

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики
Кафедра программного обеспечения и администрирования
информационных систем

Автоматизация администрирования разработки ETL-процессов

Бакалаврская работа

Направление 02.03.03 Математическое обеспечение и администрирование
информационных систем

Профиль Информационные системы и базы данных

Зав. кафедрой _____ д. ф.-м. н., проф. М. А. Артемов _____. 2019 г.

Обучающийся _____ Д. А. Леденев

Руководитель _____ ст. преп. Г. Э. Воцинская

Воронеж 2019

Аннотация

Работа посвящена разработке механизма, позволяющего систематизировать внедрение доработок ETL-системы.

Цель работы – создать клиентское приложение, которое будет предоставлять возможности для объединения скриптов Oracle SQL и объектов Informatica PowerCenter в структуру, пригодную для автоматической установки.

Приложение реализуется с использованием языка C#. При разработке были использованы различные возможности, предоставляемые соответствующим языком для работы с базой данных Oracle.

Содержание

Введение.....	4
1. Постановка задачи.....	6
2. Анализ задачи	7
2.1. Анализ функциональности приложения.....	7
2.2. Структуры данных приложения	8
2.3. Предполагаемые структуры программы.....	10
3. Средства реализации.....	12
4. Требования к аппаратному и программному обеспечению	13
5. Интерфейс пользователя	14
5.1. Управление релизом	14
5.2. Проверка релиза	19
5.3. Редактирование сценария.....	20
6. Реализация.....	22
6.1. Структура базы данных	22
6.2. Описание сущностей.....	22
6.2.1. Сущность «Патч»	23
6.2.2. Сущность «Поставка».....	24
6.2.3. Сущность «Релиз».....	24
6.3. Программные модули	25
7. План тестирования	27
Заключение	34
Литература	35
Приложение 1. Диаграмма деятельности	36
Приложение 2. Класс «CPatch».....	37

Введение

В настоящий момент установка всех скриптов Oracle и файлов Informatica PowerCenter осуществляется путем объединения их в структуру, называемую патчем.

Патчем является набор объектов:

- скриптов Oracle;
- объектов Informatica PowerCenter;
- текстового файла-сценария, указывающего, в каком порядке необходимо устанавливать остальные объекты патча.

Далее набор патчей объединяется в поставку. Поставкой является объединение патчей, к которому прилагаются следующие документы:

- excel-файл, в котором хранится информация о каждом патче: название патча, автор, номер изменения, название затрагиваемой сущности в хранилище данных, название документа, описывающего функциональные требования, описание, список патчей, от которых зависит патч и список патчей, на который патч влияет;
- текстовый файл-сценарий;
- word-файл – заметки о выпуске. В нем хранятся введение, основание для разработки – нормативный документ, описывающий спецификацию реализации хранилища, пререквизиты – список поставок, которые требуется установить до установки поставки, объекты, добавленные в поставку в предыдущих версиях, объекты, добавленные в поставки в текущей версии, объекты, исключенные из поставки, описание, инструкция по установке поставки, инструкция по деинсталляции, действия, которые необходимо выполнить после установки патча;

Набор поставок объединяется в релиз. Релизом является совокупность задач, проходящая этапы анализа бизнес-требований и функциональных требований, разработки, тестирования и внедрения в производство.

Существующий подход структурирования имеет несколько недостатков:

1. Ненадежность. Работник, ответственный за установку релиза на тестовую или производственную среду может ошибиться и забыть установить пререквизиты.
2. Низкая скорость сборки и установки релиза. Документы поставки формируются вручную, что уменьшает надежность и замедляет процесс внедрения релиза.

В результате возникла необходимость написания приложения, позволяющего автоматизировать работу по систематизации доработок ETL-системы, что существенно повысит скорость и сократит количество ошибок во время работы с данными.

1. Постановка задачи

Реализовать механизм, позволяющий систематизировать работу по доработкам ETL-системы (см. [11], [12], [13]).

Для этого необходимо решить следующие задачи:

- для каждого релиза экспортировать из системы Atlassian Jira название патча, затрагиваемую сущность, автора патча, статус готовности и зависимость от других патчей;
- реализовать возможность объединить патчи в поставку;
- реализовать автоматическое определение зависимостей между поставками по зависимостям патчей;
- реализовать возможность автоматической установки поставки на выбранную среду тестирования.
- реализовать проверку на то, что поставка была установлена предыдущую среду тестирования;
- реализовать определение времени установки поставки на выбранную среду тестирования;
- реализовать определение объектов, затрагиваемых в патче.

2. Анализ задачи

2.1. Анализ функциональности приложения

На текущий момент работа по организации релиза производится следующим образом:

Разработчик выкладывает в систему контроля версий Microsoft Visual SourceSafe скрипты базы данных Oracle, организуя их соответственно по типу объектов, над которыми производится операции DML или DDL (см. [8], [9]).

Приоритет по установке объектов БД следующий:

- 1) последовательности;
- 2) таблицы;
- 3) индексы;
- 4) представления;
- 5) функции;
- 6) процедуры;
- 7) пакеты.

Также разработчик может выкладывать объекты Informatica PowerCenter – общие для всех папок или объекты, доступные только для конкретной папки. В первую очередь устанавливаются общие объекты.

Приоритет по установке объектов Informatica PowerCenter следующий:

- 1) источники;
- 2) приемники;
- 3) определенные пользователем функции;
- 4) выражения;
- 5) последовательности;
- 6) трансформации «Справочная таблица»;
- 7) карты-шаблоны;
- 8) карты;

- 9) задачи;
- 10) потоки-шаблоны;
- 11) потоки.

Такое упорядочение обеспечивает разрешение конфликтов при установке для большинства типичных доработок и является приемлемым в рамках одного патча (см. [10]). Ввиду того, что между патчами также существуют зависимости, возникает ряд проблем, которые необходимо решить:

1. Найти способ организации, при котором все доработки могли бы устанавливаться за приемлемое время.
2. Порядок установки на всех средах должен быть схожим, в противном случае неучтенные зависимости могут привести к ошибкам, обнаруженным на поздних стадиях релиза, в худшем случае – в производственной эксплуатации.
3. Разработать способ предварительного анализа поставки на неучтенные зависимости.

Общая схема работы пользователя с программным продуктом представлена в приложении 1.

2.2. Структуры данных приложения

Для того, чтобы к данным приложения был предоставлен удобный для пользователей доступ, информация должна храниться в базе данных. Так как проект, для которого реализуется приложение, использует базу данных Oracle, приложение также будет хранить свои данные на одной из сред.

Структура данных должна поддерживать версиюность, то есть в таблицах должны присутствовать поля, содержащие информацию о том, какие изменения производились над записью: добавление, изменение или удаление, и интервал времени, в который запись была актуальна. Такой подход предоставит возможность в случае появления некорректных данных определить, в какой момент они появились.

Данные будут организованы в следующие таблицы:

1. Таблица «Патч». Хранит все метаданные патча и содержит следующие поля:

- идентификатор_патча – суррогатный ключ;
- название_патча – название, экспортируемое из системы Atlassian Jira;
- дата_начала_действия_записи – нижняя граница актуальности записи, техническая дата;
- дата_окончания_действия_записи – верхняя граница актуальности записи, техническая дата;
- статус_патча – статус, выгружаемый из системы Atlassian Jira, хранящий информацию о готовности патча и принимающий значения «Открыт», если патч не готов, «Ожидает подтверждения», если патч готов, протестирован, ожидает подтверждения на другой среде тестирования, а также запуска автоматических тестов;
- флаг_действия – техническая информация о записи, принимает значения «У» – удалена, «Д» – добавлена или «И» – изменена;
- CR – номер задачи в системе Application Lifecycle Management;
- DF – номер дефекта в системе Application Lifecycle Management;
- сущность – затрагиваемая сущность в хранилище данных;
- FSD – функциональные требования для задачи или дефекта;
- описание – текстовое описание доработок;
- идентификатор_поставки – внешний ключ, связывает с таблицей «Поставка»;

2. Таблица «Поставка». Хранит все метаданные поставки и содержит следующие поля:

- идентификатор_поставки – суррогатный ключ;

- название_поставки – название, экспортируемое из системы Atlassian Jira;
- дата_начала_действия_записи – нижняя граница актуальности записи, техническая дата;
- дата_окончания_действия_записи – верхняя граница актуальности записи, техническая дата;
- статус_поставки – статус, определяемый администратором тестовой среды;
- флаг_действия – техническая информация о записи. Принимает значения «У» – удалена, «Д» – добавлена или «И» – изменена;
- идентификатор_релиза – внешний ключ, связывает с таблицей «Релиз».

3. Таблица «Релиз». Хранит все метаданные релиза и содержит следующие поля:

- идентификатор_релиза – суррогатный ключ;
- название_релиза – название, экспортируемое из системы Atlassian Jira;
- дата_начала_действия_записи – нижняя граница актуальности записи, техническая дата;
- дата_окончания_действия_записи – верхняя граница актуальности записи, техническая дата;
- флаг_действия – техническая информация о записи. Принимает значения «У» – удалена, «Д» – добавлена или «И» – изменена.

2.3. Предполагаемые структуры программы

Предполагается, что приложение будет поделено на следующие логические элементы:

1. Oracle Database – база данных, в которой хранится информация об объектах релиза.

2. Модель данных – отвечает за связь с базой данных и предоставляет методы объектов, соответствующих записям в базе данных.
3. Отображение – набор форм, предоставляющих взаимодействие приложения с пользователем.
4. Набор утилит, обеспечивающих взаимодействие с системой контроля версий Microsoft Visual SourceSafe.
5. Простой анализатор SQL-кода, позволяющий определить тип объекта БД и его идентификатор.

3. Средства реализации

В качестве языка программирования использовался C#, для хранения метаданных релиза использовалась СУБД Oracle. Доступ к записям базы данных Oracle осуществлялся с помощью Oracle Data Provider for .NET и хранимых процедур на PL/SQL.

Возможности языка предоставляют все необходимые инструменты для реализации поставленных в работе целей и задач. При написании программы использовалась среда разработки Microsoft Visual Studio 2017.

4. Требования к аппаратному и программному обеспечению

Основное приложение предназначено для использования на IBM PC-совместимых компьютерах с операционной системой Windows 7, Windows 8, Windows 10.

Для работы программного комплекса необходим сервер баз данных Oracle для хранения метаданных релиза, проект в системе Atlassian Jira, программные продукты Microsoft Excel и Informatica PowerCenter Client. Физически СУБД может находиться на том же компьютере, что и сервер программного комплекса. Размер необходимого дискового пространства зависит от количества хранимых данных.

Минимальные системные требования приложения:

- процессор с частотой 1,6 ГГц или выше;
- 1024 МБ ОЗУ.

Других особых требований к аппаратному и программному обеспечению клиента не предъявляется.

5. Интерфейс пользователя

5.1. Управление релизом

При запуске приложения открывается форма «Управление релизом», в которой содержится структура релизов. На этой форме можно просмотреть, добавить и удалить зависимости для патчей и поставок.

Образец внешнего вида просмотра поставок представлен на рис. 5.1.

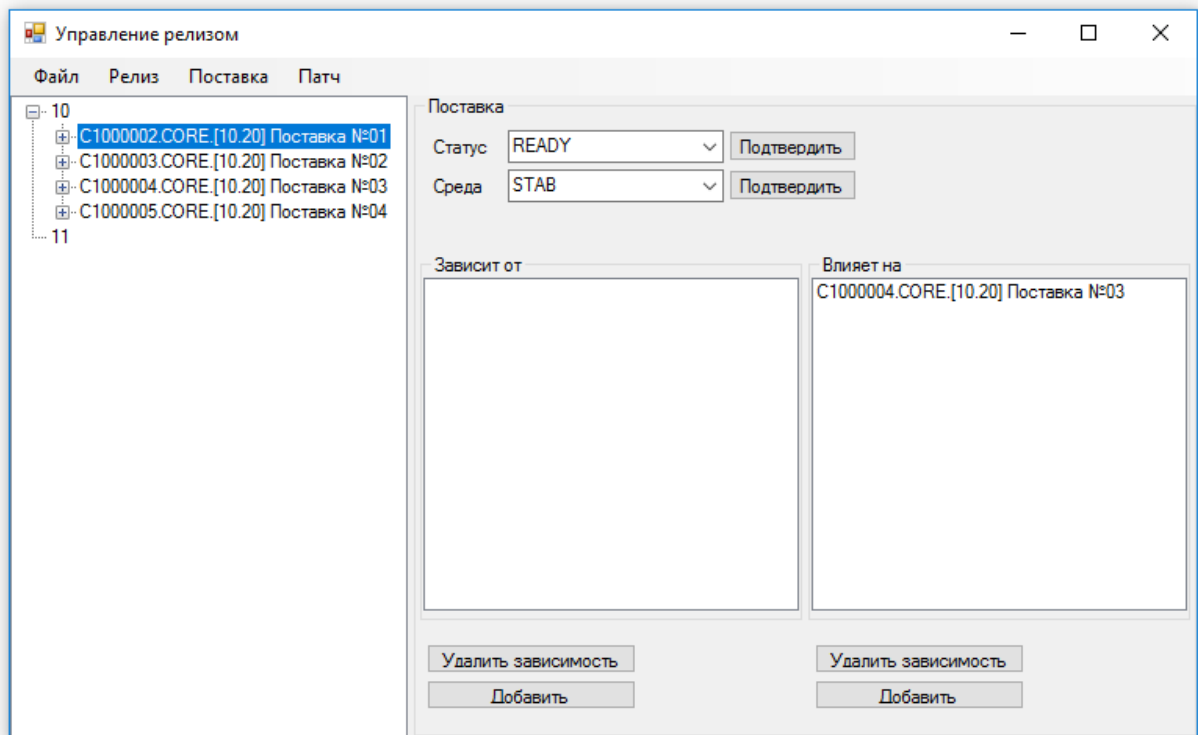


Рис. 5.1. Форма «Управление релизом» для поставок

Образец внешнего вида просмотра патчей представлен на рис. 5.2.

Форма содержит дерево выбора, в котором корневыми элементами являются релизы. В релизах содержатся поставки, в поставках – патчи.

Также при выборе узла поставки на правой панели расположены два выпадающих списка, в которых можно отметить среду, на которую установлена поставка и статус установки.

Панель содержит два списка, в которых отмечены поставки, связанные с выбранной. В каждый список можно добавить поставку и удалить.

При этом релиз выбранной и добавляемой поставки должен быть одинаковым.

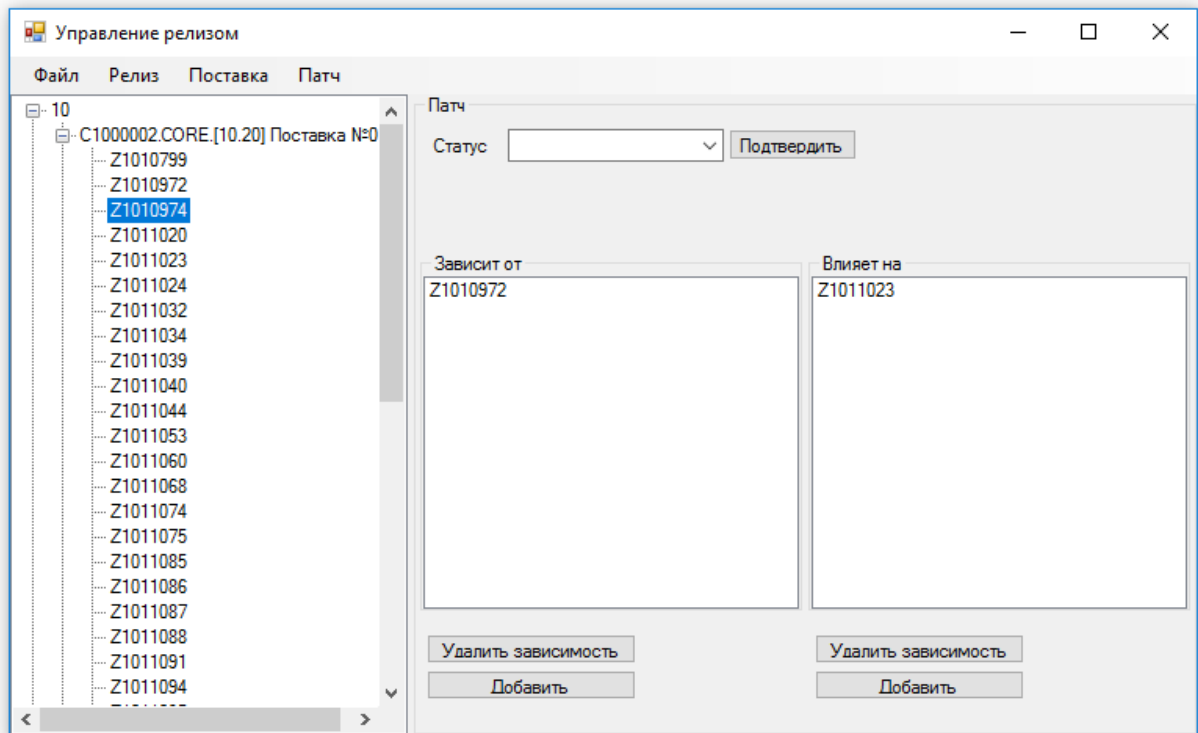


Рис. 5.2. Форма «Управление релизом» для патчей

Помимо этого, существуют и другие ограничения. Например, нельзя удалить зависимость по поставкам, если существует зависимость по содержащимся в этих поставках патчам. При попытке удалить такую зависимость пользователю будет выведено сообщение об ошибке, представленное на рис. 5.3.

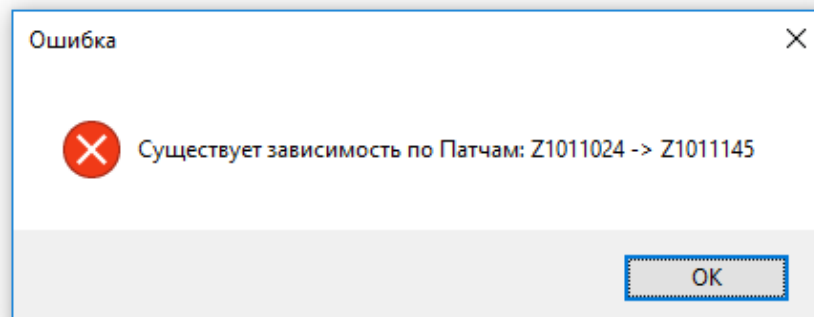


Рис. 5.3. Ошибка при удалении зависимости между поставками

Во избежание циклических зависимостей существует проверка, определяющая, образует ли цепочка зависимостей цикл. При попытке добавить такую зависимость пользователь получает ошибку, представленную на рис. 5.4.

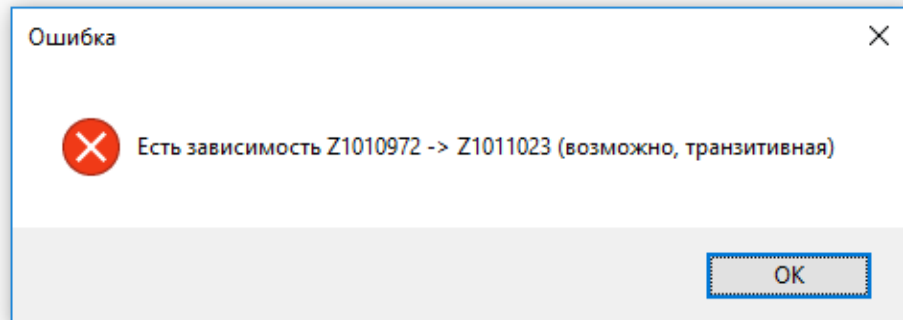


Рис. 5.4. Ошибка при добавлении циклической зависимости между патчами

Такие зависимости учитываются также, если попытаться поменять порядок установки патчей в поставке.

Изменение порядка патчей осуществляется на форме, представленной на рис. 5.5.

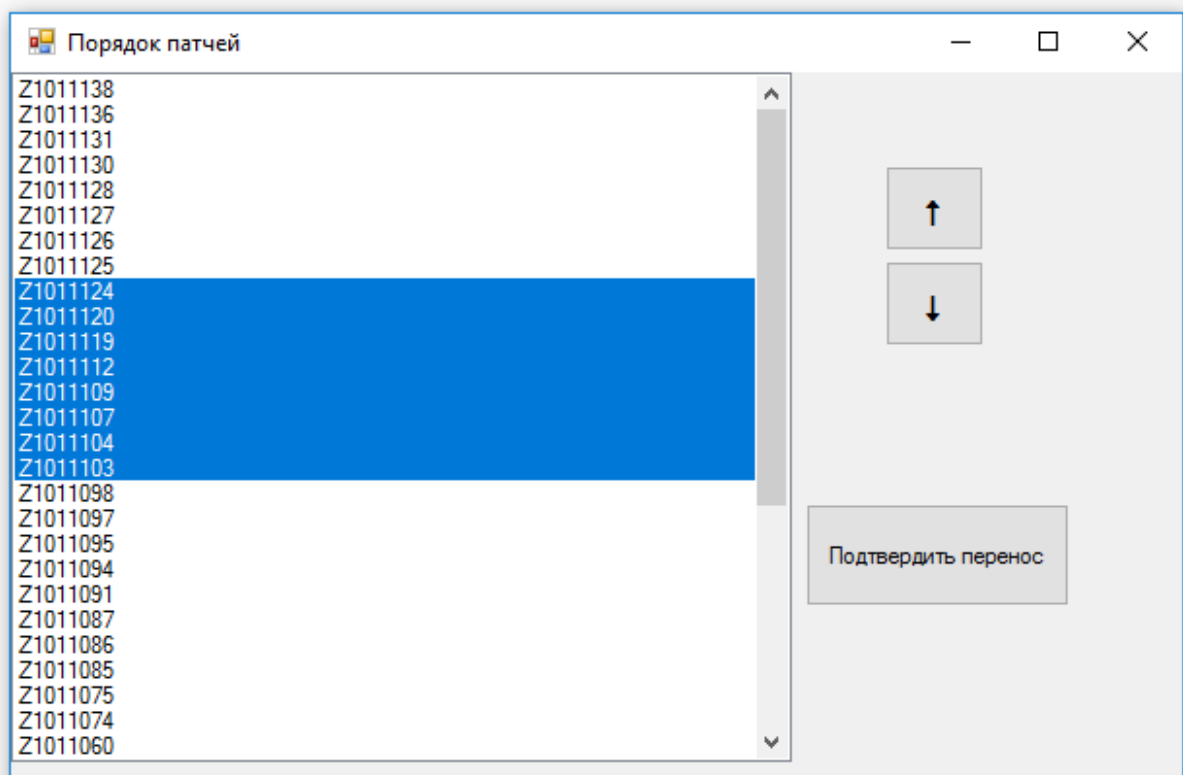


Рис. 5.5. Форма «Порядок патчей»

На форме представлены две кнопки, с помощью которых пользователь может перенести патчи выше и ниже по списку.

Чтобы изменения вступили в силу, пользователь нажимает кнопку «Подтвердить перенос».

В случае, если изменение порядка патчей противоречит зависимостям между ними, пользователь получает ошибку, представленную на рис. 5.6.

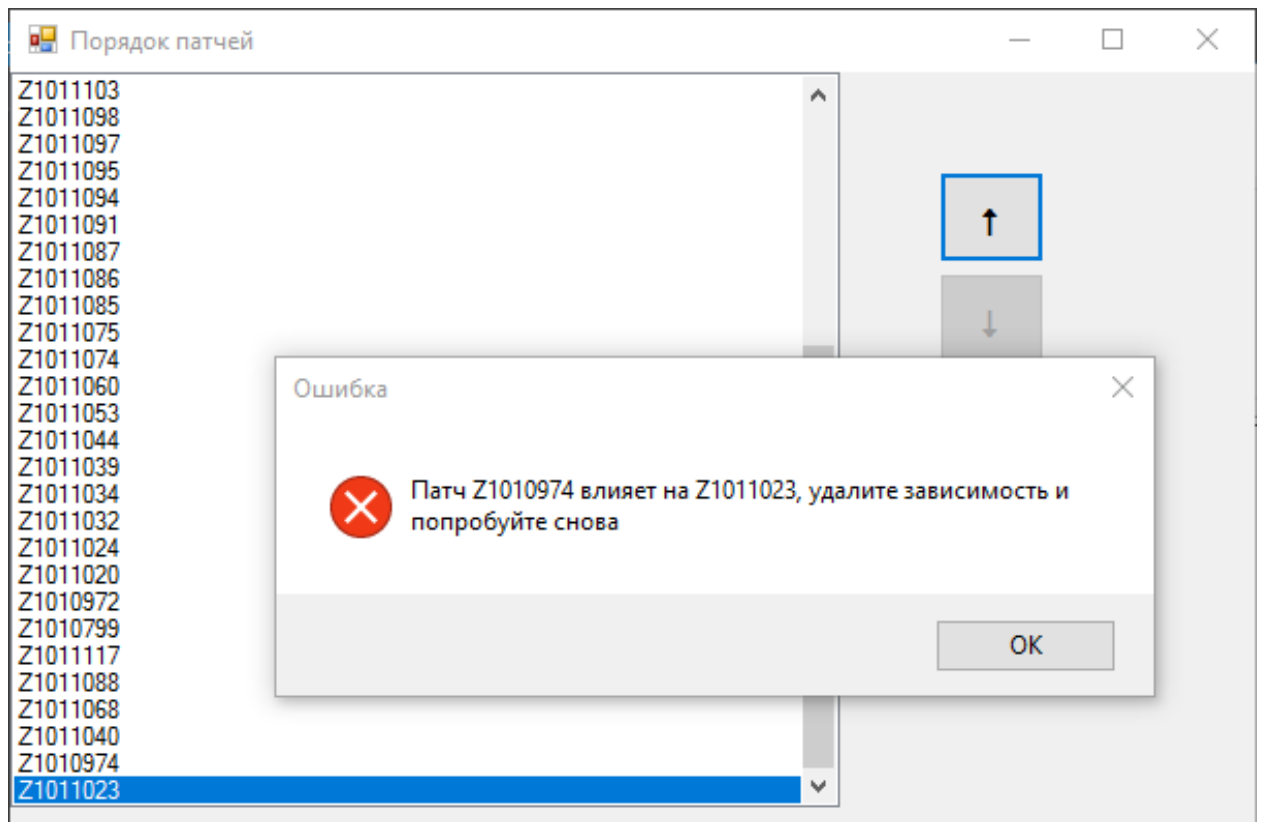


Рис. 5.6. Ошибка при попытке изменения порядка, если полученный порядок противоречит зависимостям

Для того, чтобы наглядно продемонстрировать зависимости между поставками или патчами, существует форма «Граф зависимостей».

Образец внешнего вида просмотра графа зависимостей для патчей представлен на рис. 5.7.

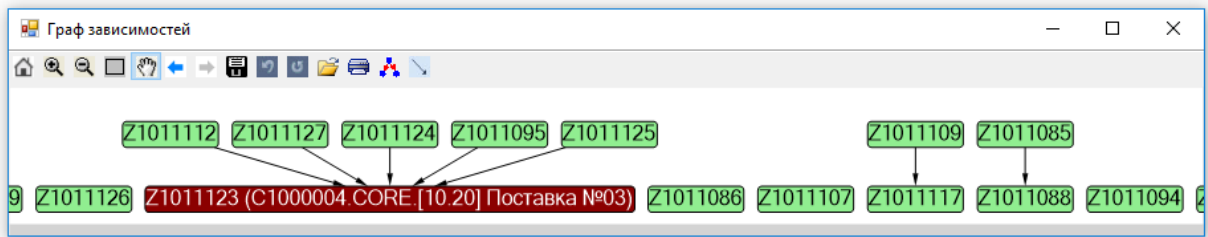


Рис. 5.7. Форма «Граф зависимостей» для патчей

На представленной форме зелеными узлами отмечены патчи выбранной поставки. Если есть зависимости из патчей другой поставки, такие патчи также указаны в графе. Для таких патчей в скобках указывается название поставки, а сам узел отмечается красным цветом.

Образец внешнего вида просмотра графа зависимостей для поставок представлен на рис. 5.8.

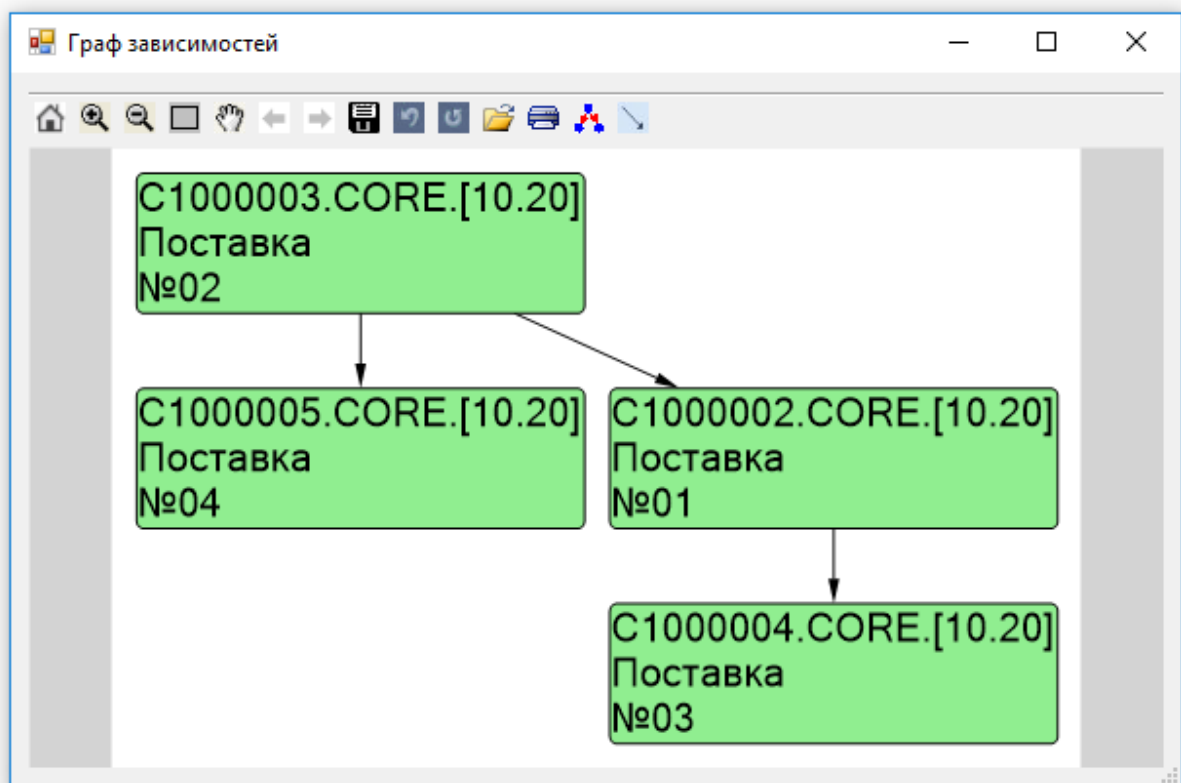


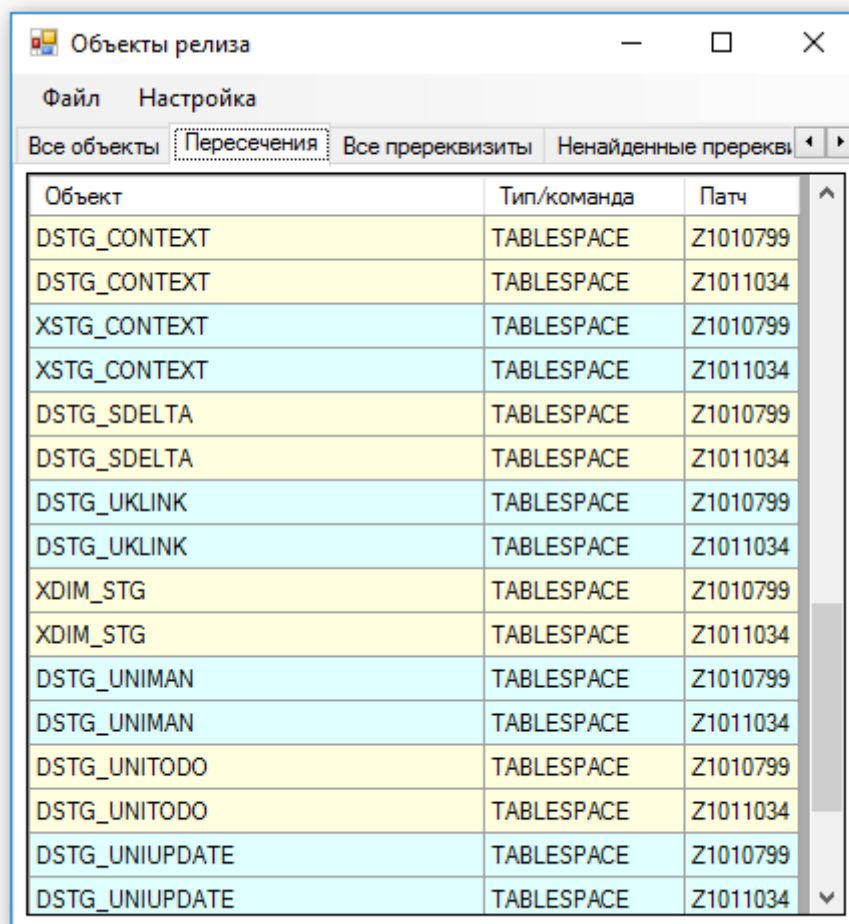
Рис. 5.8. Форма «Граф зависимостей» для поставок

На представленной форме узлами являются поставки выбранного релиза.

5.2. Проверка релиза

Пользователь может проверить, какие объекты релиза использовались несколько раз или найти зависимые объекты релиза.

Образец внешнего вида проверки релиза представлен на рис. 5.9.



The screenshot shows a window titled 'Объекты релиза' (Release Objects) with a menu bar containing 'Файл' (File) and 'Настройка' (Settings). Below the menu bar are four tabs: 'Все объекты' (All objects), 'Пересечения' (Intersections), 'Все пререквизиты' (All prerequisites), and 'Ненайденные пререквизы' (Missing prerequisites). The 'Пересечения' tab is selected. The main area contains a table with three columns: 'Объект' (Object), 'Тип/команда' (Type/command), and 'Патч' (Patch). The table lists 18 rows of data, alternating between yellow and light blue background colors.

Объект	Тип/команда	Патч
DSTG_CONTEXT	TABLESPACE	Z1010799
DSTG_CONTEXT	TABLESPACE	Z1011034
XSTG_CONTEXT	TABLESPACE	Z1010799
XSTG_CONTEXT	TABLESPACE	Z1011034
DSTG_SDELTA	TABLESPACE	Z1010799
DSTG_SDELTA	TABLESPACE	Z1011034
DSTG_UKLINK	TABLESPACE	Z1010799
DSTG_UKLINK	TABLESPACE	Z1011034
XDIM_STG	TABLESPACE	Z1010799
XDIM_STG	TABLESPACE	Z1011034
DSTG_UNIMAN	TABLESPACE	Z1010799
DSTG_UNIMAN	TABLESPACE	Z1011034
DSTG_UNITODO	TABLESPACE	Z1010799
DSTG_UNITODO	TABLESPACE	Z1011034
DSTG_UNIUPDATE	TABLESPACE	Z1010799
DSTG_UNIUPDATE	TABLESPACE	Z1011034

Рис. 5.9. Форма «Объекты релиза»

На форме представлено меню, в котором находятся вкладки:

- «Все объекты» – объекты, включенные в релиз;
- «Пересечения» – скрипты, в которых содержатся изменения этих объектов могут конфликтовать;
- «Все пререквизиты» – все зависимости, выявленные приложением;
- «Ненайденные пререквизиты» – если объект «А» влияет на объект «Б», и при этом объект «А» не был найден, это будет отображено на данной вкладке.

5.3. Редактирование сценария

Пользователю может потребоваться изменить порядок строк в общем сценарии для поставки. Редактирование осуществляется на форме «Сценарий».

Образец внешнего вида формы «Сценарий» представлен на рис. 5.10.

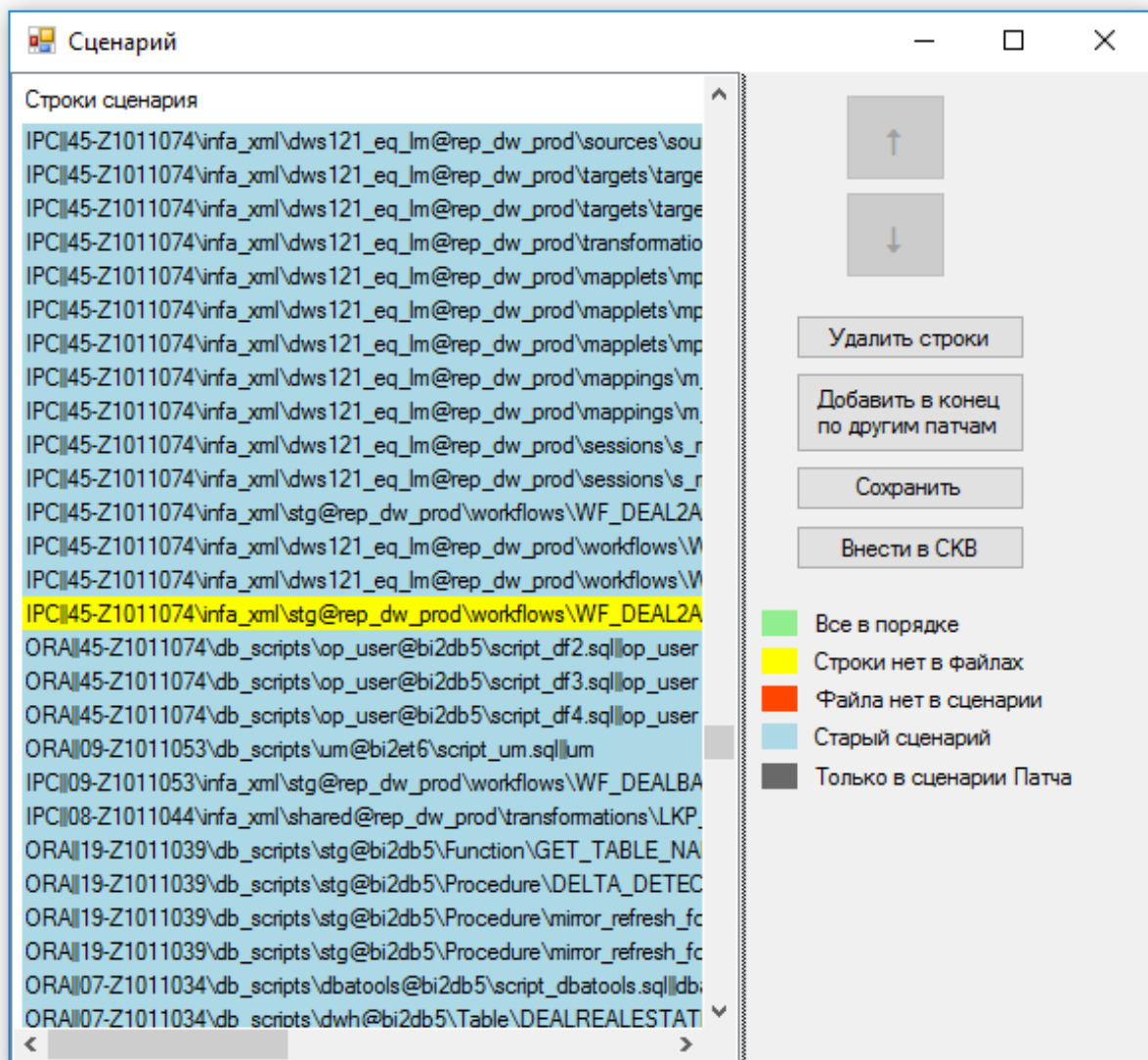


Рис. 5.10. Форма «Сценарий»

Форма содержит следующие элементы управления и отображения информации:

1. Список строк сценария. Каждая строка отмечена определенным цветом, в зависимости от состояния. Легенда представлена на панели справа.

2. Две кнопки, с помощью которых пользователь может перенести строки выше или ниже по списку.
3. Кнопку «Удалить строки», с помощью которой пользователь может исключить строки из сценария.
4. Кнопку «Добавить в конец по другим патчам», с помощью которой пользователь может обновить сценарий, добавив в него строки для патчей, которые были добавлены после последней сборки сценария.
5. Кнопку «Сохранить», с помощью которой пользователь подтверждает изменения.
6. Кнопку «Внести в СКВ», с помощью которой пользователь вносит изменения в систему контроля версий Microsoft Visual SourceSafe.

6. Реализация

6.1. Структура базы данных

На рис. 6.1 представлена логическая модель организации базы данных.

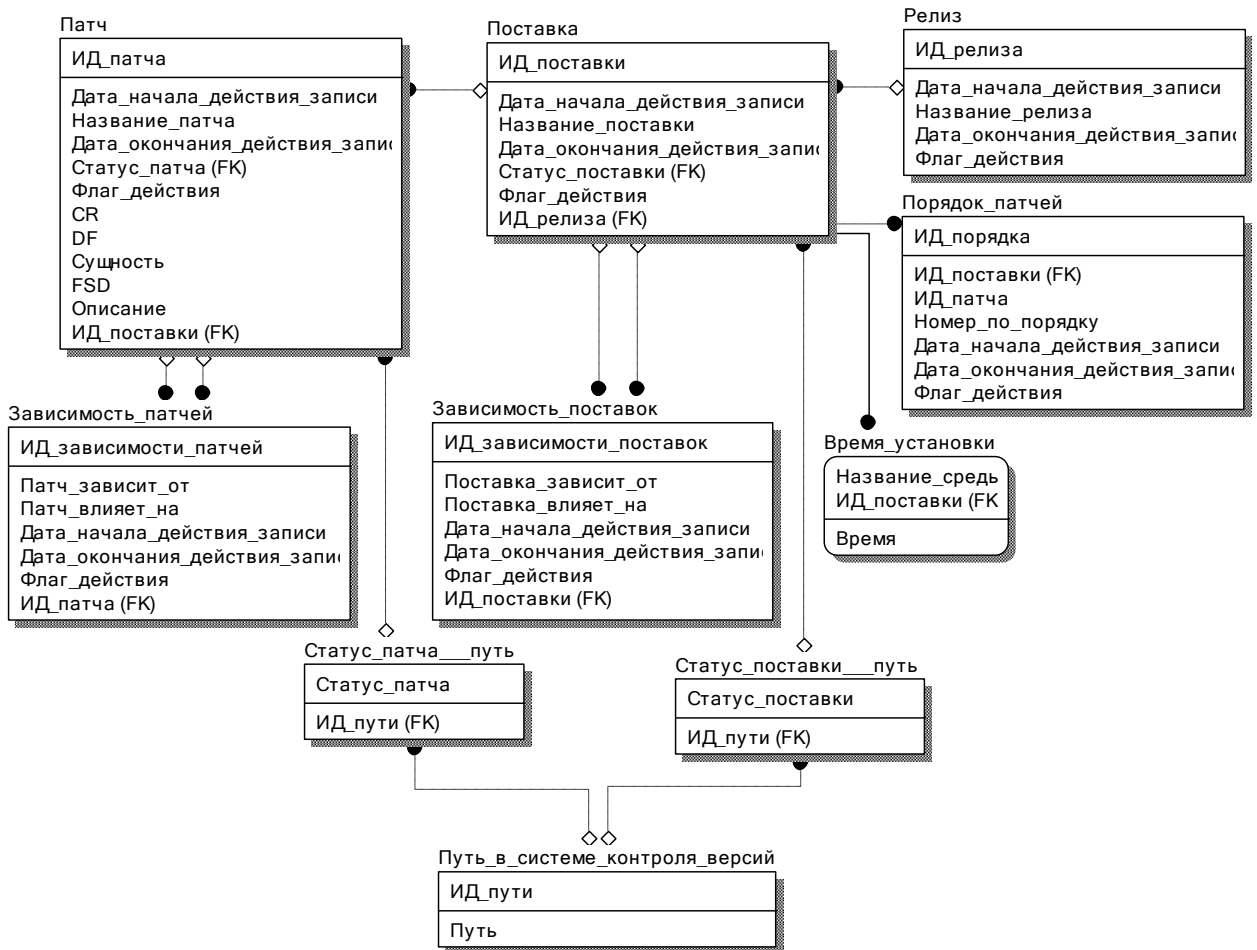


Рис. 6.1. Логическая модель организации БД

6.2. Описание сущностей

Для всех сущностей используются технические атрибуты:

- дата начала действия записи;
- дата окончания действия записи.

Так, например, при изменении записи, в БД фактически вставляется новая запись, и для старой дата окончания действия записи определяется как дата логического изменения старой. Таким образом обеспечивается версионность данных.

Также для всех сущностей используется флаг действия над записью, который определяет, какое действие произведено над записью: добавление, удаление или изменение.

6.2.1. Сущность «Патч»

Сущность «Патч» содержит основную информацию о патче (идентификатор, идентификатор родительского патча, идентификатор поставки, название, дата начала действия записи, дата окончания действия записи, флаг действия).

Таблица 6.1. Сущность «Патч»

Имя	Домен	PK	FK	Описание
Идентификатор патча	Number	+	-	
Идентификатор поставки	Number	-	+	
Название патча	String	-	-	
CR	Number	-	-	Номер изменения (для новых сущностей)
DF	Number	-	-	Номер изменения (для исправления дефектов)
FSD	String	-	-	Ссылка на функциональные требования
Сущность	String	-	-	
Дата начала действия записи	Date	+	-	
Дата окончания действия записи	Date	-	-	
Флаг действия	String	-	-	

6.2.2. Сущность «Поставка»

Сущность «Поставка» содержит основную информацию о поставке (идентификатор, идентификатор родительской поставки, идентификатор релиза, название поставки, идентификатор, идентификатор родительского патча, идентификатор поставки, название, дата начала действия записи, дата окончания действия записи, флаг действия).

Таблица 6.2. Сущность «Поставка»

Имя	Домен	PK	FK	Описание
Идентификатор поставки	Number	+	-	
Идентификатор релиза	Number	-	+	
Статус	String	-	-	
Название поставки	String	-	-	
Дата начала действия записи	Date	+	-	
Дата окончания действия записи	Date	-	-	
Флаг действия	String	-	-	

6.2.3. Сущность «Релиз»

Сущность «Релиз» содержит основную информацию о релизе (идентификатор, название релиза, дата начала действия записи, дата окончания действия записи, флаг действия).

Таблица 6.3. Сущность «Релиз»

Имя	Домен	PK	FK	Описание
Идентификатор релиза	Number	+	-	

Название релиза	String	-	-	
Дата начала действия записи	Date	+	-	
Дата окончания действия записи	Date	-	-	
Флаг действия	String	-	-	

6.3. Программные модули

На рис. 6.2 представлена диаграмма компонентов приложения.

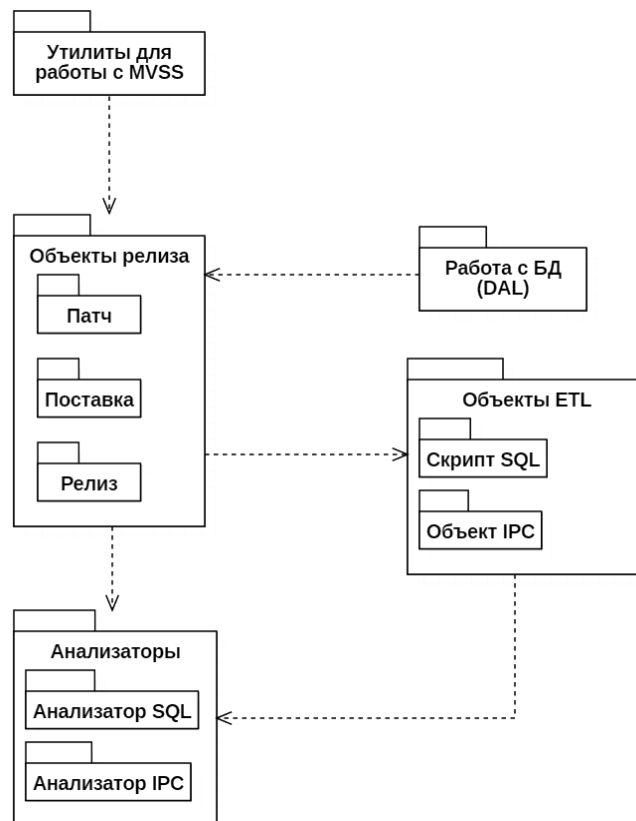


Рис. 6.1. Диаграмма компонентов

Программа состоит из классов, обеспечивающих доступ к базе данных, классов – моделей объектов патча, классов, определяющих зависимости патчей, классов, структурирующих зависимости патчей и форм, предоставляющих взаимодействие пользователя.

Классы для доступа к БД:

- DBManager включает в себя методы для создания строки подключения, открытия и закрытия подключения, выполнения DML операций;
- CPatchDAL, ReleaseDAL, ZPatchDAL включает в себя методы для получения списка записей из базы данных, вставки, добавления, изменения, удаления, поиска записей о поставках, релизах и патчах из базы данных;

Классы моделей CPatch, Release, ZPatch, поля которых соответствуют полям таблиц в базе данных.

Классы, определяющие зависимости:

- SqlParser – класс находит названия и тип объектов СУБД Oracle в скриптах релиза;
- InfaParser – класс определяет зависимости между объектами Informatica PowerCenter.

Классы, организующие зависимости:

- ObjectDict – базовый класс, формирующий список зависимостей, конфликтующие и потерянные зависимости, а также ненайденные влияющие на поставку файлы;
- OraObjectDict – класс-наследник, работающий с зависимостями скриптов Oracle;
- InfaObjectDict – класс-наследник, работающий с зависимостями объектов Informatica PowerCenter.

Утилиты для работы с Microsoft Visual SourceSafe.

Формы:

- ReleaseManagerForm – управление релизом;
- CheckReleaseForm – проверка зависимостей релиза;
- ZPatchOrderForm – управление последовательностью патчей в поставке.

7. План тестирования

Тест 1. Тестирование добавления релиза.

Цель. Проверить корректность добавления записи о релизе в дерево выбора.

Порядок проведения.

1. Выбрать в меню пункт «Релиз».
2. Выбрать подпункт «Добавить».
3. Вписать название в открывшуюся форму.

Результат. В дерево выбора должен добавиться новый пункт с введенным названием, при последующих запусках приложения пункт должен сохраниться.

Тест 2. Тестирование удаления релиза.

Цель. Проверить корректность удаления записи о релизе из дерева выбора.

Порядок проведения.

1. Выбрать удаляемый релиз в дереве выбора.
2. Выбрать в меню пункт «Релиз».
3. Выбрать подпункт «Удалить».

Результат. Из дерева выбора должен удалиться пункт для выбранного узла, при последующих запусках приложения пункт не должен появляться.

Тест 3. Тестирование переименования релиза.

Цель. Проверить корректность переименования записи о релизе в дереве выбора.

Порядок проведения.

1. Выбрать переименовываемый релиз в дереве выбора.
2. Выбрать в меню пункт «Релиз».
3. Выбрать подпункт «Переименовать».

Результат. В дереве выбора название узла должно стать редактируемым. По нажатию кнопки «Ввод» узел должен переименоваться,

при последующих запусках приложения пункт должен сохранить новое название.

Тест 4. Тестирование добавления поставки.

Цель. Проверить корректность добавления записи о поставке в дерево выбора.

Порядок проведения.

1. Выбрать в меню пункт «Релиз».
2. Выбрать подпункт «Добавить поставку в релиз».
3. Вписать название в открывшуюся форму.

Результат. В дерево выбора должен добавиться новый пункт с введенным названием, при последующих запусках приложения пункт должен сохраниться.

Тест 5. Тестирование удаления поставки.

Цель. Проверить корректность удаления записи о поставке из дерева выбора.

Порядок проведения.

1. Выбрать удаляемую поставку в дереве выбора.
2. Выбрать в меню пункт «Поставка».
3. Выбрать подпункт «Удалить».

Результат. Из дерева выбора должен удалиться пункт для выбранного узла, при последующих запусках приложения пункт не должен появляться.

Тест 6. Тестирование переименования поставки.

Цель. Проверить корректность переименования записи о поставке в дереве выбора.

Порядок проведения.

1. Выбрать переименовываемую поставку в дереве выбора.
2. Выбрать в меню пункт «Поставка».
3. Выбрать подпункт «Переименовать».

Результат. В дереве выбора название узла должно стать редактируемым. По нажатию кнопки «Ввод» узел должен переименоваться,

при последующих запусках приложения пункт должен сохранить новое название.

Тест 7. Тестирование переноса поставки.

Цель. Проверить корректность переноса записи о поставке в дереве выбора.

Порядок проведения.

1. Выбрать переносимую поставку в дереве выбора.
2. Выбрать в меню пункт «Поставка».
3. Выбрать подпункт «Перенести».
4. В появившемся списке выбрать релиз, в который переносится поставка.

Результат. В дереве выбора узел поставки должен стать подпунктом релиза, в который поставка переносится. Из релиза, откуда была перенесена поставка, узел должен быть удален. При последующих запусках приложения пункт должен сохранить новое положение в дереве выбора.

Тест 8. Тестирование удаления патча.

Цель. Проверить корректность удаления записи о патче из дерева выбора.

Порядок проведения.

1. Выбрать удаляемый патч в дереве выбора.
2. Выбрать в меню пункт «Патч».
3. Выбрать подпункт «Удалить».

Результат. Из дерева выбора должен удалиться пункт для выбранного узла, при последующих запусках приложения пункт не должен появляться.

Тест 9. Тестирование переименования патча.

Цель. Проверить корректность переименования записи о патче в дереве выбора.

Порядок проведения.

1. Выбрать переименовываемую поставку в дереве выбора.
2. Выбрать в меню пункт «Патч».

3. Выбрать подпункт «Переименовать».

Результат. В дереве выбора название узла должно стать редактируемым. По нажатию кнопки «Ввод» узел должен переименоваться, при последующих запусках приложения пункт должен сохранить новое название.

Тест 10. Тестирование переноса патча.

Цель. Проверить корректность переноса записи о патче в дереве выбора.

Порядок проведения.

1. Выбрать переносимый патч в дереве выбора.
2. Выбрать в меню пункт «Патч».
3. Выбрать подпункт «Перенести».
4. В появившемся списке выбрать поставку, в которую переносится патч.

Результат. В дереве выбора узел патча должен стать подпунктом поставки, в которую патч переносится. Из поставки, откуда был перенесен патч, узел должен быть удален. При последующих запусках приложения пункт должен сохранить новое положение в дереве выбора.

Тест 11. Тестирование построения графа для поставок.

Цель. Проверить корректность построения графа.

Порядок проведения.

1. Выбрать в меню пункт «Релиз».
2. Выбрать подпункт «Построить граф».

Результат. Граф должен содержать все поставки выбранного релиза. Других узлов на графе не должно присутствовать. Для каждой поставки, имеющей зависимости, должна быть стрелка, направленная из влияющей поставки на зависимую.

Тест 12. Тестирование построения графа для патчей.

Цель. Проверить корректность построения графа.

Порядок проведения.

1. Выбрать в меню пункт «Поставка».
2. Выбрать подпункт «Построить граф».

Результат. Граф должен содержать все патчи выбранной поставки, а также патчи, имеющие зависимости с выбранной поставкой. Других узлов на графе не должно присутствовать. Для каждого патча, имеющего зависимости, должна быть стрелка, направленная из влияющего патча на зависимый. Помимо этого, патчи, для патчей, не включенных в выбранную поставку в скобках должно указываться название поставки, а сам узел должен отмечаться красным цветом.

Тест 13. Тестирование изменения порядка патчей.

Цель. Проверить корректность изменения порядка патчей.

Порядок проведения.

1. Выбрать в меню пункт «Поставка».
2. Выбрать подпункт «Порядок патчей».
3. Выбрать патчи.
4. Нажать несколько раз кнопку переноса вверх или вниз по списку.
5. Нажать кнопку «Подтвердить перенос».

Результат. Записи о патчах должны переноситься соответственно для выбранных действий. При этом, если пользователь осуществляет попытку некорректного переноса, приложение должно вывести сообщение об ошибке, а сам перенос не должен быть осуществлен. Перенос считается некорректным, если полученный порядок противоречит зависимостям.

Тест 14. Тестирование создания сценария поставки.

Цель. Проверить создания сценария поставки.

Порядок проведения.

1. Выбрать в меню пункт «Поставка».
2. Выбрать подпункт «Сценарий».

Результат. Файлы поставки должны быть выгружены на компьютер из системы контроля версий, при этом поиск файлов поставки должен осуществляться согласно указанному статусу и готовности поставки. Если

файлы поставки не были найдены, приложение должно вывести сообщение об ошибке, а сам сценарий не должен начать формироваться. Все строки, указанные в старом файле сценария, должны отмечаться синим цветом в списке строк сценария. Если для строки сценария не был обнаружен соответствующий файл, такая строка должна быть отмечена желтым цветом. Если для файла поставки не была обнаружена соответствующая строка в сценарии, такая строка автоматически добавляется в список строк сценария согласно приоритету, при этом строка отмечается красным цветом.

Тест 15. Тестирование изменения сценария поставки.

Цель. Проверить возможность изменения сценария поставки.

Порядок проведения.

1. Выбрать в меню пункт «Поставка».
2. Выбрать подпункт «Сценарий».
3. Нажать на кнопку «Добавить в конец по другим патчам».

Результат. В конец списка строк сценария должны добавиться строки, соответствующие строкам сценария патчей, добавленных после последней сборки сценария. При этом, если для строк сценария были обнаружены соответствующие файлы, такие строки отмечаются зеленым. Так же, если после последней сборки сценария изменились файлы сценария патчей, новые строки должны отмечаться серым цветом и добавляться на позицию, соответствующую сценарию патча.

Тест 16. Тестирование проверки релиза.

Цель. Протестировать корректность проверок релиза.

Порядок проведения.

1. Выбрать в меню пункт «Релиз».
2. Выбрать подпункт «Проверить».

Результат. В появившемся окне в пункте меню «Все объекты» должны присутствовать все объекты релиза. Других записей в списке быть не должно. Для каждого объекта должен быть корректно указан его тип и патч, в котором выбранный объект изменяется.

В пункте меню «Пересечения» должны все находиться объекты, имеющие одинаковое наименование и тип, изменения по которым производятся как минимум в двух патчах. Других записей в списке быть не должно.

В пункте меню «Все пререквизиты» должны присутствовать все объекты Informatica PowerCenter, в которых присутствуют ссылки на другие объекты, и объекты, на которые ведут эти ссылки.

В пункте меню «Ненайденные пререквизиты» должны присутствовать все объекты Informatica PowerCenter, в которых присутствуют ссылки на другие объекты, при этом объекты по этим ссылкам в релизе не выкладываются.

Заключение

Реализована система автоматической работы с патчами.

Для этого были решены следующие задачи:

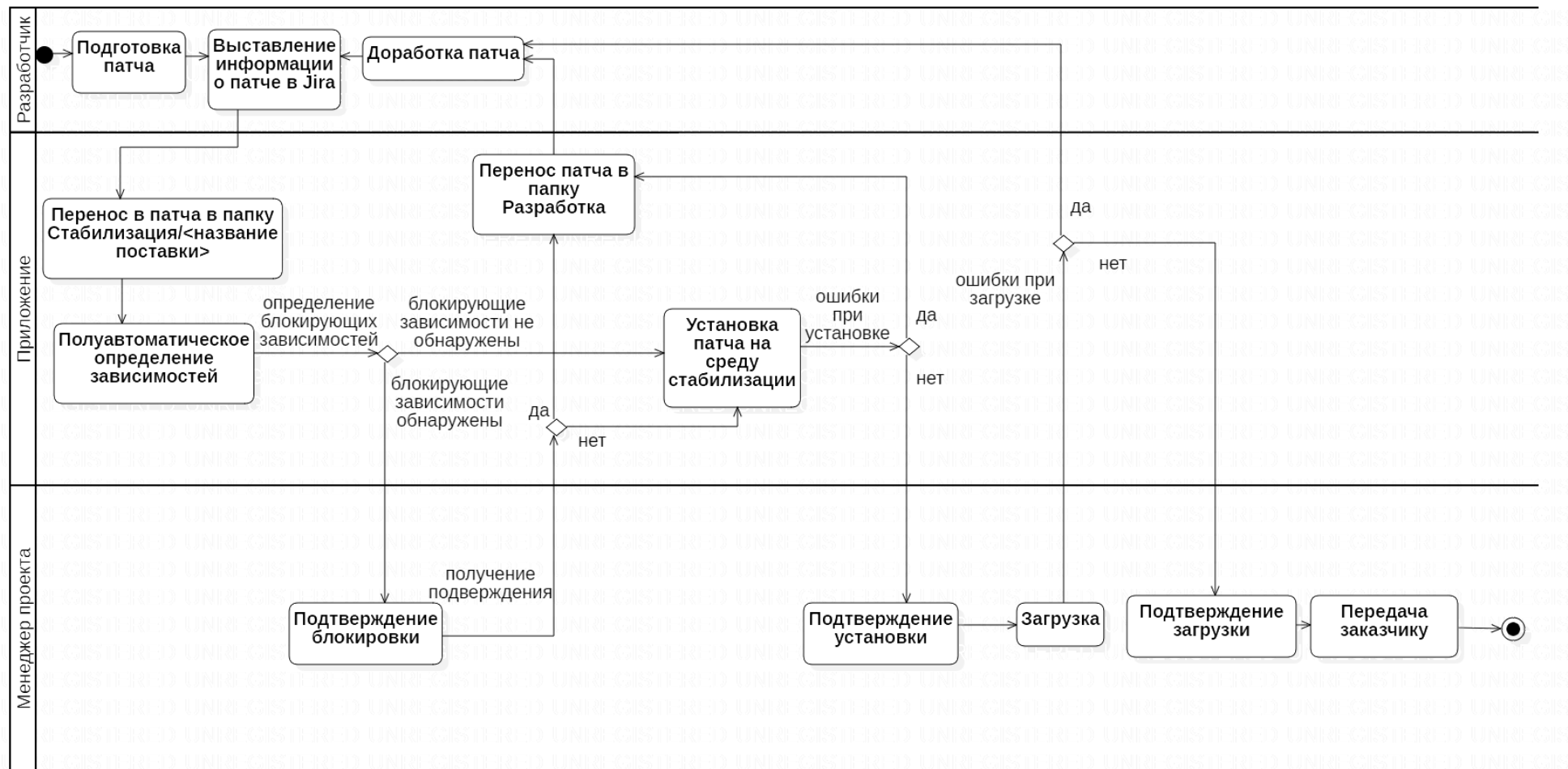
- реализован экспорт из системы Atlassian Jira, для каждого релиза можно получить название патча, затрагиваемую сущность, автора патча, статус готовности и зависимость от других патчей;
- реализована возможность объединить патчи в поставку;
- реализовано автоматическое определение зависимостей между поставками по зависимостям патчей;
- реализована возможность автоматической установки поставки на выбранную среду тестирования;
- осуществлена проверка на то, что поставка была установлена на предыдущую среду тестирования;
- реализовано определение времени установки;
- реализовать определение объектов, затрагиваемых в патче.

В дальнейшем планируется работа по усовершенствованию проекта и внедрению информационных средств.

Литература

1. Язык программирования C# / А. Хейлсберг [и др.]. – 4-е изд. – Санкт-Петербург : Питер, 2012. – 784 с.
2. Ликнесс Д. Приложения для Windows 8 на C# и XAML / Д. Ликнесс – Санкт-Петербург : Питер, 2013. – 368 с.
3. Стиллмен Э. Head First. Изучаем C# / Э. Стиллмен, Д. Грин. – Санкт-Петербург : Питер, 2017. – 816 с.
4. Албахари Д. C# 6.0. Справочник. Полное описание языка / Д. Албахари, Б. Албахари. – Москва : Издательский дом «Вильямс», 2016. – 1040 с.
5. Шилдт Г. C# 4.0: полное руководство / Г. Шилдт. – Москва : Издательский дом «Вильямс», 2011. – 1056 с.
6. Культин Н. Б. Microsoft Visual C# в задачах и примерах / Н. Б. Культин. – Санкт-Петербург : Питер, 2009. – 322 с.
7. Гринвальд Р. Oracle 11g. Основы / Р. Гринвальд, Р. Стаковьяк, Д. Стерн. – Санкт-Петербург : Символ-Плюс, 2009. – 464 с.
8. Кайт Т. Oracle для профессионалов. Архитектура, методики программирования и особенности версий 9i, 10g и 11g / Т. Кайт, Д. Кун – 3-е изд. – Москва ; Санкт-Петербург ; Киев : Вильямс, 2011. – 960 с.
9. Oracle PL/SQL для профессионалов: практические решения / К. МакДональд, Х. Кац [и др.] . – Москва : ДиаСофт, 2005. – 560 с.
10. Informatica Documentation Portal – URL: <https://docs.informatica.com/> (дата обращения 04.05.2019).
11. Kimball R. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data / J. Caserta, R. Kimball – New Jersey : Wiley, 2004. – P. 467
12. Основные функции ETL-систем – URL: <https://habrahabr.ru/post/248231/> (дата обращения 04.05.2019).
13. Коновалов М. В. ETL: обзор и роль в развитии компаний – URL: <https://moluch.ru/conf/tech/archive/286/13149/> (дата обращения: 12.06.2019).

Приложение 1. Диаграмма деятельности



Приложение 2. Класс «CPatch»

```
public class CPatch
{
    public string C { get; private set; }
    public string FullName { get; private set; }
    public string CPatchName { get; private set; }
    public string CPatchStatus { get; private set; }
    public DirectoryInfo Dir { get; private set; }
    public string cvsPath;
    public static CVS.CVS cvs;

    public List<ZPatch> ZPatches { get; protected set; }
    public Dictionary<int, ZPatch> ZPatchesDict { get; protected
set; }
    public HashSet<CPatch> DependenciesFrom { get; protected set; }
}

    public HashSet<CPatch> DependenciesTo { get; protected set; }
    public Release release;
    public string KodSredy { get; set; }
    public SortedList<int, ZPatch> ZPatchOrder { get; private set; }
}

    public int CPatchId { get; private set; }

    public static IEnumerable<string> cpatchStatuses;
    public static IEnumerable<string> cpatchEnvCodes;

    static CPatch()
    {
        cpatchStatuses = CPatchStateDAL.GetStatuses();
        cpatchEnvCodes = CPatchStateDAL.GetEnvCodes();
    }

    public override int GetHashCode()
    {
        return CPatchId.GetHashCode();
    }

    public override bool Equals(object obj)
    {
        if (obj == null) return false;
        if (obj.GetType() != typeof(CPatch)) return false;
        return ((CPatch)obj).CPatchId == CPatchId;
    }

    public ZPatch GetZPatchById(int id)
    {
        foreach(CPatch cpatch in release.CPatches)
        {
            if (cpatch.ZPatchesDict.ContainsKey(id))
            {
                return cpatch.ZPatchesDict[id];
            }
        }
        return null;
    }
}
```

```

    public override string ToString()
    {
        return CPatchName;
    }

    private void InitFromDB(int CPatchId, string CPatchName,
string CPatchStatus, string KodSredy)
    {
        this.CPatchId = CPatchId;
        this.CPatchName = CPatchName;
        this.CPatchStatus = CPatchStatus;
        this.KodSredy = KodSredy;
    }

    public void Delete()
    {
        foreach(CPatch cpatchFrom in DependenciesFrom)
        {
            cpatchFrom.DependenciesTo.Remove(this);
            CPatchDAL.DeleteDependency(cpatchFrom.CPatchId,
CPatchId);
        }

        foreach(CPatch cpatchTo in DependenciesTo)
        {
            cpatchTo.DependenciesFrom.Remove(this);
            CPatchDAL.DeleteDependency(CPatchId,
cpatchTo.CPatchId);
        }

        CPatchDAL.DeleteCPatch(CPatchId);
        release.CPatches.Remove(this);
        release.CPatchesDict.Remove(CPatchId);
    }

    public static bool CanDeleteCPatchDependency(CPatch
cpatchFrom, CPatch cpatchTo, out ZPatch zpatchFromDependency, out
ZPatch zpatchToDependency)
    {
        foreach (var zpatchFrom in cpatchFrom.ZPatches)
        {
            foreach (var zpatchTo in cpatchTo.ZPatches)
            {
                if (zpatchFrom.DependenciesTo.Contains(zpatchTo))
                {
                    zpatchFromDependency = zpatchFrom;
                    zpatchToDependency = zpatchTo;

                    return false;
                }
            }
        }

        zpatchFromDependency = zpatchToDependency = null;
        return true;
    }

```

```

public void DeleteDependencyFrom(CPatch cpatchFrom)
{
    DependenciesFrom.Remove(cpatchFrom);
    cpatchFrom.DependenciesTo.Remove(this);
    CPatchDAL.DeleteDependency(cpatchFrom.CPatchId, CPatchId);
}

public void DeleteDependencyTo(CPatch cpatchTo)
{
    DependenciesTo.Remove(cpatchTo);
    cpatchTo.DependenciesFrom.Remove(this);
    CPatchDAL.DeleteDependency(CPatchId, cpatchTo.CPatchId);
}

public void AddDependencyFrom(CPatch cpatchFrom)
{
    DependenciesFrom.Add(cpatchFrom);
    cpatchFrom.DependenciesTo.Add(this);
    CPatchDAL.AddDependency(cpatchFrom.CPatchId, CPatchId);
}

public void AddDependencyTo(CPatch cpatchTo)
{
    DependenciesTo.Add(cpatchTo);
    cpatchTo.DependenciesFrom.Add(this);
    CPatchDAL.AddDependency(CPatchId, cpatchTo.CPatchId);
}

public static bool HaveTransitiveDependency(CPatch cpatchFrom,
CPatch cpatchTo)
{
    foreach (CPatch subPatch in cpatchFrom.DependenciesTo)
    {
        if (subPatch.Equals(cpatchTo))
            return true;
        else
            return HaveTransitiveDependency(subPatch,
cpatchTo);
    }

    return false;
}

public void Move(Release newRelease)
{
    if (release != newRelease)
    {
        release.CPatches.Remove(this);
        release.CPatchesDict.Remove(CPatchId);

        newRelease.CPatches.Add(this);
        newRelease.CPatchesDict.Add(CPatchId, this);

        CPatchDAL.UpdateRelease(CPatchId,
newRelease.ReleaseId);
    }
}

```

```

public void InitZPatches()
{
    ZPatches = new List<ZPatch>();
    ZPatchesDict = new Dictionary<int, ZPatch>();
    var oraZPatchesRecords =
ZPatchDAL.GetZPatchesByCPatch(CPatchId);

    foreach (var oraZPatchRecord in oraZPatchesRecords)
    {
        ZPatch zpatch = new ZPatch(
            this,
            oraZPatchRecord.ZPatchName,
            oraZPatchRecord.ZPatchId,
            oraZPatchRecord.ZPatchStatus);

        ZPatches.Add(zpatch);
        ZPatchesDict.Add(zpatch.ZPatchId, zpatch);

        zpatch.cpatch = this;
    }
}

public void ResetStatusesByLog()
{
    foreach(ZPatch zpatch in ZPatches)
    {
        if (ZPatchDAL.IsZPatchInstalled(zpatch.ZPatchName,
KodSredy))
        {
            zpatch.ZPatchStatus = "INSTALLED";
            ZPatchDAL.UpdateStatus(zpatch.ZPatchId,
zpatch.ZPatchStatus);
        }
    }
}

public void SetDependencies()
{
    DependenciesFrom = new HashSet<CPatch>();

    var oraCPatchesDependenciesFrom =
CPatchDAL.GetDependenciesFrom(CPatchId);

    foreach (var oraCPatchRecord in
oraCPatchesDependenciesFrom)
    {
        DependenciesFrom.Add(release.GetCPatchById(oraCPatchRecord.CPatchId));
    }

    DependenciesTo = new HashSet<CPatch>();

    var oraCPatchesDependenciesTo =
CPatchDAL.GetDependenciesTo(CPatchId);

    foreach (var oraCPatchRecord in oraCPatchesDependenciesTo)
    {

```



```

DependenciesTo.Add(release.GetCPatchById(oraCPatchRecord.CPatchId));
    }

    private string GetCVSPath()
    {
        string cvsRoot = CPatchStateDAL.GetCVSPath(CPatchStatus,
KodSredy);
        string shortName = Regex.Match(CPatchName, @"C\d+").Value;

        try
        {

            string cvsFolder = cvs.FirstInEntireBase(cvsRoot, out
string cpatchmatch, new Regex(shortName), 2);

            foreach (ZPatch zpatch in ZPatches)
            {
                if (zpatch.ZPatchStatus != "OPEN")
                {
                    string patchRoot = $"{cvsRoot}/{cpatchmatch}";
                    try
                    {
                        cvs.FirstInEntireBase(patchRoot, out
string zpatchmatch, new Regex(zpatch.ZPatchName), 1);
                        zpatch.Dir = new
DirectoryInfo(Path.Combine(Dir.FullName, zpatchmatch));
                    }
                    catch
                    {
                        throw new
DirectoryNotFoundException($"Патч {zpatch.ZPatchName} не найден.
"Добавьте его в папку C-патча или переведите статус в OPEN");
                    }
                }
            }

            return cvsFolder;
        }
        catch (ArgumentException e)
        {
            throw new DirectoryNotFoundException("Папка с C-патчем
не найдена. Возможно, назначен неправильный статус или среда");
        }
    }

    public string Download()
    {
        cvsPath = GetCVSPath();
        SetAttributesNormal(Dir);
        cvs.Pull(cvsPath, Dir);
        return cvsPath;
    }

    private void DeleteLocal()
    {

```

```

        if (Dir.Exists)
        {
            SetAttributesNormal(Dir);
            Dir.Delete(true);
        }
    }

    private void SetAttributesNormal(DirectoryInfo dir)
    {
        if (dir.Exists)
        {
            foreach (var subDir in dir.GetDirectories())
                SetAttributesNormal(subDir);
            foreach (var file in dir.GetFiles())
            {
                file.Attributes = FileAttributes.Normal;
            }
        }
    }

    public CPatch(int CPatchId, string CPatchName, string
CPatchStatus, string KodSredy, Release release)
    {
        this.release = release;
        InitFromDB(CPatchId, CPatchName, CPatchStatus, KodSredy);

        Dir = new
DirectoryInfo(Path.Combine(release.rm.homeDir.FullName, CPatchName));

        InitZPatches();
        InitZPatchOrder();
    }

    private void InitZPatchOrder()
    {
        ZPatchOrder = new SortedList<int, ZPatch>();
        var zpatchOrderRecords =
ZPatchOrderDAL.GetZPatchOrdersByCPatch(CPatchId);
        foreach (ZPatchOrderRecord patchOrderRecord in
zpatchOrderRecords)
        {
            ZPatchOrder.Add(patchOrderRecord.zpatchOrder,
ZPatchesDict[patchOrderRecord.zpatchId]);
        }
    }

    private void AddNewDependenciesToList(List<ZPatch>
addedPatches)
    {
        var fullList = AllDependenciesToList();
        int i = fullList.Count - addedPatches.Count;

        foreach (var patch in fullList.Values)
        {
            if (addedPatches.Contains(patch))
            {
                ZPatchOrder.Add(i, patch);
                ZPatchOrderDAL.Insert(patch.ZPatchId, i);
            }
        }
    }

```

```

        i++;
    }
}

private void ResetAllDependenciesToList()
{
    var sourceList = AllDependenciesToList();

    SortedList<int, ZPatch> list = new SortedList<int,
ZPatch>();

    int i = 0;
    foreach (var item in sourceList)
    {
        list.Add(i++, item.Value);
    }

    foreach (var item in list)
    {
        ZPatchOrderDAL.Insert(item.Value.ZPatchId, item.Key);
    }
}

public void UpdateZPatchOrder(ZPatch zpatch, int order)
{
    if (ZPatchOrder[order] != zpatch)
    {
        ZPatchOrder[order] = zpatch;
        ZPatchOrderDAL.Update(zpatch.ZPatchId, order);
    }
}

private SortedList<int, ZPatch> AllDependenciesToList()
{
    List<ZPatch> roots = new List<ZPatch>();
    foreach (ZPatch patch in ZPatches)
    {
        if (patch.DependenciesFrom.Count == 0)
        {
            roots.Add(patch);
        }
    }

    foreach (ZPatch root in roots)
    {
        root.rank = 0;
        SetChildrenRanks(root);
    }

    SortedList<int, ZPatch> sourceList = new SortedList<int,
ZPatch>(Comparer<int>.Create((x, y) => x < y ? -1 : 1));

    foreach (ZPatch zpatch in ZPatches)
    {
        sourceList.Add(zpatch.rank, zpatch);
    }
}

```

```

        return sourceList;
    }

    public void Rename(string newName)
    {
        if(newName != null && newName != CPatchName)
        {
            CPatchName = newName;
            CPatchDAL.UpdateName(CPatchId, CPatchName);
        }
    }

    public void ChangeRelease(Release newRelease)
    {
        if(release.ReleaseId != newRelease.ReleaseId)
        {
            release.CPatches.Remove(this);
            release.CPatchesDict.Remove(CPatchId);

            newRelease.CPatches.Add(this);
            newRelease.CPatchesDict.Add(CPatchId, this);

            CPatchDAL.UpdateRelease(CPatchId,
newRelease.ReleaseId);
        }
    }

    private void SetChildrenRanks(ZPatch currPatch)
    {
        foreach (ZPatch subpatch in currPatch.DependenciesTo)
        {
            subpatch.rank = Math.Max(subpatch.rank, currPatch.rank
+ 1);

            SetChildrenRanks(subpatch);
        }
    }

    public void ReopenExcelColumns(FileInfo excelFile)
    {
        Workbook wb =
ReleaseManager.excelApp.Workbooks.Open(excelFile.FullName);
        Worksheet ws = wb.Sheets[1];
        Range res = ws.UsedRange;

        CreateCPatchDelta(res, out List<ZPatch> addedPatches);
        CreateCPatchDependenciesDelta(
            res,
            out List<Tuple<ZPatch, ZPatch>>
deletedDependenciesFrom,
            out List<Tuple<ZPatch, ZPatch>> addedDependenciesFrom,
            out List<Tuple<ZPatch, ZPatch>> deletedDependenciesTo,
            out List<Tuple<ZPatch, ZPatch>> addedDependenciesTo);

        InsertCPatchDelta(
            deletedDependenciesFrom,

```

```

        addedDependenciesFrom,
        deletedDependenciesTo,
        addedDependenciesTo,
        addedPatches);

    ExcelCleanup(wb, ws);
}

private void InsertCPatchDelta(
    List<Tuple<ZPatch, ZPatch>> deletedDependenciesFrom,
    List<Tuple<ZPatch, ZPatch>> addedDependenciesFrom,
    List<Tuple<ZPatch, ZPatch>> deletedDependenciesTo,
    List<Tuple<ZPatch, ZPatch>> addedDependenciesTo,
    List<ZPatch> addedPatches)
{
    foreach (var deletedDependency in deletedDependenciesFrom)
    {
        ZPatchDAL.DeleteDependency(deletedDependency.Item2.ZPatchId,
            deletedDependency.Item1.ZPatchId);

        if (deletedDependency.Item2.cpatch.CPatchId !=
            CPatchId)
        {
            bool canDeleteCDependency = true;
            foreach (ZPatch zPatch in ZPatches)
            {
                foreach (ZPatch dependency in
                    zPatch.DependenciesFrom)
                {
                    if (dependency.cpatch.CPatchId ==
                        deletedDependency.Item2.cpatch.CPatchId)
                    {
                        canDeleteCDependency = false;
                        break;
                    }
                }
                if (!canDeleteCDependency) break;
            }

            if (canDeleteCDependency)
            {
                DependenciesFrom.Remove(deletedDependency.Item2.cpatch);

                deletedDependency.Item2.cpatch.DependenciesTo.Remove(this);

                CPatchDAL.DeleteDependency(CPatchId,
                    deletedDependency.Item2.cpatch.CPatchId);
            }
        }
    }

    foreach (var addedDependency in addedDependenciesFrom)
    {

```

```

ZPatchDAL.AddDependency (addedDependency.Item2.ZPatchId,
addedDependency.Item1.ZPatchId);
    if (addedDependency.Item2.cpatch.CPatchId != CPatchId)
    {
        if
(!DependenciesFrom.Contains (addedDependency.Item2.cpatch))
        {

DependenciesFrom.Add (addedDependency.Item2.cpatch);

addedDependency.Item2.cpatch.DependenciesTo.Add (this);

CPatchDAL.AddDependency (addedDependency.Item2.cpatch.CPatchId,
CPatchId);
        }
    }

    foreach (var deletedDependency in deletedDependenciesTo)
    {

ZPatchDAL.DeleteDependency (deletedDependency.Item1.ZPatchId,
deletedDependency.Item2.ZPatchId);

    if (deletedDependency.Item2.cpatch.CPatchId !=
CPatchId)
    {
        bool canDeleteCDependency = true;
        foreach (ZPatch zPatch in ZPatches)
        {
            foreach (ZPatch dependency in
zPatch.DependenciesTo)
            {
                if (dependency.cpatch.CPatchId ==
deletedDependency.Item2.cpatch.CPatchId)
                {
                    canDeleteCDependency = false;
                    break;
                }
            }
            if (!canDeleteCDependency) break;
        }

        if (canDeleteCDependency)
        {

DependenciesTo.Remove (deletedDependency.Item2.cpatch);

deletedDependency.Item2.cpatch.DependenciesFrom.Remove (this);

CPatchDAL.DeleteDependency (deletedDependency.Item2.cpatch.CPatchId,
CPatchId);
        }
    }
}

```

```

    }

    foreach (var addedDependency in addedDependenciesTo)
    {
        ZPatchDAL.AddDependency(addedDependency.Item1.ZPatchId,
            addedDependency.Item2.ZPatchId);
        if (addedDependency.Item2.cpatch.CPatchId != CPatchId)
        {
            if
            (!DependenciesFrom.Contains(addedDependency.Item2.cpatch))
            {
                DependenciesTo.Add(addedDependency.Item2.cpatch);

                addedDependency.Item2.cpatch.DependenciesFrom.Add(this);

                CPatchDAL.AddDependency(CPatchId,
                    addedDependency.Item2.cpatch.CPatchId);
            }
        }

        foreach(ZPatch z1 in ZPatches)
        {
            foreach(ZPatch z2 in ZPatches)
            {
                if(ZPatch.HaveTransitiveDependency(z1, z2) &&
                    ZPatch.HaveTransitiveDependency(z2, z1))
                {
                    throw new Exception($"Одновременные
зависимости {z1} -> {z2} и {z2} -> {z1}");
                }
            }
        }

        bool oldDependenciesAreCorrect = true;

        foreach (var newDependency in addedDependenciesFrom)
        {
            if (addedPatches.Contains(newDependency.Item1) &&
                !addedPatches.Contains(newDependency.Item2))
            {
                oldDependenciesAreCorrect = false;
                break;
            }

            if (ZPatchOrder.IndexOfValue(newDependency.Item1) >
                ZPatchOrder.IndexOfValue(newDependency.Item2))
            {
                oldDependenciesAreCorrect = false;
                break;
            }
        }

        if (oldDependenciesAreCorrect)
            foreach (var newDependency in addedDependenciesTo)

```

```

        {
            if (!addedPatches.Contains(newDependency.Item1) &&
addedPatches.Contains(newDependency.Item2))
            {
                oldDependenciesAreCorrect = false;
                break;
            }

            if (ZPatchOrder.IndexOfValue(newDependency.Item1)
< ZPatchOrder.IndexOfValue(newDependency.Item2))
            {
                oldDependenciesAreCorrect = false;
                break;
            }
        }

        if (oldDependenciesAreCorrect)
        {
            AddNewDependenciesToList(addedPatches);
        }
        else
        {
            ResetAllDependenciesToList();
        }
    }

    private void ExcelCleanup(Workbook wb, Worksheet ws)
    {
        GC.Collect();
        GC.WaitForPendingFinalizers();

        Marshal.FinalReleaseComObject(ws);

        wb.Close(Type.Missing, Type.Missing, Type.Missing);
        Marshal.FinalReleaseComObject(wb);
    }

    private CPatch()
    { }

    public static CPatch CreateNewFromExcel(Release release,
FileInfo excelFile)
    {
        CPatch empty = new CPatch
        {
            ZPatches = new List<ZPatch>(),
            ZPatchesDict = new Dictionary<int, ZPatch>(),

            DependenciesFrom = new HashSet<CPatch>(),
            DependenciesTo = new HashSet<CPatch>(),
            ZPatchOrder = new SortedList<int, ZPatch>(),
            release = release
        };

        empty.release.CPatches.Add(empty);
    }

```



```

        empty.ReopenExcelColumns(excelFile);
        empty.release.CPatchesDict.Add(empty.CPatchId, empty);
        return empty;
    }

    private static readonly string regexZPatchName = @"(Z) [0-9]+";
    private static readonly string regexCPatchName = @"(C) [0-9]+";

    private static readonly string regexFrom =
"зависит.*?ALFAM.*?([0-9]+)";
    private static readonly string regexTo =
"влияет.*?ALFAM.*?([0-9]+)";

    private HashSet<ZPatch> GetDependenciesFrom(string rawString)
    {
        HashSet<ZPatch> res = new HashSet<ZPatch>();
        MatchCollection matchesFrom = Regex.Matches(rawString,
regexFrom);
        foreach (Match m in matchesFrom)
        {
            if (FindPatchByShortName(m.Groups[1].Value, out ZPatch
zPatch))
            {
                res.Add(zPatch);
            }
        }
        return res;
    }

    private HashSet<ZPatch> GetDependenciesTo(string rawString)
    {
        HashSet<ZPatch> res = new HashSet<ZPatch>();
        MatchCollection matchesTo = Regex.Matches(rawString,
regexTo);
        foreach (Match m in matchesTo)
        {
            if (FindPatchByShortName(m.Groups[1].Value, out ZPatch
zpatch))
            {
                res.Add(zpatch);
            }
        }
        return res;
    }

    private bool SameEnding(string s1, string s2)
    {
        for (int i = 1; i <= Math.Min(s1.Length, s2.Length); i++)
        {
            if (s1[s1.Length - i] != s2[s2.Length - i]) return
false;
        }

        return true;
    }

    private bool FindPatchByShortName(string shortName, out ZPatch
patch)

```

```

{
    foreach (CPatch cpatch in release.CPatches)
    {
        try
        {
            patch = cpatch.ZPatches.First(x =>
SameEnding(x.ZPatchName, shortName));
            return true;
        }
        catch { }
    }
    patch = null;
    return false;
}

private int GetPatchNameIndex(Range columns)
{
    for (int i = 1; i <= columns.Columns.Count; ++i)
    {
        string currCell = ((Range)columns.Cells[1, i]).Value2;
        if (currCell == "Тема")
        {
            return i;
        }
    }
    throw new KeyNotFoundException("Колонка с именем патча не
найдена");
}

private int GetPatchStatusIndex(Range columns)
{
    for (int i = 1; i <= columns.Columns.Count; ++i)
    {
        string currCell = ((Range)columns.Cells[1, i]).Value2;
        if (currCell == "Статус")
        {
            return i;
        }
    }
    throw new KeyNotFoundException("Колонка со статусом не
найдена");
}

private int GetLinkIndex(Range columns)
{
    for (int i = 1; i <= columns.Columns.Count; ++i)
    {
        string currCell = ((Range)columns.Cells[1, i]).Value2;
        if (currCell == "Issue Link Type")
        {
            return i;
        }
    }
    throw new KeyNotFoundException("Колонка с зависимостями не
найдена");
}

public void UpdateStatus(string newStatus)

```

```

    {
        if (newStatus != CPatchStatus)
        {
            CPatchStatus = newStatus;
            CPatchDAL.UpdateStatus(CPatchId, newStatus);
        }
    }

    public void UpdateEnvCode(string newEnvCode)
    {
        if (newEnvCode != KodSredy)
        {
            KodSredy = newEnvCode;
            CPatchDAL.UpdateEnvCode(CPatchId, newEnvCode);
        }
    }

    private void CreateCPatchDelta(Range columns, out List<ZPatch>
addedPatches)
    {
        addedPatches = new List<ZPatch>();
        HashSet<ZPatch> deletedPatches = new
HashSet<ZPatch>(ZPatches);

        int patchNameIndex = GetPatchNameIndex(columns);

        int patchStatusIndex = GetPatchStatusIndex(columns);
        if (CPatchId == 0)
        {
            CPatchName = "NOT DEFINED";
            CPatchId = CPatchDAL.Insert(release.ReleaseId, null,
CPatchName, null, null);
        }

        for (int i = 2; i <= columns.Rows.Count; ++i)
        {
            string patchCell = ((Range)columns.Cells[i,
patchNameIndex]).Value2 ?? "";
            string excelStatus = ((Range)columns.Cells[i,
patchStatusIndex]).Value2;
            string ZPatchStatus = "UNDEFINED";

            if (excelStatus == "Открытый")
                ZPatchStatus = "OPEN";
            else if (excelStatus == "Testing" || excelStatus ==
"Waiting bank confirm")
                ZPatchStatus = "READY";
            else if (excelStatus == "Installed to STAB" ||
excelStatus == "Installed to STAB2")
                ZPatchStatus = "INSTALLED";

            MatchCollection matches = Regex.Matches(patchCell,
regexZPatchName);
            if (matches.Count == 0)
            {
                matches = Regex.Matches(patchCell,
regexCPatchName);
                if (matches.Count > 0)

```

```

        {
            if (CPatchName == "NOT DEFINED")
            {
                CPatchName = patchCell;
                CPatchDAL.UpdateName(CPatchId, patchCell);
            }
        }
    else
    {
        string patchName = Regex.Match(patchCell,
regexZPatchName).Value;
        if (!FindPatchByShortName(patchName, out ZPatch
currPatch))
        {
            if (ZPatchStatus != "OPEN")
            {
                ZPatch zpatch = new ZPatch(
                    patchName,
                    CPatchId,
                    new HashSet<ZPatch>(),
                    new HashSet<ZPatch>(),
                    ZPatchStatus)
                {
                    cpatch = this
                };

                addedPatches.Add(zpatch);
                zpatch.excelFileRowId = i;
            }
            else
            {
                deletedPatches.Remove(currPatch);
                currPatch.excelFileRowId = i;
            }
        }
    }

    foreach(ZPatch zpatch in addedPatches)
    {
        zpatch.ZPatchId = ZPatchDAL.Insert(CPatchId, null,
zpatch.ZPatchName, null);
        ZPatches.Add(zpatch);
        ZPatchesDict.Add(zpatch.ZPatchId, zpatch);
    }

    foreach(ZPatch zpatch in deletedPatches)
    {
        ZPatchDAL.DeleteZPatch(zpatch.ZPatchId);
        ZPatches.Remove(zpatch);
        ZPatchesDict.Remove(zpatch.ZPatchId);
    }
}

public Graph DrawGraph()
{
    Graph graph = new Graph();

```

```

foreach (ZPatch zpatch in ZPatches)
{
    Microsoft.Msagl.Drawing.Node node = new
Microsoft.Msagl.Drawing.Node(zpatch.ZPatchId.ToString());
    node.Attr.FillColor = Color.LightGreen;
    node.Label.Text = zpatch.ZPatchName;
    graph.AddNode(node);
}

foreach (ZPatch zpatch in ZPatches)
{
    foreach (ZPatch depFrom in zpatch.DependenciesFrom)
    {
        if(depFrom.cpatch != this)
        {
            Microsoft.Msagl.Drawing.Node node = new
Microsoft.Msagl.Drawing.Node(depFrom.ZPatchId.ToString());
            node.Attr.FillColor = Color.DarkRed;
            node.Label.FontColor = Color.White;
            node.LabelText = $"{depFrom.ZPatchName}
({depFrom.cpatch.CPatchName})";
            graph.AddNode(node);
        }
        graph.AddEdge(depFrom.ZPatchId.ToString(),
zpatch.ZPatchId.ToString());
    }

    foreach (ZPatch depTo in zpatch.DependenciesTo)
    {
        if (depTo.cpatch != this)
        {
            Microsoft.Msagl.Drawing.Node node = new
Microsoft.Msagl.Drawing.Node(depTo.ZPatchId.ToString());
            node.Attr.FillColor = Color.DarkRed;
            node.Label.FontColor = Color.White;
            node.LabelText = $"{depTo.ZPatchName}
({depTo.cpatch.CPatchName})";
            graph.AddNode(node);
            graph.AddEdge(zpatch.ZPatchId.ToString(),
depTo.ZPatchId.ToString());
        }
    }
}

return graph;
}

private void CreateCPatchDependenciesDelta(
    Range columns,
    out List<Tuple<ZPatch, ZPatch>> deletedDependenciesFrom,
    out List<Tuple<ZPatch, ZPatch>> addedDependenciesFrom,
    out List<Tuple<ZPatch, ZPatch>> deletedDependenciesTo,
    out List<Tuple<ZPatch, ZPatch>> addedDependenciesTo)
{
    deletedDependenciesFrom = new List<Tuple<ZPatch,
ZPatch>>();
    addedDependenciesFrom = new List<Tuple<ZPatch, ZPatch>>();
    deletedDependenciesTo = new List<Tuple<ZPatch, ZPatch>>();
}

```

```

addedDependenciesTo = new List<Tuple<ZPatch, ZPatch>>();

int linkIndex = GetLinkIndex(columns);
if (linkIndex != -1)
{
    foreach (ZPatch zpatch in ZPatches)
    {
        string dependenciesCell =
            ((Range) columns.Cells[zpatch.excelFileRowId, linkIndex]).Value2 ?? "";
        var dependenciesFrom =
            GetDependenciesFrom(dependenciesCell);

        foreach (ZPatch excelFromDependency in
            dependenciesFrom)
        {
            if
                (!zpatch.DependenciesFrom.Contains(excelFromDependency))
            {
                addedDependenciesFrom.Add(new
                    Tuple<ZPatch, ZPatch>(zpatch, excelFromDependency));

                zpatch.DependenciesFrom.Add(excelFromDependency);
                excelFromDependency.DependenciesTo.Add(zpatch);
            }
        }

        foreach (ZPatch patchFromDependency in
            zpatch.DependenciesFrom)
        {
            if
                (!dependenciesFrom.Contains(patchFromDependency))
            {
                deletedDependenciesFrom.Add(new
                    Tuple<ZPatch, ZPatch>(zpatch, patchFromDependency));

                zpatch.DependenciesFrom.Remove(patchFromDependency);
                patchFromDependency.DependenciesTo.Remove(zpatch);
            }
        }

        var dependenciesTo =
            GetDependenciesTo(dependenciesCell);

        foreach (ZPatch excelToDependency in
            dependenciesTo)
        {
            if
                (!zpatch.DependenciesTo.Contains(excelToDependency))
            {
                addedDependenciesTo.Add(new Tuple<ZPatch,
                    ZPatch>(zpatch, excelToDependency));
            }
        }
    }
}

```

```

zpatch.DependenciesTo.Add(excelToDependency);

excelToDependency.DependenciesFrom.Add(zpatch);
    }
}

        foreach (ZPatch patchToDependency in
zpatch.DependenciesTo)
        {
            if
(!dependenciesTo.Contains(patchToDependency))
            {
                deletedDependenciesFrom.Add(new
Tuple<ZPatch, ZPatch>(zpatch, patchToDependency));

zpatch.DependenciesTo.Remove(patchToDependency);

patchToDependency.DependenciesFrom.Remove(zpatch);
            }
        }
    }
}

```