок на объекты подписки
  - event MethodContainer onAgendaUpdates(string uuid) — событие обновления по подпискам
- **Методы:**
  - void Join(string IP, int Port) — подключение к ИП
  - bool Disconnect() — отключение от ИП
  - void InitializeSubscription() — инициализация подписок
  - string PublishSection(Section section) — публикация объекта секции в ИП
  - Section GetSectionByUuid(string uuid) — получение объекта секции из ИП по идентификатору
  - ObservableCollection<Section> GetAllSections() — получение списка всех объектов секций
  - string GetCurrentSectionUuid() — получение идентификатора текущей секции
  - string GetCurrentTimeslotUuid() — получение идентификатора текущего таймслота
  - void RemoveSectionByUuid(string uuid) — удаление индивида секции из ИП по идентификатору
  - void SetCurrentSection(Section section) — установка текущей секции в ИП

    - void StartConference(Section section) — нотификация начала конференции
    - void EndConference(Section section) — нотификация конца конференции
    - bool EndCurrentPresentation() — конец текущей презентации
    - void PrevNextSlide(int offset) — переключение слайдов
    - void SubscriptionHandler(object sender, EventArgs e) — обработчик подписок
    - void updateAgendaConference(Section section) — нотификация обновления расписания
    - void Resubscribe() — переподписывание
    - void subscribeOnCurrentTimeslot(string uuid) — инициализация подписки на текущий таймслот по идентификатору секции

- WPF (отображает графический интерфейс пользователя)

Тип bool принимает значения true/false. Для типа string ограничение составляет 2 ГБ или 1 миллиард символов, чего достаточно при обычном использовании приложения. В связи с этим тесты на переполнение проводится не будут.

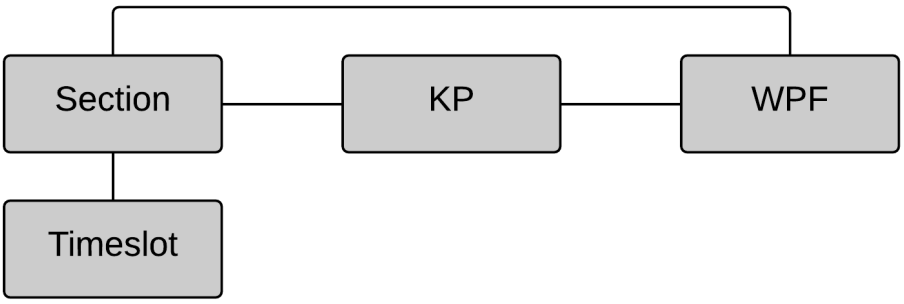

Рис. 3: Диаграмма классов *AdminClient*

На Рис. 3 представлены связи между классами логики приложения. Класс KP является главным классом, в котором создаются и используются объекты классов Section, Timeslot, Person, Presentation. Таким образом класс KP осуществляет взаимодействие между классами. Класс KP используется в интерфейсе пользователя WPF GUI.

Классы Timeslot, Person, Presentation не будут тестироваться отдельно, т.к. они являются классами-структурами для хранения данных о выступлениях, участниках и презентациях соответственно.

В классе KP не будут тестироваться функции, которые для полной проверки работоспособности, требуют взаимодействия с другими компонентами ИЗ. А это GetCurrentSectionUuid, GetCurrentTimeslotUuid, SetCurrentSection, StartConference, EndConference, EndCurrentPresentation, PrevNextSlide. Они будут проверятся при аттестационном тестировании.

Класс WPF не будет тестироваться методом блочного тестирования, т.к. все функции используемые в WPF являются интеграцией с классами KP и Section.

Стратегия тестирования: блочное тестирование будет применено ко всем функциям классов, кроме оговоренных ранее.

Схема интеграционного тестирования: На Рис. 4 представлена схема взаимодействия между классами.

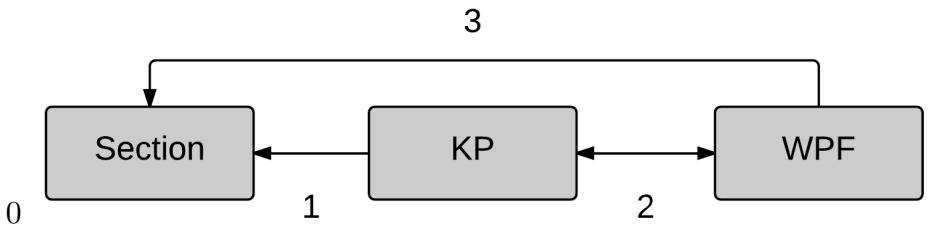

Рис. 4: Схема взаимодействия между классами

Интеграционное тестирование будет проходить в следующем порядке:

1. KP $\longrightarrow$ Section по типу вызов одного из другого
2. WPF $\longleftrightarrow$ KP по типу вызов друг друга
3. WPF $\longrightarrow$ Section вызов

Еще одним методом тестирования является Аттестационное тестирование, в котором будут проверятся функциональные качества программы.

3 Стратегия тестирования

3.1 Подход к тестированию

Тестирование будет проводиться в автоматическом режиме для блочных и интеграционных тестов, с помощью Visual Studio Test Tools. Тестировщику необходимо запускать проект для тестирования и ожидать результат. В случае ошибки, тестировщик самостоятельно формирует отчет об ошибке.

При аттестационном тестировании, необходимо разворачивание других компонент: Content-service, Presentation-service, Conference-service. Способы их запуска приведены в их README файлах. Тесты, где имеется зависимость от других тестов, следует выполнять в соответствующей последовательности.

Тестировщик имеет право выполнения тестов выборочным способом.

При получении исправленной версии приложения, производится регрессионное тестирование.

3.2 Критерий прохождения тестов

Тест считается пройденным, если конечный результат соответствует ожидаемому результату, Тест считается пройденным, если конечный результат соответствует ожидаемому результату, представленному для каждого варианта блочного, интеграционного и аттестационного теста в таблицах в столбце *Результат*.

3.3 Критерий приостановления/возобновления работы

Необходимо приостановить тестирование, если количество непройденных тестов превысит 15 процентов от их общего количества.

После получения новой версии приложения, необходимо начать тестирование с самого начала. Тестирование с места, когда была приостановлена работа, считается недопустимым, т.к. вероятно появление новых ошибок.

3.4 Требуемая документация

Необходим отчет о проведении тестирования, содержащий подробный отчет об ошибках после проведения тестирования.

4 Тесты

Типы тестов:

- О - общий
- С - специальный
- Н - негативный
- К - краевой

Формулировка <все свойства заданны> означает, что необходимо обратиться к описанию объекта в разделе 2, для уточнения о каких свойствах идет речь.

4.1 Блочные тесты

4.1.1 Класс KP

void Join(string IP, int Port)

Описание: инициализирует соединение с ИП

Входные данные: string IP, int Port

Косвенные данные: bool isJoined, внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join с параметрами

Результат: вызов функции не возвращает ошибки и происходит соединение с ИП, isJoined = true, иначе — isJoined = false.

№	Исходные данные	Результат	Тип
1.1.1	IP = "194.85.173.9" Port = 10011 Подключение к существующему ИП	isJoined = true	О
1.1.2	IP = "194.85.173.9" Port = 10012 Подключение к несуществующему ИП	isJoined = false	Н

Продолжение таблицы

1	2	3	4
1.1.3	IP = NULL Port = 10011 IP не задан	isJoined = false	Н
1.1.4	IP = "194.85.173.9" Port = NULL Port не задан	isJoined = false	Н
1.1.5	IP = NULL Port = NULL IP и Port не заданы	isJoined = false	Н
1.1.6	IP = "194.85.173.9" Port = 10011 ИП отключен	isJoined = false	С

bool Disconnect()

Описание: разъединяет соединение с ИП

Входные данные: нет

Косвенные данные: внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса КР
2. У объекта вызвать функцию Join
3. У объекта вызвать функцию disconnect

Результат: вызов функции не возвращает ошибки и происходит разъединение соединения с ИП, возвращение true, иначе — возвращает false.

№	**Исходные данные**	**Результат**	**Тип**
1.1.7	Отключение при существующем соединении с ИП	true	О
1.1.8	Отключение при проблемах с соединением с сетью или ИП	false	С

void InitializeSubscription()

Описание: устанавливает подписку на необходимые индивиды в ИП

Входные данные: нет

Косвенные данные: List<Subscription> subscriptionList, внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. У объекта вызвать функцию InitializeSubscription

Результат: вызов функции не возвращает ошибки, устанавливается подписка, subscriptionList.C
= 2, иначе — subscriptionList.Count $<$ 2.

№	Исходные данные	Результат	Тип
1.1.9	Функция не вызывает исключений	subscriptionList.Count = 2	O
1.1.10	Функции SmartSlog вызвали исключение	subscriptionList.Count $<$ 2	C

void subscriptionHandler(Subscription sender)

Описание: обработчик подписки, вызывается автоматически, обрабатывает и передаёт дальше запросы

Входные данные: Subscription sender (структура SmartSlog)

Косвенные данные: Subscription subscriptionCurrentTimeslot, внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. У объекта вызвать функцию InitializeSubscription
4. Назначить обработчика события onAgendaUpdates(string uuid)
5. Опубликовать индивиды:
   (a) класса AgendaNotification, со свойством UpdateAgenda

(b) класса ConferenceService со свойством CurrentSection

(c) класса Section (с uuid использованным для CurrentSection), со свойством CurrentTimeslot

Результат: подписка обрабатывается, если было добавлено одно из используемых свойств - вызывается onAgendaUpdates или subscribeOnCurrentTimeslot (при этом subscriptionCurrentTimeslot != NULL), иначе — ничего не происходит.

№	**Исходные данные**	**Результат**	**Тип**
1.1.11	Публикация индивида AgendaNotification со свойством UpdateAgenda с указанием правильного uuid индивида секции	События onAgendaUpdates с параметром uuid класса Section	O
1.1.12	Публикация индивида AgendaNotification со свойством UpdateAgenda с указанием неправильного(неполного) uuid индивида секции	Событие onAgendaUpdates не вызывается с параметром uuid класса Section	H
1.1.13	Публикация индивида AgendaNotification со свойством UpdateAgenda без указания uuid индивида секции	Событие onAgendaUpdates вызывается без параметра uuid класса Section	O
1.1.14	Публикация индивида ConferenceService со свойством CurrentSection с указанием правильного uuid индивида секции	Вызов subscribeOnCurrentTimeslot с параметром uuid класса Section, subscriptionCurrentTimeslot != NULL	O

Продолжение таблицы

1	2	3	4
1.1.15	Публикация индивида ConferenceService со свойством CurrentSection с указанием неправильного(неполного) uuid индивида секции	subscribeOnCurrentTimeslot не вызывается с параметром uuid класса Section, subscriptionCurrentTimeslot = NULL	H
1.1.16	Публикация индивида ConferenceService со свойством CurrentSection без указания uuid индивида секции	subscribeOnCurrentTimeslot не вызывается с пустым параметром uuid класса Section, subscriptionCurrentTimeslot = NULL	H
1.1.17	Изменение свойства CurrentTimeslot индивида Section с указанием uuid индивида таймслота	Событие onAgendaUpdates вызывается с параметром uuid класса Section	O
1.1.18	Изменение свойства CurrentTimeslot индивида Section с указанием неправильного(неполного) uuid индивида таймслота	Событие onAgendaUpdates не вызывается с параметром uuid класса Section	H
1.1.19	Изменение свойства CurrentTimeslot индивида Section без указания uuid индивида таймслота	Событие onAgendaUpdates не вызывается с параметром uuid класса Section	H
1.1.20	Публикация индивида AgendaNotification со свойством StartConference	Событие onAgendaUpdates не вызывается	C

Продолжение таблицы

1	2	3	4
1.1.21	Публикация индивида ConferenceService со свойством StartConference	subscribeOnCurrentTimeslot не вызывается	С
1.1.22	Публикация индивида Section со свойством CurrentTimeslot	Событие onAgendaUpdates не вызывается	С

string PublishSection(Section section)

Описание: публикует объект класса Section в виде соответствующего индивида в ИП

Входные данные: Section section

Косвенные данные: внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. Создать объект класса Section с заполненными значениями (использование средств автозаполнения)
4. У объекта вызвать функцию PublishSection

Результат: в ИП размещена заданная секция, возвращает uuid созданной секции, иначе — возвращает NULL.

№	**Исходные данные**	**Результат**	**Тип**
1.1.23	section.uuid = NULL , Timeslot.Count = 10 все свойства таймслотов и секции заданы	В ИП опубликован индивид секции с 5 таймслотами, все свойства заданны, возвращает uuid созданной секции	О
1.1.24	section.uuid != NULL , Timeslot.Count = 10 все свойства таймслотов и секции заданы	В ИП опубликован индивид секции с 5 таймслотами с заданным section.uuid, все свойства заданны, возвращает uuid созданной секции	О

Продолжение таблицы

1	2	3	4
1.1.25	section.uuid = NULL , Timeslot.Count = 0 заданы свойства Title, StartTime, StartDate	В ИП опубликован индивид секции со свойствами Title, StartTime, StartDate, возвращает uuid созданной секции	O
1.1.26	section = NULL	В ИП не публикуется индивид секции, возвращает NULL	H
1.1.27	section.uuid != NULL , Timeslot.Count = 10 все свойства заданы, в ИП такой индивид уже существует	В ИП публикуется индивид секции с 10 таймслотами, все свойства заданны, старый удаляется, возвращает uuid созданной секции	O
1.1.28	section.uuid = NULL , Timeslot.Count = 100 все свойства заданы	В ИП публикуется индивид секции со 100 таймслотами, все свойства заданы, возвращает uuid созданной секции	K
1.1.29	section.uuid = NULL , Timeslot.Count = 10 все свойства заданы, нет подключения к ИП	В ИП не публикуется индивид секции, возвращает NULL, ошибка в лог	C

Section GetSectionByUuid(string uuid)

Описание: возвращает из ИП объект класса Section для заданного uuid

Входные данные: string uuid

Косвенные данные: внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. Опубликовать индивид Section с заданными значениями (в зависимости от теста)
4. У объекта вызвать функцию GetSectionByUuid

Результат: возвращает объект Section с заданным uuid, иначе — возвращает NULL.

№	Исходные данные	Результат	Тип
1.1.30	uuid != NULL идентификатор существующего индивида у индивида Section заданны все свойства, Timeslot.Count = 5	возвращает объект Section с заданным uuid, Timeslot.Count = 5, все свойства заданны	O
1.1.31	uuid != NULL идентификатор существующего индивида у индивида Section заданны все свойства, Timeslot.Count = 0	возвращает объект Section с заданным uuid, Timeslot.Count = 0, все свойства заданны	O
1.1.32	uuid != NULL идентификатор существующего индивида у индивида Section не заданны свойства StartTime, StartDate, Title, Timeslot.Count = 5	возвращает объект Section с заданным uuid с пустыми свойствами StartTime, StartDate, Title, Timeslot.Count = 5	O
1.1.33	uuid != NULL идентификатор несуществующего индивида	возвращает NULL	H
1.1.34	uuid = NULL	возвращает NULL	H
1.1.35	uuid != NULL удаление секции во время считывания	возвращает NULL, ошибка в лог	C
1.1.36	uuid != NULL У Section заданы все свойства, Timeslot.Count = 100	Получение объекта секции, Timeslot.Count = 100, все свойства заданы	K

void RemoveSectionByUuid(string uuid)

Описание: удаляет из ИП индивид класса Section для заданного uuid

Входные данные: string uuid

Косвенные данные: внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. Опубликовать индивид Section с заданными значениями (в зависимости от теста)
4. У объекта вызвать функцию RemoveSectionByUuid

Результат: В ИП удаляется индивид секции с заданным uuid, иначе — объект не удаляется, не удаляются некоторые таймслоты.

№	Исходные данные	Результат	Тип
1.1.37	uuid != NULL идентификатор существующего индивида у индивида секции заданны все свойства, Timeslot.Count = 5	удаляет индивид секции с заданным uuid со всеми относящимися Timeslot и свойствами	O
1.1.38	uuid != NULL идентификатор существующего индивида у индивида Section заданны все свойства, Timeslot.Count = 0	удаляет индивид секции с заданным uuid и свойствами	O
1.1.39	uuid != NULL идентификатор существующего индивида у индивида Section не заданны свойства, Timeslot.Count = 5	удаляет индивид секции с заданным uuid	O
1.1.40	uuid != NULL идентификатор несуществующего индивида	ничего не удаляет	O
1.1.41	uuid = NULL	ничего не делает	H
1.1.42	uuid != NULL удаление таймслота во время выполнения	ошибка удаления, запись в лог	H

Продолжение таблицы

1	2	3	4
1.1.43	uuid != NULL У Section заданы все свойства, Timeslot.Count = 100	удаление секции и всех таймслотов и свойств	К

4.1.2 Класс Section

Section ReadXML(string filename)

Описание: считывает файл секции из xml файла

Входные данные: string filename

Косвенные данные: нет

Алгоритм:

1. Создать объект класса Section
2. У класса вызвать функцию ReadXML

Результат: возвращает объект секции, иначе — возвращает NULL.

№	**Исходные данные**	**Результат**	**Тип**
1.2.1	filename Верный путь У объекта секции заданны все свойства, Timeslot.Count = 5	Возвращает объект секции с заданными свойствами со всеми относящимися Timeslot	О
1.2.2	filename Верный путь У объекта секции заданны все свойства, Timeslot.Count = 100	Возвращает объект секции с заданными свойствами со всеми относящимися Timeslot	К
1.2.3	filename Неверный путь	Возвращает NULL	Н
1.2.4	filename Верный путь Неверная структура файла	Возвращает NULL	Н

void WriteXML(Section section, string filename)

Описание: записывает объект класса Section в xml файл

Входные данные: Section section, string filename

Косвенные данные: нет

Алгоритм:

1. Создать объект класса Section
2. У класса вызвать функцию WriteXML

Результат: возвращает объект секции, иначе — возвращает NULL.

№	Исходные данные	Результат	Тип
1.2.5	filename Верный путь, файла по пути не существует У объекта секции заданны все свойства, Timeslot.Count = 5	Записывает в файл объект секции с заданными свойствами со всеми относящимися Timeslot	O
1.2.6	filename Верный путь У объекта секции заданны все свойства, Timeslot.Count = 100	Возвращает индивид секции с заданными свойствами со всеми относящимися Timeslot	K
1.2.7	filename Неверный путь	Возвращает NULL	H
1.2.8	filename Верный путь Неверная структура файла	Возвращает NULL	H

4.2 Интеграционные тесты

4.2.1 KP ⟶ Section

List<Section> GetAllSection()

1. Получаем список всех индивидов через SmartSlog
2. Вызываем для каждой секции GetSectionByUuid(string uuid)

Описание: возвращает из ИП список объектов класса Section

Входные данные: нет

Косвенные данные: внутренние переменные SmartSlog

Алгоритм:

1. Создать объект класса KP
2. У объекта вызвать функцию Join
3. Опубликовать список индивидов Section с заданными значениями (в зависимости от теста)
4. У объекта вызвать функцию GetAllSection

Результат: возвращает список объектов Section, иначе — возвращает NULL.

№	Исходные данные	Результат	Тип
2.1.1	sectionList.Count = 5 Все свойства секций заданы, в каждой секции Timeslot.Count = 5	Возвращает список секций, sectionList.Count = 5, для каждого Timeslot.Count = 5	O
2.1.2	sectionList.Count = 0 Секций нет	Возвращает NULL	O
2.1.3	sectionList.Count = 5 Все свойства секций заданы, в каждой секции Timeslot.Count = 0	Возвращает список секций, sectionList.Count = 5, для каждого Timeslot.Count = 0	O

Продолжение таблицы

1	2	3	4
2.1.4	sectionList.Count = 100 Все свойства секций заданы, в каждой секции Timeslot.Count = 100	Возвращает список секций, sectionList.Count = 100, для каждого Timeslot.Count = 100	К
2.1.5	sectionList.Count = 10 Все свойства секций заданы, в каждой секции Timeslot.Count = 10 Удаляем секцию во время считывания	Возвращает список секций, sectionList.Count = 9, для каждого Timeslot.Count = 10	С
2.1.6	sectionList.Count = 10 Все свойства секций заданы, в каждой секции Timeslot.Count = 10 Пропадает соединение с ИП	Возвращает список секций, sectionList.Count = 10, для каждого Timeslot.Count = 10	С

4.2.2 WPF $\longleftrightarrow$ KP

В тестах полагается что объект KP создан, и организовано подключение к ИП через Join.

void saveSection()

1. Проверяется есть ли изменения, которые нужно сохранить
2. Если секция не началась, то секция полностью удаляется функцией KP.RemoveSectionBy
   uuid), а затем публикуется через KP.PublishSection(Section section)
3. Если секция идет, то вызывается UpdateSection(Section section)

Описание: выполняет сохранение секции в ИП из интерфейса пользователя.

Входные данные: нет

Косвенные данные: выбранный объект Section, внутренние переменные SmartSlog

Алгоритм:

1. Получить список всех секций

2. Изменить что-либо в секции

3. Нажать кнопку сохранения

Результат: сохранение в ИП актуального объекта Section, иначе — возвращает сообщение об ошибке.

№	Исходные данные	Результат	Тип
2.2.1	секция, с переменной порядка таймслотов Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 5	O
2.2.2	секция, с добавлением нового таймслота Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 6	O
2.2.3	секция, с переменной порядка таймслотов и добавлением нового Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 6	O
2.2.4	Попытка сохранения без выбранной секции	Сообщение об ошибке	H
2.2.5	секция, с переменной порядка таймслотов секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 5	O

Продолжение таблицы

1	2	3	4
2.2.6	секция, с добавлением нового таймслота секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 6	О
2.2.7	секция, с переменной порядка таймслотов и добавлением нового секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции сохранен верно, Timeslot.Count = 6	О

void refreshSection(string uuid)

1. Проверяем значение uuid, если NULL, то вызывается KP.GetCurrentSectionUuid()
2. Получаем объект Section section = KP.GetSectionByUuid(uuid)
3. Получаем для секции текущий таймслот через KP.GetCurrentTimeslotUuid(uuid)
4. Определяется какие таймслоты уже выступили, и передача измененного списка в интерфейс

Описание: выполняет обновление секции из ИП в интерфейсе пользователя.
Входные данные: string uuid
Косвенные данные: List<Section> sectionList, внутренние переменные SmartSlog
Алгоритм:

1. Получить список всех секций
2. Изменить секцию
3. Нажать кнопку обновления или получение события по подписке (изменение текущего таймслота)

Результат: получение обновленного объекта секции из ИП, иначе — возвращает сообщение об ошибке.

№	Исходные данные	Результат	Тип
2.2.8	секция, с переменной порядка таймслотов Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 5	O
2.2.9	секция, с добавлением нового таймслота Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 6	O
2.2.10	секция, с переменной порядка таймслотов и добавлением нового Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 6	O
2.2.11	Попытка обновления без выбранной секции	Сообщение об ошибке	H
2.2.12	секция, с переменной порядка таймслотов секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 5	O
2.2.13	секция, с добавлением нового таймслота секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 6	O

Продолжение таблицы

1	2	3	4
2.2.14	секция, с переменной порядка таймслотов и добавлением нового секция уже начата Все свойства секции заданы, в каждой секции Timeslot.Count = 5	Индивид секции обновлен верно, Timeslot.Count = 6	O

void Connection()

1. Получение глобального объекта KP
2. Проверка верности введенных данных (корректность IP адреса и порта)
3. Выполнение Join(IP,Port)
4. Если подключились, инициализируем подписки, сохраняем параметры в настройки

Описание: Подключение к ИП и инициализация подписок через интерфейсе пользователя.
Входные данные: нет
Косвенные данные: string IP, int Port, внутренние переменные SmartSlog
Алгоритм:

1. Ввести IP и Port в соотвествующие поля
2. Нажать кнопку подключения

Результат: подключение к ИП и инициализация подписок, иначе — возвращает сообщение об ошибке.

№	**Исходные данные**	**Результат**	**Тип**
2.2.15	Верные значения IP, Port	Подключение к ИП, скрытие полей подключения	O
2.2.16	IP и Port = NULL	Сообщение об ошибке	O
2.2.17	IP или Port из недопустимого формата/диапазона	Сообщение об ошибке	K

4.2.3 WPF ⟷ Section

void saveSectionFile()

1. Получение выбранной в интерфейсе секции
2. Указание имени файла в зависимости от названия секции или идентификатора секции
3. Запись файла секции через Section.WriteXML

Описание: Сохранение файла секции через интерфейс пользователя.
Входные данные: нет
Косвенные данные: List<Section> sectionList, filename, внутренние переменные SmartSlog
Алгоритм:

1. Добавить в список объект класса Section
2. Указать его как выбранный
3. Вызвать метод saveSectionFile

Результат: файл выбранной секции, иначе — возвращает сообщение об ошибке.

№	Исходные данные	Результат	Тип
2.3.1	section.Uuid != NULL, section.Title != NULL	Файл сохранен с именем section.Title	О
2.3.2	section.Uuid != NULL, section.Title = NULL	Файл сохранен с именем section.Uuid	О
2.3.3	section.Uuid = NULL, section.Title != NULL	Файл сохранен с именем section.Title	О
2.3.4	section.Uuid = NULL, section.Title = NULL	Файл сохранен с именем по умолчанию	О
2.3.5	section.Uuid != NULL, section.Title = "Test?"	Файл не сохранен	Н
2.3.6	section.Uuid != NULL, section.Title.Length = "257"	Файл сохранен с именем section.Title, урезанным до допустимого	К

void loadSectionFile()

1. Получение пути до файла

2. Загрузка объекта класса Section через метод Section.ReadXML

3. Добавление объекта секции в общий список

Описание: Загрузка объекта секции из файла в интерфейсе пользователя.
Входные данные: нет
Косвенные данные: List<Section> sectionList, filename, внутренние переменные SmartSlog
Алгоритм:

1. Указать путь до файла

2. Вызвать метод loadSectionFile

Результат: список секций sectionList пополненный загруженной секцией, иначе — возвращает сообщение об ошибке.

№	**Исходные данные**	**Результат**	**Тип**
2.3.7	filename, верный путь	Список секций sectionList пополнен загруженной секцией	О
2.3.8	filename, неправильный файл	Список секций sectionList не пополнен загруженной секцией, сообщение об ошибке	Н
2.3.9	filename = "*?"	Список секций sectionList не пополнен загруженной секцией, сообщение об ошибке	Н

4.3 Аттестационные тесты

В аттестационном тесте запускается приложение *Admin Client* и разворачиваются остальные элементы ИЗ (Content-service, Presentation-service, Conference-service). Проверяется функционал приложения.

4.3.1 Создание программы мероприятия

№	Функция	Результат
3.1.1	Добавление секции: - задание названия, времени и даты - сохранение в ИП	Секция добавляется в ИП с заданным названием, временем и датой
3.1.2	Добавление участников в секцию созданную в тесте 3.1.1: - из списка зарегистрированных с Content-service - создание шаблона участника - сохранение в ИП	Секция добавляется в ИП с указанными типами таймслотов
3.1.3	Сортировка и перемещение таймслотов полученных в тесте 3.1.2: - сортировка таймслотов внутри секции - перемещение таймслотов между секциями (требуется создание дополнительной секции, как в тесте 3.1.1) - сохранение в ИП	Секции изменяются в ИП в соответствии с указанным порядком таймслотов и таймслоты находятся в тех секциях, куда они были перемещены.

Продолжение таблицы

1	2	3
3.1.4	Сохранение/загрузка файла для одной из секций в тесте 3.1.3: - сохранение выбранной секции в файл - удаление секции из ИП - загрузка удаленной секции из файла - сохранение в ИП	Секции изменяются в ИП в соответствии с указанным порядком таймслотов и таймслоты находятся в тех секциях, куда они были перемещены.

4.3.2 Управление ходом мероприятия:

№	**Функция**	**Результат**
3.2.1	Начало секции созданной в тесте 3.1.3	Секция началась на Projector-service и Conference-service
3.2.2	Завершение текущей презентации для секции начатой тесте 3.2.1	Презентация завершилась на Projector-service и Conference-service, началась следующая
3.2.3	Управление показом текущей презентации для секции начатой в тесте 3.2.1: - переключение слайдов вперед (случайное количество) - переключение слайдов назад (случайное количество)	Слайды презентации показаны на Projector-service в верном порядке и количестве
3.2.4	Смена порядка выступающих для секции начатой в тесте 3.2.1: - переместить еще не выступившего в конец - сохранение изменений в ИП	Измененный порядок есть в ИП и Conference-service получил новый порядок

Продолжение таблицы

1	2	3
3.2.5	Добавление выступающего для секции начатой в тесте 3.2.1: - добавить новый таймслот - сохранение изменений в ИП	Измененная секция есть в ИП и Conference-service получил новое содержимое
3.2.6	Завершить секцию начатую в тесте 3.2.1	Секция закончилась на Projector-service и Conference-service

5 Примеры реализации тестов

В качестве примера тестирования был выбран Блочный тип тестов.

Для реализации этих тестов использовался **Visual Studio Test Tools**, а именно тип тестов **Unit Test**.

Unit test – это модульный программный тест, при выполнении которого вызываются методы класса и проверяются возвращаемые ими значения.

Для начала работы над тестированием, необходимо создать тестовый проект для решения, которое далее будет тестироваться.

После создания проекта, в него необходимо добавить исходный код Unit-теста.

В коде 1 представлен пример объявления тестового класса.

```
namespace AdminClient.Tests
{
        [TestClass()]
        public class UnitTest
        {
        // Here is tests body
        }
}
```

Код 1: Объявление тестового класса

Для некоторых тестов необходимо иметь некоторую базу для проверки или действия выполненные перед началом тестирования, для этого есть специальные директивы *ClassInitialize()* и *ClassCleanup()*. Функции обозначенные данными директивами выполняются каждый раз перед началом тестирования и в конце когда закончился последний тест.

В качестве примера инициализации можно рассмотреть код 2. В данном методе инициализируется глобальное соединение с интеллектуальными пространствами и публикуются тестовые данные, работа с которыми потом будет проверятся.

```
[ClassInitialize()]
public static void Initialize(TestContext context)
{
        try
        {
                kp_g.Join("194.85.173.9", 10011);
```

```
        }
        catch
        {
                return;
        }

        kp_g.PublishTestSection(section_uuid, "Test 1");
        kp_g.PublishTestSection();

        Section section = new Section();

        section.Uuid = empty_section_uuid;
        section.Title = "Test 2";

        kp_g.PublishSection(section);

        section = new Section();
        section.Uuid = totaly_empty_section_uuid;

        kp_g.PublishSection(section);
}
```

Код 2: Пример инициализирующего метода

В качестве примера очистки, приведен код 3. В данном примере все действия сделанные при инициализации отменяются.

```
[ClassCleanup()]
public static void CleanAll()
{
        kp_g.RemoveSectionByUuid(section_uuid);
        kp_g.RemoveSectionByUuid(empty_section_uuid);
        kp_g.RemoveSectionByUuid(totaly_empty_section_uuid);

        kp_g.Disconnect();
}
```

Код 3: Пример деструктивного метода

В коде 4 приведен пример реализации теста для функции получения секции. Тест считывает входные параметры из файла *GetSectionTestData.xml*. В коде 5 приведен пример того как выглядят тестовые данные.

```
[TestMethod()]
[DeploymentItem("GetSectionTestData.xml")]
[DataSource("Microsoft.VisualStudio.TestTools.DataSource.XML",
                        "|DataDirectory|\\GetSectionTestData.xml",
                        "Row",
                        DataAccessMethod.Sequential)]
public void GetSectionByUuidTest()
{
        string Uuid = TestContext.DataRow["Uuid"].ToString();
        string Title = TestContext.DataRow["Title"].ToString();

        bool result =
           bool.Parse(TestContext.DataRow["Result"].ToString());
        int TimeslotCount;
        int.TryParse(TestContext.DataRow["TimeslotCount"].ToString(),
           out TimeslotCount);

        Section section = kp_g.GetSectionByUuid(Uuid);

        if (result)
                Assert.IsNotNull(section);
        else
        {
                Assert.IsNull(section);
                return;
        }

        Assert.AreEqual(Title, section.Title);
        Assert.AreEqual(TimeslotCount, section.Timeslots.Count);
}
```

Код 4: Пример теста: функция получения секции

```
<Rows>
```

```
        <Row>
                <Uuid>test_section_for_testing</Uuid>
                <Title>Test 1</Title>
                <TimeslotCount>3</TimeslotCount>
                <Result>true</Result>
        </Row>
        <Row>
                <Uuid>test_section_for_testing_1</Uuid>
                <Title>Test 1</Title>
                <TimeslotCount>0</TimeslotCount>
                <Result>false</Result>
        </Row>
        <Row>
                <Uuid>totaly_empty_section_for_testing</Uuid>
                <Title></Title>
                <TimeslotCount>0</TimeslotCount>
                <Result>true</Result>
        </Row>
        <Row>
                <Uuid></Uuid>
                <Title></Title>
                <TimeslotCount>0</TimeslotCount>
                <Result>false</Result>
        </Row>
</Rows>
```

Код 5: Пример тестовых данных

6 Методы покрытия

Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводился по формуле:

$$T_{cov} = \frac{L_{tc}}{L_{code}} * 100\%, \text{ где:}$$

- T_{cov} — тестовое покрытие,
- L_{tc} — количество строк кода, покрытых тестами (исключая комментарии и пробелы),
- L_{code} — общее количество строк кода (исключая комментарии и пробелы).

Для нашего проекта, эта метрика принимает следующие значения:

$$L_{code} = 1726, L_{tc} = 1436 \longrightarrow T_{cov} = 83,2\%$$

7 Отчет о проведенном тестировании

В ходе разработки и выполнения тестов были получены следующие результаты:
Блочные тесты

Класс	**Номера тестов**	**Количество запусков**	**Количество уникальных ошибок**
KP	1.1.1 – 1.1.8	15	0
	1.1.9 – 1.1.22	15	1
	1.1.23 – 1.1.43	15	6
Section	1.2.1 – 1.2.8	15	0

Перечень обнаруженных ошибок:

1. KP.subscriptionHandler, Тест 1.1.16 — ошибочно вызывается subscribeOnCurrentTimeslot(NULL), что проводило к необработанному исключению.

2. KP.PublishSection, Тест 1.1.28 — некоторые таймслоты не были опубликованы, что привело к сбитому списку секции.

3. KP.PublishSection, Тест 1.1.29 — возникло необработанное исключение, из-за отстутствия соединения с ИП .

4. KP.GetSectionByUuid, Тест 1.1.33 — возвращает объект секции, с единственным заданным свойством uuid.

5. KP.GetSectionByUuid, Тест 1.1.35 — получены не все таймслоты, возникло необработанное исключение.

6. KP.GetSectionByUuid, Тест 1.1.36 — получены не все таймслоты.

7. KP.RemoveSectionByUuid, Тест 1.1.42 — возникло необработанное исключение.

Подробный отчет о найденных ошибках смотреть в разделе 8.

После устранения большинства ошибок, было проведено регрессионное тестирование. Ниже приведена таблица отчета проведенных запусков тестов.

Класс	**Номера тестов**	**Количество запусков**	**Количество уникальных ошибок**
KP	1.1.1 – 1.1.8	5	0
	1.1.9 – 1.1.22	5	0
	1.1.23 – 1.1.43	5	2
Section	1.2.1 – 1.2.8	5	0

Повторное тестирование показало, что исправленные ошибки не привели к новым, а оставшиеся ошибки находятся на стадии исправления.

Интеграционные тесты

Классы	**Номера тестов**	**Количество запусков**	**Количество уникальных ошибок**
KP $\longrightarrow$ Section	2.1.1 – 2.1.8	15	3
WPF $\longleftrightarrow$ KP	2.2.1 – 2.2.14	15	0
	2.2.15 – 2.2.17	15	0
WPF $\longleftrightarrow$ Section	2.3.1 – 2.3.9	15	0

Перечень критических ошибок:

1. KP.GetAllSection, Тест 2.1.4 — получены не все секции или не все таймслоты.
2. KP.GetAllSection, Тест 2.1.5 — получены все секции, но не все таймслоты, а удаленная секция, должна была быть исключена.
3. KP.GetAllSection, Тест 2.1.6 — возникло необработанное исключение.

Подробный отчет о найденных ошибках смотреть в разделе 8.

После устранения большинства ошибок, было проведено регрессионное тестирование. Ниже приведена таблица отчета проведенных запусков тестов.

Классы	**Номера тестов**	**Количество запусков**	**Количество уникальных ошибок**
KP $\longrightarrow$ Section	2.1.1 – 2.1.8	5	2

Продолжение таблицы

1	2	3	4
WPF $\longleftrightarrow$ KP	2.2.1 – 2.2.14	5	0
	2.2.15 – 2.2.17	5	0
WPF $\longleftrightarrow$ Section	2.3.1 – 2.3.9	5	0

Повторное тестирование показало, что исправленные ошибки не привели к новым, а оставшиеся ошибки находятся на стадии исправления.

Аттестационные тесты

Класс функций	**Номера тестов**	**Количество запусков**	**Количество уникальных ошибок**
Создание программы мероприятия	3.1.1 – 3.1.4	10	0
Управление ходом мероприятия	3.2.1 – 3.2.6	10	0

При аттестационном тестирование ошибок в работе ПО не найдено.

После проведения тестов были исправлены ошибки в классе KP, для необработанных исключений были добавлены обработчики, но некоторые ошибки не были исправлены, по причине стороннего ПО SmartSlog или Smart-M3, остальные ошибки планируется исправить.

8 Отчеты об ошибках

Для занесения отчетов об найденных ошибках используется инструмент Bugzilla. Приведем пример одного отчета об ошибке.

1. **KP.subscriptionHandler, Тест 1.1.16**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

   На предыдущих версиях тест не проводился

   Серьезность: Серьезная

   Приоритет: P3

   Состояние: Закрыта

   Исполнитель: Вдовенко А.С.

   Решение: Исправлено

   Описание ошибки:

   Ошибочно вызывается subscribeOnCurrentTimeslot(NULL), что проводит к необработанному исключению.

   Ожидаемое решение: Не вызывать subscribeOnCurrentTimeslot(NULL), проверять перед вызовом.

   Повторимость: всегда.

   Алгоритм воспроизведения ошибки:

   (a) Запуск теста

   (b) Необработанное исключение

2. **KP.PublishSection, Тест 1.1.28**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

   На предыдущих версиях тест не проводился

   Серьезность: Серьезная

   Приоритет: P2

   Состояние: Закрыта

   Исполнитель: Вдовенко А.С.

   Решение: Исправлено

Описание ошибки:

Некоторые таймслоты не были опубликованы, что привело к сбитому списку секции.

Ожидаемое решение: Публикация секции и всех таймслотов.

Повторимость: периодическая.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Секция опубликована не полностью.

3. **KP.PublishSection, Тест 1.1.29**:

Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

Версия приложения: Version 1.1.0

На предыдущих версиях тест не проводился

Серьезность: Серьезная

Приоритет: P2

Состояние: Закрыта

Исполнитель: Вдовенко А.С.

Решение: Исправлено

Описание ошибки:

Возникло необработанное исключение, из-за отстутствия соединения с ИП.

Ожидаемое решение: Корректная обработка ситуация с отсутствием соединения с ИП.

Повторимость: периодическая.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Получение не всех секций, либо неполностью.

4. **KP.GetSectionByUuid, Тест 1.1.33**:

Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

Версия приложения: Version 1.1.0

На предыдущих версиях тест не проводился

Серьезность: Серьезная

Приоритет: P2

Состояние: Закрыта

Исполнитель: Вдовенко А.С.

Решение: Исправлено

Описание ошибки:

Возвращает объект секции, с единственным заданным свойством uuid.

Ожидаемое решение: Возвращение NULL

Повторимость: всегда.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Возвращение объекта секции, с единственным заданным свойством uuid.

5. **KP.GetSectionByUuid, Тест 1.1.35**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

   На предыдущих версиях тест не проводился

   Серьезность: Серьезная

   Приоритет: P4

   Состояние: Назначена

   Исполнитель: Вдовенко А.С.

   Решение: -

   Описание ошибки:

   Получены не все таймслоты, возникло необработанное исключение.

   Ожидаемое решение: Возвращение NULL, сообщение об ошибке

   Повторимость: периодическая.

   Алгоритм воспроизведения ошибки:

   (a) Запуск теста

   (b) Получение не всей секции.

6. **KP.GetSectionByUuid, Тест 1.1.36**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

На предыдущих версиях тест не проводился

Серьезность: Серьезная

Приоритет: P4

Состояние: Назначена

Исполнитель: Вдовенко А.С.

Решение: -

Описание ошибки:

Получены не все таймслоты.

Ожидаемое решение: Получение объекта секции, Timeslot.Count = 100, все свойства заданы

Повторимость: периодическая.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Получение не всех секций, либо неполностью.

7. **KP.RemoveSectionByUuid, Тест 1.1.42**:

Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

Версия приложения: Version 1.1.0

На предыдущих версиях тест не проводился

Серьезность: Серьезная

Приоритет: P3

Состояние: Закрыта

Исполнитель: Вдовенко А.С.

Решение: Исправлено

Описание ошибки:

Возникло необработанное исключение.

Ожидаемое решение: Ошибка удаления, запись в лог.

Повторимость: периодическая.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Возникает необработанное исключение

8. **KP.GetAllSection, Тест 2.1.4**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

   На предыдущих версиях тест не проводился

   Серьезность: Серьезная

   Приоритет: P4

   Состояние: Назначена

   Исполнитель: Вдовенко А.С.

   Решение: -

   Описание ошибки:

   Получены не все секции или не все таймслоты.

   Ожидаемое решение: Получение всех секций и всех таймслотов.

   Повторимость: периодическая.

   Алгоритм воспроизведения ошибки:

   (a) Запуск теста

   (b) Получение не всех секций, либо неполностью.

9. **KP.GetAllSection, Тест 2.1.5**:

   Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

   Версия приложения: Version 1.1.0

   На предыдущих версиях тест не проводился

   Серьезность: Серьезная

   Приоритет: P2

   Состояние: Закрыта

   Исполнитель: Вдовенко А.С.

   Решение: Исправлено

   Описание ошибки:

   Получены все секции, но не все таймслоты, а удаленная секция, должна быть исключена.

   Ожидаемое решение: Исключение удаленной секции, за счет проверки в конце списка на удаленные индивиды.

   Повторимость: периодическая, зависит от удаляемой секции.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Удаление случайно выбранной секции из списка

(c) Получение не всех секций, либо неполностью.

10. **KP.GetAllSection, Тест 2.1.6**:

Плафторма и ОС: Приложение было протестировано под ОС Windows 8, с использованием Visual Studio 2012, .Net v4.0

Версия приложения: Version 1.1.0

На предыдущих версиях тест не проводился

Серьезность: Серьезная

Приоритет: P4

Состояние: Назначена

Исполнитель: Вдовенко А.С.

Решение: -

Описание ошибки:

Возникает необработанное исключение.

Ожидаемое решение: Корректная обработка пропадания соединения с ИП.

Повторимость: всегда.

Алгоритм воспроизведения ошибки:

(a) Запуск теста

(b) Отключение сетевого интерфейса

(c) Возникает необработанное исключение

Список литературы

1. Create and run unit tests - Visual Studio [Электронный ресурс], 2014 URL: http://www.visualstudio.com/en-us/get-started/create-and-run-unit-tests-vs.aspx (дата обращения - 18.12.2014).

2. How To: Create a Data-Driven Unit Test - MSDN - Microsoft [Электронный ресурс], 2014 URL: http://msdn.microsoft.com/en-us/library/ms182527.aspx (дата обращения - 18.12.2014).

3. Сергей Синицын, Никита Налютин, Верификация программного обеспечения, Издательство Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2008.