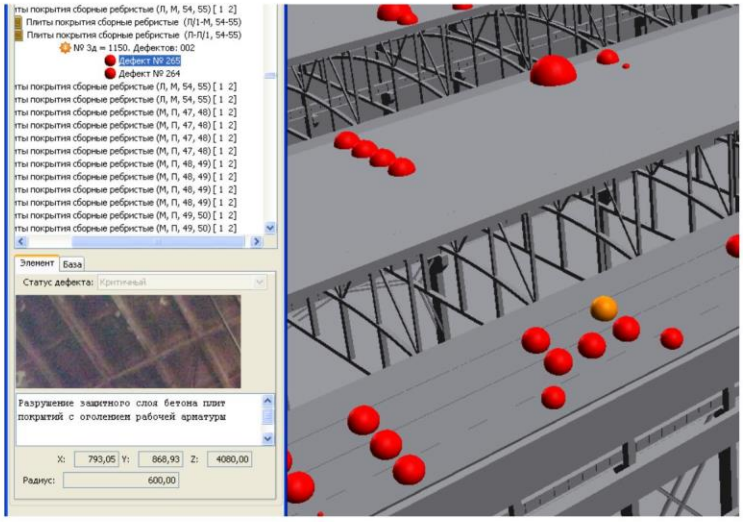
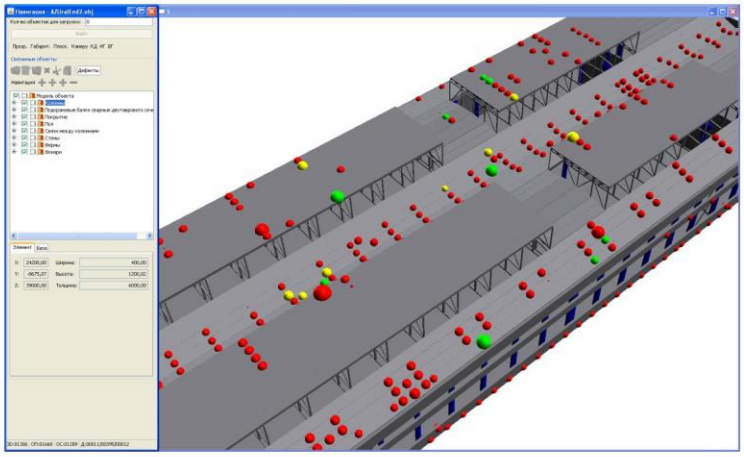
5/1 Декабрь 2012 г.

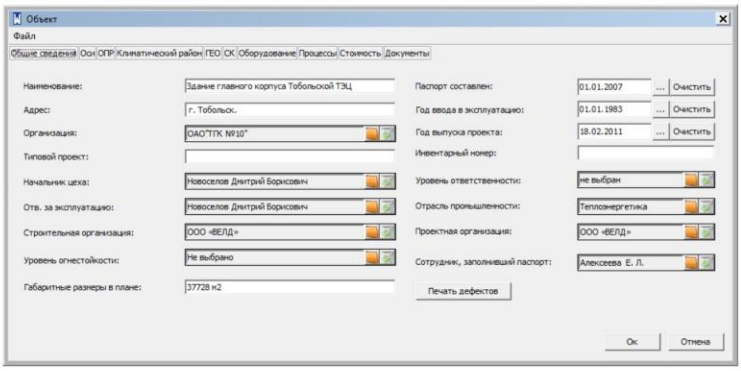
НАУКА И БЕЗОПАСНОСТЬ

Science and Safety

Ежеквартальный научный журнал







Разрушающиеся строения

Электронный паспорт

Оценка вероятностей аварий на плотинах

Восстановление культовых зданий

Сборно-монолитная составная свая

АНО "НИИ "ПРОМЫШЛЕННАЯ БЕЗОПАСНОСТЬ"
НИИ ПБ
г.Магнитогорск

НАУКА БЕЗОПАСНОСТЬ

ISSN 2225-0360

СОДЕРЖАНИЕ

Шишкина Н.А.
Разрушающиеся строения – угроза для подростков **3**

Сорокин М.О., Макшанов Е.О., Алексеев О.Г.
Концепция и система электронного паспорта строительного объекта **7**

Сорокин М.О., Ткаченко М.С.
Методология и инструментарий разработки информационной системы электронного паспорта **33**

Сорокин М.О., Гаврилкович В.И.
Проектирование и реализация информационной системы электронного паспорта **62**

Сорокин М.О., Алексеев О.Г.
Опыт применения информационных моделей электронного паспорта на промышленных объектах **89**

Стефанишин Д.В.
Сценарный подход к оценке вероятностей аварий на плотинах **103**

Пунгин В.Л., Пунгина Т.В.
Возможности восстановления культовых зданий **112**

Пермяков М.Б., Веселов А.В., Токарев А.А.
Сборно-монолитная составная свая **116**

Наши авторы **124**

РАЗРУШАЮЩИЕСЯ СТРОЕНИЯ – УГРОЗА ДЛЯ ПОДРОСТКОВ

УДК 69.059
Шишкина Н.А.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область
Shishkina N.A.
LLC «WELD», Magnitogorsk, Chelyabinsk region

Аннотация
В работе заостряется внимание на незаконном использовании детского труда. Приведены несколько кратких описаний трагических случаев, где при демонтаже строительных конструкций гибнут подростки. Предлагаются способы борьбы с безответственностью.
Summary
The work emphasizes the unlawful use of child labour. Are some brief descriptions of tragic cases where the dismantling of building structures are dying teenagers. Suggests ways of dealing with irresponsibility.

Ключевые слова:
Причины смерти подростков, социальная проблема, обрушение, заброшенное здание, использование детского труда, контроль состояния зданий, смерть по неосторожности.
Key words:
The cause of death of teenagers, a social problem, collapse, abandoned building, child labour, monitoring the condition of buildings, death by negligence.

В наше время, когда речь заходит о причинах смерти подростков, мы, не задумываясь, называем наркотические средства. Да! Никто не спорит. Никто не спорит и о том, что очень много жизней забирают дорожно-транспортные происшествия. И ненадлежащий медицинский уход. И разного рода подростковые войны, неразделенная первая любовь и еще многое другое. Но, увы, это далеко не все.

Но мы сегодня затронем несколько иную тему. Кратко опишем одну из социальных проблем, следствием которой является смерть подростков по неосторожности. Да. Как ни странно, смерть при выполнении трудовых обязательств.

Не так давно в электронных средствах массовой информации прочитала новостную заметку «В Оренбуржье из-за обрушения стены мастерской погиб ученик». Трагедия произошла в декабре 2011 года в поселке Вьюжный при демонтаже строительных конструкций заброшенного здания мастерской. Во время извлечения железнодорожного рельса, вмонтированного в качестве балки-перекрытия над проёмом гаражных дверей, произошло обрушение стены здания. Заброшенная мастерская была одноэтажным строением, выполненным из бетонных плит и шлакоблоков, высота дверного проёма в месте обрушения составляла пять метров. Под завалами оказались двое подростков, один из которых погиб на месте происшествия. Факт того, что демонтаж был незаконным, налицо. Кто отправит на такого рода работы подростков? Возникает вопрос. С какой целью ребята разбирали стены?



Рис. 1. Обрушение стены заброшенного здания мастерской в пос.Вьюжный (Кувандыкский район, Оренбургская область) 09 декабря 2011 года

Заинтересовавшись этим вопросом, увидела, что эта трагедия – далеко не единичный случай. Понятно, что при демонтаже вышедших из эксплуатации зданий гибнут рабочие. Это зачастую банальное несоблюдение техники безопасности, невнимательность, расторопность… Но что на таких объектах делают дети? Ответ не заставил себя долго ждать. Введя в строку поисковика слова Демонтаж и Подростки, нашла еще одно описание подобного несчастного случая. В августе 2011 года в селе Шапошниково той же, кстати, области во время разбора строительных конструкций погиб подросток. Двое семнадцатилетних юношей разбирали здание с целью добычи строительных материалов для перепродажи. Часть железобетонной стены обрушилась и придавила одного из юных старателей.



Рис. 2. Обрушение стены двухэтажного заброшенного здания в селе Шапошниково (Первомайский район, Оренбургская область) 22 августа 2011 года

Или еще вот случай из найденных без особого труда. Привожу из новостной заметки почти дословно. «…В ноябре 2010 года в селе Мирное Ставропольского края обрушилась стена нежилого дома. Инициаторами аварии стали подростки, которые самовольно демонтировали стену с целью добычи строительных материалов. Под грудой обвалившегося кирпича оказался четырнадцатилетний подросток. От полученных травм он скончался на месте.»

Обратите внимание, что подобные трагедии происходят совсем не в городской местности. Конечно, ментальности городских и сельских поселений несколько

различаются, в селах дети имеют больше свободы и, рано взрослея, самостоятельно стараются заработать на карманные расходы, а порою и на жизнь, ведь в последние несколько десятков лет жизнь в деревне оставляет желать лучшего. Из-за отсутствия работы многие родители спиваются, что наталкивает на определенные выводы. Детьми заниматься особенно некому, они предоставлены сами себе. И хорошо, если они самоорганизованно гоняют мяч на школьном стадионе. Но нередки случаи, описанные выше. Много ли можно выручить за строительные материалы, добытые таким вот образом? Для знающих людей не секрет, что кирпич, бывший в потреблении, стоит половину, а то и треть стоимости нового. Немного. Но все же выручка. Куда тратят подростки эти деньги? Пиво? Сигареты? Наркотики? Или реальная помощь безработным родителям? Почему же тогда сами родители не идут на разборки заброшенных зданий?

«…Май 2011 года. Вологодская область. В поселке Октябрьский при демонтаже старой котельной обрушилась стена. Под завалами оказались трое подростков. Двое из них, мальчики десяти и четырнадцати лет, погибли на месте. Третьего пострадавшего с травмами ног госпитализировали. Авария произошла, когда дети пытались вытаскивать из тела каменной кладки кирпичи. Таким образом они пытались заработать. Ведется расследование инцидента. Ищут заказчика работ.» Это дословная информация из ленты происшествий одного из специальных сайтов. Здесь уже констатируется факт заказа дешевых строительных материалов. Не скрывается это и при описании трагедий произошедших в июне 2010 года в г.Кунгуре Пермского края, где предприимчивый гражданин даже сколотил бригаду из подростков, и в июле 2011 года в Красном Сулине Ростовской области, где для заготовки блоков в помещении бывшего огнеупорного цеха школьники были наняты самим хозяином помещения.



Рис. 3. Обрушение крыши помещения неработающего цеха в г. Красный Сулин (Ростовская область) 21 июля 2011 года

Только подумайте! Налицо использование детского труда. Люди, желающие получить почти бесплатный строительный материал для своих дач, сараек и прочего, сознательно отправляют детей на опасный промысел. Дети в большинстве случаев не понимают, что ради копеечной выручки рискуют жизнью?

А что думают взрослые? Преследуя получение дешевого строительного материала, они забывают, что используют детский труд и толкают детей на смерть. Это причинение вреда и даже смерти по неосторожности. Наплевательское отношение к

чужим жизням говорит о полнейшей деградации населения. Если взрослые ни в чем, кроме выгоды, не заинтересованы, что же можно говорить о подростках, о детях? Каким будет поколение, растущее в развалинах? Ведь, взрослея и набираясь опыта, каждый из нас ориентируется на поколение, показывающее направление.

Что тут делать? Как предотвратить подобные преступления? Понятно, что объекты незавершенного строительства и выведенные из эксплуатации различного рода сооружения необходимо охранять с большей тщательностью. Свободного доступа к ним и быть не должно. Усилив контроль за ними и наложив на их владельцев уголовную ответственность за возможные подобные несчастные случаи, можно сохранить не одну детскую жизнь. Кроме того, надо срочно ограждать растущее поколение с еще слабо сформировавшимся мировоззрением от жестоких и алчных личностей. Ужесточив уголовное законодательство в части преступлений против детей, уже можно многого добиться. Страх лишения свободы на долгие годы зачастую отрезвляет мошенников и заставляет их задуматься над своими желаниями. То же касается и педофилов, и малоответственных родителей.

Библиографический список

1. Информационный портал «Наука и безопасность». Электронный ресурс: http://pamag.ru/fatal/1

КОНЦЕПЦИЯ И СИСТЕМА ЭЛЕКТРОННОГО ПАСПОРТА СТРОИТЕЛЬНОГО ОБЪЕКТА

УДК 004.9

Сорокин М.О.
ФГБОУ ВПО «Московский государственный строительный университет», г.Москва
Макшанов Е.О., Алексеев О.Г.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Sorokin M.O.
Moscow State Construction University, Moscow
Makshanov E.O., Alekseev O.G.
«WELD» company, Magnitogorsk, Chelyabinsk region

Аннотация
В работе обосновывается концепция электронного паспорта строительного объекта, описано, какие возможности предоставляет электронный паспорт, каким интерфейсом он обладает, и какие важные моменты обеспечивает.

Ключевые слова:
Электронный паспорт, Интерфейс программы, Дефекты и повреждения, 3D-модуль, Здания и сооружения, Обработка информации, База данных, Справочник, Техническое состояние, Работа с объектами
Key words:
The electronic passport, the program Interface, Defects and damage, 3D module, Buildings and structures, information Processing, Database, Guide, the Technical condition, Working with objects

В 2009 году специалисты ООО «ВЕЛД» разработали, запатентовали и зарегистрировали право на выпуск программного продукта «Электронный паспорт зданий и сооружений». Этот продукт стал первым из ряда планируемых к выпуску программных комплексов, направленных на эффективное управление промышленными технологическими рисками на предприятиях всех отраслей экономики. Толчком для создания новой программы стала актуальная проблема аварий эксплуатируемых промышленных зданий, связанных с повреждением оборудования и нарушением технологического процесса.

Что представляет собой электронный паспорт? Как он работает? Что нужно для его полноценного функционирования? Рассмотрим по порядку.

На всех этапах жизненного цикла необходима информационная поддержка. Ее высокая эффективность обусловлена функционированием и взаимодействием различных информационных систем, образующих единое информационное пространство. Отдельные системы формируют и оперируют данными частной информационной модели объекта, актуальной в контексте функционального назначения системы. Совокупность частных информационных моделей представляет собой интегрированную модель. Таким образом, первым этапом предлагаемой новой методики является разработка частной информационной модели промышленного здания.

Исходной информацией для инженерных исследований технического состояния здания является проектная документация и результаты натурных исследований. На их основе предлагается формирование частной информационной модели эксплуатируемого промышленного здания (рис. 1), представляющей «электронный паспорт» промышленного здания. Технические аспекты разработки системы, реализующей представленную модель, рассмотрены в [1].



Рис. 1. Электронный паспорт промышленного здания

Разработка данной модели позволяет в структурированном и четко определенном формате получить полный набор данных, характеризующий состояние конструкций строительного объекта, включая информацию о дефектах и повреждениях. Эти данные являются исходной информацией для анализа технического состояния, поэтому их необходимо поддерживать в актуальном состоянии. Например, устранение дефектов в ходе ремонтов или возникновение новых должно отражаться в системе. Кроме того, для обеспечения безопасности необходима своевременная достоверная оценка технического состояния, а также обнаружение перехода в опасное состояние. Поэтому при обнаружении дефекта необходимо оперативно оценить его значимость и категорию опасности. Процесс автоматизации обработки информации о дефекте и принятия решения о категории дефекта представлен на рис. 2.

При первоначальном вводе информации выполняется идентификация конструкции, вида дефекта и его параметров. Таким образом, выполняется актуализация частной информационной модели: строительная конструкция в совокупности с дефектами отражает её действительное состояние. После выполнения актуализации необходима оценка опасности дефекта. Такая оценка может быть осуществлена на основании сопоставления вида дефекта с типом строительной конструкции. Например, если конструкции, имеющие различное функциональное назначение имеют однотипные дефекты, то и влияние таких дефектов на безопасность здания будет различна. Однако в ряде случаев решение об опасности и категории дефекта может быть принято только на основании поверочного расчета. Так, влияние изменения прочностной характеристики несущей конструкции можно оценить, лишь

выполнив соответствующий расчет. Расчет осуществляется во внешней системе, для этого из частной информационной модели выполняется экспорт расчетной схемы и фактических параметров конструкций с учетом дефектов. Результат расчета загружается в систему, а назначение категории дефекта осуществляется вручную, со ссылкой на данные соответствующего расчета. Следующим этапом анализа состояния объекта является оценка влияния дефекта на безопасность здания в целом (рис. 3).



Рис. 2. Автоматизированная аналитическая обработка сведений о дефекте

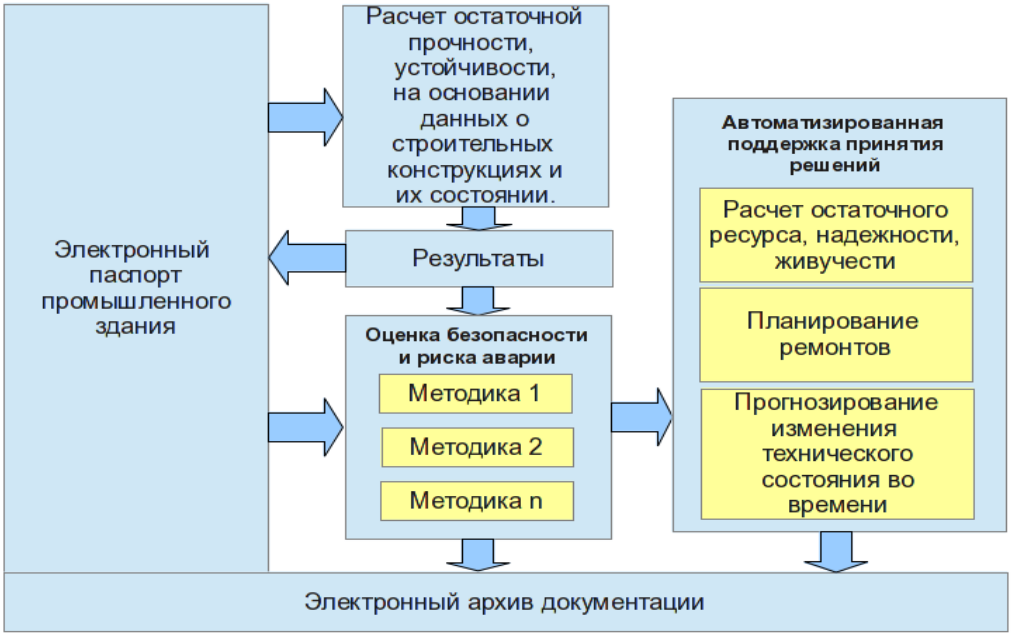


Рис. 3. Аналитическая обработка и актуализация модели

Для этого выполняются расчеты остаточной прочности, устойчивости. Полученные результаты загружаются в систему и служат исходной информацией для оценки безопасности и риска аварии на основании авторских методик. Результаты аналитической обработки фиксируются как в численном виде в качестве параметров модели, так и в виде соответствующих документов в архиве документации. Блок по оценке безопасности и риска аварии предлагает возможности оценки состояния здания по нескольким авторским методикам. При реализации возможна как разработка модулей, предоставляющих автоматизированную обработку данных о состоянии здания, так и модулей, реализующих механизмы взаимодействия с существующими программами. Благодаря применению различных методик достигается повышение достоверности результата.

Представленные схемы позволяют разработать автоматизированную систему оценки безопасности эксплуатируемого здания. Минимизируя информационную энтропию, связанную с неопределенностью технического состояния здания, представляется возможным прогнозировать изменение состояния и принимать решения, связанные с управлением комплексом мероприятий, направленных на поддержание технического состояния здания на приемлемом уровне. Кроме того, благодаря единому информационному пространству, такие прогнозы могут корректироваться после выполнения ремонтов и устранения дефектов.

Для реализации представленного подхода предлагается разработка сетевой информационной системы. Сервер системы должен находиться в распоряжении службы эксплуатации объекта. Представители экспертных организаций получают доступ к данным системы через сеть передачи данных (локальные или глобальные сети). Кроме того, возможно предоставление авторизованного доступа к данным системы страховым организациям и надзорным службам (рис. 4).

Таким образом, становится возможной разработка единых стандартов информационного обмена между участниками процессов декларирования и контроля безопасности объектов. Кроме того, использование интегрированных систем позволит повысить эффективность работ. С точки зрения безопасности и защищенности объектов от аварий, положительный эффект достигается за счет механизмов поддержки принятия решений, включающих:

1. Обработку количественных и качественных значений показателей технического состояния по различным методикам.
2. Применение и апробацию авторских методик для обработки данных информационных моделей и оценки безопасности, риска, ресурса, надежности, живучести.
3. Классификацию дефектов по категории опасности и оценку влияния дефектов на безопасность здания.

Система электронной паспортизации очень логична. Рассмотрим, какие она предоставляет возможности, каким интерфейсом обладает, затронем еще некоторые важные моменты.

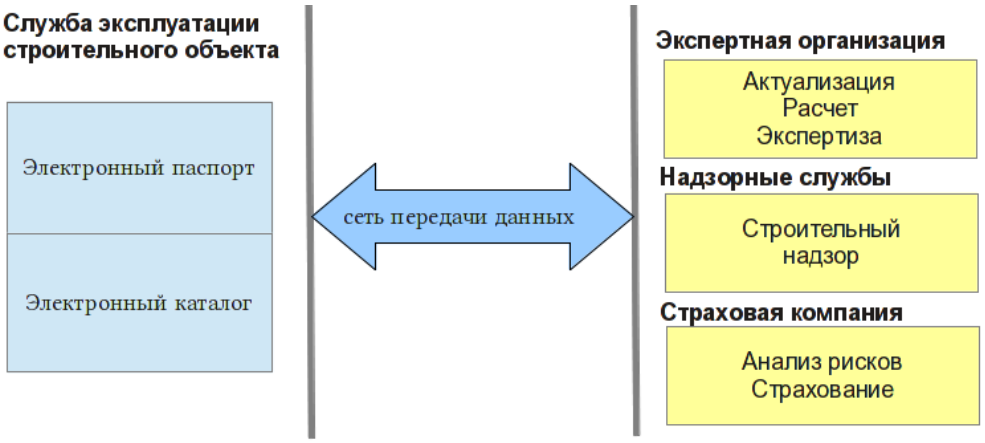


Рис. 4. Организационная схема использования системы

1. Описание программного комплекса «Электронный паспорт»

Итак, что же такое электронный паспорт и как он работает?

Электронный паспорт – это программный комплекс, предназначенный для накопления, хранения и анализа общей, технической, технологической, эксплуатационной информации по зданиям и сооружениям, информации о техническом состоянии строительных конструкций, ведения учета, контроля и управления процессами, связанными с оценкой и проверкой технического состояния строительных конструкций, анализа и прогнозирования технического состояния строительных конструкций.

Цель комплекса – организация эффективной системы сбора, хранения и отображения данных о зданиях и сооружениях. Для выполнения данных функций созданы единое централизованное хранилище документов, модуль просмотра объекта в трехмерном режиме, а также модуль паспортных данных.

Единое хранилище документов (Электронный каталог) позволяет избежать многократное копирование документов и, как следствие, нерациональное использование ресурсов, обеспечивает автоматизированный ввод и регистрацию документов и файлов. Благодаря системе распределения прав доступа, электронный каталог позволяет организовать коллективный доступ к документам, разграничив полномочия пользователей. В программе предусмотрен журнал событий, в который заносятся все значимые действия пользователей над документами.

Модуль просмотра объекта в трехмерном режиме (модуль 3D или 3D-модуль) позволяет просматривать трехмерную модель здания или сооружения, связывать строительные конструкции с выявленными дефектами и повреждениями и многое другое.

Модуль паспортных данных (Электронный паспорт) дает возможность вести электронный паспорт объекта, заносить информацию о строительных конструкциях здания или сооружения, объектно-планировочных решениях, а также автоматически создавать полностью сформированный электронный паспорт объекта, включая ведомость дефектов и повреждений.

В системе «Электронный паспорт» предоставляются возможности:

- **Заполнение паспортных данных о зданиях или сооружениях (ЗиС).** Занесение информации о вводе в эксплуатацию, объектно-планировочном решении, устройстве строительных конструкций, наличию оборудования, проведенных изысканиях, климатических условиях и т.п.
- **Внесение, хранение и отображение всей эксплуатационной информации.** Информация о наличии дефектов и повреждений в строительных конструкциях, проведенных эксплуатационных осмотрах, обследованиях, экспертизах.
- **Просмотр трехмерной модели конкретного производственного объекта.** Добавление информации о дефектах и повреждениях, привязка к элементам строительных конструкций в трехмерном режиме.

Система хранения и обработки документов «Электронный каталог» предусматривает:

- **Занесение и хранение в базе данных записи о документах.** Занесение новых документов, редактирование и хранение в базе данных записей о документах; работа со стандартным и расширенным набором реквизитов записей; работа с файлами документов.
- **Работа со связанными документами.** Связь документа с другими документами (необходима, когда документ входит в состав другого документа, в пакет документов или ссылается на текст другого документа).
- **Работа со справочниками и классификаторами.** Создание, ведение и поддержка справочников и классификаторов документов.
- **Поиск документов.** Поиск документов по наименованию или по значениям реквизитов.

2. Интерфейс программы (общие сведения)

Интерфейс программы (системы) – это программные функции, представленные такими графическими элементами, как меню, панели инструментов, закладки, вкладки, кнопки (рис.5).

Меню – элемент интерфейса программы, позволяющий выбрать одну (в простейшем случае) из нескольких перечисленных опций (см. рис. 5).

Панель инструментов – элемент графического интерфейса программы, предназначенный для размещения на нём различных элементов управления (рис. 6).

Вкладка – элемент графического интерфейса, который позволяет в одном окне приложения переключаться между разными разделами приложения (см. рис. 6). Вкладка представляет собой прямоугольник с надписью. Клик мышью по вкладке активизирует ее и делает видимым соответствующий ей документ.

Кнопка – элемент графического интерфейса программы. В узком смысле слова кнопкой называют элемент, выглядящий как прямоугольник нейтрального цвета, в широком – любой «кликабельный» элемент, при нажатии на который происходит действие.

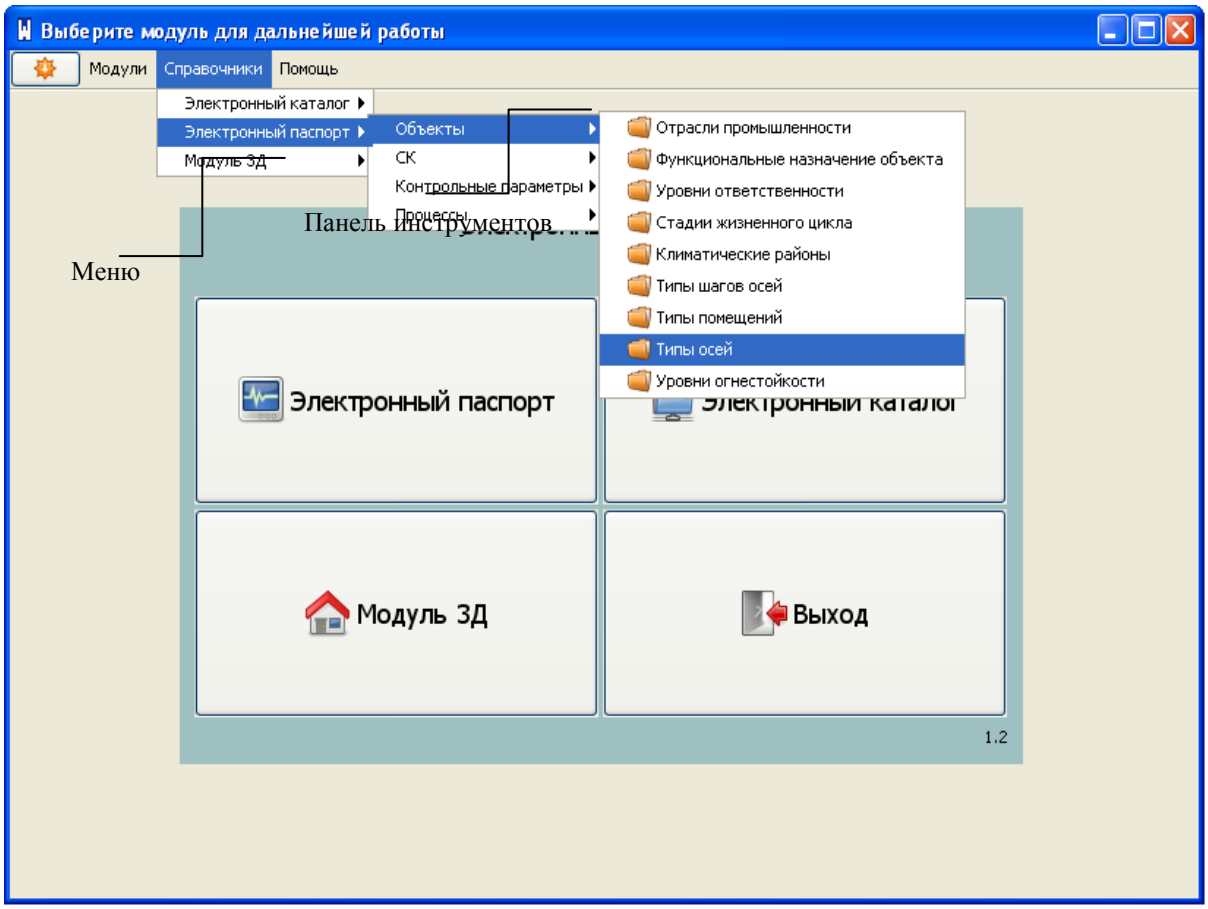


Рис. 5. Демонстрация работы с элементом интерфейса «Меню»

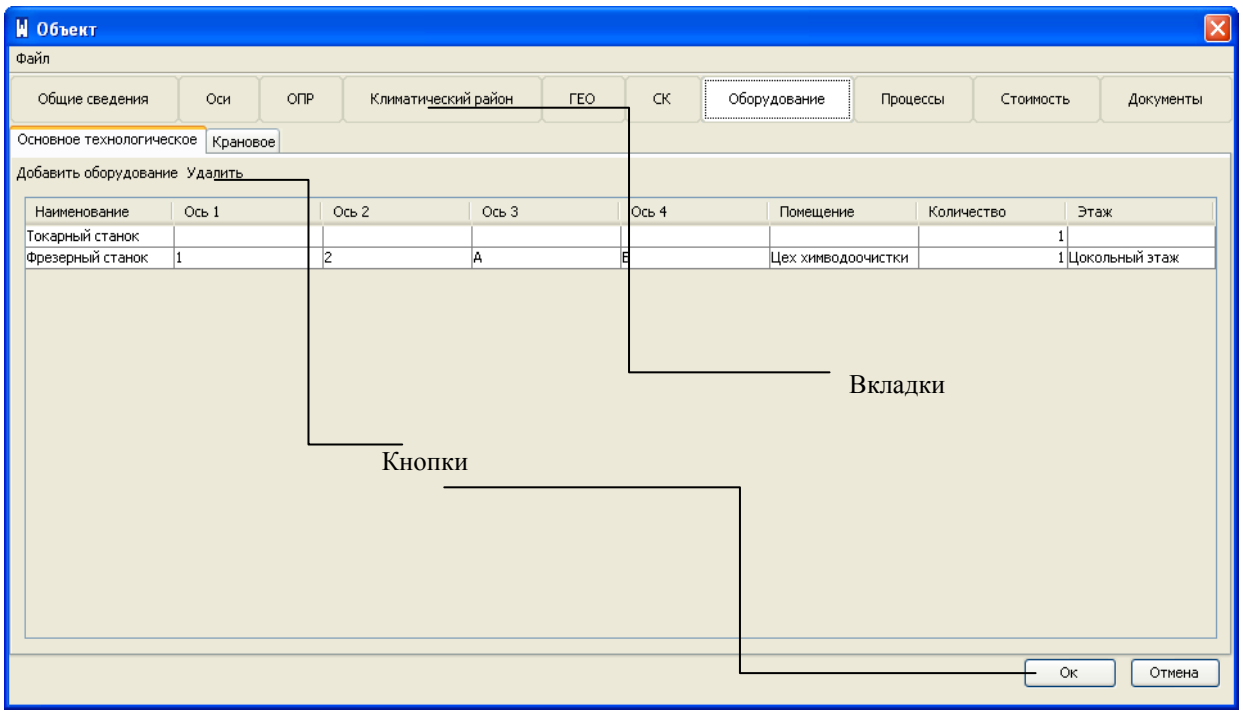


Рис. 6. Окно с информацией об объекте, распределенной по вкладкам

3. Сведения об информационных объектах системы

Информационными объектами программы являются: паспортные сведения о зданиях и сооружениях; сведения о конструктивных особенностях ЗиС; информация об объектно-планировочных решениях; климатические характеристики района; геологические сведения о грунтах основания ЗиС; сведения о процессах, проводимых при эксплуатации; списки прикрепленных документов к электронному паспорту ЗиС.

4. Запуск системы и стартовое окно программы

После запуска программы появляется стартовое окно программы (рис. 7).

На панели инструментов стартового окна программы находятся: кнопка перехода на главное меню; кнопки открытия меню «Модули», «Справочники» и «Помощь».

Меню «Модули» позволяет переходить в любой из трех разделов комплекса.

Меню «Справочники» дает возможность открывать и управлять всеми справочниками комплекса, относящимися как к электронному каталогу, так и к электронному паспорту.

Меню «Помощь» позволяет узнать версию программы, получить оперативную справочную информацию по работе с системой, отправить отзыв разработчикам.

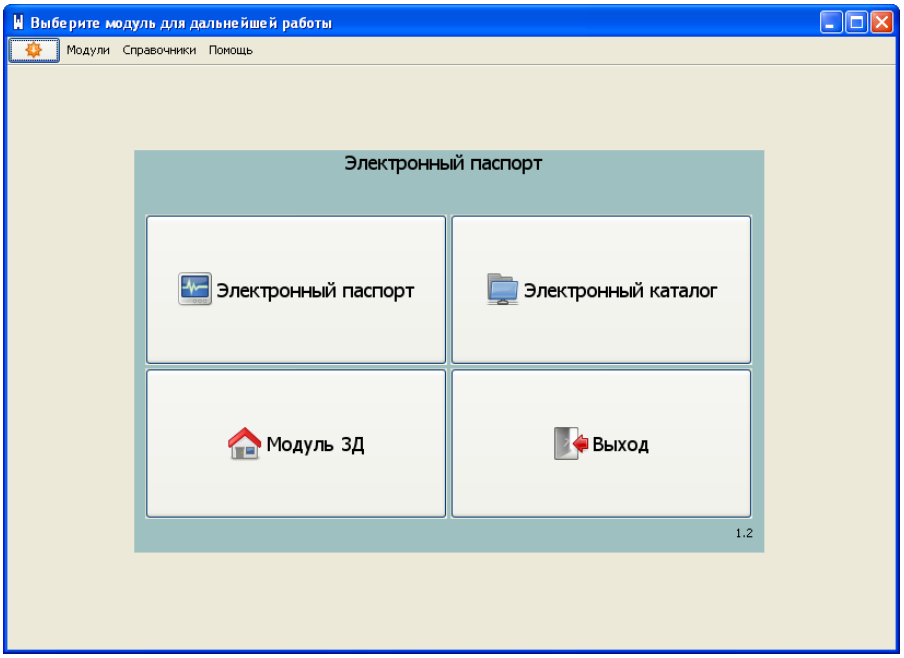


Рис. 7. Стартовое окно программы «Электронный паспорт»

5. Работа в системе

5.1. Работа со справочниками

Для хранения постоянной или условно постоянной информации однородных объектов в программе должны быть созданы справочники. Каждый справочник представляет собой список однородных объектов. Каждый такой объект называется элементом справочника. Справочники поддерживают принятые в организации или

общероссийские классификаторы. Как правило, справочники имеют многоуровневую структуру представления.

В справочниках реализованы следующие функции: создание, сортировка, редактирование и удаление элементов справочников и папок, а также функции «Вырезать» и «Вставить».

Раскладка, используемая в справочнике документов, распространяется и на остальные окна программы. Чтобы не переключаться каждый раз в окнах программы к необходимой раскладке, переключите ее в справочнике документов, и она сохранится для всех других окон программы.

На рис. 8 представлена общая структура справочников системы «Электронный паспорт».

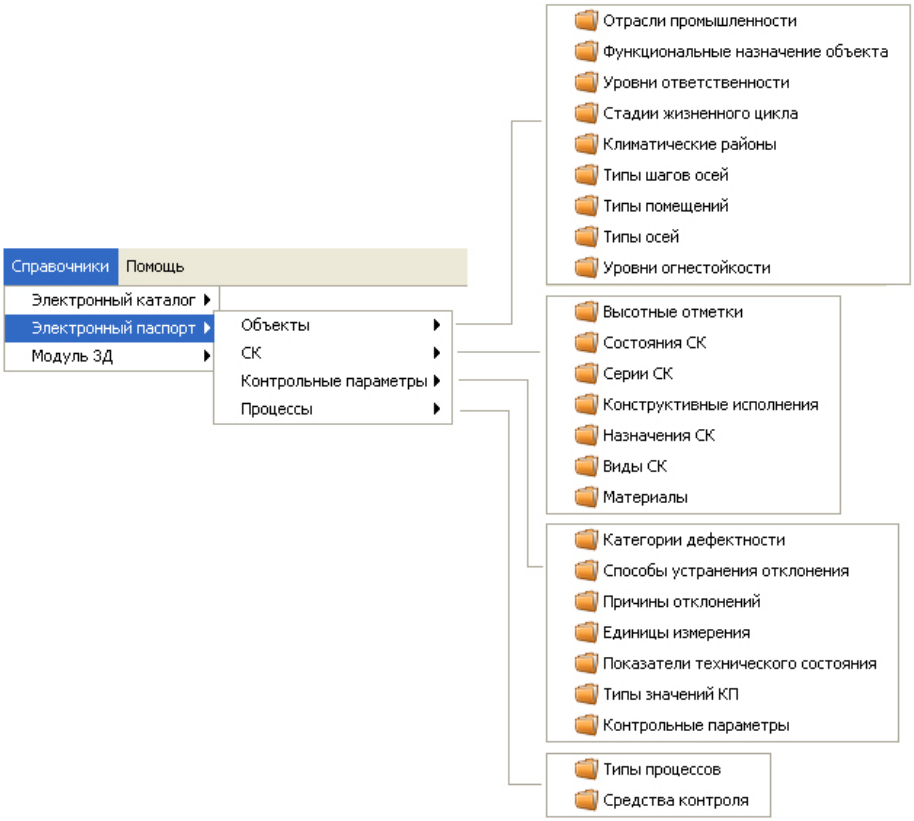


Рис. 8. Структура справочников

Описание справочников:

Отрасли промышленности. Справочник, содержащий информацию о различных отраслях промышленности, таких как машиностроительная, металлургическая, текстильная, пищевая и др.

Функциональные назначения объекта. Справочник, содержащий информацию о тех функциях, которые могут выполнять объекты (здания и сооружения, отдельные конструкции), например «Бытовые», «Производственные», «Электротехнические», «Столовые», «Складские».

Уровни ответственности. Справочник, позволяющий выбрать значения (классы) уровней ответственности здания или сооружения.

Стадии жизненного цикла. Справочник, содержащий основные стадии жизненного цикла: проектирование, строительство, эксплуатация.

Климатические районы. Многоуровневый справочник, позволяющий создавать объекты «Климатический район» и добавлять к каждому из них характеристики, которые описывают его среднюю температуру по сезонам года, снеговую и ветровую нагрузки, глубину промерзания грунтов и т.п.

Типы шагов осей. Информация о расположении осей: крайние или средние и т.п.

Типы помещений. Справочник для указания типа помещения: «Хозяйственное», «Бытовое», «Производственное» и т.п.

Уровни огнестойкости. Информация об уровнях огнестойкости здания.

Высотные отметки. Информация о типах высотных отметок (абсолютные, относительные, базовые).

Состояние СК. Техническое состояние строительных конструкций: работоспособное, неработоспособное, ограниченно работоспособное и т.п.

Серии СК. Многоуровневый справочник серий строительных конструкций, позволяющий создавать группы серий, добавлять к каждой серии выпуски строительных конструкций. Каждая серия и выпуск имеет свое название и номер и статус (действует или нет).

Конструктивные исполнения. Справочник конструктивных исполнений строительных конструкций, содержащий общеупотребительные названия конструктивных исполнений: Балочные конструкции, вертикальные конструкции, пол, плиты, фермы, фонари и др.

Назначения СК. Функциональные назначения строительных конструкций: несущие, ограждающие, фундаменты и др.

Виды СК. Многоуровневый справочник, содержащий информацию о видах строительных конструкций, включающую в себя (рис. 9):

- Наименование вида.
- Принятое обозначение.
- Функциональное назначение (выбирается из справочника «Назначения СК»).
- Конструктивное исполнение (выбирается из справочника «Конструктивные исполнения).
- Материал (выбирается из справочника «Материалы»).

Материалы. Справочник, содержащий сведения о материалах, из которых изготовлены строительные конструкции.

Категории дефектности. Справочник, содержащий список категорий дефектности, к которым относятся дефекты и повреждения строительных конструкций.

Способы устранения отклонений. Многоуровневый справочник, позволяющий создавать способы устранения отклонений (дефекты и повреждения, несоответствия номинальным параметрам) и группировать их (рис. 10).

Причины отклонений. Многоуровневый справочник, позволяющий создавать причины отклонений (дефекты и повреждения, несоответствия номинальным параметрам) и группировать их.

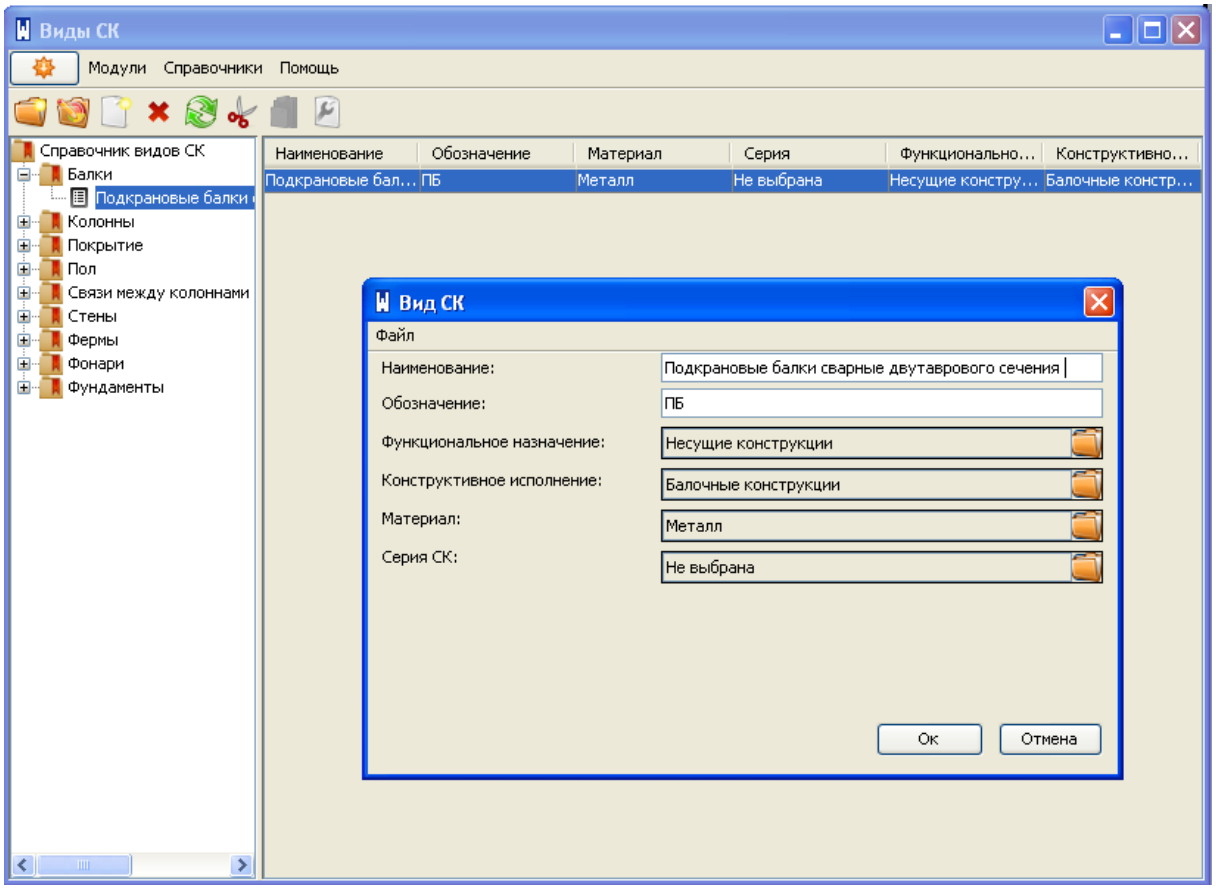


Рис. 9. Справочник «Виды СК»

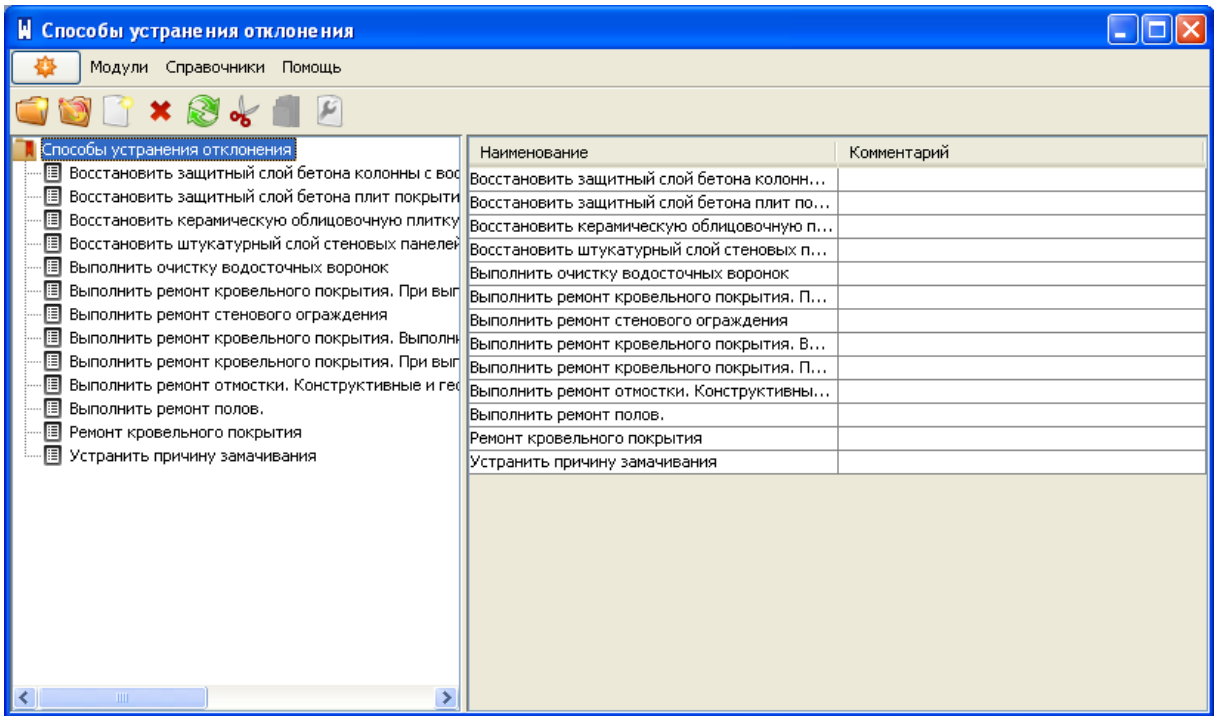


Рис. 10. Справочник «Способы устранения отклонения»

Единицы измерения. Многоуровневый справочник, позволяющий создавать списки единиц измерения (расстояний, напряжений, давлений и т.п.) и группировать их.

Показатели технического состояния. Многоуровневый справочник, позволяющий создавать показатели технического состояния строительных конструкций и группировать их. Основными показателями, характеризующими техническое состояние строительных конструкций, являются:

- Соответствие конструктивных характеристик СК проектной и нормативной документации.
- Соответствие характеристик материалов СК нормируемым значениям.
- Наличие дефектов и повреждений СК.
- Наличие отклонений пространственного положения СК от нормативных значений.
- Соответствие несущей способности СК усилиям от эксплуатационных нагрузок.

Типы значений КП. Справочник типов значений контрольных параметров: нормативное, измеренное, допустимое, проектное и т.д.

Контрольные параметры. Многоуровневый справочник, содержащий информацию о контрольных параметрах, сгруппированных по папкам и включающих в себя следующие сведения (рис. 11):

- Наименование контрольного параметра.
- Единица измерения (берется из одноименного справочника).
- Показатель технического состояния (берется из одноименного справочника).
- Комментарий.
- Способы устранения отклонения (берутся из одноименного справочника).
- Причины возникновения отклонений (берутся из одноименного справочника).

Типы процессов. Справочник, содержащий сведения о типах проводимых процессов на объекте: осмотр, обследование, экспертиза промышленной безопасности.

Средства контроля. Справочник средств, приборов для измерения различных параметров: дальномеры, тепловизоры. Содержит сведения о наименовании, назначении и поверке прибора.

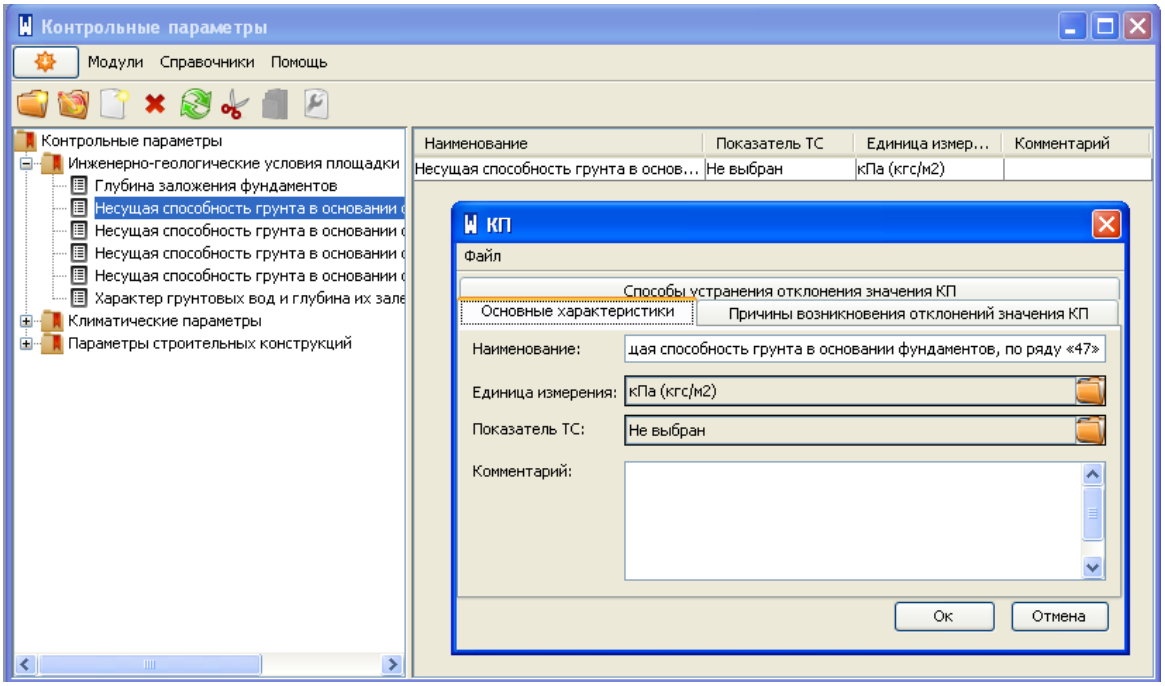


Рис. 11. Справочник «Контрольные параметры»

5.2. Работа с объектами

Создание объекта

Для создания объекта нужно запустить программу. Для этого в открывшемся главном меню выбрать кнопку «Электронный паспорт», в открывшемся после этого окне надо выбрать кнопку «Здания и сооружения», откроется окно, показанное на рис. 12. В данном окне можно создавать объекты, группировать их по папкам, проводить операции копирования и удаления объектов.

Для редактирования свойств объекта необходимо войти в режим просмотра двойным щелчком левой клавиши мыши по названию объекта.

Окно редактирования объекта содержит следующие вкладки:

- Общие сведения (общие сведения об объекте, сведения об организациях и т.п.);
- Оси (сведения о координационных осях объекта; шаг осей, расположение относительно самого объекта);
- ОПР (объектно-планировочное решение объекта – сведения о пролетах, помещениях, этажах объекта);
- Климатический район (указывается климатический район из соответствующего справочника);
- ГЕО (сведения о геологическом строении грунтов основания объекта);
- СК (сведения о строительных конструкциях, использованных при строительстве и эксплуатации объекта);
- Оборудование (сведения о используемом в процессе эксплуатации объекта оборудовании, его местонахождении, количестве, назначении и т.п.);
- Процессы (сведения о всех проведенных процессах, таких как осмотры, обследования, экспертизы и др., собранные в иерархичную древовидную структуру);
- Стоимость (сведения о показателях балансовой стоимости, физическом износе на определенный год);
- Документы (строительная документация объекта).

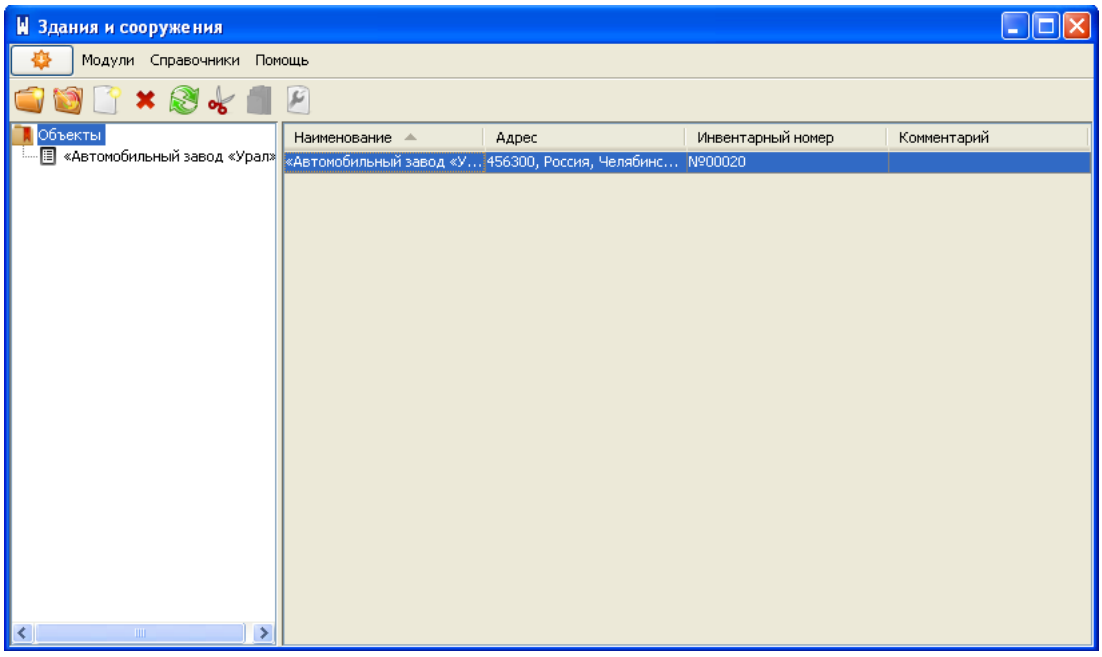


Рис.12. Форма управлением объектами

Общие сведения

Общие сведения об объекте вносятся на вкладке, показанной на рис. 13.

Данная вкладка содержит информацию о наименовании объекта, его расположении (адрес), организации, эксплуатирующей этот объект, ответственных лицах, времени сдачи в эксплуатацию, инвентарный номер. Эта информация заполняется специализированным сотрудником согласно данным паспорта объекта.

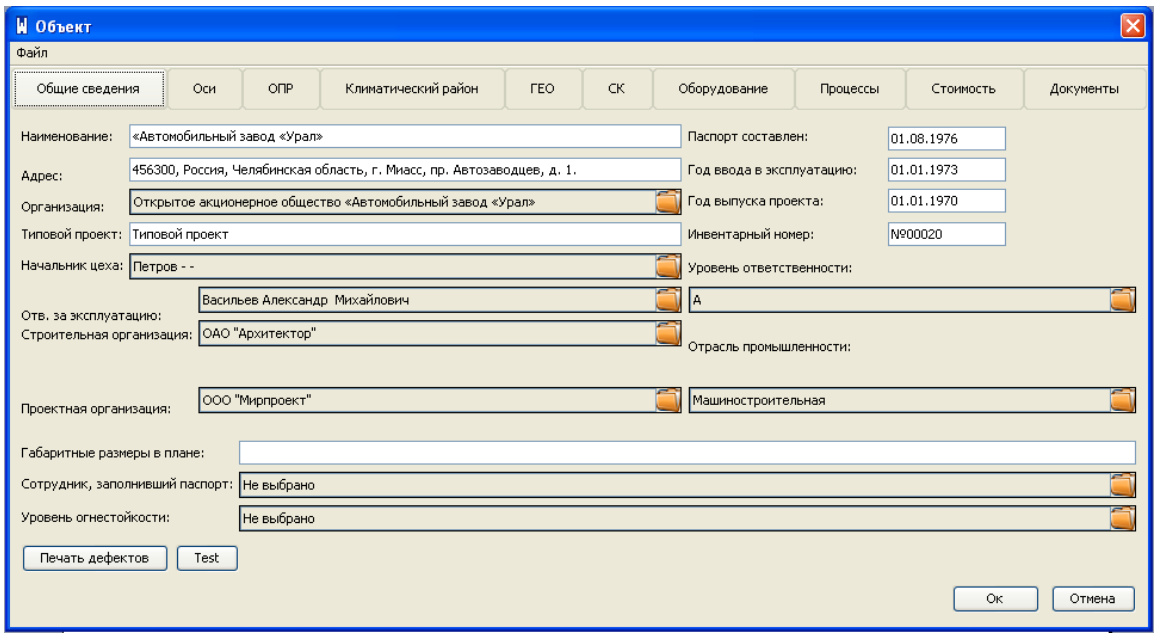


Рис. 13. Занесение общих сведений об объекте

Оси

Вид окна с открытой вкладкой «Оси» представлен на рис. 14. При добавлении оси требуется указать ее обозначение (цифра, буква), размер шага, выбрать тип шага (крайняя, средняя) и тип оси (буквенная или цифровая).

Для удаления оси нужно выделить нужную ось и нажать кнопку «Удалить ось».

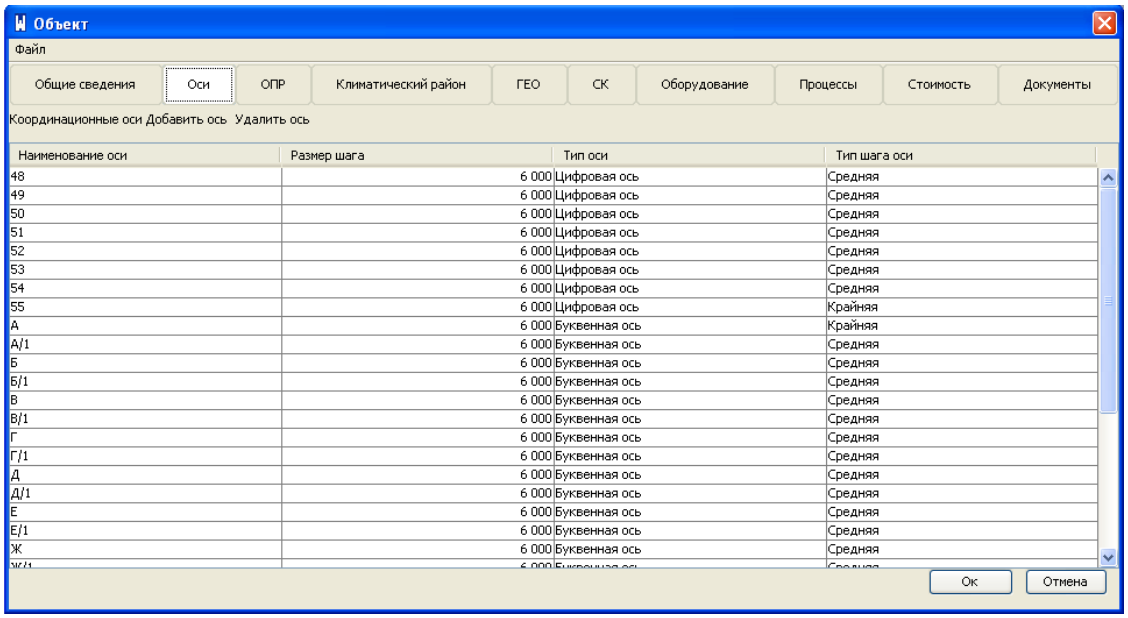


Рис. 14 – Вид вкладки «Оси»

Объектно-планировочное решение объекта

Вид окна с открытой вкладкой «ОПР» представлен на рис. 15. На этой вкладке требуется указать пролеты, этажи и помещения производственного объекта, их расположение, размеры, высотные отметки.

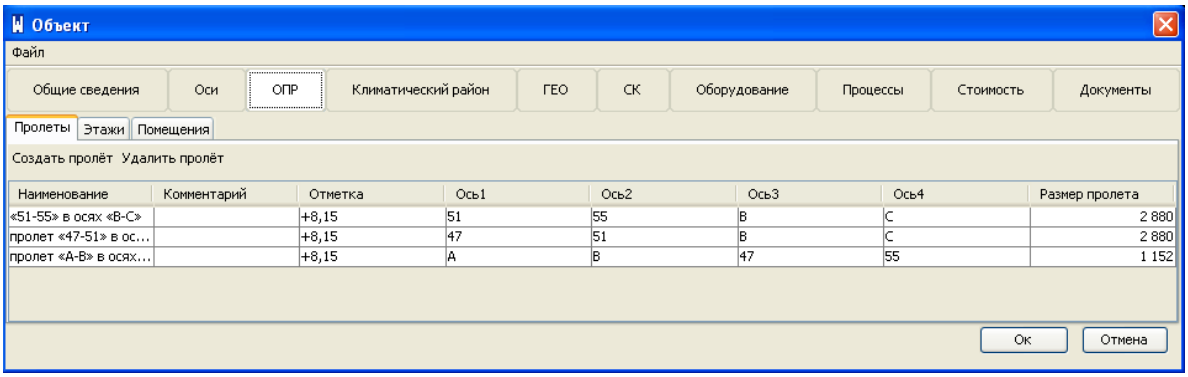


Рис. 15. Вид вкладки «ОПР»

Климатический район

Климатический район выбирается из соответствующего справочника. На соответствующей вкладке отображаются только характеристики выбранного района без возможности их изменения.

Сведения о геологическом строении грунтов основания

В этом разделе паспорта вносятся геологические характеристики грунтов основания объекта (рис. 16).

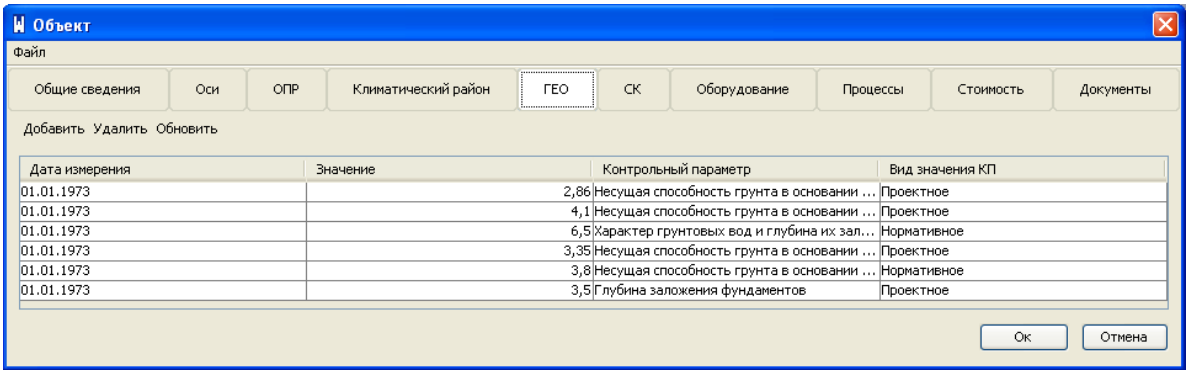


Рис. 16. Вид вкладки «ГЕО»

Сведения о строительных конструкциях

В эту вкладку вносится информация о всех строительных конструкциях, использованных при строительстве и эксплуатации объекта. При добавлении конструкции указываются вид, состояние, дата ввода в эксплуатацию, расположение относительно координационных осей, высотные отметки (рис. 17).

Оборудование

На вкладке «Оборудование», показанной на рис. 18, указывается используемое в процессе эксплуатации объекта оборудование, его расположение и характеристики. Оборудование подразделяется на основное технологическое и крановое.

При добавлении основного технологического оборудования требуется указать наименование оборудования, расположение относительно координационных осей, этаж

и помещение из справочников. Также нужно ввести количество оборудования, при необходимости можно указать воздействия на строительные конструкции.

При добавлении кранового оборудования требуется указать вид кранового оборудования, грузоподъемность, количество оборудования и площадь обслуживания. Также необходимо указать расположение оборудования относительно координационных осей и выбрать из справочника помещения, где находится оборудование (рис. 19).

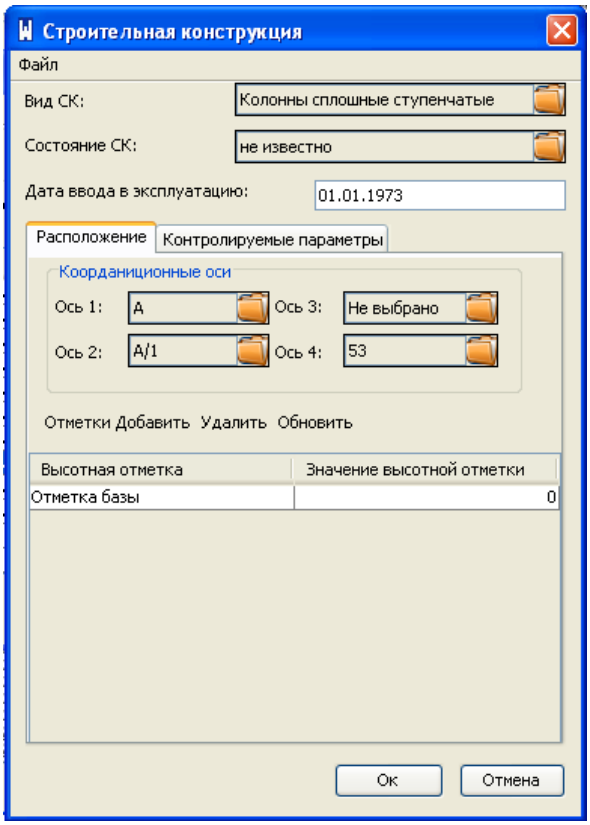


Рис. 17. Вид формы добавления информации о строительной конструкции

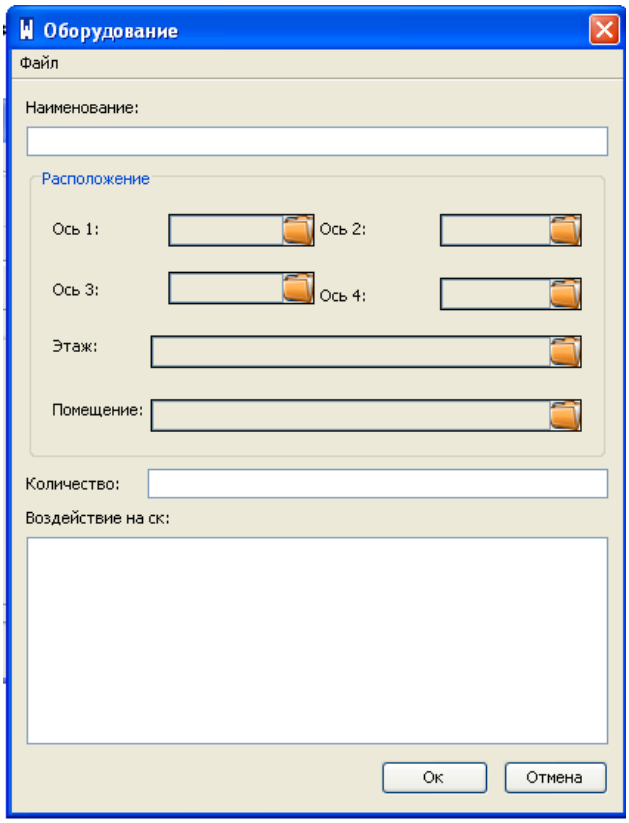


Рис. 18. Вид формы добавления информации об основном технологическом оборудовании

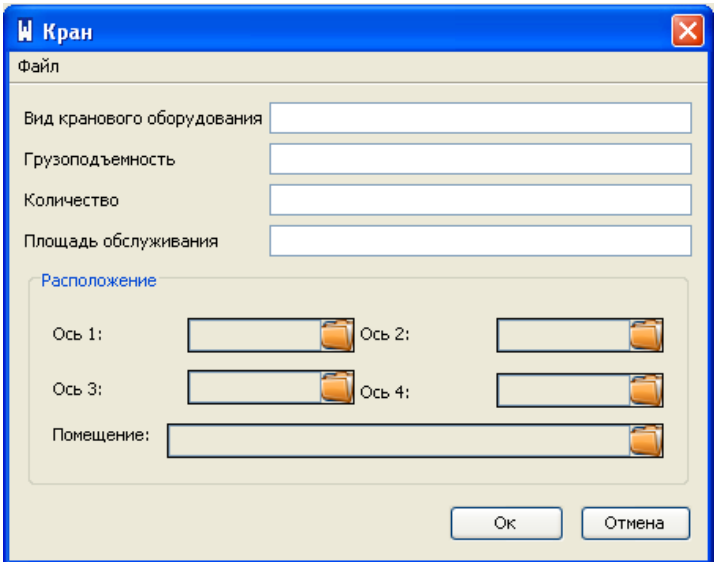


Рис. 19. Вид формы добавления информации о крановом оборудовании

Процессы

На вкладке «Процессы» содержится информация обо всех процессах, происходящих на объекте. Основные процессы: осмотр, обследование, экспертиза промышленной безопасности. Пользователь может сам создавать необходимые процессы.

Процессы состоят из операций, операции – из измерений. Следовательно, последовательность действия при заполнении вкладки «Процессы» такова: создать процесс, создать в этом процессе одну из операций этого процесса, создать в этой операции одно из измерений (рис. 20).

Для создания процесса необходимо указать наименование процесса, выбрать стадию жизненного цикла, тип процесса, даты начала и окончания, ответственное лицо (рис. 21).

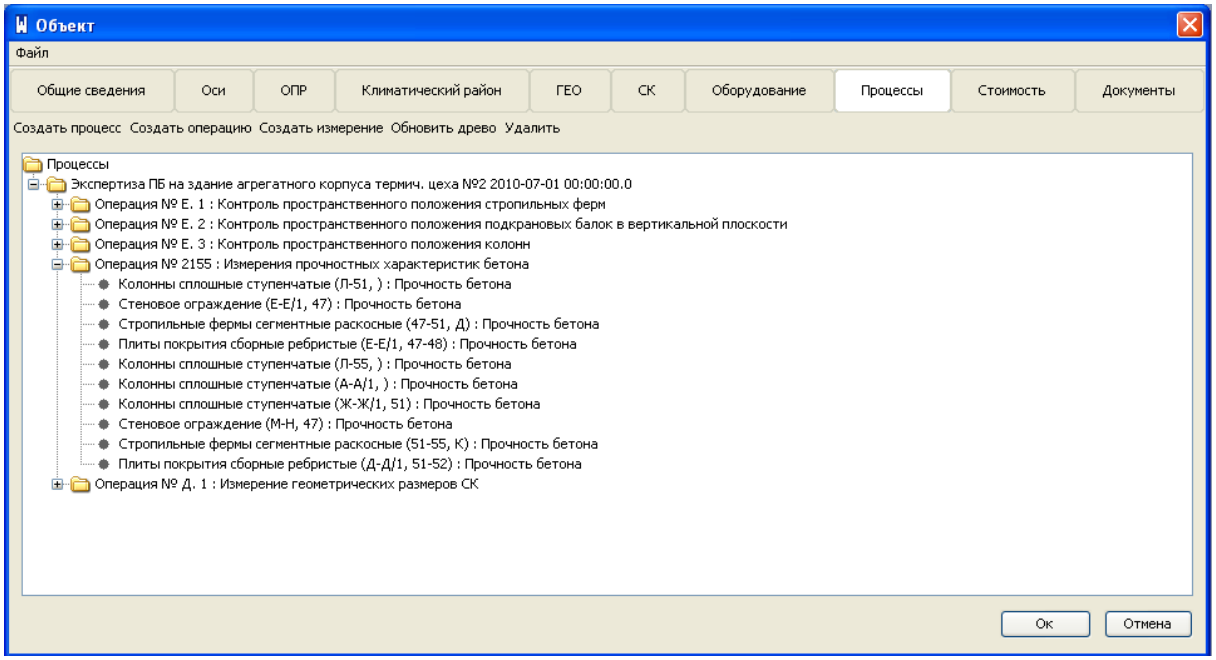


Рис. 20. Дерево процессов

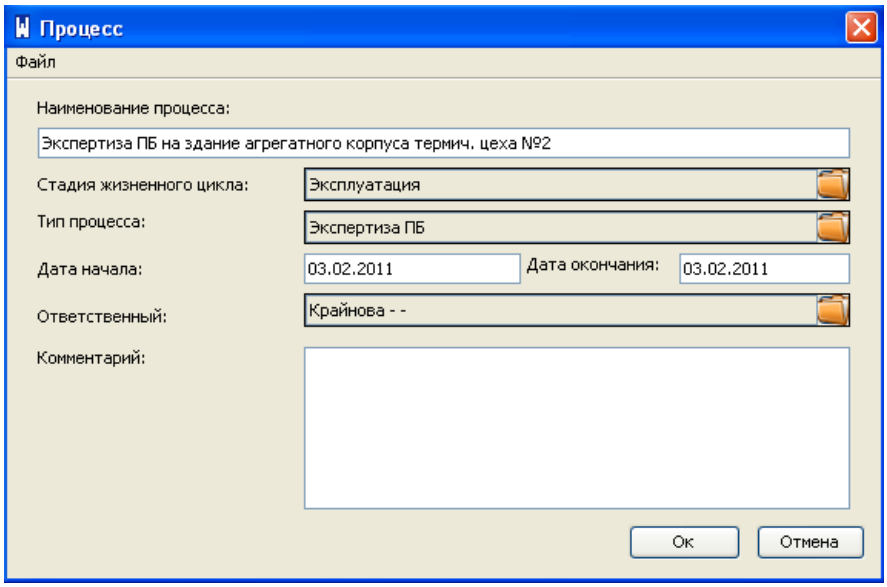


Рис. 21. Создание процесса

Для создания операции необходимо указать номер операции, контрагента (исполнителя по договору), даты начала и окончания, наименование операции, основания для выполнения работ.

В нижней части окна показаны вкладки, где указываются нормативные документы, средства (приборы) контроля, ответственные лица, а также делается вывод по результатам измерений (рис. 22).

Для создания измерения необходимо указать дату измерения, строительную конструкцию, которая подвергалась измерению, контрольный параметр и его фактическое значение. При необходимости можно добавить комментарий (рис. 23).

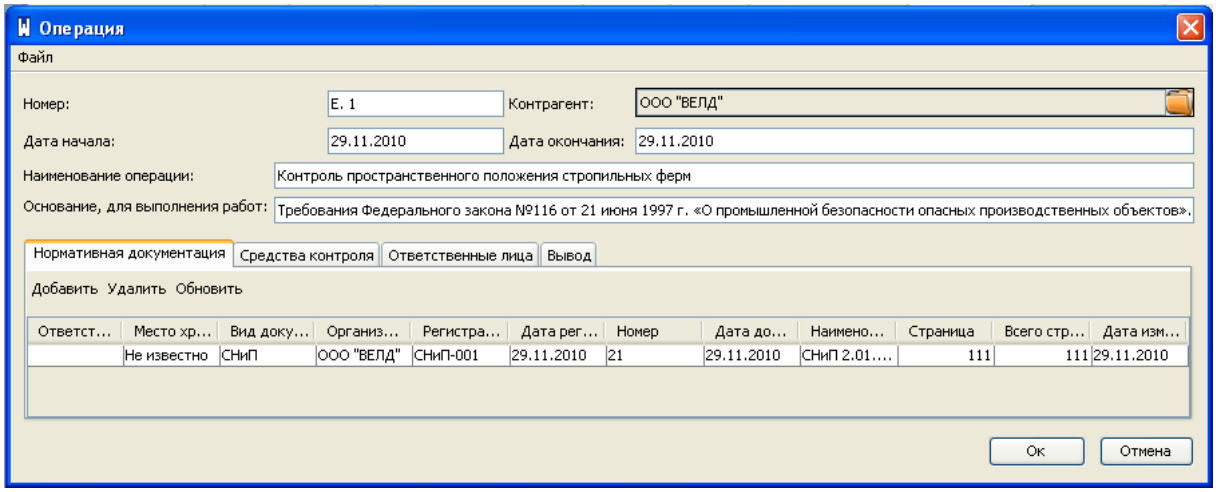


Рис. 22. Создание операции

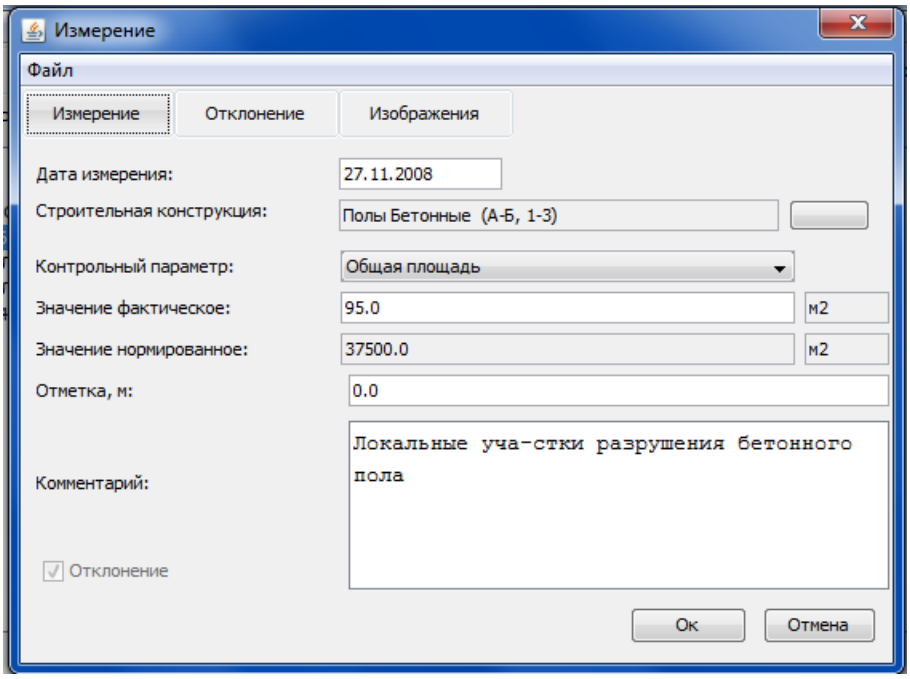


Рис. 23. Добавление измерения в операцию

Если фактическое значение контрольного параметра не входит в диапазон нормированных значений, то для данного измерения существует возможность указать параметры отклонения, способы устранения отклонения и причины его возникновения (рис. 24).

При необходимости можно добавить фотографии объекта измерения, открыв вкладку «Изображения».

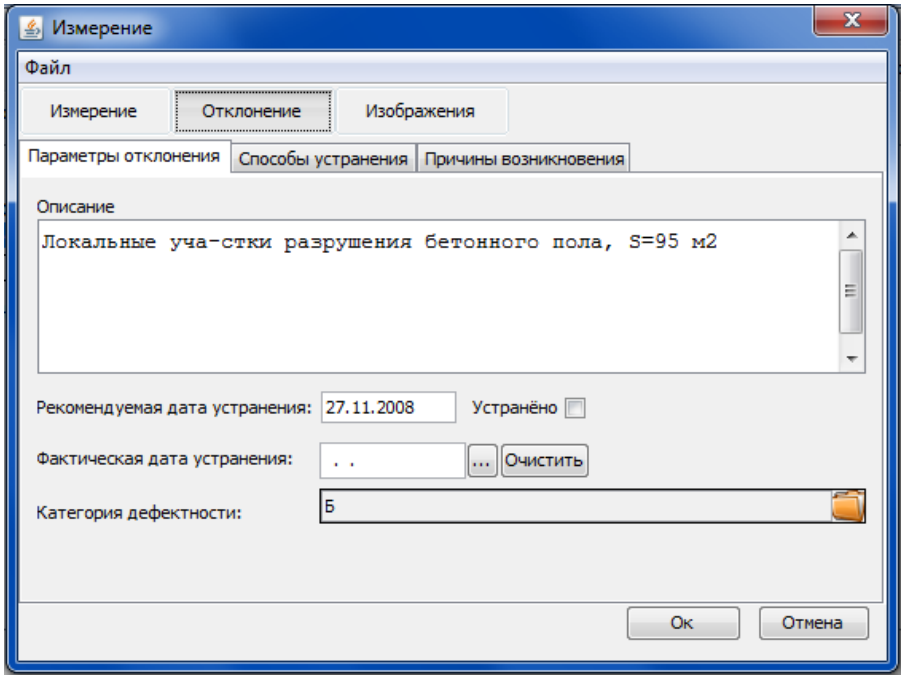


Рис. 24. Указание параметров отклонения

Стоимость

На данной вкладке содержится информация о балансовой стоимости объекта с учетом износа на определенный период. Для добавления записи о стоимости требуется указать год, стоимость объекта в рублях, физический износ в процентах, при необходимости комментарий (рис. 25).

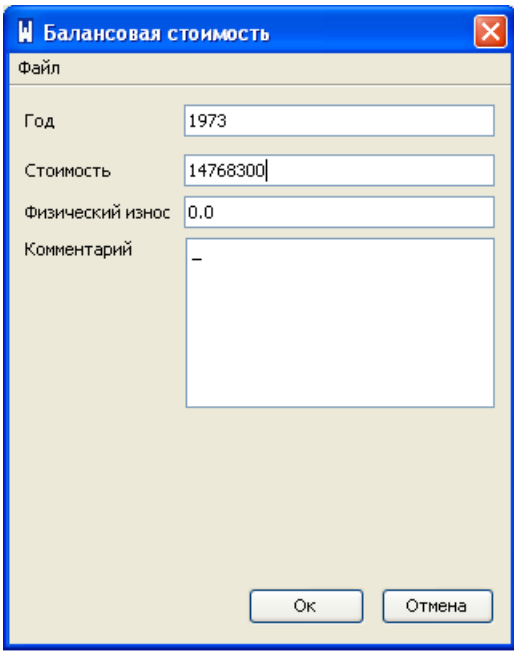


Рис. 25. Добавление записи о стоимости

Документы

Во вкладку «Документы» из подсистемы «Электронный каталог» добавляются документы, относящиеся к строительной документации объекта (рис. 26).

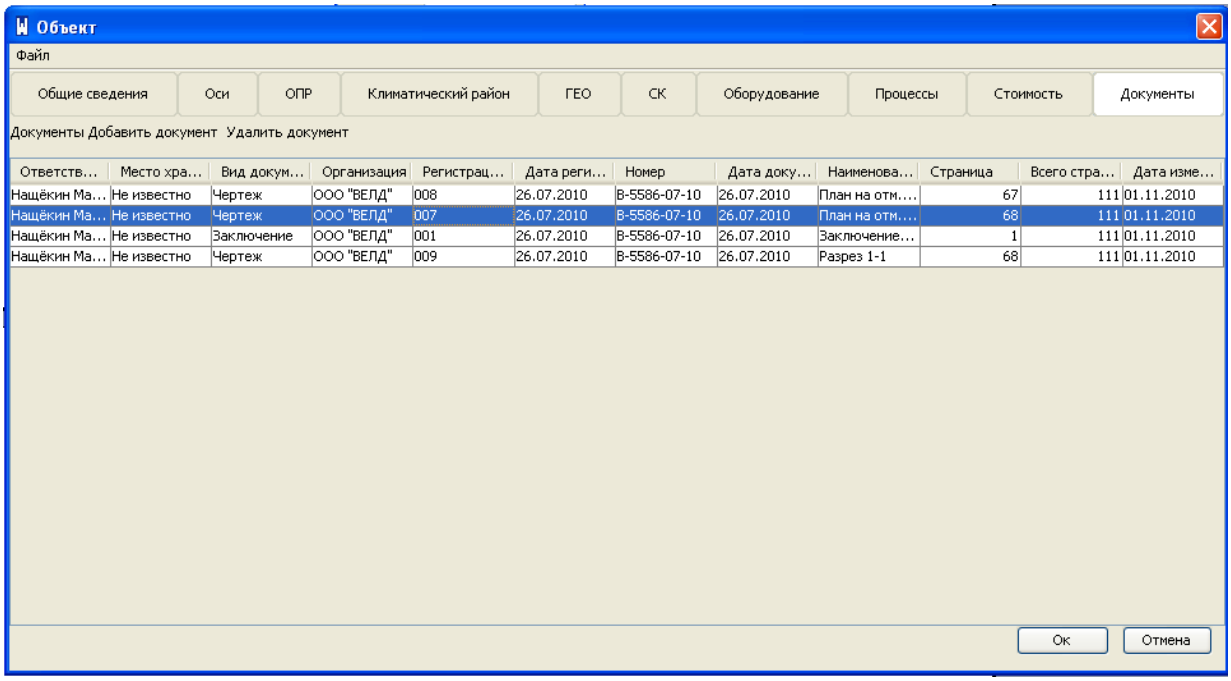


Рис. 26. Вкладка «Документы»

5.3 Работа с 3Д-модулем

Подготовка к работе

Для начала работы с модулем трехмерной визуализации объекта необходимо нажать на главном окне программы кнопку «Модуль 3Д» (см. рис. 5).

В открывшемся окне необходимо выбрать производственный объект в поле со списком и нажать кнопку «Выбрать» (рис. 27).

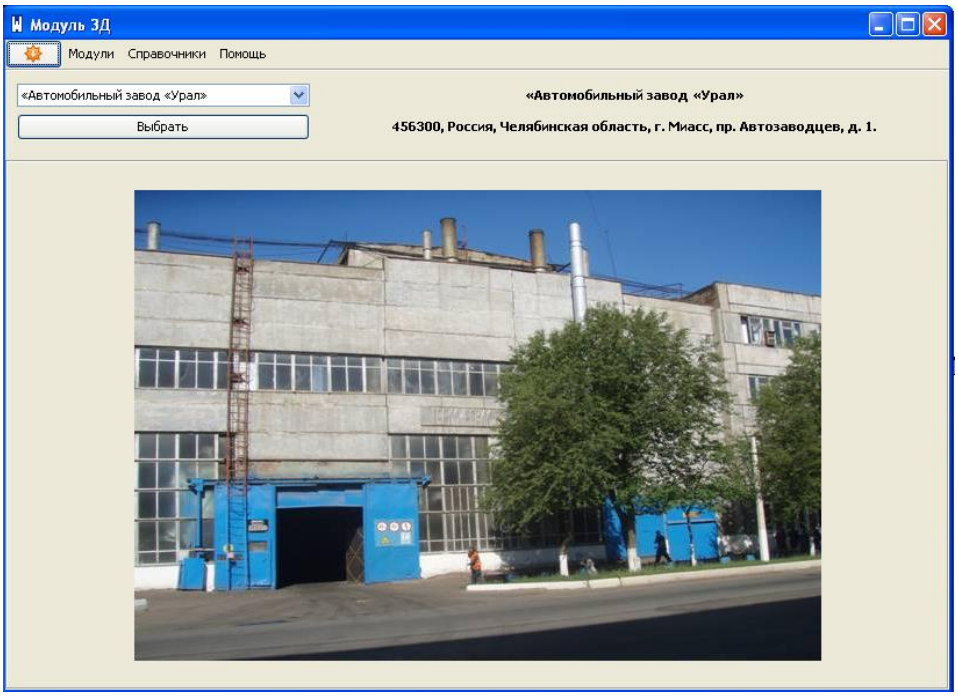


Рис. 27. Выбор объекта визуализации

После этого в левой части окна откроется панель навигации (рис.28).

При нажатии на кнопку «Файл» откроется окно выбора файлов, где необходимо найти объектный файл трехмерной модели (рис. 29). В данном справочном руководстве этот файл называется AZUral.obj.

После выбора необходимого файла окно примет вид, показанный на рис. 30.

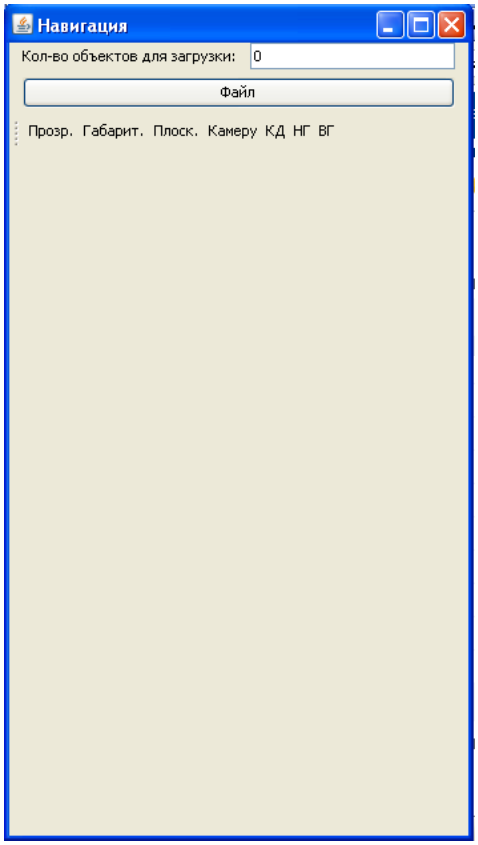


Рис. 28. Панель навигации

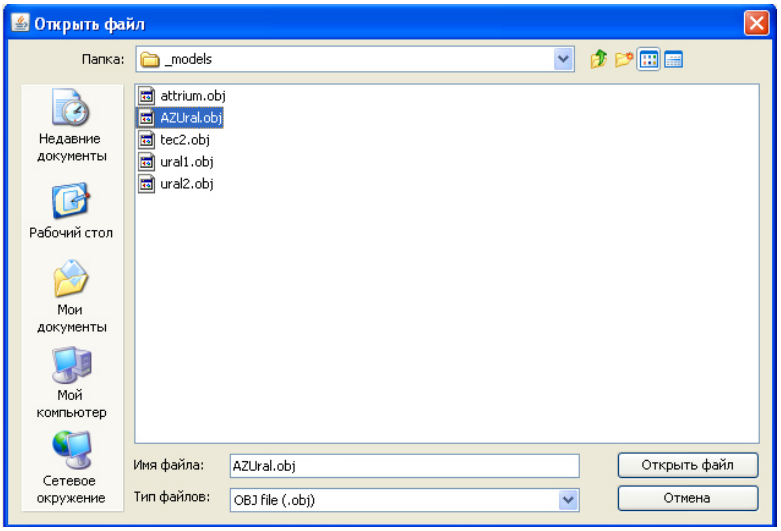


Рис. 29. Выбор файла

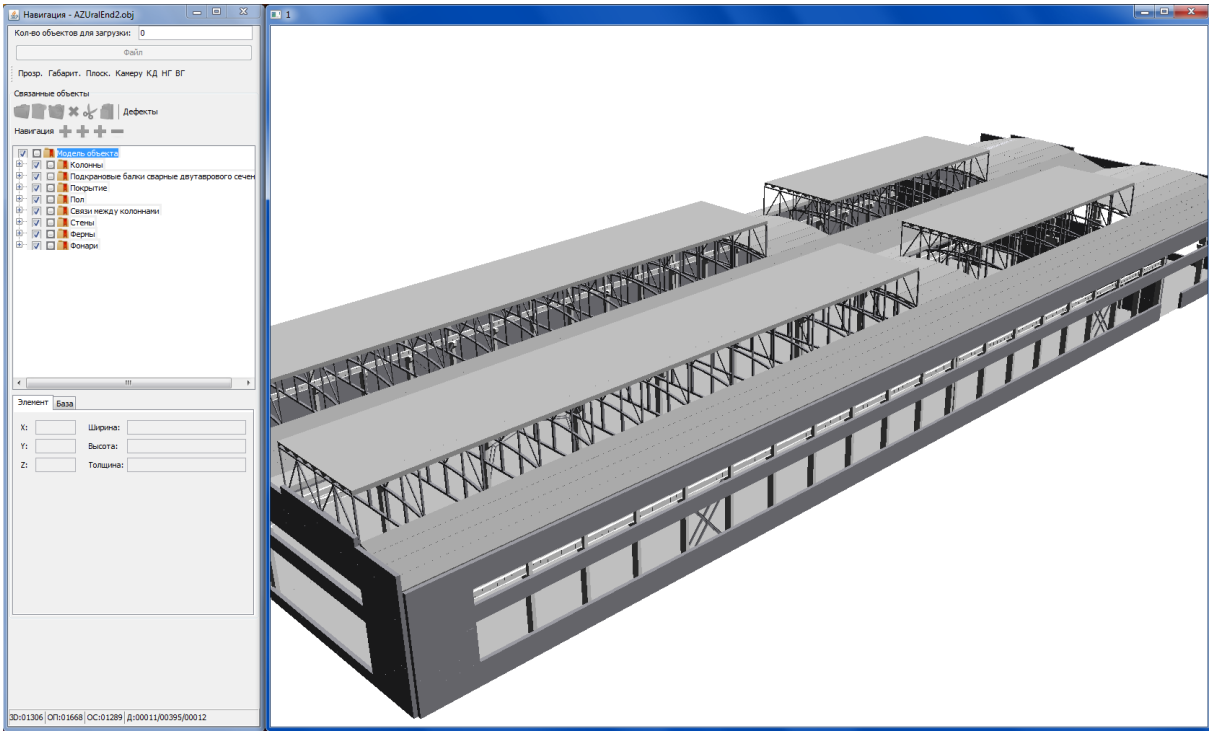


Рис.30. Общий вид 3Д-модуля с загруженным объектом

Навигация в окне визуализации

Для навигации в окне трехмерной визуализации используйте манипулятор «Мышь» и управляющие кнопки клавиатуры:

Действие	***Управление***
Приблизить модель	В области окна 3D удерживать нажатой правую клавишу мыши и перемещать курсор вправо
Отдалить модель	В области окна 3D удерживать нажатой правую клавишу мыши и перемещать курсор влево
Вращать модель относительно камеры наблюдателя	В области окна 3D удерживать нажатой левую клавишу мыши и перемещать курсор в нужном вам направлении, вращать модель
Выделить элемент в модели	Выделить необходимый элемент или строительную конструкцию с помощью нажатия на среднюю клавишу мыши (колесико) и позиционирования курсора над выделяемым элементом
Перемещение камеры наблюдателя в пространстве направо, налево, вниз, вверх	Управляющие кнопки клавиатуры: «Влево», «Вверх», «Вниз», «Вправо»

Навигация в дереве объектов

При выделении в дереве объектов какой-либо группы объектов в окне визуализации объекты, входящие в эту группу, выделяются синим цветом (рис. 31).

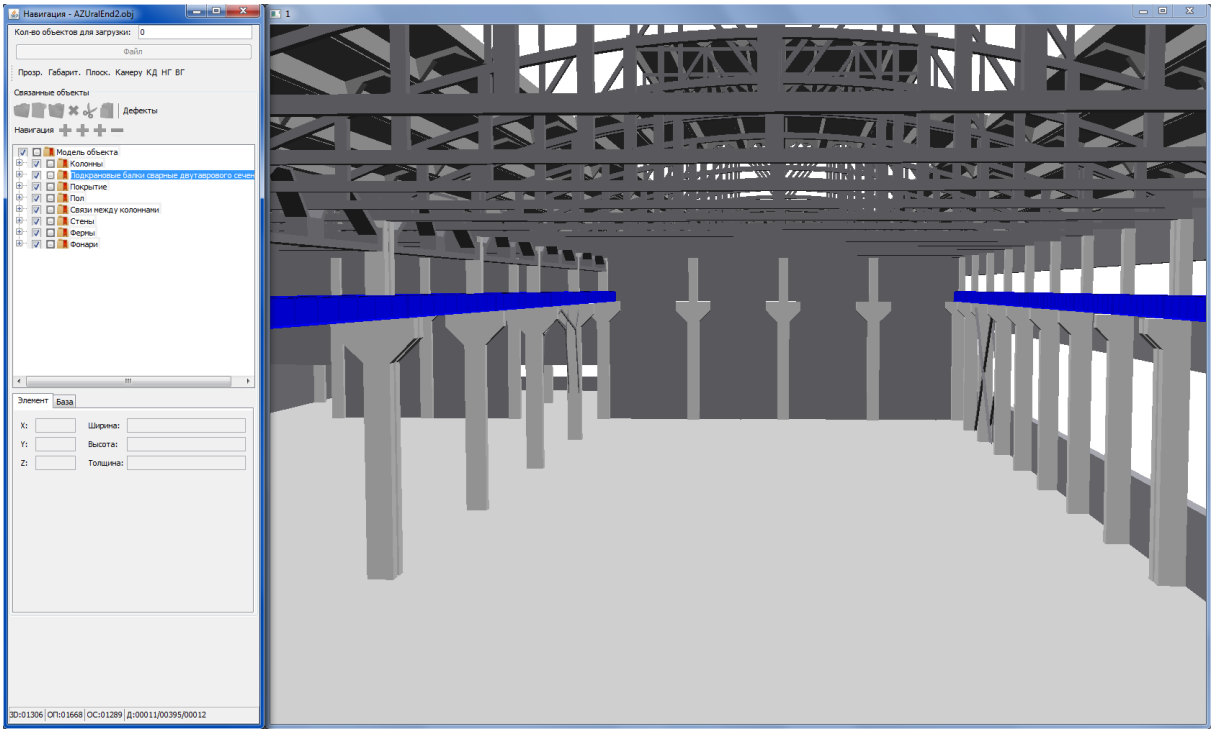


Рис. 31. Выделенные объекты

Просмотр дефектов и повреждений

Для того чтобы увидеть общую картину по локализации и состоянию дефектов и повреждений, необходимо нажать на кнопку «Дефекты» (рис. 32).

Дефекты и повреждения обозначаются несколькими цветами:

- красный (критичные дефекты и повреждения);
- желтый (обычные дефекты и повреждения);
- зеленый (устраненные дефекты и повреждения);
- оранжевый (выделенный мышкой дефект или повреждение).

Для того чтобы просмотреть подробную информацию о имеющемся дефекте (повреждении), необходимо выделить нужный дефект (повреждение) с помощью средней клавиши мыши (колесика). Дефект (повреждение) поменяет цвет на оранжевый, в левой панели можно будет просмотреть подробную информацию о нем: строительная конструкция, на которой он возник, техническое состояние этой строительной конструкции, наличие на ней других дефектов (повреждений) и др. (рис. 33).

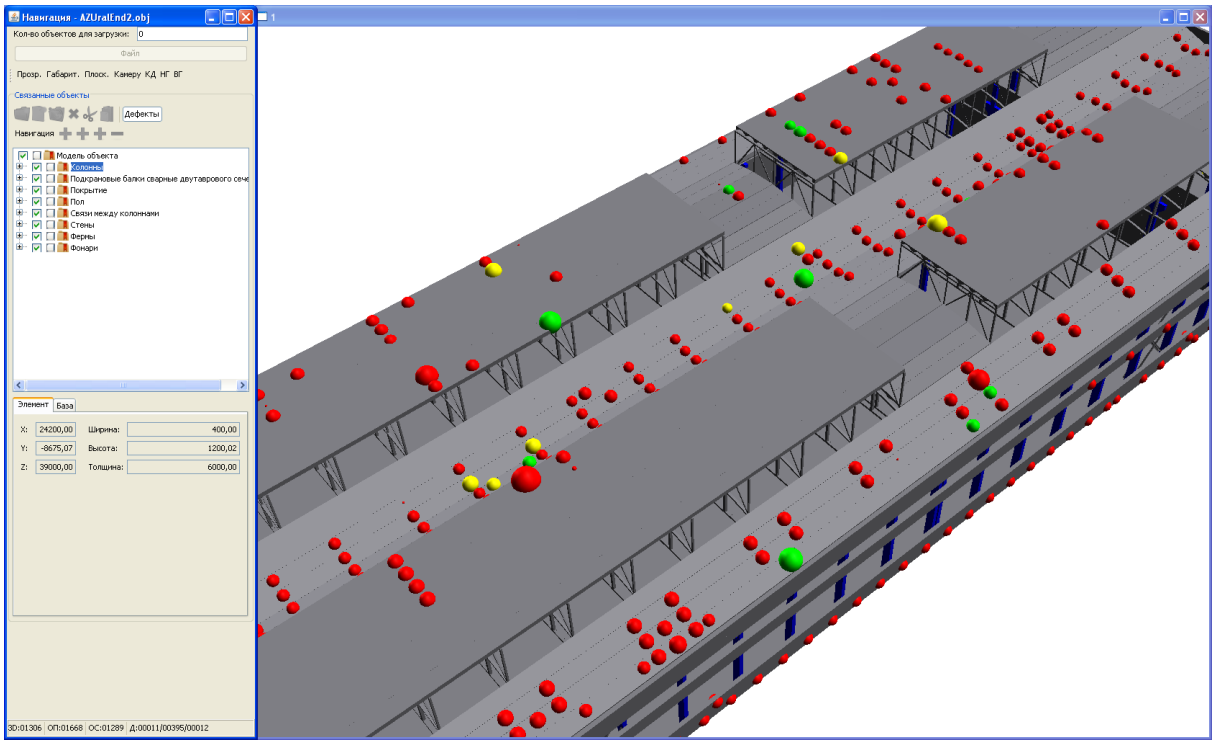


Рис.32. Просмотр общего расположения дефектов и повреждений

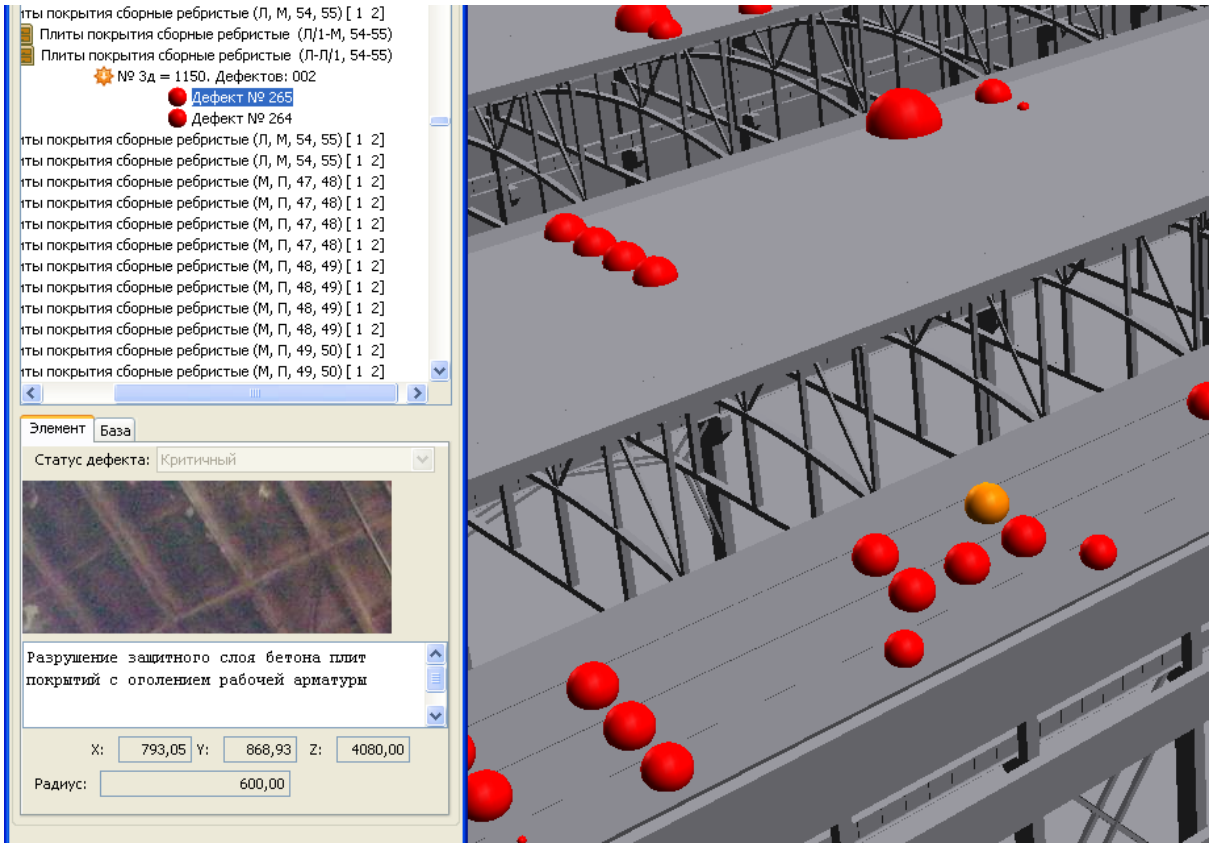


Рис. 33. Просмотр подробной информации о дефекте (повреждении)

Добавление информации о дефекте (повреждении) на 3Д-модель

Для добавления информации о дефекте или повреждении необходимо с помощью средней клавиши мыши выделить строительную конструкцию, на которой он появился (рис. 34). Затем необходимо включить режим «Связывание», нажав на кнопку «Навигация». Название кнопки изменится – оно отражает текущий режим (рис. 35). При этом станут активны элементы управления, позволяющие добавлять и удалять информацию о дефектах или повреждениях.

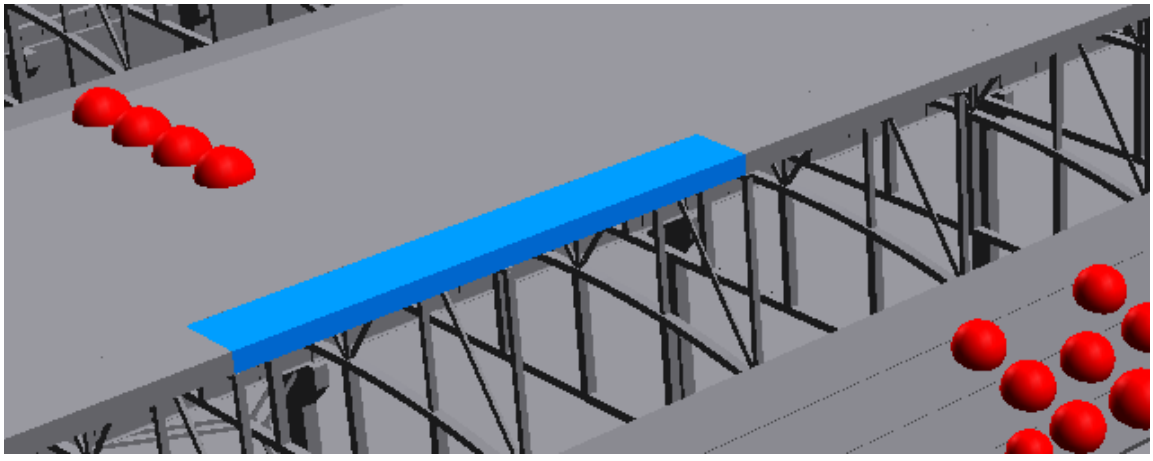


Рис. 34. Выделение строительной конструкции для добавления информации о дефекте или повреждении

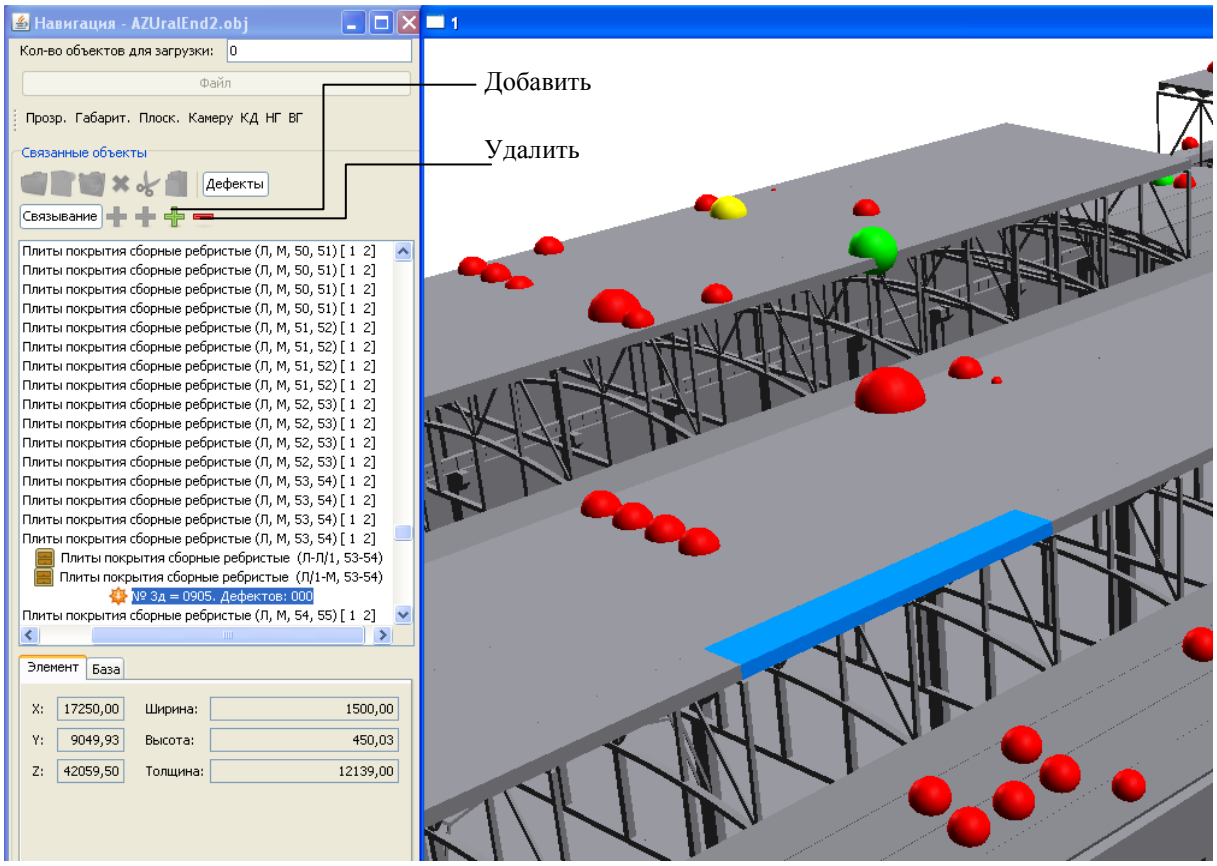


Рис. 35. Режим связывания

После нажатия кнопки «Добавить», в открывшемся диалоговом окне, необходимо выбрать условный размер дефекта (повреждения), отражающий его значимость и расположение относительно выбранной строительной конструкции, вид дефекта или повреждения (критичный, обычный, устраненный), добавить его описание в поле «Комментарии» (рис. 36).

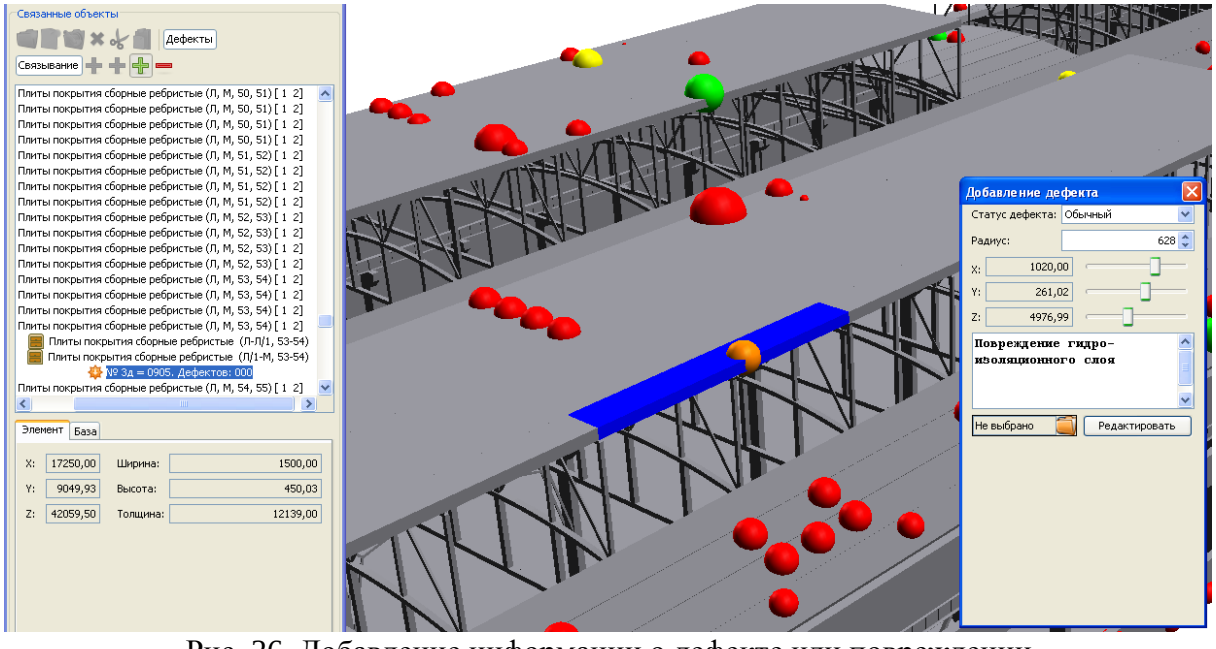


Рис. 36. Добавление информации о дефекте или повреждении

Удаление информации о дефекте (повреждении)

Для удаления информации о дефекте (повреждении), его необходимо выделить средней клавишей мыши в окне трехмерного просмотра. При этом должен быть включен режим отображения дефектов (повреждений) и включен режим «Связывание». Также дефект (повреждение) можно выделить в дереве объектов.

После этого необходимо нажать кнопку «Удалить» (рис. 37).

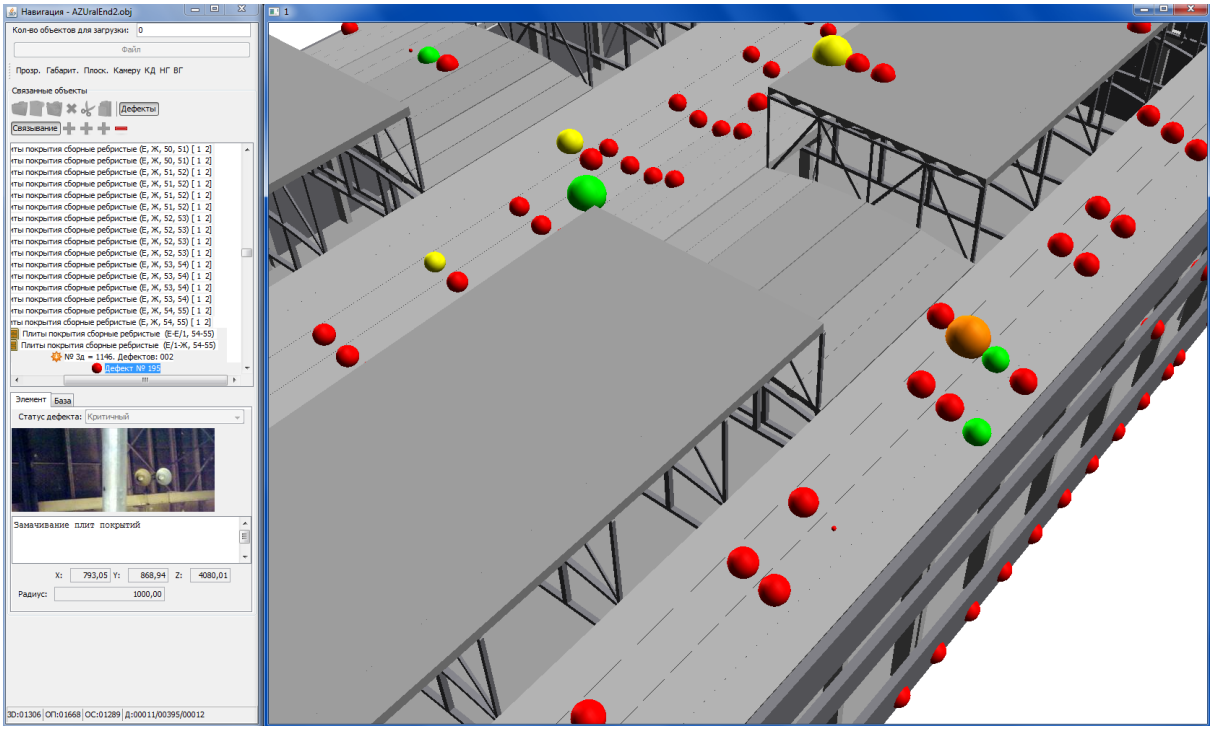


Рис. 33. Удаление информации о дефекте (повреждении)

Библиографический список

1. Анализ причин аварий и повреждений строительных конструкций / Под. ред. д.т.н. А.А. Шишкина. – М.: Стройиздат, 1964. - 290 с.

МЕТОДОЛОГИЯ И ИНСТРУМЕНТАРИЙ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ ЭЛЕКТРОННОГО ПАСПОРТА

УДК 004.9
Сорокин М.О.
ФГБОУ ВПО «Московский государственный строительный университет», г.Москва
Ткаченко М.С.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Sorokin M.O.
Moscow State Construction University, Moscow
Tkachenko M.S.
«WELD» company, Magnitogorsk, Chelyabinsk region

Аннотация
В работе рассмотрены методология и инструментарий разработки информационной системы электронного паспорта.

Ключевые слова:
Программное обеспечение, Жизненный цикл, Проектирование, Язык UML, объектно-реляционное проецирование, Библиотека Hibernate, Разработка информационной системы, Технология Java, Инструментальные средства, Реализация принципа непрерывной интеграции
Key words:
The software Life cycle, Design, UML, object-relational projection, Hibernate, Development of information system, Java Technology, tools, realization of the principle of continuous integration

Электронный паспорт зданий и сооружений, запатентованный специалистами ООО «ВЕЛД» в 2009 году, стал первым из ряда планируемых к выпуску программных комплексов, которые направлены на эффективное управление промышленными технологическими рисками на предприятиях всех отраслей экономики. Здесь мы рассмотрим методологию и инструментарий разработки информационной системы.

1. Этапы жизненного цикла программного обеспечения

Программные продукты, как и другие продукты производства за время своего существования проходят различные этапы. Под этапом понимается определенный временной интервал, на котором для программного средства характерен определенный вид деятельности. Совокупность всех таких этапов представляет его жизненный цикл. По определению Лешек А. Мацяшек в [1] жизненный цикл - «...упорядоченный набор видов деятельности, осуществляемый и управляемый в рамках каждого проекта по разработке ПО...». В литературе [1-4] прослеживается следующий перечень этапов жизненного цикла программного продукта:

1. Установление требований.

2. Спецификация требований.
3. Проектирование архитектуры.
4. Детализированное проектирование.
5. Реализация
6. Интеграция
7. Сопровождение (и окончательный вывод из эксплуатации).

При более укрупненном рассмотрении и акцентировании на разработке, можно выделить три этапа: анализ, проектирование и реализация. Рассмотрим этапы подробнее.

Первым, и одним из самых ответственных этапов является **установление требований.** Задачей этапа является выявление, явное определение, анализ и обсуждение требований к системе с заказчиком. При этом применяются анкеты, заполняемые заказчиком и будущими пользователями, изучается документация. Для прояснения сложных и неоднозначных аспектов системы используется быстрая разработка прототипа. Результатом этапа является документ, содержащий изложение требований. В терминологии, применяемой В.В. Липаевым в [5] результатом является **концепция информационной системы.**

Задачей этапа **спецификации требований** является моделирование требований и формирование детального описания данных и функций по их обработке. Кроме того, формализуются и «не функциональные» требования к системе: производительность, пригодность к использованию/сопровождению, безопасность, юридические требования, требования к эргономичности и удобству пользовательского интерфейса. Результатом этапа, согласно [5] является **техническое задание.**

В рамках **проектирования архитектуры** разрабатываются детализированные модели системной архитектуры и ее внутренние механизмы. При этом определяются используемые технологии (язык разработки, инструментальные библиотеки, целевая платформа, использ уемая СУБД, и другие). Кроме того, в рамках этапа Л.А. Мацяшек [1] различает:

- *Архитектурное проектирование* (описание системы в терминах составляющих ее модулей;
- *Детализированное проектирование* (подробные алгоритмы и структуры данных для каждого модуля);
- *Выбор стратегии решения* (выбор общего архитектурного решения разрабатываемой системы (файл-сервер, клиент-сервер, трех и многозвенная архитектура).

При **детализированном проектировании** внимание акцентируется на модулях системы. На данном этапе выполняется проектирование базы данных и пользовательского интерфейса системы.

На **этапе реализации** выполняется создание (настройка в случае наличия) инфраструктуры разработки и непосредственно реализация системы. Выполняется установка и настройка необходимого программного обеспечения. Производится организация совместных работ программистов и тестирующего персонала.

Интеграция подразумевает интеграцию реализованных модулей в систему. Интеграция модулей должна быть спланирована на ранних стадиях жизненного цикла. Последовательность реализации модулей должна позволять выполнять последовательную интеграцию реализованных модулей. При этом реализованные модули должны отличаться минимальной связанностью с другими модулями. В [1] подчеркивается, что «... Если система спроектирована недостаточно качественно, этап интеграции приведет к хаосу и поставит под угрозу весь проект по разработке системы».

После успешной передачи заказчику программного продукта начинается **этап сопровождения**. Согласно данным S.Schach [6] сопровождение является самым продолжительным этапом жизненного цикла и занимает 67% времени. Сопровождение состоит из трех стадий [7]:

- *Поддержка эксплуатации* (рутинные задачи сопровождения, необходимые для поддержания системы в состоянии готовности);
- *Адаптивное сопровождение* (отслеживание и анализ работы системы, настройка ее функциональных возможностей применительно к изменениям внешней среды. Настройка производительности и пропускной способности);
- *Улучшающее сопровождение* (перепроектирование и модификация системы для удовлетворения новых или изменившихся требований).

Заключительным мероприятием в жизненном цикле системы является **вывод из эксплуатации.**

Материалы настоящей главы посвящены методологии, применяемой на этапах проектирования архитектуры и детального проектирования. Кроме того, приводится подробное описание инструментария разработки и программного обеспечения, необходимого для организации инфраструктуры проекта на этапе реализации.

2. Объектно-ориентированный подход в проектировании и разработке системы

Л.А. Мацяшек в [1] констатирует: «На сегодняшний день объектно-ориентированный подход – единственный известный метод, позволяющий совладать с разработкой нового управляемого событиями отличающегося высоким уровнем интерактивности ПО». Объектно-ориентированный подход к разработке систем получил распространение в 1990-х годах. За последнее десятилетие XX века произошел бесповоротный переход от систем с командным режимом работы (командная строка) к системам с интерактивным, событийно-ориентированным графическим интерфейсом. В то же время информационные системы охватывали множество отраслей и сфер деятельности: возрастала их сложность, классические принципы системного подхода оказались несостоятельными.

Базовыми понятиями объектно-ориентированного подхода являются класс и объект. Под **классом** понимается общее описание сущности, содержащее описание его атрибутов и операций для их обработки. **Объект** – конкретный экземпляр сущности, содержащий конкретное значение атрибутов. С помощью атрибутов в классе определяются типы и набор данных, характеризующих данный класс. Операции класса служат для взаимодействия объектов и изменения значений атрибутов. Кроме того,

объект обладает состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяется в общем для них классе. Состояние объекта определяется перечнем всех свойств данного объекта и текущими значениями каждого из этих свойств. Поведение – это действия и реакции объекта, выраженные через изменения состояния и передачу сообщений. Для осуществления взаимодействия между объектами используется их связь: конкретное соединение, через которое объект (клиент) запрашивает услугу у другого объекта (сервера) или управляет им. Участвуя в связи, объект может выполнять одну из следующих трех ролей [8]:

- **Контроллер** (controller). Объект может выполнять операции с другими объектами, но сам никогда не подвергается воздействию других объектов.
- **Сервер** (server). Объект не выполняет операций с другими объектами, а лишь подвергается воздействию со стороны других объектов.
- **Агент** (proxy). Объект может выполнять операции над другими объектами и подвергаться воздействию с их стороны.

В рамках объектно-ориентированного подхода, различают: объектно-ориентированный анализ, проектирование и программирование [8]:

- *Объектно-ориентированный анализ* – метод анализа, исследующий требования к системе с точки зрения классов и объектов, относящихся к словарю предметной области. Актуальной задачей анализа является идентификация сущностей, их классификация и выявление механизмов взаимодействия.
- *Объектно-ориентированное проектирование* – метод проектирования, сочетающий процесс объектно-ориентированной декомпозиции и систему обозначения для представления логической и физической, а также статической и динамической моделей проектируемой системы.
- *Объектно-ориентированное программирование* – метод программирования, основанный на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определенного класса, а классы являются членами определенной иерархии наследования.

Ядром объектно-ориентированной технологии является *объектная модель* [8]. Основными принципами этой модели являются: абстракция, инкапсуляция, модульность, иерархия, контроль типов, параллелизм и постоянство (персистентность):

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко описывает его концептуальные границы с точки зрения наблюдателя. Различают [8]:

- *Абстракция сущности* – объект, представляющий собой полезную модель некой сущности в предметной области.
- *Абстракция действия* – объект, состоящий из обобщенного множества операций, каждая из которых выполняет однотипные функции.
- *Абстракция виртуальной машины* – объект, группирующий операции, которые либо вместе используются на более высоком уровне управления, либо сами используют некоторый набор операций более низкого уровня.
- *Произвольная абстракция* – объект, включающий в себя набор операций, не имеющих друг с другом ничего общего.

Инкапсуляция **–** процесс разделения элементов абстракции, определяющих ее структуру и поведение; инкапсуляция предназначена для изоляции контрактных обязательств абстракции от их реализации.

Модульность – свойство системы, разложенной на цельные, но слабо связанные между собой модули. Разбиение системы на модули дает ряд преимуществ: упрощается повторное использование кода, уменьшается связанность системы, что в свою очередь позволяет разрабатывать модули параллельно различными разработчиками. Кроме того, модуль служит отдельной единицей документирования и администрирования.

Иерархия **–** ранжирование, или упорядочивание абстракций. Примерами иерархии является **одиночное, множественное наследование и агрегация. Одиночное наследование** создает иерархию классов. При этом подкласс реализует или переопределяет существующую структуру и поведение суперкласса. Другими словами, наследование порождает иерархию обобщения/специализации, в которой подкласс конкретизирует более общую структуру или поведение своего суперкласса. Соответственно, при **множественном наследовании** подкласс наследует структуру и поведение нескольких базовых классов. Реализация множественного наследования связана с рядом трудностей: возможны конфликты имен атрибутов или операций суперкласса. Различные объектно-ориентированные языки программирования предоставляют разные реализации этой концепции.

Агрегация – описание отношения «целое/часть». Например, атрибутом класса может быть объект другого класса. Агрегация поддерживается всеми языками программирования, имеющими структуры, подобные записям.

Контроль типов – это правила использования объектов, не допускающие или ограничивающие взаимную замену объектов разных классов. Этот принцип реализуется на уровне языка программирования. Различают языки с сильным (строгим) и слабым контролем типов. Сильному контролю типов соответствует раннее связывание, слабому – позднее связывание. При раннем связывании контроль типов осуществляется на этапе компиляции, при котором фиксируется тип переменных и выражений. В случае позднего связывания типы переменных и выражений до этапа выполнения программы остаются неизвестными.

Параллелизм – свойство, отличающее активные объекты от пассивных. Под активным объектом понимается понятие, объединяющее две точки зрения: каждый объект (построенный на основе абстракции реального мира) может представлять собой отдельный поток управления (абстракцию процесса).

Персистентность – это способность объекта преодолевать временные рамки (то есть продолжать свое существование после исчезновения своего создателя) или пространственные пределы (то есть выходить за пределы своего первоначального адресного пространства).

Одним из преимуществ объектно-ориентированной технологии является простота повторного использования как отдельных классов, так целых модулей. Однако помимо повторного использования конкретных классов значительную практическую пользу несет использование так называемых **паттернов проектирования.** Под **паттерном проектирования** понимается описание взаимодействия объектов и классов,

адаптированных для решения общей задачи проектирования в конкретном контексте [9]. Применение паттернов позволяет реализовать гибкую архитектуру приложения и минимизировать связи между классами благодаря инкапсулированию абстракций. Кроме того, при использовании паттернов при проектировании упрощается общая сложность проекта: паттерны скрывают локальные аспекты реализации, таким образом предоставляя более высокоуровневые «блоки» при разработке проекта.

Показатели качества классов и объектов

В [9] определяются пять показателей качества классов и объектов:

1. Связанность (на уровне модулей) (coupling). В [10] определяется как «сила связей между отдельными модулями. Сильная связанность усложняет систему, поскольку тесно связанные модули гораздо сложнее воспринимать и модифицировать. Сложность можно уменьшить, разрабатывая системы с как можно более слабыми связями между модулями».
2. Связность (внутри модуля) (cohesion). Понятие аналогичное связанности, но в контексте отдельного класса или объекта: сила связей между элементами отдельного модуля.
3. Достаточность (sufficiency). Достаточность определяет, что класс или модуль содержит достаточно много характеристик, для обеспечения эффективного взаимодействия. Другими словами, количество атрибутов класса достаточно для описания сущности.
4. Полнота (completeness). Полнота означает что интерфейс класса или модуля описывает все операции, необходимые для эффективного взаимодействия.
5. Элементарность (primitiveness). Отражает наличие операций, для осуществления которых необходим доступ к внутреннему представлению класса.

3. Язык UML

UML (англ. Unified Modeling Language – унифицированный язык моделирования) [11] – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью.

Появление UML в 1994 году стало результатом совместной работы Гради Буча (Grady Booch) и Джима Рамбо (Jim Rumbaugh) в результате объединения их методов: метода Буча и OMT (Object Modeling Technique) [12]. Затем, эти методологии были объединены Айваром Якобсоном (Ivar Jacobson). Позднее, была создана компания Rational Software Corporation [8], в рамках деятельности, которой осуществлялось пропагандирование новых методов. Это привело к созданию первой версии UML. Со временем UML получил широкой распространение и привлек внимание многих специалистов. Стандартный язык моделирования был предложен консорциуму Object Modeling Group (OMG), создающему и поддерживающему стандарты в компьютерной индустрии [13]. В ноябре 1997 года консорциум OMG принял UML в качестве стандарта и с тех пор контроль над языком и его дальнейшее развитие относится к компетенции группы OMG. На русский язык переведены и изданы основные труды авторов UML. Например, можно отметить Руководство пользователя UML [14].

Проектирование информационной системы невозможно выполнить без моделирования. Моделирование является типичной практикой множества инженерных дисциплин. Например, в строительстве моделирование позволяет оценить воздействие окружающей среды (ветры, землетрясения) на сооружения. Моделирование необходимо для понимания системы. Моделирование позволяет определить необходимый уровень абстракции, и сосредоточиться на решении основных задач. При этом единственной модели никогда не бывает достаточно. Наоборот, для понимания любой нетривиальной системы необходимо разрабатывать большое число взаимосвязанных моделей. Язык UML позволяет описывать представления архитектуры системы на протяжении цикла ее разработки с различных точек зрения. Кроме того, UML закрепляет хорошо определенную семантику [15]: таким образом модель, написанная одним разработчиком, может быть однозначно интерпретирована другим, или даже инструментальной программой. Так как UML – язык графического моделирования, его словарь состоит из графических обозначений.

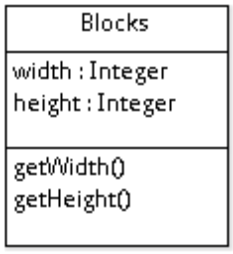
Согласно [14],словарь UML включает три вида строительных блоков: *сущности, связи, диаграммы*. При этом, *сущности* – это абстракции, которые являются основными элементами модели; *связи* соединяют их между собой; *диаграммы* группируют представляющие интерес наборы сущностей. Кроме того, различают четыре вида сущностей UML: *структурные*, *поведенческие*, *группирующие*, *аннотирующие*. *Структурные сущности* - «имена существительные» в моделях UML. Это в основном статические части модели, представляющие либо концептуальные модели, либо физические элементы. *Поведенческие сущности* – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение во времени и пространстве. *Группирующие сущности* – организационная часть моделей UML. Это «ящики», по которым можно разложить модель. *Аннотирующие сущности* – это поясняющие части UML моделей, иными словами, комментарии, которые применяются для описания, выделения и пояснения любого элемента модели. Табл. 1 содержит графическое описание обозначения структурных сущностей и их описание.

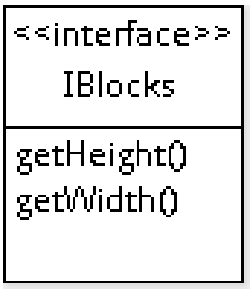
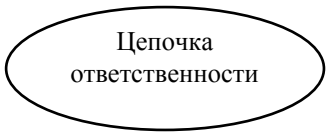
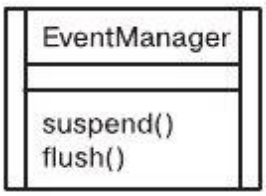
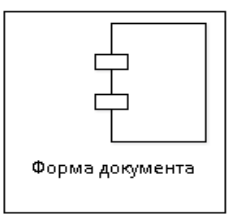
В табл. 2 представлены поведенческие сущности.

Табл. 3 объединяет сведения о *группирующих* и *аннотирующих* сущностях.

Существует четыре типа связей в UML: зависимость, ассоциация, обобщение, реализация. Описание каждой и графическое обозначение приведены в табл. 4.

Таблица 1

Графическое обозначение элемента	**Описание**
	*Класс (class)* – описание набора объектов с одинаковыми атрибутами, операциями, связями и семантикой. Графически класс изображается в виде прямоугольника, включающего имя, атрибуты и операции

Графическое обозначение элемента	Описание
	*Интерфейс (interface)* – набор операций, который специфицирует сервис класса или компонента. Интерфейс описывает видимое извне поведение элемента. Существенно, что интерфейс никогда не определяет детали реализации. Интерфейс изображается как класс с ключевым словом «interface» над его именем; атрибуты несущественны, за исключением констант. Однако интерфейс редко существует сам по себе. Интерфейс, представляемый классом для внешнего мира, изображается в виде маленького круга, соединенного линией с рамкой класса. Интерфейс, запрашиваемый классом от некоторого другого класса, представлен маленьким полукругом, соединенным с рамкой класса линией
	*Кооперация (collaboration)* – определяет взаимодействие и представляет собой совокупность ролей и других элементов, которые функционируют вместе, обеспечивая некоторое совместное поведение, представляющее нечто большее, чем сумма поведений отдельных элементов. Кооперация изображается в виде эллипса, нарисованного пунктирной линией, иногда включающего в себя лишь ее имя
	*Вариант использования (use case)* – описание последовательности действий, выполняемых системой и приносящих значимый результат конкретному *действующему лицу*. Варианты использования применяются для структурирования поведенческих сущностей модели. Графически вариант использования представлен эллипсом, нарисованным сплошной линией (обычно включает в себя только имя)
	*Активный класс (active class)* – класс, объекты которого являются владельцами одного или нескольких процессов или потоков и, таким образом, могут инициировать управляющие воздействия. *Активный класс* во всем подобен простому классу, за исключением того, что его объекты представляют собой элементы, поведение которых осуществляется параллельно с поведением других элементов. Изображается как класс с двойными боковыми линиями; обычно включает в себя имя, атрибуты и операции
	*Компонент (component)* – модульная часть системы, которая скрывает свою реализацию за набором внешних интерфейсов. Компоненты системы, разделяющие общие интерфейсы, могут замещать друг друга, сохраняя при этом одинаковое логическое поведение. Реализация компонента может быть выражена объединением частей и коннекторов; при этом части могут включать в себя более мелкие компоненты. Графически компонент представлен как класс со специальной пиктограммой в правом верхнем углу

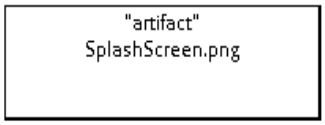
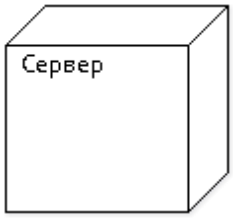
Графическое обозначение элемента	Описание
	*Артефакт (artifact)* – физическая и замещаемая часть системы, несущая физическую информацию. (Исходный код, исполняемые программы и скрипты. Изображается как прямоугольник, снабженный ключевым словом «artifact», расположенный над его именем
	*Узел (node)* – физический элемент, который существует во время исполнения и представляет вычислительный ресурс, обычно имеющий по меньшей мере некоторую память и часто – вычислительные возможности. Набор компонентов может находиться на узле, а также мигрировать с одного узла на другой. Узел изображается в виде куба, обычно содержащего лишь его имя

Таблица 2

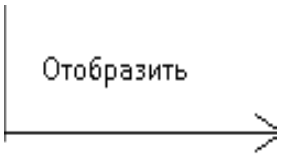
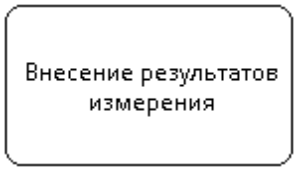
Графическое обозначение элемента	Описание
	*Взаимодействие (interaction)* – представляет собой поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели. Сообщение изображается в виде линии со стрелкой, почти всегда сопровождаемой именем операции
	*Автомат (state machine)* – представляет собой поведение, характеризуемое последовательностью состояний объекта, в которых он оказывается на протяжении своего жизненного цикла в ответ на события, вместе с его реакцией на эти события. Автомат включает в себя множество других элементов: состояния, переходы (из одного состояния в другое), события (сущности, которые инициируют переходы), а так же действия (реакции на переходы). Графически состояние представлено прямоугольником с закругленными углами, обычно с указанием имени и подсостояний, если таковые есть
	*Деятельность (activity)* – специфицирует последовательность шагов процесса вычислений. Во взаимодействии внимание сосредоточено на наборе взаимодействующих объектов, в автомате – на жизненном цикле одного объекта; для деятельности же в центре внимания – последовательность шагов безотносительно к объектам, выполняющим каждый шаг. Отдельный шаг деятельности называется *действием*. Изображается оно в виде прямоугольника с закругленными углами, включающего имя, которое отражает его назначение. Состояния и действия различаются по контекстам

Таблица 3

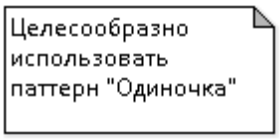
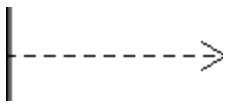
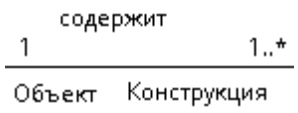
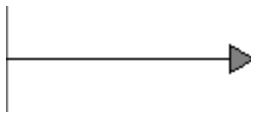
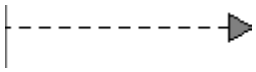
Группирующие сущности	
Графическое обозначение элемента	**Описание**
	*Пакет (package)* – механизм общего назначения для организации проектных решений, который упорядочивает конструкции реализации. Структурные сущности, поведенческие сущности и даже другие группирующие сущности могут быть помещены в пакет. В отличие от *компонентов* пакеты полностью концептуальны, то есть существую лишь на этапе разработки. Пакет изображается в виде папки с закладкой, обычно только с указанием имени, но иногда и содержимого
Аннотирующие сущности	
	*Примечание (note)* – простой символ, служащий для описания ограничений и комментариев, относящихся к элементу либо набору элементов. Графически представлен в виде прямоугольника с загнутым углом. Внутри помещается текстовый или графический комментарий

Таблица 4

Графическое обозначение элемента	**Описание**
	*Зависимость (dependency)* – семантически представляет собой связь между двумя элементами модели, в которой изменение одного элемента (независимого) может привести к изменению семантики другого элемента (зависимого). Графически представлена пунктирной линией, иногда со стрелкой; может быть снабжена меткой
	*Ассоциация (associacion)* – структурная связь между классами, которая описывает набор связей, существующих между объектами – экземплярами классов. *Агрегация (agregation)* – особая разновидность ассоциации, представляющая структурную связь целого с его частями. Изображается сплошной линией, иногда со стрелкой; иногда снабжена меткой и часто содержит другие пометки, такие как мощность и конечные имена
	*Обобщение (generalization)* – выражает специализацию или обобщение, в котором специализированный элемент (потомок) строится по спецификациям обобщенного элемента (родителя). Потомок разделяет структуру и поведение родителя. Графически обобщение представлено в виде сплошной линии с пустой стрелкой, указывающей на родителя

Графическое обозначение элемента	Описание
	*Реализация (realization)* – семантическая связь между классификаторами, когда один из них специфицирует соглашение, которого другой обязан придерживаться. Пример: реализация между интерфейсами и классами или компонентами, которые реализуют эти интерфейсы, или между вариантами использования и реализующими их кооперациями. Связь реализации в графическом исполнении – гибрид связей обобщения и зависимости

Диаграммы UML являются графическим представлением набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Назначение диаграммы состоит в визуализации системы с различных точек зрения, поэтому отдельная диаграмма – это проекция системы. Диаграмма может включать в себя любую комбинацию сущностей и связей, однако на практике используется лишь небольшое число общих комбинаций, состоящих из пяти наиболее часто применяемых представлений архитектуры программных систем. В «Руководстве пользователя UML» [14], составленными авторами языка, представлено 13 видов диаграмм. Краткое описание и назначение этих диаграмм приводится ниже.

Диаграмма классов (class diagram) – отражает набор классов, интерфейсов и кооперaций, а также их связи. Такие диаграммы чаще всего применяются для моделирования объектно-ориентированных систем. Предназначены для статического представления системы. Диаграммы классов, включающие активные классы, представляют статическое представление процессов системы.

Диаграмма объектов (object diagram) – отражает набор объектов и их связи. Диаграммы объектов представляют статические копии состояний экземпляров сущностей, описанных в диаграмме классов. Кроме того, диаграмма объектов представляет статическое представление дизайна или статическое представление процессов системы (аналогично диаграммы классов, но с точки зрения реальных или прототипных ситуаций).

Диаграмма компонентов (component diagram) – отражает инкапсулированные классы и их интерфейсы, порты и внутренние структуры, состоящие из вложенных компонентов и коннекторов. Диаграммы компонентов описывают статическое представление дизайна системы.

Диаграмма вариантов использования (use case diagram) – отражает набор вариантов использования и действующих лиц (которые являются специальным видом классов), а также их связи. Диаграммы этого типа описывают статическое представление вариантов использования системы. Большую значимость такие диаграммы имеют для организации и моделирования поведения системы.

Диаграмма взаимодействия (interaction diagram) – отражает взаимодействие, состоящее из набора объектов и ролей, включая сообщения, которые могут передаваться между ними. Диаграммы взаимодействия предназначены для динамического представления системы. *Диаграмма последовательности (sequence*

diagram) – это разновидность диаграммы взаимодействия, показывающая временную последовательность сообщений. *Диаграмма коммуникаций (communication diagram)* – разновидность диаграммы взаимодействия, показывающая структурную организацию объектов или ролей, отправляющих и принимающих сообщения.

Диаграмма состояний (state diagram) – показывает автомат (*state machine),* включающий в себя состояния, переходы, события и деятельности. Диаграммы состояний описывают динамическое представление объекта. Они особенно важны для моделирования поведения интерфейсов, классов или кооперации и подчеркивают событийно-зависимое поведение объекта.

Диаграмма деятельности (activity diagram) – показывает структуру процесса или других вычислений как пошаговый поток управления и данных. Диаграммы деятельности описывают динамическое представление системы. Они особенно важны при моделировании функций системы и выделяют поток управления между объектами.

Диаграмма размещения (deployment diagram) – показывает конфигурацию узлов-процессоров, а также размещаемые на них компоненты. Диаграммы размещения дают статическое представление размещения архитектуры. Узлы, как правило, содержат один или несколько артефактов.

Диаграмма артефактов (artifact diagram) показывает физический состав компьютерной системы. Артефакты представляют собой файлы, базы данных и подобные им физические наборы битов. Диаграммы данного типа часто применяются в сочетании с диаграммами размещения. Также показывают классы и компоненты, реализованные ими.

Временная диаграмма (timing diagram) – это диаграмма взаимодействий, показывающая реальное время жизни различных объектов или ролей, в противовес простой последовательности сообщений.

Диаграмма обзора взаимодействий (interaction overview diagram) – это гибрид диаграммы деятельности и диаграммы последовательности.

4. Понятие процесса разработки. «Гибкие» методологии

Филипп Крачтен (Philippe Kruchten)[1] в предисловии к [12] констатирует: «Программирование — это развлечение, но разработка качественных программ – тяжелый труд». Схожую мысль высказывает и В.В. Липаев в [5]: «Многие молодые программисты - «художники» лихо утверждают, что они могут писать программы со скоростью тысяч строк в неделю. Однако такие программы способны решать относительно небольшие, автономные задачи с неизвестным и, как правило, низким качеством». Так что же еще, кроме качественного проектирования и владением технологиями реализации является залогом разработки качественных программ? Ответ подсказывает модель технологической зрелости CMM: важным необходимым фактором является наличие формализованного процесса разработки.

1 Руководитель разработки Rational Unified Process, в 90-х годах руководил разработкой системой управления воздушными перевозками. Для успешной реализации проекта было привлечено 150 программистов.

Методологически, различают понятия «тяжелого» (heavy) и «гибкого» (agile) процесса. Согласно [16], «тяжелый» процесс характеризуется следующими особенностями:

- Множество артефактов, создаваемых в бюрократической атмосфере;
- Отсутствие гибкости и управляемости;
- Долгосрочное детальное планирование;
- Распределение человеческих ресурсов на длительный срок с охватыванием большей части проекта;

«Гибкий» (agile) процесс напротив, разрабатывался как адаптивный и устойчивый к изменениям требований, для него характерны [12]:

- Выбор небольшого набора видов деятельности и артефактов процесса. Для каждого проекта этот набор может быть своим, но всегда он должен оставаться небольшим;
- Анализ требований и проектирование не завершаются к моменту начала реализации. Требования и модель проектирования адаптивно настраиваются в течение нескольких итераций на основе обратной связи;
- Не существует подробного плана всего проекта. Разрабатывается лишь план высокого уровня или так называемый укрупненный план, в котором указывается дата завершения проекта и даты выполнения основных этапов, однако этот план не детализируется. Подробный план строится для каждой итерации в отдельности.

В настоящее время большую популярность получили «гибкие» методологии. В целом для них характерна «спиральная» модель разработки, противопоставляемая классической модели «водопада». На практике распространение получили конкретные «гибкие» модели реализации процесса разработки.

Одной из первых реализаций «гибкой» методологии является «Рациональный унифицированный процесс (Rational Unified Process – «RUP») [17], [18]. В рамках RUP различают 4 фазы разработки проекта: начало (определение границ проекта, принятие решение о целесообразности дальнейшей работы, оценка масштабов системы), проектирование (создание базовой архитектуры, минимизация главных технических рисков, оценка затрат на реализацию системы), построение (разработка рабочих версий продукта), внедрение (создание окончательной версии и поставка заказчику). При этом каждая фаза содержит одну или несколько итераций, которые направлены исключительно на достижение целей фазы. Кроме того, в рамках RUP определены четыре ключевых модельных элемента [18]: роли, задачи, артефакты, рабочие процессы. Роль – определяет виды деятельности и область компетенции и ответственности, которые должен иметь человек, выполняющий данную роль. Один человек, как правило, выполняет несколько ролей, и напротив – одну роль может выполнять несколько человек. Задача конкретной роли – единица работы, выполнения которой можно потребовать от человека, выполняющего данную роль. Как правило, задача имеет ясную цель по созданию или выполнению работ с артефактами проекта. Кроме того, задачи подразделяются на шаги, которые разделены на три главные категории [18]:

- шаги для *продумывания,* когда человек, выполняющий определенную роль, осознает суть задачи, проводит сбор и проверку входящих артефактов и формулирует результат;
- *исполняющие* шаги, когда ролью создается или модифицируются некоторые артефакты;
- *проверочные* шаги, когда ролью выполняется оценка результатов по определенному критерию;

Рабочий процесс – описание осмысленных последовательностей задач, создающих какой-либо полезный результат и описывающий взаимодействия между ролями.

В заключение, все элементы процесса сгруппированы в логические контейнеры, называемые дисциплинами. В рамках RUP предлагается 9 дисциплин: бизнес-моделирование, управление требованиями, анализ и проектирование, реализация, развертывание, тестирование, управление проектом, управление изменениями. Стоит отметить, что понятие RUP выходит за пределы теоретической методологии организации процесса. Компанией IBM [19] предлагается ряд программных продуктов, позволяющих обеспечить инфраструктурное обеспечение функционирования процесса. Авторы методологии RUP [17], [18] определяют ее следующим образом:

1. RUP – четко определенная и четко структурированная технология программной инженерии. В нем явно указываются все вехи проекта, кто и за что отвечает, как нужно все делать и когда.
2. RUP – технологический продукт, предоставляющий настраиваемую технологическую основу для программной инженерии. Продукт RUP обеспечивает настройку, а также разработку авторских процессов.
3. RUP – это подход к разработке программного обеспечения: итеративный, архитектурно-центрический и основанный на прецедентах использования. Он описывается во многих официальных документах и книгах, но наиболее полную информацию можно найти в самом продукте RUP.

Кроме того, отличительной чертой RUP является гибкость и адаптивность для проектов различного масштаба и команд с разным опытом разработки. В зависимости от конкретной ситуации RUP позволяет реализовать как более формализованный процесс, так и напротив, менее формализованный и более гибкий, и адаптивный к внешним изменениям. В любом случае, основными идеями RUP являются: минимизация рисков, нацеленность на исполняемую программу даже на ранних стадиях, выявление и детализация требований и их выполнение.

Другой методологией организации процесса разработки программного обеспечения является Microsoft Solution Framework (MSF). В этом подходе сосредоточен огромный опыт разработки программного обеспечения другого лидера отрасли – компании Microsoft. В целом, приоритеты близки к методологии RUP, однако существуют некоторые различия. Например, участники команды разработки имеют равные полномочия. Таким образом, обеспечивается равная ответственность участников проекта и поддерживается дисциплина обязательств. Как отмечается в [20], модель команды MSF реализует подход компании Microsoft к структуризации

участников команды и их действий, приводящих к успеху проекта. Фундаментальными принципами модели команды MSF являются:

- команда – группа равнозначных сотрудников с четко подотчетностью, разделяемой ответственностью и свободным общением;
- защита интересов каждого ключевого представителя, вовлечённого в проект, голос которого может повлиять на успех;
- необходимая гибкость в масштабировании команды.

Кроме того, в рамках MSF предложены роли участников проекта, причем каждая роль рассмотрена с точки зрения выполняемых операций и действий. В рамках командной модели MSF (MSF Team Model) различают следующие роли: бизнес-аналитик, менеджер проекта, архитектор, разработчик, тестировщик, релиз-менеджер, администратор базы данных и разработчик баз данных. Практическая ценность MSF заключается в наличии интегрированных в среду разработки Visual Studio [21] инструментов, позволяющую организовать единую информационную среду для управления и совместной работы команды над проектом. Подробная информация о программном обеспечении, его архитектуре и настройки содержится в книге Д.В. Кознова [22].

MSF и RUP по существу являются существенно большими, чем теоретическая методология и концепция. Фактически, данные технологии представляют собой комплексные решения, позволяющие реализовать и автоматизировать реализацию процесса разработки, характерную для конкретной компании.

В середине 2000х годов популярность получили и ряд других гибких методологий, например, Scrum и XP (extremal programming).

По существу, для Scrum характерны следующие черты:

- итерации являются четко фиксированными по времени;
- команда состоит из участников и так-называемого Scrum-мастера, выполняющего общее координирование работ участников группы;
- Scrum команда является основной единицей разработки, участники являются мультифункциональными (могут выполнять как программирование, так тестирование и другие необходимые функции);
- команда (в том числе Scrum-мастер) подчиняется руководителю проекта;
- оценка объемов работ, сложности и других решающих факторов производится командой. В задачи руководителя проекта входит создание списка задач проекта (основанного на прецедентах использования) и расстановка важности каждой задачи;
- регламентированы обязательные мероприятия: встречи для планирования итерации, ежедневные встречи команды для оценки результатов планирования работы, демонстрация результата по окончанию итерации;
- результатом каждой итерации должна являться готовая версия продукта, с определенной новой функциональностью, разработанной в рамках итерации;
- в процессе разработки главными являются два документа: список историй проекта (описание прецедентов использование), который составляется руководителем проекта и план итерации (Sprint backlog), который содержит описание историй и

задач, реализуемых в рамках текущей итерации (составляется командой, под руководством Scrum-мастера).

Вопросы практического применения методологии Scrum подробно описаны в книге [23]. Кроме того, распространение получила процедура сертификации, при которой программисты изучают методологию и получают подтверждающие повышение квалификации сертификаты.

Другой, не менее популярной методологией стало «экстремальное программирование» (XP). Основным предположением, допускаемым в рамках XP, является отказ от признания экспоненциальной зависимости стоимости внесения изменений в систему от времени. Напротив, допускается, что стоимость внесения изменений является примерно одинаковой с течением времени. Таким образом, во всей методологической базе прослеживается тенденция к решению текущих краткосрочных задач проекта, решение которых приводит к удовлетворению текущих же нужд заказчика. Кроме того, во главу артефактов проекта ставится исходный код, потому что только качественный исходный код, а не обилие иных документов составляет рабочее решение, которое должно работать и приносить выгоды заказчику. В книге Кента Бека [24] обозначаются четыре ценности XP: коммуникация, простота, обратная связь, храбрость. Благодаря оперативной и эффективной *коммуникации* между участниками проекта и представителя заказчика достигается существенное снижение уровня информационной энтропии, что в свою очередь положительно сказывается на снижении рисков проекта, связанных с ней. Для обеспечения коммуникации в рамках XP обязательным является наличие в команде разработчиков представителя заказчика, уполномоченного принимать решения. Вследствие ориентированности на решении краткосрочных задач *простота* при реализации решений является одним из ключевых моментов. Простую систему проще модифицировать в будущем, поэтому для решения текущих задач используется наиболее простое работающее решение.

Обратная связь – обеспечивает качество системы. Под обратной связью подразумевается обязательная практика написания модульных тестов: с помощью тестов, программист может оценить, как работоспособность написанного кода, так и влияние внесенных изменений на общую функциональность системы. Для обеспечения обратной связи так же используется подход непрерывной интеграции: после внесения изменений в программный код специализированными средствами обеспечивается автоматизированная сборка проекта, выполнение тестов и, при необходимости, развертывание на тестовом сервере. Методология XP подразумевает коллективное владение кодом: содержимое хранилища исходного кода является общим, и члены команды обладают равными правами на изменение произвольных частей/модулей системы. По этой причине возможны ситуации, когда «... у кого-нибудь из вашей команды возникает сумасшедшая идея, в результате реализации которой сложность всей системы существенно уменьшится, он обязательно попробует реализовать свою идею» [24]. В таких случаях важное значение имеет решительность и *храбрость*. Однако, храбрость должна быть обоснована и принятие решений принесет положительный эффект только в сочетании с коммуникацией, простотой и надежной обратной связью.

Методология «экстремального» программирования основной деятельностью при разработке программных систем признает написание исходного кода, то есть непосредственно программирование. В то же время для того, чтобы при решении текущих задач минимизировать риски в долгосрочной перспективе, методология XP декларирует ряд подходов, комплексное применение которых делает исходный код максимально простым, гибким и главное управляемым. Эти подходы не являются принципиально новыми и разработанными непосредственно в рамках XP, напротив они известны и проверены временем. XP лишь декларирует их совместное и обязательное использование. Таковыми подходами являются: *разработка через тестирование, парное программирование, непрерывная интеграция.* Ниже они рассмотрены подробнее.

Разработка через тестирование. Тестирование является неотъемлемой частью процесса разработки, независимо от методологии. Благодаря тестированию возможно оценить качество программной системы. Однако в классических подходах к разработке, тестирование выполняется стихийно: по результатам завершения определенных этапов или перед передачей программы заказчику. Подход *разработки через тестирование* переводит тестирование из отдельного процесса в повседневную задачу программиста. Этот подход буквально определяет, что перед тем как написать программный код, решающий конкретную задачу, необходимо предварительно написать тест, с помощью которого можно проверить корректность решения задачи еще не написанным кодом. Затем, реализуется программный код решения задачи и проверяется написанным тестом. Операция повторяется итеративно: код теста совершенствуется (добавляются новые тестовые случаи), в результате чего написанный код решения задачи не проходит проверку тестом. После этого модифицируется решение. Практическая ценность покрытия кода тестами можно сформулировать следующим образом: покрытие кода тестами позволяет объективно оценить текущее состояние системы и влияние на нее конкретных наборов изменений программного кода в процессе их внесения. Таким образом, снижается общая информационная энтропия, связанная с неопределенностью текущего состояния системы в конкретный момент времени. Подробнее данный подход освящен в литературе: например, в книге К. Бека [25] или [26].

Парное программирование – подход, при котором решением одной задачи занимается два программиста. Причем решение задачи происходит совместно: в распоряжении пары только один компьютер с одним комплектом клавиатура/мышь. Согласно известной русской поговорке: «Одна голова хорошо, а две лучше». Настоящий подход подтверждает ее на практике. При работе двух человек над одной задачей – велика вероятность принятия более простых и изящных программных решений. Кроме того, программирование в паре, согласно [24], повышает трудовую дисциплину: меньше вероятность, что будет нарушен регламент разработки и цикл написание, тестирование, интеграция. И наконец, программирование в паре с периодичной сменой состава пар положительно способствует информационному обмену между участниками: компетентность участников команды относительно любых частей или модулей системы выравнивается. Актуальность и интерес к данной

технологии отражены в актуальной на момент исследования в статье в The Wall Street Journal в августе 2012 года [27].

Непрерывная интеграция – обеспечение сборки проекта и выполнения автоматизированного тестирования с помощью тестов, с последующим развертыванием на тестовом сервере при каждом внесении изменений в исходный код проекта. Выполнение непрерывной интеграции необходимо для получения программистами «обратной связи» от системы при внесении изменений. Выполнение комплекса этих мероприятий должно проходить в относительно короткие сроки, так как эта операция выполняется программистами множество раз в течении разработки. Таким образом, внося изменения в исходный код системы, после применения тестов, программист получают информацию о влиянии их на корректную работоспособность системы. Организация непрерывной интеграция – задача организации инфраструктуры разработки: для решения этой задачи существуют специализированные программы, интегрируемые с системой контроля версий. Как правило, такое программное обеспечение устанавливается на отдельный сервер. Настоящая работа содержит сведения о таком сервере, сконфигурированном для разработки информационной системы учета параметров безопасности зданий и сооружений.

5. Технология объектно-реляционного проецирования. Библиотека Hibernate

Л.А. Мацяшек в книге [1] одним из существенных недостатков объектного подхода к разработке программ отмечает так называемый *семантический разрыв* между концепцией и реализацией при использовании реляционной базы данных на этапе реализации проектного решения. В частности, отмечается, что «Хотя анализ и проектирование могут проводиться итеративно с наращиванием возможностей, в конце концов разработка достигает этапа реализации, которая требует трансформации решения применительно к реляционной базе данных». Однако, в настоящее время разработана технология *объектно-реляционного проецированя (object-relational mapping – ORM* [28]*)*, назначение которой состоит в представлении возможности прозрачного хранения объектов в реляционной базе данных. Использование ORM значительно упрощает реализацию: в отдельных случаях не требуется даже написание SQL запросов для элементарных операций сохранения, изменения и удаления объекта из базы данных.

В общем виде, принцип ORM можно описать следующим образом: объекты в объектно-ориентированной системе есть совокупность данных и методов их обработки; совокупность данных (атрибутов и их значений) объекта представляет собой описание таблицы в реляционной базе данных для хранения объектов одного класса; назначение ORM состоит в предоставлении простого механизма для сохранения объектов в таблице реляционной базы данных таким образом, что бы одна строка таблицы представляла собой набор данных конкретного объекта. Кроме того, слой ORM обычно абстрагирует приложение от конкретной используемой СУБД: инструментарий ORM поддерживает несколько различных СУБД с возможностью настройки без изменения кода, использующего непосредственно механизм ORM. Для выполнения сложных запросов, характерных для реляционных баз данных может использоваться внутренний

диалект языка SQL, который так же позволит исключить использование в коде приложения SQL кода, специфичного для конкретной СУБД. Таким образом, с использованием ORM приложение получает преимущество в виде легкого переноса между различными системами СУБД.

Реализации ORM существуют для многих языков программирования. Табл. 5 содержит сведения о библиотеках ORM, составленные на основе данных, представленных в [29].

Таблица 5

Реализации ORM

Название	**Платформа**	**Лицензия**	**Сайт проекта**	**Текущая версия (09.2012)**
Hibernate	java	GNU LGPL	http://hibernate.org	4.1.7 Final от 6 сентября 2012 г.
Apache Cayenne	java	Apache License 2.0	http://cayenne.apache.org	3.1 от 2 декабря 2012 г.
NHibernate	.net	GNU LGPL	http://nhforge.org	3.2 (июль 2011 г.)
Django (одним из компонентов содержит ORM)	python	BSD	http://djangoproject.com	1.4.1 (от 7 августа 2012 г)
Doctrine	php	GNU LGPL	http://doctrine-project.org	2.2.0 (от 29 января 2012 г.)
ADO.NET Entity Framework	.net	Ms-RSL	http://msdn.microsoft.com/en-us/library/e80y5yhx.aspx	4.3 (февраль 2012 г.)
Circumflex ORM	scala	BSD-like	http://circumflex.ru	2.3 (от 3 апреля 2012 г.)

Практический интерес при использовании ORM представляют следующие два вопроса:

1. Низкоуровневая реализация хранения связей и ассоциаций между объектами (отображение в схеме базы данных связей между объектами).
2. Отображение иерархии классов – принципы формирования схемы базы данных для хранения иерархий классов.

Для реализации информационной системы в рамках настоящего исследования использована библиотека Hibernate. Ниже рассмотрены ответы на поставленные вопросы, а также освящены некоторые аспекты использования Hibernate.

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного проецирования. Представляет собой свободное программное обеспечение с открытым исходным кодом. Распространяется по лицензии GNU Lesser General Public License. Проект имеет десятилетнюю историю: первая версия библиотеки по данным [30] датируется июлем 2002 года. Стоит отметить

качественную документацию [31], покрывающую различные аспекты использования. Кроме того, издана книга [32].

Одной из первых задач при использовании средства ORM является его настройка. Настройка Hibernate возможна двумя способами: формирование xml файла, содержащего параметры настройки и их значения, и программная конфигурация во время выполнения программы. Возможна так же и комбинация: базовые настройки (параметры подключения к базе данных) можно хранить в конфигурационном файле, а параметры, влияющие на поведение библиотеки изменять при необходимости во время выполнения. Другим аспектом конфигурации является непосредственно настройка проецирования: при её выполнении осуществляется указание классов, экземпляры которых должны проецироваться в базу данных. Hibernate поддерживает 2 варианта настройки отображения классов: первый – создание для каждого проецируемого класса файла с индивидуальной конфигурацией; второй – использование компонента Hibernate Annotations. Компонент Hibernate Annotations позволяет избавиться от дополнительных xml файлов для конфигурации отображения конкретного класса: вместо них используется механизм аннотаций языка java. В процессе разработки информационной системы автором был использован второй вариант: использование аннотаций позволяет в одном файле иметь информацию как о классе, так и параметры его проецирования. Такой подход упрощает сопровождение исходного кода и внесение модификаций: при удалении или модификации полей необходимая для проецирования информация находится в том же файле, таким образом, снижается вероятность случайных ошибок, связанных с несоответствием структуры класса параметрам проецирования.

Следующим аспектом технологии, представляющим интерес с практической точки зрения, является проецирование наследования. Hibernate поддерживает три стратегии проецирования наследования, описанные в спецификации Enterprise Java Beans 3.1: отдельная таблица на класс (Table per Class), простая таблица для всей иерархии классов (Single Table per Class Hierarchy) и стратегия присоединяемых подклассов (Joined Subclass).

Отдельная таблица для класса. Используется на высшем уровне иерархии классов. Стратегия поддерживает ассоциации один ко многим, в случае, если они двунаправленны.

Простая таблица для всей иерархии классов. При использовании данной стратегии, атрибуты всех классов (подклассов и класса-родителя) располагаются в одной таблице. Для определения класса конкретной записи используется отдельный столбец. Значение в данном столбце индивидуально для каждого класса.

Стратегия присоединяемых подклассов. Особенностью данной стратегии является наличие отдельной таблицы для хранения специфичных атрибутов подкласса. Значения атрибутов, наследуемых от родительского класса, размещаются в таблице родительского класса. Связь между таблицами осуществляется с помощью внешнего ключа: идентификатор записи в таблице родительского класса является первичным ключом записи в таблице подкласса.

6. Технические средства разработки информационной системы

В **разделе 1.5** настоящей работы рассмотрена философия свободного программного обеспечения. Кроме того, отражена нормативная база, показывающая актуальность и перспективы внедрения и развития программного обеспечения в нашей стране. Настоящий раздел посвящен вопросу выбора технологии разработки информационной системы. Кроме того, рассмотрено программное обеспечение, необходимое для организации инфраструктуры разработки: учета ошибок, система контроля версий. Приведены сведения о реализации принципа непрерывной интеграции в рамках разрабатываемой системы.

Технология Java

Java [33] – объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт код, поэтому они могут работать на любой виртуальной Java машине, вне зависимости от компьютерной архитектуры. Однако в широком смысле Java является несколько большим, чем очередной язык программирования и набор инструментальных средств для сборки программ. В рамках Java существуют несколько основных семейств:

- Java SE – (Java Standart Edition) – основное издание Java, содержит компиляторы, SPI, Java Runtime Environment; Предназначена для создания пользовательских приложений, в первую очередь — настольных систем.
- Java EE – (Java Enterprise Edition) – набор спецификаций для создания программного обеспечения уровня предприятия.
- Java ME – (Java Micro Edition) – редакция Java, созданная для использования в устройствах, ограниченных по вычислительной мощности, например, в мобильных телефонах, КПК и встроенных системах.
- Java FX – перспективная технология, являющаяся следующим шагом в эволюции Java в качестве платформы для клиентских приложений корпоративных систем и настольных приложений.
- Java Card – технология, предоставляющая безопасную среду для приложений, работающих на смарт-картах и других устройствах с очень ограниченным объемом памяти и возможностями обработки.

Платформа Java хорошо документирована: в рамках технологии существует инструмент JavaDoc, позволяющий генерировать документацию к исходному коду в удобном представлении. Кроме того, издан ряд книг, часть из которых получили широкую популярность, например двухтомник «Java 2. Библиотека профессионала» К. Хорстманна и Г. Корнелла [34], [35], «Полный справочник по Java» [36] Г. Шилдта, «Искусство программирования на Java» [37] Г. Шилдта и Д. Холмса. Так же стоит отметить оперативное появление документации для новых версий платформы: уже изданы книги, освещающие новую редакцию платформы Java 7 от 28 июля 2011 года [33], например, [38]

Использование семейства технологий Java SE позволяет разрабатывать приложения, работающие без изменений на любой современной операционной

системе, поддерживающей запуск виртуальной Java машины. Для реализации графического интерфейса пользователя, согласно [33] еще в 1998 году была разработана библиотека Swing. Эта библиотека содержит реализации необходимых элементов управления, интерфейса и непосредственно различных окон. При ее использовании, внешний вид приложений не зависит от используемой платформы. Внешний вид компонентов определяется наборами Look&Feel, часть из которых поставляется в комплекте с jvm, часть доступна в виде продуктов внешних разработчиков. Построение пользовательских интерфейсов с использованием Swing рассмотрено в [39], [40].

Однако стоит отметить, что библиотека Swing по ряду объективных причин признана устаревшей[2]. Для использования в новых проектах специалистами компании Oracle рекомендуется использовать технологию JavaFX. JavaFX [41] – следующий шаг в эволюции Java как клиентской платформы. JavaFX разработан для предоставления легкой, аппаратно-ускоренной платформы для построения пользовательского интерфейса для бизнес-приложений промышленного уровня. JavaFX поддерживает 3 различных подхода к разработке пользовательского интерфейса:

1. Проектирование интерфейса с помощью визуального редактора форм. Компанией Oracle предлагается продукт JavaFX Scene Builder, с помощью которого возможно осуществить быстрый дизайн интерфейса без необходимости написания кода. В настоящий момент (сентябрь 2012 года) продукт доступен для платформ Windows и MacOS. Подробное описание возможностей программы доступно в статье [42]. Кроме того, на конференции JavaOne/OracleDevelop была продемонстрирована ранняя версия модуля для среды разработки Netbeans, предоставляющего аналогичные возможности.
2. Декларативное описание интерфейса. JavaFX позволяет описать структуру пользовательского интерфейса в отдельном файле (FXML). Данный подход позволяет на технологическом уровне разделить реализацию логики приложения и графического интерфейса пользователя. Стоит отметить, что результатом работы средств визуального проектирования является непосредственное описание интерфейса в формате XML.
3. Программное создание пользовательского интерфейса. В данном случае не используется FXML файл. Для реализации пользовательского интерфейса программист использует классы, предоставляемые пакетом разработчика JavaFX SDK. Например, для создания кнопки существует класс Button, для создания древовидной структуры и табличного представления данных предназначены классы TreeView и TableView соответственно.

В настоящее время издано несколько книг, посвященных практическому применению данной технологи: [43] посвящена JavaFX версии 1.3; [44] рассматривает примеры использования JavaFX версии 2.0.

[2] Например, по причине не использования аппаратного ускорения при прорисовке интерфейса, особенностей внутренней архитектуры библиотеки, исчерпавшей потенциал развития. По данным информации от специалистов компании Oracle на конференции JavaOne/OracleDevelop в Москве в 2011,2012, 2013 годах.

В новой версии, Java SE 7, помимо исправления большого количества ошибок представлено несколько новшеств. Например, в качестве эталонной реализации Java SE 7 использован не проприетарный пакет JDK, а его открытая реализация OpenJDK. Кроме того, все поставляемые Oracle бинарные файлы эталонной реализации Java SE 7 собраны на основе кодовой базы OpenJDK. Сама эталонная реализация полностью открыта под лицензией GPLv2 с исключениями GNU ClassPath, разрешающими динамическое связывание с проприетарными продуктами [33]. Таким образом, Java в настоящее время является современной, гибкой, активно-развивающейся платформой для разработки программного обеспечения, распространяемого как по коммерческим, так и по свободным лицензиям.

Инструментальные средства

На этапе формирования требований и концептуального проектирования системы необходимо средство UML моделирования. Среди специализированных свободных инструментов построения UML диаграмм стоит отметить ArgoUml. Программа написана на Java и может функционировать на любой современной операционной системе, поддерживающей данную технологию. Распространяется по лицензии Eclipse Public License. Полностью поддерживает стандарт UML 1.4 для разработки диаграмм. Стоит отметить возможность генерации программного кода на следующих языках: C++, C#, Java, PHP4, PHP5. Сохранение результатов возможно, как в виде графического файла (актуально для импорта в документацию), так и в виде файла собственного формата, с возможностью последующего редактирования. Проект имеет более чем 10-летнюю историю: развитие программы происходило параллельно с развитием языка UML. Одна из первых версий была продемонстрирована в апреле 1995 г. [45].

Другим характерным инструментом, используемым на этапе формирования требований, является текстовый процессор. Одним из флагманских продуктов в данной области является LibreOffice, разрабатываемый организацией The Document Foundation (http://documentfoundation.org). Данный проект основан на кодовой базе проекта OpenOffice.org. Ответвление данного проекта от OpenOffice.org произошло после приобретения компанией Oracle компании Sun. Пакет программ LibreOffice включает средство для создания текстовых документов со сложным форматированием, таблицами, диаграммами и др., средство для создания таблиц (аналог Excel), средство для создания презентаций и средство для создания баз данных.

Наиболее специфичными являются средства, применяемые на этапе реализации и тестирования. Ключевым инструментом взаимодействия разработчиков при написании программного кода является система контроля версий. Такая система позволяет разработчику в произвольный момент времени фиксировать в хранилище наборы изменений написанного им программного кода. При этом система предоставляет возможность в произвольный момент времени получить зафиксированную ранее версию исходного кода. Ключевой является возможность объединения наборов изменений, сделанных в разное время различными разработчиками. Использование такой системы – залог успеха в случае, когда в написании программного кода задействовано более 1 человека. В настоящее время популярна категория распределенных систем контроля версий. Они пришли на смену централизованным

системам. В них хранилище располагалось на отдельной системе, и все разработчики работали с ним напрямую. В случае отсутствия соединения с хранилищем, фиксирование изменений было невозможно. Распределенные системы контроля версий лишены данного недостатка: в них каждый участник разработки имеет полную копию хранилища. Одной из современных систем контроля версий является Mercurial. Она разработана на языке python, поэтому функционирует на любых платформах, где доступен интерпретатор. Основным интерфейсом является интерфейс командной строки. Однако существуют сборки с интегрированной программой TortoiseHg, которая предоставляет удобный графический интерфейс. Программа поставляется с набором скриптов для организации доступа к хранилищу по протоколу http. Возможен запуск программы в режиме локального сервера для предоставления доступа к хранилищу по локальной сети. Mercurial используется такими крупными проектами, как OpenJDK, Netbeans, Python, Symbian OS, что подтверждает отсутствие проблем с производительностью при большом объеме кода. Программа распространяется по лицензии GNU GPL v2 или выше.

Реализация принципа непрерывной интеграции

При рассмотрении методологии экстремального программирования затронута техническая дисциплина – принцип непрерывной интеграции. В рамках разработки рассматриваемой в работе информационной системы данный принцип был реализован с использованием свободного программного обеспечения.

Ключевым программным продуктом, реализующим выполнение задач сборки, является сборочный сервер. В функции данного сервера входит отслеживание изменений в хранилище исходного кода проекта и выполнение сборок проекта, при обнаружении изменений. Для нужд проекта был выбран проект Jankins. Jenkins представляет собой Java EE приложение с web интерфейсом. В качестве сервера приложений был выбран Apache Tomcat, ввиду своей простоты и наличия в репозитории пакетов дистрибутива Debian Linux, на базе которого был построен сборочный сервер.

Стоит отметить, что настройка автоматизированной сборки на сервере не является тривиальной задачей. Во-первых, необходимо обеспечить наличие на сервере необходимых для сборки библиотек и ресурсов (файлы изображений, шаблоны отчетов). Во-вторых, при сборке в среде разработки (Netbeans) сборка осуществляется с помощью внутренних инструментов среды разработки, на сервере же сборка не предполагает их использование. Таким образом, сформирована первая задача: написание скриптов для обеспечения корректной сборки проекта, включая библиотеки и используемые ресурсы.

Другой, не менее важной с практической точки зрения задачей является сборка установочного пакета проекта. Результат компиляции программы разработчиком – набор исполняемых файлов, требующий для функционирования корректно установленных переменных окружения. Практика распространения программ подразумевает упаковку компонентов программы и ее ресурсов в программу установки: приложение, позволяющее пользователю самостоятельно корректно установить программное обеспечение на компьютер. Важным моментом является

использование в проекте технологии Java: система может функционировать без изменений на различных платформах, поэтому к программному обеспечению для формирования программы-установщика предъявляются дополнительные требования: оно должно быть разработано на Java и должно поддерживать установку программы на платформы, поддерживаемые самой программой.

В рамках работ по разработке системы данные вопросы были успешно решены. Далее рассмотрено решение поставленных задач.

Одним из специализированных инструментов для автоматизации процесса сборки программного продукта является Apache Ant [46]. Apache Ant – это библиотека и утилита командной строки для управления процессом сборки проектов. Основное применение утилиты – сборка Java приложений. В более общем случае, Ant может быть использован для управления любыми процессами, которые могут быть описаны в терминах целей и задач. Использование Ant подразумевает написание сценария для сборки в формате xml. При этом используются набор собственных тегов Ant, которые позволяют сформировать описание целей и использовать задачи Ant, таких как сборка исходного кода, формирование Jar файла или копирование файлов в файловой системе. Стоит отметить, что среда разработки Netbeans для сборки Java SE проектов так же использует Ant: сценарий для сборки формируется средой разработки на основе внутренних шаблонов, а также пользовательских параметров, например, путей расположения библиотек, зависимостей проекта и ряда других. Кроме того, в сценариях, сформированных средой разработки, используются параметры, передаваемые средой разработки. В целом, изучение сборочных сценариев проекта показал, что для автоматизации сборки в среде сборочного сервера целесообразнее реализовать собственные сценарии, чем адаптировать служебные сценарии среды Netbeans.

Статьи [47], [48] содержат исчерпывающую информацию для написания собственного сценария сборки. Структура разрабатываемой информационной системы подразумевает наличие нескольких проектов в среде Netbeans. На момент разработки, существовало 3 проекта: PassportModule – главный компонент клиентского приложения: запускается непосредственно пользователем, осуществляет загрузку необходимых модулей и формирование пользовательского интерфейса; PassportLib – инструментальный пакет библиотек, содержащий общие для различных модулей классы; CatalogModule – модуль электронного каталога документации; TubeModule – модуль электронного паспорта промышленных труб. Реализация такого сценария является тривиальной задачей. Результатом выполнения скрипта является каталог, содержащий собранные jar файлы проекта, а также требуемые для корректного функционирования библиотеки и ресурсы. Кроме того, сценарий формирует конфигурационный файл с параметрами подключения к базе данных.

Следующей задачей является формирование установочного пакета информационной системы. Для решения этой задачи был использован проект IzPack [49]. Установочные пакеты, сформированные с помощью IzPack, являются кроссплатформенными и функционируют на любой операционной системе, поддерживающей Java SE 6 и выше. Кроме того, IzPack является программным

обеспечением с открытым исходным кодом, и распространяется по лицензии Apache Software License 2.0. Настройка параметров сборки установочного пакета формируется с помощью XML файла. Конечные пакеты позволяет запускать установку как в режиме графического интерфейса, так и в режиме командной строки (при использовании специфичных опций). В базовой поставке IzPack содержится тестовый проект, который можно использовать как основу, для создания собственных конфигураций. В статье [50] Поп Дюваль рассматривает практически все вопросы использования IzPack: загрузка и установка, редактирование тестовой конфигурации, компиляция установочного пакета. Применив данное средство для упаковки системы, можно получить удобное средство её распространения, с установкой которого может справиться обычный пользователь.

Следующим логичным шагом автоматизации является вызов задачи построения установочного пакета из Ant скрипта сборки проекта. Это является тривиальной задачей благодаря поддержки Ant со стороны IzPack. IzPack поставляется с библиотекой для Ant, предоставляющий реализацию задачи запуска сборки программы-установщика. Таким образом, запуск единственного Ant скрипта позволяет решить задачу сборки проекта, копирования ресурсов и библиотек и формирование программы-установщика.

Практическое использование принципа непрерывной интеграции показало необходимость наличия как в программе-установщике, так и в самом приложении информации о версии исходного кода в хранилище системы контроля версий из которого собрана текущая бинарная версия. Такая информация необходима при тестировании: при обнаружении ошибки можно легко получить версию исходного кода, соответствующую данной сборке. Информация о текущей версии доступна в системе контроля версий, поэтому получения этой информации возможно автоматизировать. Ввиду использования при разработке проекта системы контроля версий Mercurial, для получения информации о версии была использована библиотека Ant4hg [51].

Результатом выполненных работ является автоматизированная среда непрерывной интеграции:

- исполняемый на сервере экземпляр системы Jenkins отслеживает изменения в хранилище исходного кода проекта;
- при обнаружении изменений исходного кода, выполняется разработанный в настоящем разделе скрипт, формирующий установочный пакет для системы;
- командами командной оболочки bash выполняется перемещение собранного установочного пакета в каталог, доступный пользователям из локальной сети;
- по результатам выполнения сборки сервером Jenkins осуществляется рассылка уведомлений разработчикам с использованием протокола Jabber: разработчики в режиме реального времени получают оперативную обратную связь о успешности процесса сборки проекта.

7. Замечания по выбору инструментов разработки

В настоящем разделе приведены сведения о том стеке технологий, которые использовались при разработке системы электронной паспортизации «Электронный паспорт». Стоит отметить, что в целом, существуют и другие технологии: в частности, .net, разрабатываемая корпорацией Microsoft, технологии html5, позволяющие реализовать интерфейсы информационных систем в рамках интернет – браузера и многие другие. Выбор технологии, как правило, основывается на личном опыте конкретной команды разработчиков, перспектив развития рынка программного обеспечения и в целом. Важно понимать, что применение гибких методологий является актуальной тенденцией. Гибкие команды являются достаточно самостоятельными и высокоэффективными. На примере методологии экстремального программирования рассмотрены технические дисциплины и подходы, используемые при разработке: непрерывная интеграция и автоматизированное тестирование. В вопросах разработки информационной системой стоит различать следующие принципиальные моменты:

- методология: необходимый набор методик, лучших практик и подходов к решению той или иной задачи;
- инструменты: языки программирования, средства сборки, тестирования и проектирования для целевой платформы.
- инфраструктура: технологии и средства обеспечения работы процесса разработки с точки зрения гибких методологий.

Библиографический список

1. Мацяшек, Лешек, А. Анализ требований и проектирование систем. Разработка, информационных систем с использованием UML / Перевод с англ. и редакция к.т.н. В.М. Неумоина. – М.: Вильямс, 2002.
2. Думаков А.А. Проектирование информационных систем: учебное пособие. – Томск: Изд-во Томского политехнического университета, 2011. – 258 с.
3. Гвоздеева Т.В., Баллод Б.А. Проектирование информационных систем: учебное пособие. - Ростов-на-Дону: Феникс, 2009 - 508 с.
4. Управление программными проектами: достижение оптимального качества при минимуме затрат / Шафер, Ф. Дональд, Роберт Фатрелл, Т., Шафер, И. Линда // Пер с англ. – М.: Издательский дом «Вильямс», 2003. - 1136 с.
5. Липаев В.В. Качество программных средств: Методические рекомендации / Под общей ред. проф., д.т.н.. А.А. Полякова. – М.: Янус-К, 2002. 400 с, 47 ил.
6. Schach S. Object-oriented & classical software engineering sixth edition. Mc Graw Hill / Stephen R. Schach. - New York, 2010 - 688 p.
7. Maciaszek, L.A. Database design and implementation. Prentice Hall/ Maciaszek L.A. - New York, 1990 - 383 p.
8. Объектно-ориентированный анализ и проектирование с примерами приложений / Буч, Гради, Максимчук, Роберт А., Энгл, Майкл У., Янг, Бобби Дж., Коналлен, Джим, Хьюстон, Келли А.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2008. – 720 с.: ил. - Парал. тит. англ.
9. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. – СПб.: Питер, 2012. - 368 с.: ил.
10. Левин В.И. История информационных технологий. – М.: Национальный Открытый Университет «ИНТУИТ», 2007 - 336 с.

11. ISO/IEC 19505-2:2012. Information technology - Object Management Group Unified Modeling Language (OMG UML). - Part 2: Superstructure.
12. Ларман К. Применение UML и шаблонов проектирования. 2-е издание / Пер с англ. – М.: Издательский дом «Вильямс», 2004. - 624 с.: ил. - Парал. тит. англ.
13. Object Management Group [Электронный ресурс] - Режим доступа: http://www.omg.org/
14. Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Второе издание. ДМК. - М., 2006 - 496 с.
15. Rambaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual. Addison-Wesley / J. Rambaugh, I. Jacobson, G. Booch 1999 - 568 p.
16. Fowler M. Put your process on a Dieet, Software Development, CMP Media, Dec., 2000.
17. Kruchten P. The Rational unified Process. An introduction, 2nd edition, MA: Addison-Wesley, 2000. - 336 p.
18. Кролл П., Крачтен Ф. Rational Unified Process - это легко. Руководство по RUP. Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2004. - 432 с.
19. IBM Rational Unified Process (RUP) [Электронный ресурс] - Режим доступа: http://www-01.ibm.com/software/rational/rup/
20. Microsoft Corporation Microsoft Solutions Framework: Гибкая методология разработки программного обеспечения / Русская редакция, 2008 - 127 с.
21. Visual Studio 2013 Preview | Microsoft Visual Studio [Электронный ресурс] - Режим доступа: http://www.microsoft.com/visualstudio/
22. Д.В. Кознов. Введение в программную инженерию - М.: Национальный Открытый Университет «ИНТУИТ», 2009 - 325 с.
23. Kniberg H. Scrum and XP from the Trenches How we do Scrum. InfoQ, 2007 - 168 p.
24. Бек К. Экстремальное программирование. – СПб.: Питер, 2002. - 224 с.
25. Бек К. Экстремальное программирование: разработка через тестирование. Библиотека программиста. – СПб.: Питер, 2003. - 224 с.
26. Криспин Л., Грегори Д. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд / Пер. с англ. – М.: ООО «И.Д. Вильямс», 2010. - 464 с. ил: - Парал. тит. англ.
27. Walker J. Computer Programmers Learn Tough Lession in Sharing / The wall street journal: Europe Edition, 26 August 2012.
28. Barnes J. Object-Relational Mapping as a Persistence Mechanism for Object-Oriented Applications. Department of Mathematics and Computer Science at Macalester College in Saint Paul, Minnesota. // Jeffrey M. Barnes, 2007. - 113 p.
29. Scott W. Ambler. Mapping objects to relational databases. Ibm developerWorks [Электронный ресурс] - Режим доступа: http://www.ibm.com/developerworks/library/ws-mapping-to-rdb/
30. Elliot J., O'Brein T., Fowler R. Harnessing Hibernate. O'Relly Media / James Elliott, Timothy M. O'Brien, Ryan Fowler, Baijing Cambridge Farnham Koln Paris Sebastopol Taipei Tokyo, 2012. - 382 p.
31. Javae J., Johnson P. Jboss in Action. Manning / Javid Jamae, Peter Johnson - Greenwich, 2009 - 492 p.
32. Bauer C., King G. Java Persistence with Hibernate. Manning / C. Bauer, King G - Greenwich, 2007. - 841 p.
33. Schildt H. Java, A beginner's guide, 5th edition. MC Graw Hill / Schildt H. - New York Chikago San Francisco Lisbon London Madrid Mexico City Milan New Delphi San Juan Seoul Singapore Sydney Toronto, 2012 - 618 p.
34. Хорстманн, Кей С., Корнелл, Гари Java 2. Библиотека профессионала, том 1. Основы, 7-е изд.: Пер. с англ. - М.: Издательский дом «Вильямс», 2007. - 896 с. : ил. - Парал. тит. англ.

35. Хорстманн, Кей С., Корнелл, Гари Java 2. Библиотека профессионала, том II. Тонкости программирования, 7-е изд.: Пер. с англ. - М.: Издательский дом «Вильямс», 2007. - 1168 с. : ил. - Парал. тит. англ.
36. Шилдт Г. Java. Полное руководство, 8-е изд.: Пер. с англ. - М.: ООО «И.Д. Вильямс», 2012. - 1104 с.: ил. - Парал. тит. англ.
37. Шилдт Г., Холмс Д. Искусство прогграммирования на Java.: Пер. с англ. - М.: Издательский дом «Вильямс», 2005. - 336с.: ил. - Парал. тит. Англ.
38. Reese M., Reese J Java 7 New features Cookbook Richard M. Reese, Jennifer L. Reese, Packt Publishing / - Birmingham - Mumbai, 2012 - 385 p.
39. Шилдт, Г. SWING: руководство для начинающих: Пер. с англ. - М.: ООО «И.Д. Вильямс»,2007. - 704 с.
40. Портянкин И.А. Swing: Эффективные пользовательские интерфейсы. Библиотека программиста. – СПб.: Питер, 2005. - 528 с.
41. JavaFX Developer Home [Электронный ресурс] - Режим доступа: http://www.oracle.com/technetwork/java/javafx/overview/index.html
42. C. Castillo JavaFX Scene Builder Overview, Release 1.0 [Электронный ресурс] - Режим доступа: http://docs.oracle.com/javafx/scenebuilder/1/overview/jsbpub-overview.pdf
43. DiMarzio J.F. JavaFX A Beginners Guide. MC Graw Hill / J.F. DiMarzio - New York Chikago San Francisko Lisbon London Madrid Mexico City Milan New Delphi San Juan Seoul Singapore Sydney Toronto, 2011 - 304 p.
44. Carl P. Dea. JavaFX 2.0: Introduction by Example. Apress / Carl P. Dea - New York, 2011. - 175 p.
45. ArgoUML: ArgoUml Project History [Электронный ресурс] - Режим доступа: http://argouml.tigris.org/history.html
46. The apache ant project [Электронный ресурс] - Режим доступа: http://ant.apache.org/
47. Technology As If people mattered: Weblog - Ant in ten steps [Электронный ресурс] - Режим доступа : http://jroller.com/rolsen/entry/ant_in_ten_steps
48. Чепмен М. Apache Ant 101: Моментальная компоновка Java-программ [Электронный ресурс]: Режим доступа: http://www.ibm.com/developerworks/ru/edu/j-apant/section2.html
49. IzPack - Package once. Deploy everywhere. [Электронный ресурс] - Режим доступа : http://izpack.org
50. Paul Duvall. Automation for the people: Wielding wizard-based installers. [Электронный ресурс] - Режим доступа: http://www.ibm.com/developerworks/library/j-ap11258/index.html?S_TACT=105AGX99&S_CMP=CP
51. ANT4HG, ant task for hg - mercurial - [Электронный ресурс] - Режим доступа: http://ant4hg.free.fr/

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ЭЛЕКТРОННОГО ПАСПОРТА

УДК 004.9

Сорокин М.О.
ФГБОУ ВПО «Московский государственный строительный университет», г.Москва
Гаврилкович В.И.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Sorokin M.O.
Moscow State Construction University, Moscow
Gabrilovich V.I.
«WELD» company, Magnitogorsk, Chelyabinsk region

Аннотация
В работе рассмотрены концепция информационной системы, прецеденты использования, описаны реализация принципа модульности, методология разработки частной информационной модели, реализация модуля электронного архива документации, адаптивная многоуровневая группировка данных в модуле «Электронный каталог».

Ключевые слова:
Электронный паспорт, Концепция информационной систем, Прецеденты использования, Модульность, Принцип модульности, Метаобъекты, Информационная модель, Древовидные структуры данных, Электронный каталог, Электронный архив документации, Динамическая иерархия, Алгоритм динамической многоуровневой группировки, частная информационная модель
Key words:
The electronic passport, the Concept of information systems, use cases, Modularity, modularity Principle, the meta-objects, Information model, the Tree structure data, Electronic catalog, Electronic archiving, Dynamic hierarchy, the Algorithm for dynamic multi-level groups, private information model

Вопрос создания электронного паспорта зданий и сооружений сегодня стал одним из самых актуальных, когда речь заходит о сохранности и продолжении срока службы строительных объектов. Такой программный комплекс уже разработан, запатентован и рекомендован к реализации. Разработчиками стали инженеры ООО «ВЕЛД», г.Магнитогорск.

1. Концепция информационной системы. Прецеденты использования

В настоящее время мероприятия по снижению аварийности эксплуатируемых строительных объектов с использованием информационных технологий, можно разделить на две категории:

1. Внедрение автоматизированных систем мониторинга параметров сооружений. Современные системы автоматизированного мониторинга – сложные программно-аппаратные комплексы. Их возможности варьируются от простого фиксирования изменений измеряемого параметра, до комплексного анализа результатов изменения комплекса параметров, и их влияние на техническое состояние сооружения в целом.

2. Внедрение систем автоматизации процессов обеспечения безопасности строительных объектов. Такие мероприятия оправданы с точки зрения [1]: только качественно выполняемый процесс на всех стадиях может породить качественный результат. В приложении к безопасности эксплуатируемых объектов, разработка и внедрение таких систем позволит эффективно, в короткие сроки, с высокой точностью и достоверностью определить техническое состояние эксплуатируемого объекта.

Особый интерес представляют мероприятия второй категории: программные средства, реализующие автоматизацию по своей природе являются легко тиражируемыми, что позволяет массово применять удачные решения и получать максимальный эффект. Среди работ в данном направлении стоит отметить работы Коргина А.В. [2], Столярова А.Н. [3]. Так в работе [2] предложена методика автоматизации инженерных исследований реконструируемых сооружений. В рамках методики основной акцент сделан на использовании современного цифрового измерительного и геодезического оборудования, позволяющего автоматизировать стадию камеральной обработки измеренных значений. Кроме того, благодаря использованию современных средств передачи данных и внедрению безбумажного взаимодействия между исполнителями работ достигается общее повышение эффективности. В работе [3] элементы автоматизации применены процессу проведения экспертизы технического состояния строительного объекта. Значительным результатом работы является разработка и наполнение базы данных типовых решений по усилению строительных конструкций. Кроме того, обе работы объединяет наличие компонентов автоматизации формирования отчетных документов: системы позволяют компоновать базовые версии отчетных документов с результатами исследования, требующих минимальной ручной доработки.

Однако, с точки зрения реализации CALS технологии в целом для строительных объектов, данные технологии автоматизируют лишь отдельные процессы, реализуя информационную поддержку в контексте этих процессов. Разумеется, данные о объекте, собранные в рамках автоматизированных процессов могут быть использованы в дальнейшем. Однако общая тенденция такова, что для эксплуатируемых зданий на стадии эксплуатации отсутствуют информационные системы, позволяющие вести учет технического состояния в привязке ко времени. В следствии этого первоначальный этап (сбор сведений о объекте и его состоянии) является одним из самых продолжительных в приведенных выше методиках. Таким образом, для обеспечения возможности внедрения CALS технологии для строительных объектов необходимо наличие системы, обеспечивающий отражение изменения технического состояния объекта в целом и параметров конструкций, систем и оборудования с течением времени.

Принимая во внимание Распоряжение Правительства РФ [4] автором выполнена разработка такой системы с использованием свободного программного обеспечения. В рамках реализации проекта апробирована возможность организации процесса разработки и формирования инфраструктуры проекта исключительно на основе свободных программных продуктов [5].

Функционально для реализации принципа CALS/ИПИ система обладает следующими возможностями:

1. Прозрачная интеграция с данными о строительном объекте, накопленными за предшествующие стадии жизненного цикла объекта. (Система предназначается для эксплуатируемых зданий, однако документация и материалы стадий изысканий и строительства должны быть доступны из системы).
2. Отражение конструктивной характеристики строительного объекта. (В рамках CALS технологии информационные системы формируют частную информационную модель объекта. С точки зрения безопасности эксплуатации центральным объектом модели является информационная модель строительного объекта, отражающая его конструктивную характеристику).
3. Учет значений параметров безопасности строительного объекта, и оценка динамики их изменений во времени. Базовый анализ значений, и оповещение при выходе значения из диапазона допустимых значений.
4. Автоматизированное формирование документов, сопутствующих процессы контроля технического состояния. Формирование паспорта строительного объекта с отражением его действительного технического состояния на момент формирования документа.

Проектирование системы выполнено с использованием объектно-ориентированного подхода. Проведена объектная декомпозиция системы и сформирована объектная модель. Благодаря принципу модульности, являющимся одним из основных элементов объектно-ориентированного подхода основная функциональность системы разработана без привязки к конкретному типу строительного объекта. Базовое ядро системы содержит модели строительных конструкций, параметров безопасности и процессов измерения их значений и собственно модель процесса контроля технического состояния. С точки зрения принадлежности к объекту эти модели ссылаются на абстрактный «Строительный объект», реализациями которого в конкретных случаях являются различные виды сооружений.

Основное назначение информационной системы – обеспечить хранение и первичный анализ информации о параметрах безопасности строительного объекта. Информационная система должна стать единым источником информации как о самом строительном объекте, так и о измерениях и значениях параметров безопасности объекта с течением времени. В общем случае имеет место 2 вида информации, сопровождающей строительный объект:

1. Неструктурированная и слабоструктурированная информация. Описывает процессы, сопровождающие строительный объект (в том числе связанные с безопасностью). Примером такой информации являются документы. На разных стадиях жизненного цикла объекта, его сопровождают различные документы. На стадии проектирования, важной является проектная и исполнительная документация. В то же время эти документы являются первичным источником информации для проведения ремонтов, выполнения поверочных расчетов при изменившихся условиях эксплуатации и пр. Таким образом, информационная система должна предоставлять функционал для

хранения, структурирования и поиска образов документов – оцифрованных версий документации.

2. Информация о значениях параметров безопасности. Информация данного вида содержит количественные показатели технического состояния. Такие показатели подлежат наблюдению с течением времени. Информационная система должна позволять описать параметр безопасности и хранить значения этого параметра, измеряемые в разное время. Описание параметра должно содержать пороговые значения: при выявлении системой отклонения значения параметра от допустимого диапазона – система должна оповещать эксплуатирующий персонал.

Параметр безопасности является вполне конкретной сущностью. Например, он может представлять характеристику материала конструкции. Для разных видов конструкций один и тот же параметр может иметь разные допустимые диапазоны значений. Например, железобетонные колонны и фермы могут иметь разную нормативную прочность. Таким образом, информационная система должна обеспечить описание объекта: конструктивное решение, информацию о материалах. Фактическая конструкция строительного объекта должна иметь свое отражение в информационной системе.

С точки зрения безопасности чрезвычайно важной является информация о наличии дефектов и повреждений строительных конструкций. Система должна предоставлять возможности для хранения информации о дефектах и повреждениях, а также отражать принадлежность к конструкции.

Система должна предоставлять возможности по формированию отчетной документации. Одним из таких документов является «Паспорт строительного объекта», рекомендуемая форма и набор полей которого приведены в приложении к [6], другим – «Ведомость дефектов и повреждений» – содержащий соответствующую сводную информацию о дефектах и повреждениях объекта. Кроме того, целесообразна возможность печати информации, отражающей динамику изменения значений параметров безопасности.

Одним из принципов объектно-ориентированного подхода отмечается *модульность*. Благодаря использованию объектно-ориентированного подхода функциональность системы целесообразно разделить на модули. Так, самостоятельным модулем может служить функциональность для работы с документами. Другой модуль, может объединять функции для работы с параметрами безопасности. Изначальная разработка модульной системы позволит в дальнейшем обеспечить простое расширение функциональности: например, при разработке новых модулей автоматизирующих различные процессы. Другим направлением может быть легкое добавление новых отчетов и методов представления информации. Таким образом, разрабатываемую систему можно представить в качестве базовой платформы для автоматизации любых информационных процессов, затрагивающих строительные объекты. На рис. 1 представлена диаграмма вариантов использования разрабатываемой системы.

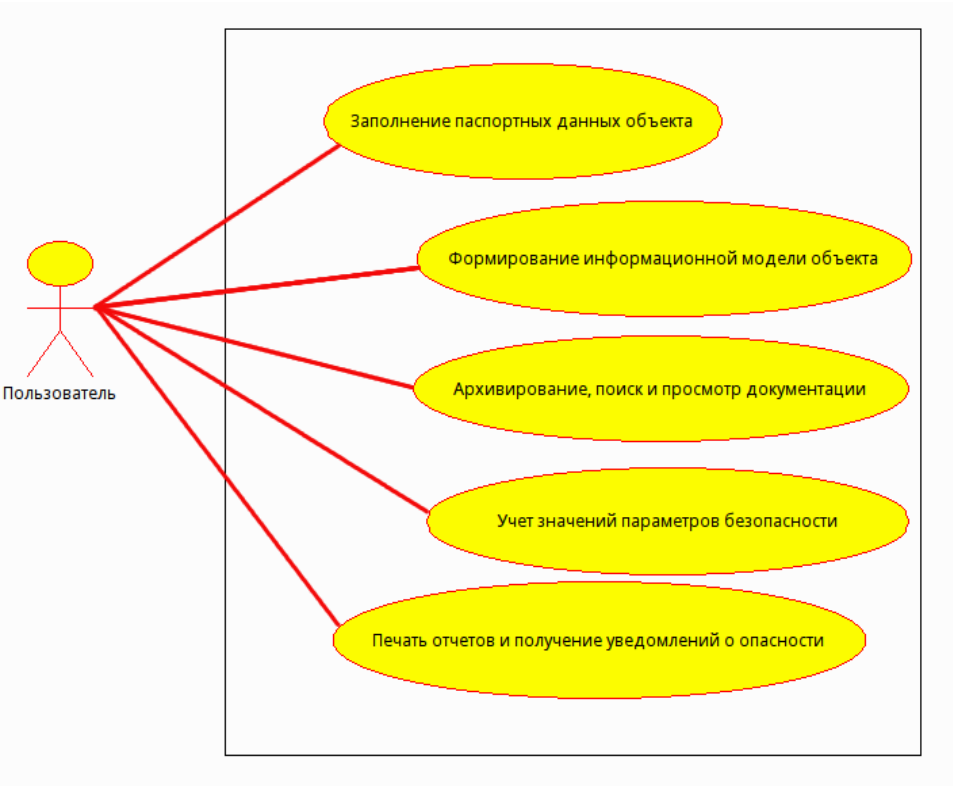


Рис. 3. Варианты использования системы

2. Реализация принципа модульности

Модульность – один из аспектов объектной модели, составляющей основу объектно-ориентированного подхода к разработке программ. Согласно [7]: «Модульность – это разделение программы на фрагменты, которые компилируются по отдельности, но связаны между собой». Кроме того, модульность тесно связана с инкапсуляцией: при использовании модулей инкапсуляция происходит на более высоком уровне абстракции. Так, инкапсуляция на уровне класса скрывает данные класса и предоставляет публичный интерфейс для изменения состояния этого класса. Инкапсуляция на уровне модулей скрывает уже внутреннее устройство модуля: скрываются класс, объекты и их связи целиком: для взаимодействия с внешними классами модуль имеет внешний интерфейс. Таким образом, модуль является «физическим контейнером» высокого уровня, возникающим при логическом проектировании системы. Согласно [7] *Модульность* определяется как «свойство системы, разложенной на цельные, но слабо связанные между собой модули».

Разбиение системы на модули – обычная практика при разработке крупных проектов. Разбиение на модули должно учитывать, как технические факторы, так и организационные. С одной стороны, разбиение на модули должно допускать повторное использование кода. С другой, разбиение на модули должно приводить к допустимому уровню связанности: это позволит разрабатывать несколько модулей параллельно разными разработчиками.

Для реализации принципа модульности предлагается архитектура приложения, представленная на рис. 3.2.

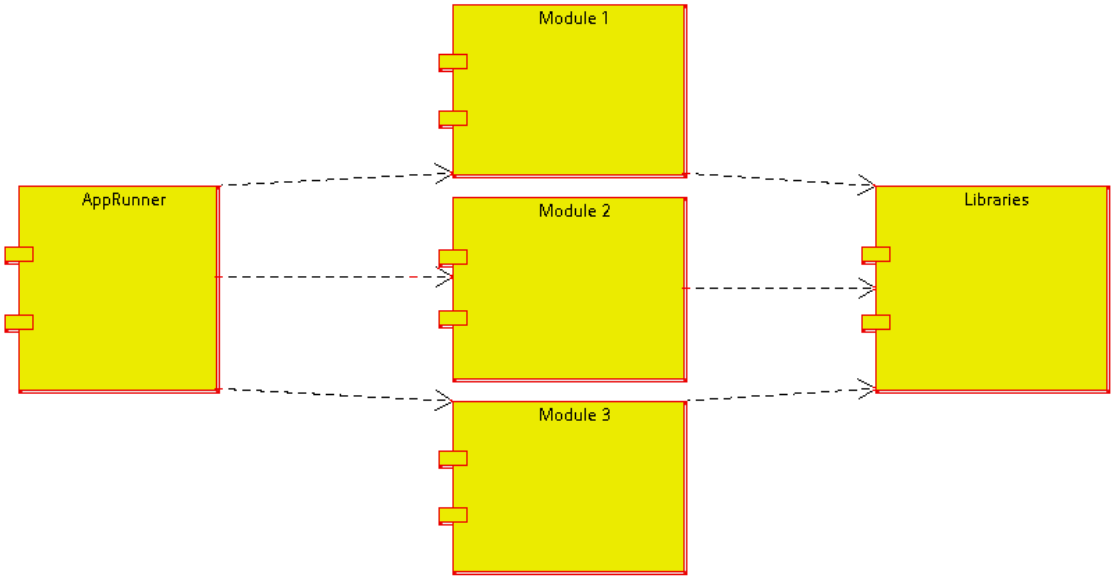


Рис. 2. Архитектура модульного приложения

Основная цель при разработке архитектуры модульного приложения – уменьшить связанность и зависимости модулей. На представленной диаграмме данная цель достигается путем вынесения общего кода, используемого всеми модулями системы в отдельный библиотечный модуль (*Libraries*). Такой модуль может содержать общие для системы элементы пользовательского интерфейса, логику для взаимодействия с базой данных или механизмы протоколирования пользовательских операций. Конкретные модули (*Module1, Module 2, Module 3*) используют функциональность из *Libraries* для реализации непосредственно функциональности модуля. Таким образом, конкретный модуль, например, Module 1, является зависимым от модуля библиотек *Libraries* как на стадии компиляции, так и во время выполнения программы.

Рассмотрим структуру модуля более детально. Для того, чтобы программный код системы не имел ссылок на конкретные модули и позволял исполнение модулей, программный код которых неизвестен во время компиляции системы, необходимо общее решение. В контексте данной задачи эффективным решением является использование паттерна проектирования «фасад», представленного в [8]. «Фасад» является паттерном, структурирующим объекты, назначение которого заключается в предоставлении унифицированного интерфейса вместо набора интерфейсов некоторой подсистемы. Таким образом, «фасад» определяет интерфейс более высокого уровня, который упрощает использование модуля. Графически «фасад» представлен на рис.3.

В рамках разработки информационной системы учета параметров безопасности строительных объектов, автором реализован абстрактный класс, определяющий внешний интерфейс класса, представляющего «фасад» модуля системы (рис. 4).

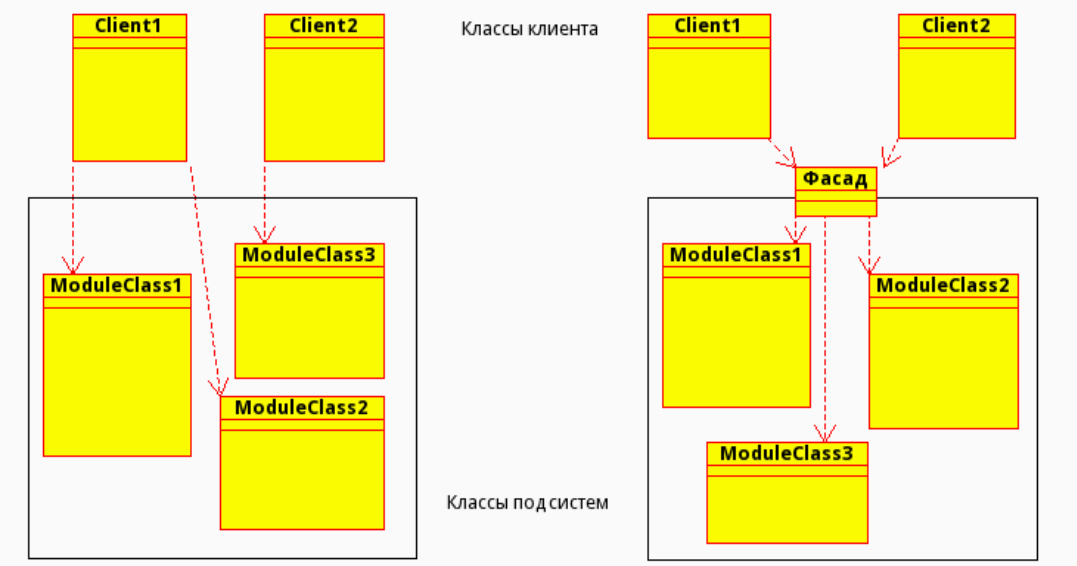


Рис. 3. Паттерн «Фасад»

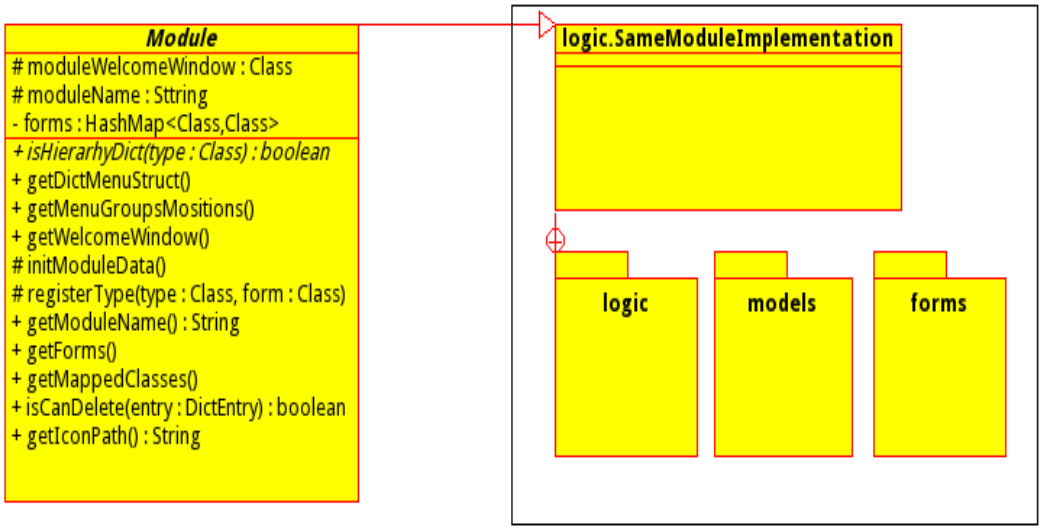


Рис.4. Спецификация интерфейса для модулей системы

Класс *Module* определяет функции, необходимые для интеграции модуля в систему. Например, метод *getMappedClasses()*– возвращает все классы, описывающие данные конкретного модуля; метод *getDictMenuStruct()* – определяет элементы меню системы, которые так же индивидуальны для каждого модуля; *getForms()* – возвращает классы экранных форм, предназначенных для отображения и редактирования данных. Разумеется, набор функций класса *Module* индивидуален для каждого проекта, и настоящая реализация приведена для наглядной иллюстрации предлагаемого подхода. Однако стоит отметить, что реализация интерфейса или класса *Module* не должна быть привязана к функциональности конкретного модуля, напротив: роль данного класса – предоставить метаинформацию о модуле, достаточную для его интеграции в систему. Для интеграции в систему уникальной функциональности модуля, при разработке интерфейса модуля необходимо предусмотреть механизмы интеграции модуля в систему. В приведенном примере для этих целей служат функции *getDictMenuStruct(), getDictMenuPositions(), getWelcomeWindow()*. Функция *getWelcomeWindow()* – возвращает класс модуля, являющейся наследником *JPanel*: при обращении

пользователя в системе к модулю, инициализируется экземпляр данного класса и формируется экранная форма, предоставляющая интерфейс к функциональности модуля. Согласно диаграмме 3 – форма является классом конкретного модуля и пакета *forms*: такой класс имеет доступ ко всей функциональности модуля. Функции *getDictMenuStruct()* и *getDictMenuPositions()* – возвращают данные, используемые системой для формирования меню системы. Теперь рассмотрим компонент *AppRunner*, представленный в правой части диаграммы, представленной на рис.2.

Назначение *AppRunner* в предложенной архитектуре модульного приложения достаточно простое: данный компонент инициализирует необходимые модули, и на основании данных полученных от них, формирует элементы пользовательского интерфейса системы. Благодаря использованию паттерна «Фасад», и определении его для всех модулей, код AppRunner может быть освобожден от ссылок на конкретные реализации модулей класса *Module.*

Не освещенным остается вопрос: как получить объекты реализаций интерфейса *Module*, без явной инициализации в коде, так что бы сборка *AppRunner* не требовала наличия связей с конкретными модулями? Ответ кроется в использовании механизма *Java Reflection*, представленного в одной из немногих книг [9], посвященных его использованию.

Механизм *Java Reflection*, согласно [9], предоставляет возможность работающей программы опрашивать себя и программную среду, а затем изменять свое поведение в зависимости от результатов. Для выполнения этого, программа содержит собственное представление: такую информацию называют метаданные. В языке Java этак информация организована в объекты, называемые **метаобъекты.**

Язык Java предоставляет следующие классы, описывающие метаобъекты:

- **java.lang.Class** – класс метаобъектов, отражающий класс объекта; Базовый класс *java.lang.Object* содержит функцию *getClass()*, возвращающую метаобъект, отражающий класс объекта. Метаобъекты данного класса предоставляют доступ к информации о полях класса, его методах, конструкторах. Кроме того, доступна информация о иерархии наследования.
- **java.lang.reflect.Method** – класс метаобъектов, отражающих методы класса. Экземпляры данного класса содержат информацию о имени метода, типов параметров, тип возвращаемого значения и выбрасываемые исключения. Кроме того, существует возможность вызова метода.
- **java.lang.reflect.Field** – класс метаобъектов, отражающих поля класса. Экземпляры данного класса содержат информацию о имени поля, класса (типа) поля и модификаторах доступа.
- **java.lang.reflect.Constructor** – класс метаобъектов, отражающих конструкторы класса. Экземпляры данного класса позволяют создавать объекты класса с помощью их конструкторов, требующих передачу параметров. Так же, как для методов и полях такие метаобъекты содержат сведения о модификаторах доступа.
- **java.lang.reflect.Proxy** – инструментальный класс, осуществляющий реализацию интерфейсов динамическим созданием класса, реализующего набор заданных интерфейсов. Данный класс не относится к метаобъектам: он не отражает

существующие структуры программы, однако он позволяет реализовать посредничество вызова метода.

Использование метаобъектов позволяет программе обрабатывать объекты, классы которых неизвестны разработчику, включая классы, разработанные после написания программы. Эта гибкость важна для таких применений как поверка утечек памяти, сериализации и редакторов свойств, когда любой объект в системе является возможным кандидатом для обработки.

Кроме того, *Java Reflection* предоставляет следующие возможности изменения поведения программы:

Динамический вызов (Dynamic Invocation) – возможность, позволяющая программе вызвать метод объекта во время исполнения программы без указывания этого метода во время компиляции.

Динамическая загрузка (Dynamic Loading) – процесс, при котором виртуальная машина Java создает класс из байткода во время исполнения. Динамическая загрузка позволяет Java приложению находить и использовать классы, недоступные во время написания приложения.

Отраженное построение (Reflective construction) – возможность, позволяющая программе инициализировать экземпляр класса, неизвестного во время компиляции программы.

Посредничество (Intercession) – возможность модифицирования поведения программы напрямую управляя ее поведением. ***Посредничество вызова метода*** – возможность перехватить вызов метода;

Очевидно, возможностей, предоставляемых Java Reflection более чем достаточно для реализации *AppRunner*, не связанной с конкретными модулями. Например, с помощью **динамической загрузки**, возможно загрузить байткод классов модулей (например, по сети), а затем с использованием **отраженного построения** инициализировать объекты. Другим вариантом, апробированным автором, является использование параметров запуска приложения: в качестве параметров, при запуске AppRunner передаются наименования классов – реализаций модулей. Во время исполнения программы эти параметры обрабатываются и с помощью **отраженного построения** создаются экземпляры классов-реализаций модулей, которые затем преобразовываются к типу *(Module)*. Ниже представлен исходный код такого решения:

```
public static void main(String[] args) throws ClassNotFoundException,
IllegalAccessException, InstantiationException {
    MainWindow mw = new MainWindow();
    for(int i = 0; i<args.length;i++) {
      String moduleName = args[i];
      Class moduleClass = Class.forName(moduleName);
      if(isModule(moduleClass)) {
        mw.enableModule((Module) moduleClass.newInstance());
      }
    }
  }
```

Цикл *for* обрабатывает массив параметров запуска приложения: для каждого параметра, вызывается фунция ***Class.forName,*** которая производит формирование класса метаданных, описывающая класс, заданный строкой. Затем, производится **отраженное построение** объекта с помощью функции **newInstance().** После этого, объект преобразуется к общему типу *Module* и передается для дальнейшей обработки.

Представленная архитектура является достаточно гибкой и масштабируемой. Она не привязана к количеству модулей, а функциональность по управлению модулями является легко расширяемой. Например, при большом количестве модулей и необходимости во взаимодействии между ними, возможно в классе *Module* реализовать указание зависимостей. Тогда в *AppRunner*, целесообразно проверять такие зависимости, и при требовании для работы модуля наличия другого модуля – прозрачно загружать его из сети или любым другим способом. Компонент *AppRunner* так же расширяем: например, в нем возможно реализовать проверку прав доступа пользователя к определенным модулям, и при отсутствии таких прав – не загружать и не инициализировать модуль.

В заключение стоит отметить, что использование паттернов проектирования в совокупности с расширенными возможностями, предоставляемыми современными языками программирования позволяет реализовать эффективные, гибкие решения. Представленный подход может быть использован в качестве основы для разработки модульного приложения. Причем, использование паттерна «фасад», позволяет сформировать единый интерфейс для расширения системы путем разработки новых модулей. Использование *Java Reflection* позволяет абстрагировать код от конкретных модулей: построенная с использованием данного подхода система сможет корректно работать с модулями, исходный код которых недоступен во время компиляции программы.

3. Методология разработки частной информационной модели

Частная информационная модель эксплуатируемого строительного объекта является ядром системы: именно с данными модели оперируют все модули системы. Для сохранения состояния модели используется СУБД. Таким образом, необходимым является разработка схемы базы данных, в которой будут отражаться все сущности модели, а также данные, характеризующие их состояние.

Использование технологий, представленных в главе 2 (UML, ORM) позволяет автоматизировать построение схемы и выполнить эту работу одновременно с проектированием. В рамках реализации системы использован подход, состоящий из трех этапов:

1. Проектирование информационных сущностей системы. Этот этап выполняется с использованием унифицированного языка моделирования (UML). Разработанные схемы являются одним из компонентов проектной документации разрабатываемой системы. С точки зрения UML разрабатываются диаграммы классов. Осуществление данного этапа позволяет принять допустимый уровень абстракции и сформировать базовую иерархию классов сущностей системы.

2. Разработка классов, описывающих сущности системы, на целевом языке программирования. Для реализации системы выбран язык программирования *Java.* В рамках данного этапа выполняется непосредственно кодирование классов. Разработанные классы являются реализацией частной информационной модели в рамках системы. При разработке классов учитывается совместимость со стандартом EJB 3.1 (разрабатываются классы, являющиеся по стандарту «*entity bean»).*
3. Выполняется конфигурирование ORM инструмента *Hibernate.* Настраивается подключение к реляционной базе данных, формируется список классов-сущностей, которые необходимо использовать для построения схемы базы данных. После этого, в автоматизированном режиме инструментом *Hibernte* выполняется построение схемы данных, необходимой для хранения состояния всех сущностей системы. Кроме того, создаются необходимые таблицы для хранения связей между сущностями, таких как «многие ко многим».

Предлагаемый подход при разработке системы показал высокую эффективность. По сравнению с классическим проектированием схемы базы данных (как при использовании структурного подхода) имеются следующие преимущества:

- Достигается более высокий уровень абстракции от конкретной реализации и производителя системы управления базой данных. (Система не является зависимой от конкретной базы данных, её выбор ограничивается лишь списком поддерживаемых конкретной реализацией ORM);
- Достигается более высокая скорость разработки: нет необходимости в поддержке актуальности версий схемы базы данных и набора моделей. Например, при добавлении поля в класс и последующей инициализации ORM инструмента, необходимый для хранения поля столбец таблицы, создастся автоматически;
- Достигается более четкое разделение уровней информационной системы. Использование ORM исключает реализацию логики на уровне БД, что в свою очередь положительно сказывается на переносимости системы и простоте поддержки.

В последующих разделах настоящей главы представлены диаграммы классов, отражающие как информационные сущности отдельных модулей (электронный каталог), так и сущности, реализующие частную информационную модель эксплуатируемого строительного объекта.

4. Реализация модуля электронного архива документации

В целях реализации инфраструктуры информационного пространства, содержащего полные сведения о строительном объекте, предлагается реализация модуля «Электронный каталог документации». Наличие данного модуля позволит интегрировать в систему документацию и данные о мероприятиях по контролю технического состояния, выполненных за годы эксплуатации объекта, предшествующие внедрению системы. Функциональность модуля «Электронный каталог документации» заключается в следующем:

- Формирование учетной карточки образа документа. (Предполагается выполнение оцифровки бумажной документации). Учетная карточка документа должна отражать его основные реквизиты для поиска и идентификации без просмотра образа.
- Классификация документации. Заключается в разработке набора справочников, содержащих значения типовых полей учетной карточки документа. Таким образом предотвращается избыточность данных и необходимость ручного ввода значений полей учетной карточки.
- Возможность добавления дополнительных полей к учетной карточки документа и классификации документов по их значениям. (Необходима для добавления характеристик документа и использованию их значений при поиске).
- Поиск документации: как по текстовому запросу, так и по сложным критериям, представляющим набор значений полей карточки документа.
- Возможность выборки массива документов для выгрузки из системы. (Необходима для передачи определенных видов документации в экспертные организации при проведении инженерных исследований технического состояния).
- Регистрация заносимых документов и присвоение уникальных в рамках системы регистрационных номеров.

Базовыми сущностями модуля являются классификатор и справочник. Под классификатором понимается пополняемый набор значений одного из полей учетной карточки образа документа. Справочник – более сложная структура, предоставляющая помимо пополняемого набора значений поля возможность построения иерархии путем предоставления возможности задания каталогов и помещения в них значений классификаторов. Стоит отметить, что значения и справочников, и классификаторов могут быть как простого типа (строка, число и т.п.), так и составного, и представлять собой сложные структуры данных, в том числе содержащие поля, так же являющиеся значениями других справочников и классификаторов. На рис. 5 представлена диаграмма классов, отражающая классификаторы модуля «Электронный каталог документации».

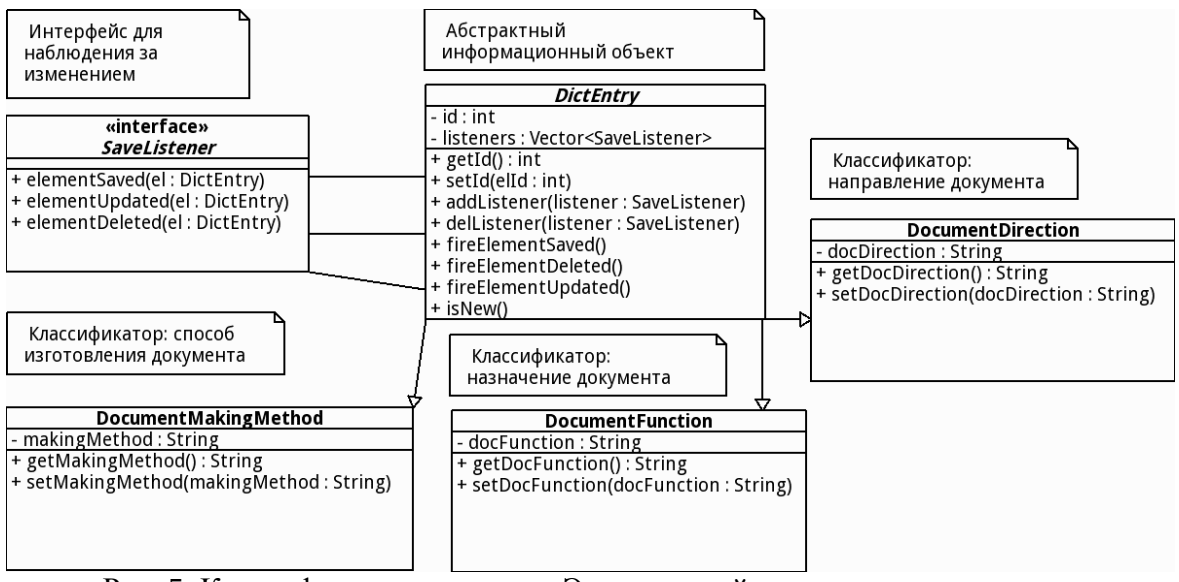


Рис. 5. Классификаторы модуля «Электронный каталог документации»

Предлагается использование следующих справочников: «Способ изготовления документа» (*DocumentMakingMethod), «*назначение документа» (*DocumentFunction)* и «направление документа» (*DocumentDirection)* (рис. 6).

Предложенные классификаторы отражают природу документа, его функциональное назначение, а также направление: получен ли документ от внешней организации, или является результатом внутренних процессов. Предлагаемые справочники предназначены для идентификации содержимого документа.

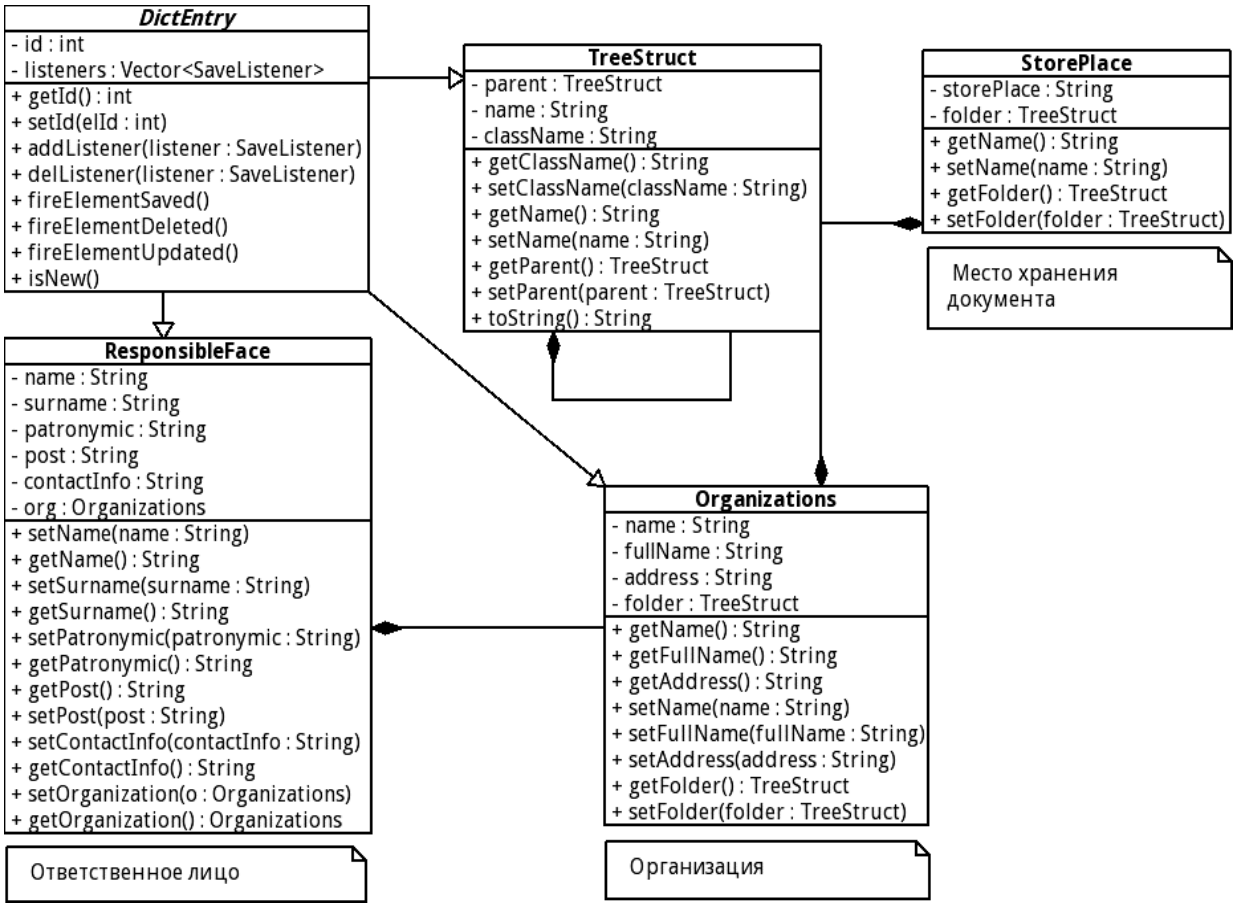


Рис. 6. Справочники модуля «Электронный каталог документации»

Класс *TreeStruct* является системным классом, реализующим иерархию и представление в справочники понятия каталог. Классы «Место хранения», «Ответственное лицо», «Организация» имеют связь с классом *TreeStruct,* таким образом, реализуя вложенность значений справочника в каталоги. Благодаря наличию поля *className* в классе *TreeStruct* иерархичные структуры различных справочников не пересекаются (в рамках одного справочника могут существовать только каталоги и вложенные в них значения). Кроме того, для классификации и характеристики образа документа предлагается справочник «Вид документа» (рис.7).

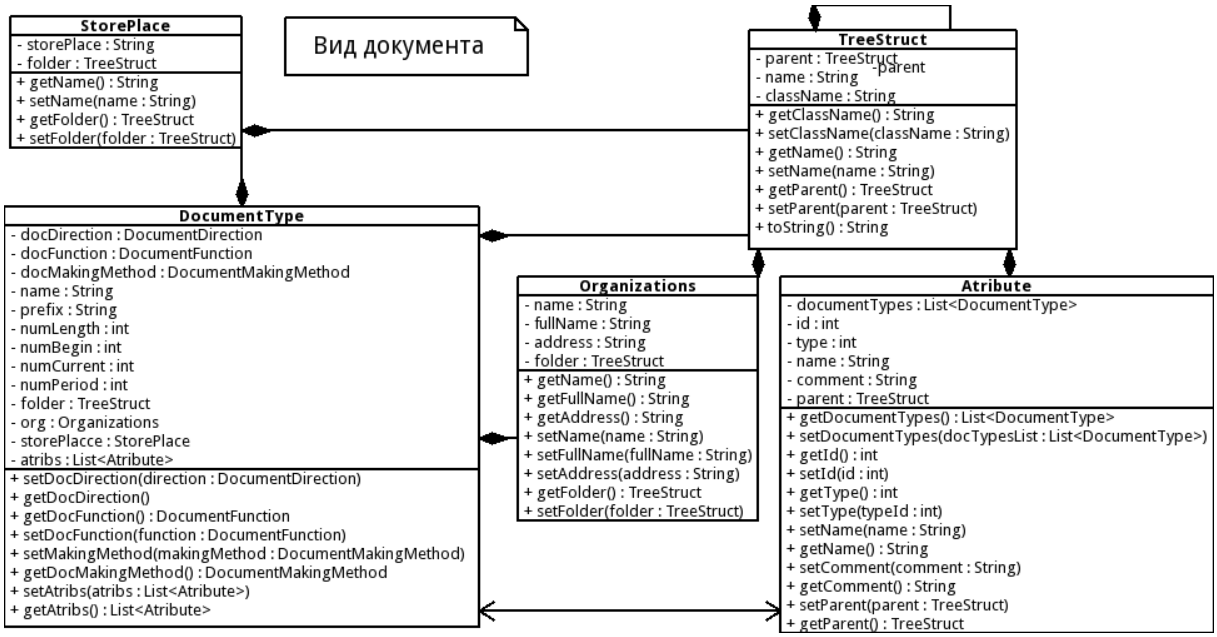


Рис. 7. Справочник «Вид документа»

«Вид документа» является одним из справочников, поля которого содержат ссылки на значение других справочников. Так, присутствует поле «Организация». Данный справочник предназначен для идентификации групп однотипных документов и предоставления возможностей по добавлению к карточке документа дополнительных полей классификации. Представленный на рис.4 класс *Atribute* предназначен для описания таких дополнительных полей. Кроме того, значение справочника «Вид документа» определяет формат формируемых регистрационных номеров, для добавляемых в систему документов данного вида. На рис. 8 представлена диаграмма классов, отражающая структуру учетной карточки документа в системе. Класс *Document,* описывающий информационную сущность «документ» связан со значениями справочников и классификаторов, кроме того содержит поля для непосредственного хранения значений характеристик документа: его наименования, номера, количества страниц, дате занесения документа в систему. Для связи учетной карточки документа с его оцифрованным графическим образом реализован класс *FileDocuments.* Данный класс содержит сведения об электронном образе документа: имени файла, размере и расположении.

Для реализации пользовательского интерфейса модуля «Электронный каталог» в работе предложен алгоритм динамической многоуровневой группировки, представленный в следующем разделе.

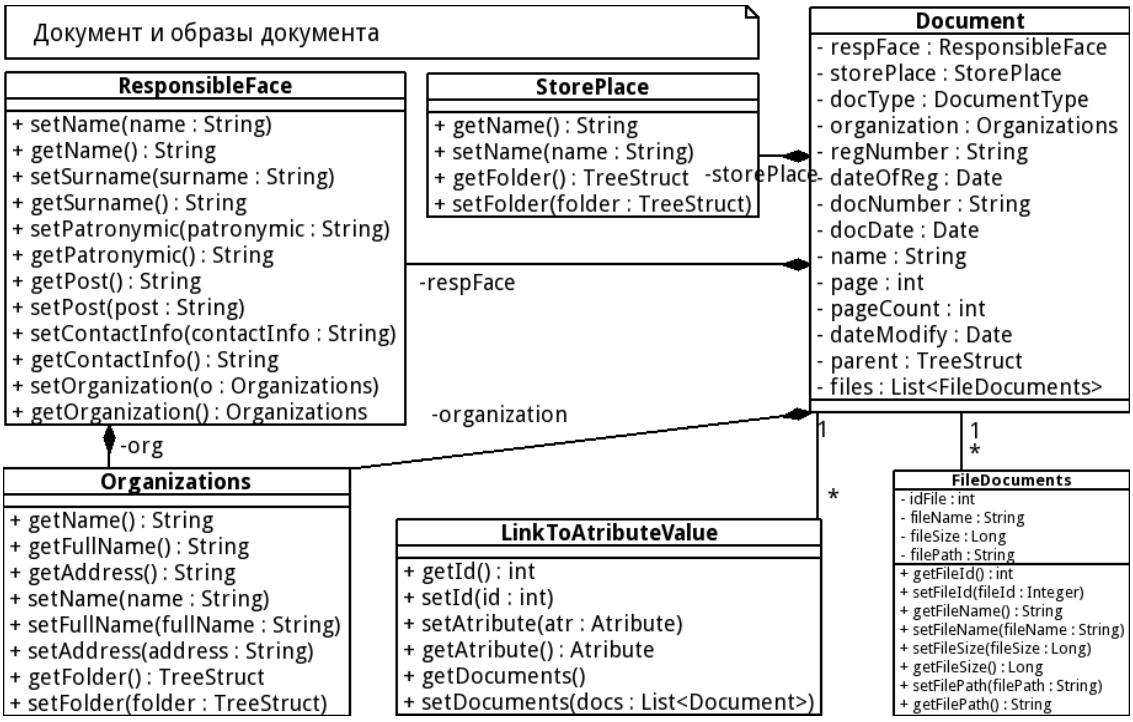


Рис. 8. Справочник документов модуля «Электронный каталог документации»

5. Адаптивная многоуровневая группировка данных в модуле «Электронный каталог»

Проблема отображения древовидных структур данных большого объема

Одной из проблем, выявленных при разработке информационной системы и модуля электронного каталога в частности, является реализация пользовательского интерфейса. Для систем, реализующих электронный архив характерен значительный объем информации. Такой объем усложняет техническую реализацию системы: в общем виде не представляется возможным отображать пользователю полное представление массива документов. Осложняет задачу и наличие иерархичных структур в массиве документов: наличие различных уровней иерархии не позволяет реализовать постраничное отображение данных. Технически, задача отображения иерархичных данных большого объема решаема за счет применения приема загрузки по требованию: при первоначальной инициализации загружаются только корневые элементы справочника. Загрузка дочерних производится непосредственно после перехода пользователя в конкретную ветку древовидной структуры. Однако, в общем виде при отсутствии ограничений на количество дочерних элементов узла дерева проблема вновь остается нерешенной. Для архивных систем для решения подобной задачи характерен следующий подход: для документов вводятся дополнительные классификаторы или тематические разделы. Пользователь находит и выбирает интересующий его тематический раздел в соответствующем справочнике тематических разделов, затем просматривает документы уже в табличном режиме с использованием постраничной загрузки данных. Как правило, такие системы имеют расширенные возможности поиска, результаты которого представляются так же в табличном виде.

Таким образом, пользователю не предоставляется возможность навигации непосредственно по всему массиву документов.

Динамическая иерархия

Модуль «Электронный каталог» изначально предоставлял возможности для хранения документов в иерархичной структуре. В предыдущих разделах настоящей главы показана техническая реализация такой структуры, предоставляющей возможность создания именованных каталогов, содержащих карточки документов. Правильность выбранного решения подтвердилась в ходе работ по формированию каталога документации для объекта «Первомайский хладокомбинат»: каталоги в системе отражали различные блоки переданной для занесения в систему информации. Такое решение позволило легко контролировать корректность введения информации в систему.

Последующая работа пользователей с каталогом и документацией показала, что работа с массивом документов и осуществление быстрой выборки необходимых при использовании статической иерархичной структуры размещения данных, не является эффективной. Более того, пользователями системы сформулированы варианты использования, при которых уровни иерархии меняются, в зависимости от потребностей пользователя при решении конкретной задачи. Например, рассмотрим 2 случая:

1. Иерархия должна иметь следующие уровни: «Строительный объект», «Организация», «Вид документации». Таким образом, на первом уровне иерархии находятся каталоги, в названиях которых перечислены конкретные строительные объекты, например, «Холодильный цех», «Рыбный цех». В каталоге, характеризующим конкретный цех, должны быть перечислены организации, связанные с процессами, связанными с данным цехом. В заключение, в каталоге с названием конкретной организации должны быть каталоги, содержащие различные виды документации: «Проектная документация», «Исполнительная документация».
2. Иерархия должна иметь уровни: «Строительный объект», «Вид документации», «Ответственное лицо». В этом случае, на первом уровне так же располагаются каталоги, отражающие строительные объекты, на втором – каталоги, отражающие различные виды документации. А в третьем уровне – каталоги, в названиях которых отражены инициалы конкретных ответственных лиц, связанных с документами.

Примечательно, что уровни иерархии могут быть произвольными, причем случаен как уровень иерархии, на котором находится характеристика документа, так и весь набор таких характеристик. Для реализации описанной функциональности системы, автором предложен алгоритм адаптивной динамической группировки, на основе которого реализован пользовательский интерфейс каталога документации. При этом, первоначальная возможность создания каталогов и хранения в них карточек документов так же остается в системе. Принято решение реализовать 2 режима работы с каталогом: режим ввода документации в систему, и режим работы пользователя. Функциональность режима пользователя и описание алгоритма динамической адаптивной группировки приведено ниже.

Ключевым моментом реализации представления массива документов, удовлетворяющим представленным выше условиям, является много вариантность иерархии. Иными словами, в момент разработки программы нет возможности определить ни уровни иерархии, ни их взаимное расположение. Кроме того, подход, при котором иерархичное представление является прямым отображением схемы данных из СУБД, так же неприменим, так как такая иерархия является статичной, и для изменения уровней иерархии необходима ручная (пусть и средствами самой программы) корректировка данных. Таким образом, иерархичное представление необходимо формировать во время выполнения программы. Для определения уровней иерархии и их взаимного расположения, необходимо предоставить пользовательский интерфейс, с использованием которого пользователь может сформировать необходимое в конкретный момент представление. Таким образом, понятие «каталог системы», будет отражать не объект системы, являющийся в конечном счете записью в таблице *TreeStructs,* а временный объект времени исполнения программы, который необходим для группировки записей по значению атрибута.

Алгоритм динамической многоуровневой группировки

Структура карточки и схема хранения дополнительных атрибутов документа позволяет осуществить группировку по произвольному атрибуту. Для этого, необходимо произвести выборку всех уникальных значений требуемого атрибута, создать объект времени исполнения, представляющий каталог группировки по каждому уникальному значению такой выборки. В заключение, в таком объекте нужно хранить либо непосредственно значение атрибута, либо ссылку на это значение (в случае дополнительных атрибутов). Таким образом, карточки документов, удовлетворяющие данному значению атрибута возможно получить выполнением SQL запроса в СУБД. Осуществляя последовательно такую группировку по различным атрибутам, будет сформировано требуемое иерархичное представление документов. Примечательно, что на этапе навигации по такой иерархичной структуре нет необходимости в загрузки самих карточек документов. Карточки документов целесообразно загружать на последнем уровне иерархии. При этом, путь в такой временной иерархии, позволит сформировать SQL запрос: так как на определенном уровне находится значение заданного атрибута и это значение известно, формирование запроса возможно осуществить так же во время исполнения программы.

Однако, набор дополнительных атрибутов документа у всех документов массива не является постоянным. Набор дополнительных реквизитов конкретного документа определяется его видом, и наличием связанных с документов значениями атрибутов. Таким образом, актуален вопрос о классификации документа по атрибуту, значение которого у данного документа отсутствует. В рамках реализации модуля «Электронный каталог» принято решение о создании специального объекта классификации «Не классифицировано», в который помещаются документы, не имеющие значения атрибута классификации на данном уровне. В случае, если данный уровень не является последним, классификация внутри такого каталога по другим атрибутам, осуществляется в обычном режиме.

Использование представленного подхода позволяет значительно сократить выборку документов при использовании нескольких критериев для группировки (на базе данных «Первомайского хладокомбината», оптимально от трех критериев). Разумеется, количество документов и количество самих критериев группировки определяются содержимым базы данных. Несмотря на это, даже в случае отсутствия в базе данных дополнительных атрибутов документа, возможно использование для группировки любых обязательных атрибутов документа (организация, вид документа, ответственное лицо, место хранения и др.).

Пользовательский интерфейс системы, разработанный на основе представленных выше положений, удовлетворяет запросом пользователя и позволяет выполнить в системе сформулированные пользователями варианты использования. Однако только лишь в описанном виде, решение не является защищенным от ситуации нехватки памяти. Например, при группировке всего массива документов по одному критерию, количество карточек в каждом каталоге будет велико: в одном каталоге может оказаться несколько тысяч карточек документов, что является слабым местом системы. В случае большого объема данных, программа может завершиться с ошибкой о нехватке памяти, или же будет иметь место значительное падение производительности и отзывчивости пользовательского интерфейса.

Для устранения этого недостатка принято решение осуществлять проверку количества карточек на конечном этапе группировки: в случае, если количество карточек велико (например, больше 1000), то необходимо осуществлять дополнительную группировку.

В качестве такой дополнительной группировки предлагается использовать группировку, основанную на первых буквах наименований карточек. Для этого выполняется запрос уникальных наборов первых n символов наименований карточек. Затем, осуществляется подсчет общего количества карточек в каждой подобной группе. Если максимальное количество карточек в группах является удовлетворительным с точки зрения производительности, значит число n выбрано корректно. Если же количество карточек остается велико, число n увеличивается. Таким образом, увеличивается количество групп и уменьшается общее количество карточек документов в них.

Реализация в рамках модуля «Электронный каталог»

Использование модуля «Электронный каталог» на различных этапах работы с документацией, показала целесообразность наличия в системе двух режимов работы для ввода и просмотра документов. При занесении документации целесообразно использование статичной иерархии, формируемой пользователем. При просмотре и навигации по массиву документов, эффективна адаптивная многоуровневая группировка, представленная выше. Для выбора режима работы на главном экране модуля размещены кнопки перехода в соответствующие режимы (рис. 9).

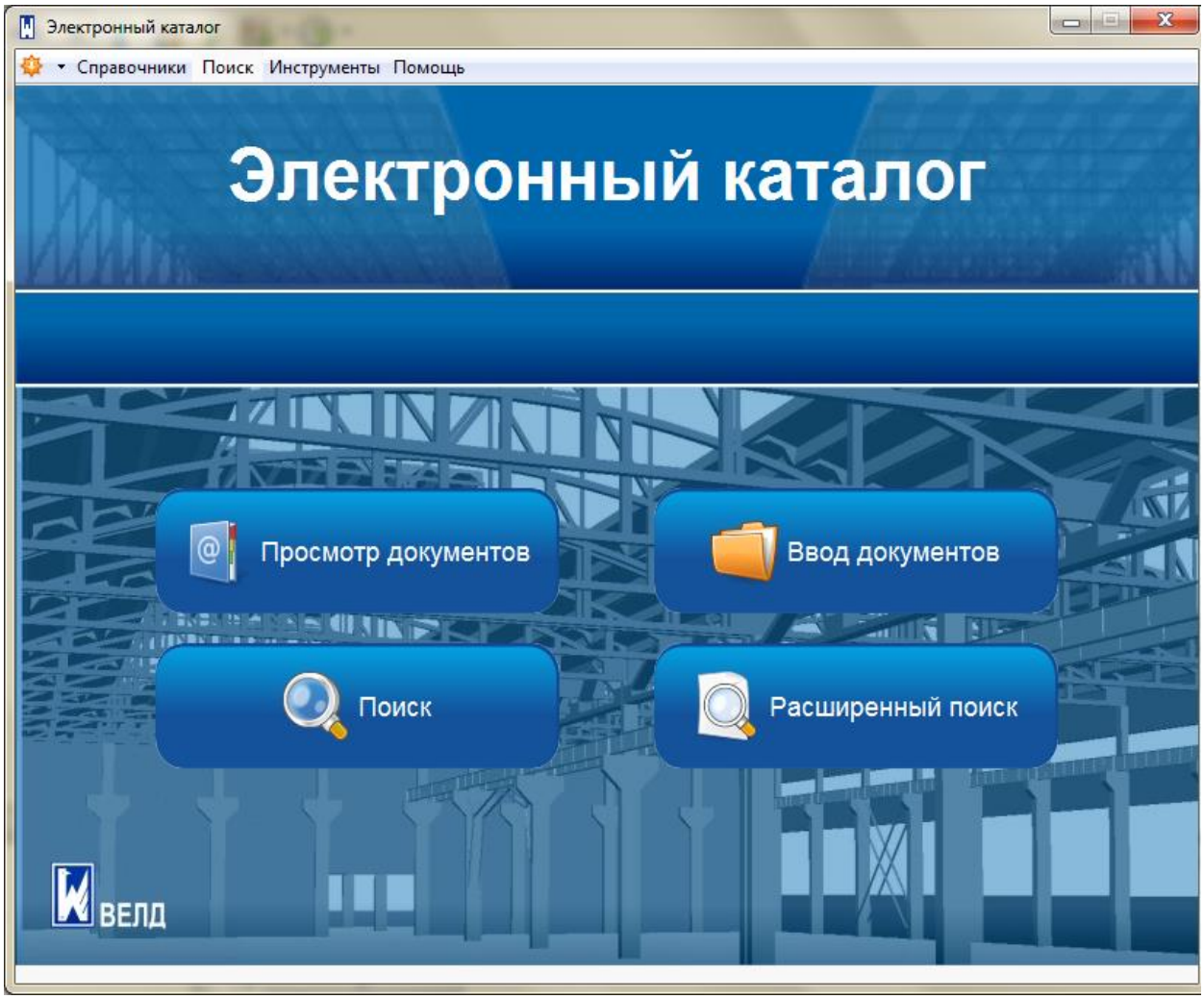


Рис. 9. Главный экран: выбор режима работы с каталогом

При выборе режима «Просмотр документов», открывается режим каталога, отражающий статичную древовидную структуру каталогов, сформированную при занесении документации в систему (рис. 10). В этом режиме содержимое окна составляют три области: статичная древовидная структура каталогов, блок просмотра значений атрибутов документа, блок просмотра образа документа. Для вызова команд работы с каталогами и документами предназначен набор кнопок на панели инструментов. Таким образом, пользовательский интерфейс режима «Ввод документов», основан на базовых компонентах операционной системы. Навигация по каталогам осуществляется привычным для пользователя способом, аналогично навигации по файловой системе в современных операционных системах. Благодаря этому, специальная подготовка пользователей системы для осуществления ввода документации в систему не требуется.

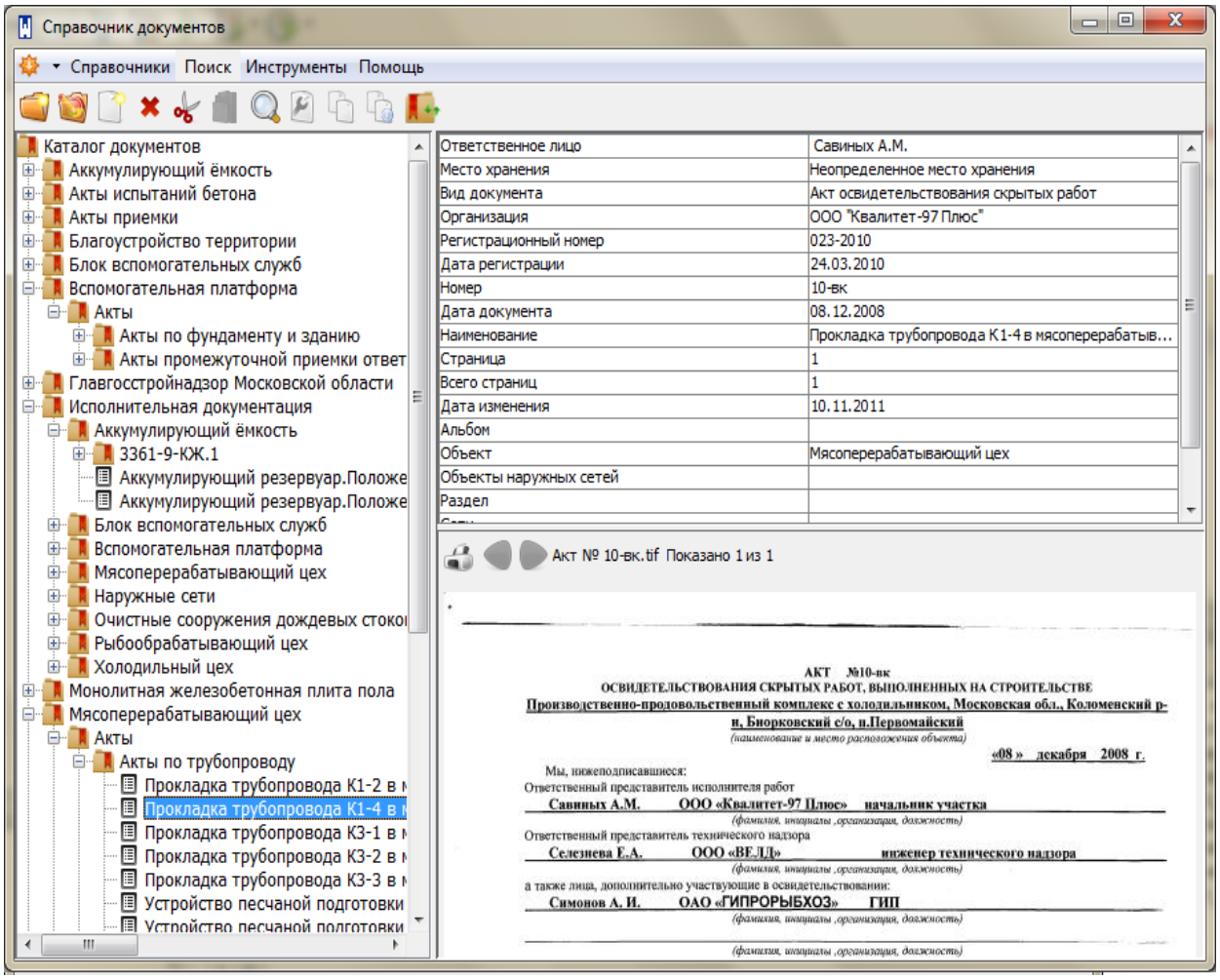


Рис. 10. Электронный каталог: режим ввода документов

Режим «Просмотр документов» напротив, требует более интерактивного взаимодействия с пользователем. Для формирования древовидной структуры требуемой пользователю, необходимо обеспечить выбор уровней иерархии пользователем. Для этого, в рамках модуля разработан компонент «Кнопка выбора уровней иерархии» (рис. 11).

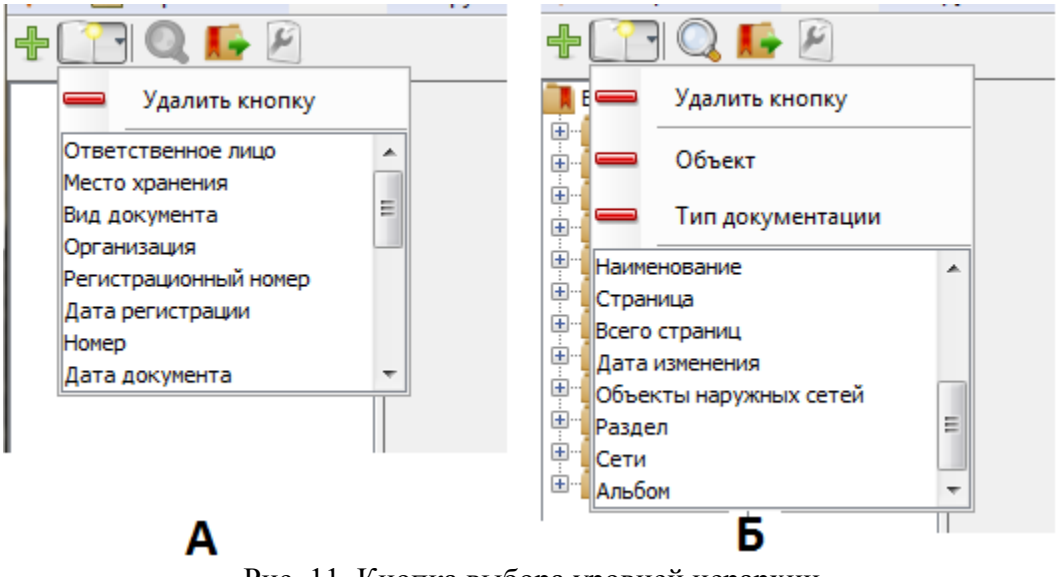


Рис. 11. Кнопка выбора уровней иерархии

Функционально, компонент представляет собой кнопку, при нажатии на которую появляется выпадающее меню с функциональными пунктами и список атрибутов документа. Такая кнопка появляется на панели инструментов, после нажатия кнопки «Добавить представление». Изначально, меню кнопки отражает только один пункт «Удалить кнопку». При нажатии на него, кнопка удаляется с панели инструментов. Одновременно на панели инструментов могут быть несколько таких кнопок, поэтому удаление кнопки необходимо, если при работе в системе пользователю более не нужна иерархия, сформированная в конкретной кнопке. Для формирования уровней иерархии, необходимо последовательно выбрать соответствующие атрибуты из списка. При выборе атрибута, в меню кнопки добавляется новый пункт, с названием атрибута. Для удаления атрибута из иерархии, необходимо нажать на соответствующий пункт меню. Иерархия выбранных атрибутов определяется последовательностью выбора: первый выбранный атрибут попадает на первый уровень иерархии, второй – на второй и т.д. Текущие уровни иерархии видны в меню: иерархичность определяется порядком следования пунктов меню с наименованиями атрибутов (сверху вниз). После формирования. После выбора уровней иерархии (критериев многоуровневой группировки) и нажатия непосредственно на кнопку, осуществляется построение требуемой иерархии и отображение её пользователю (рис. 12).

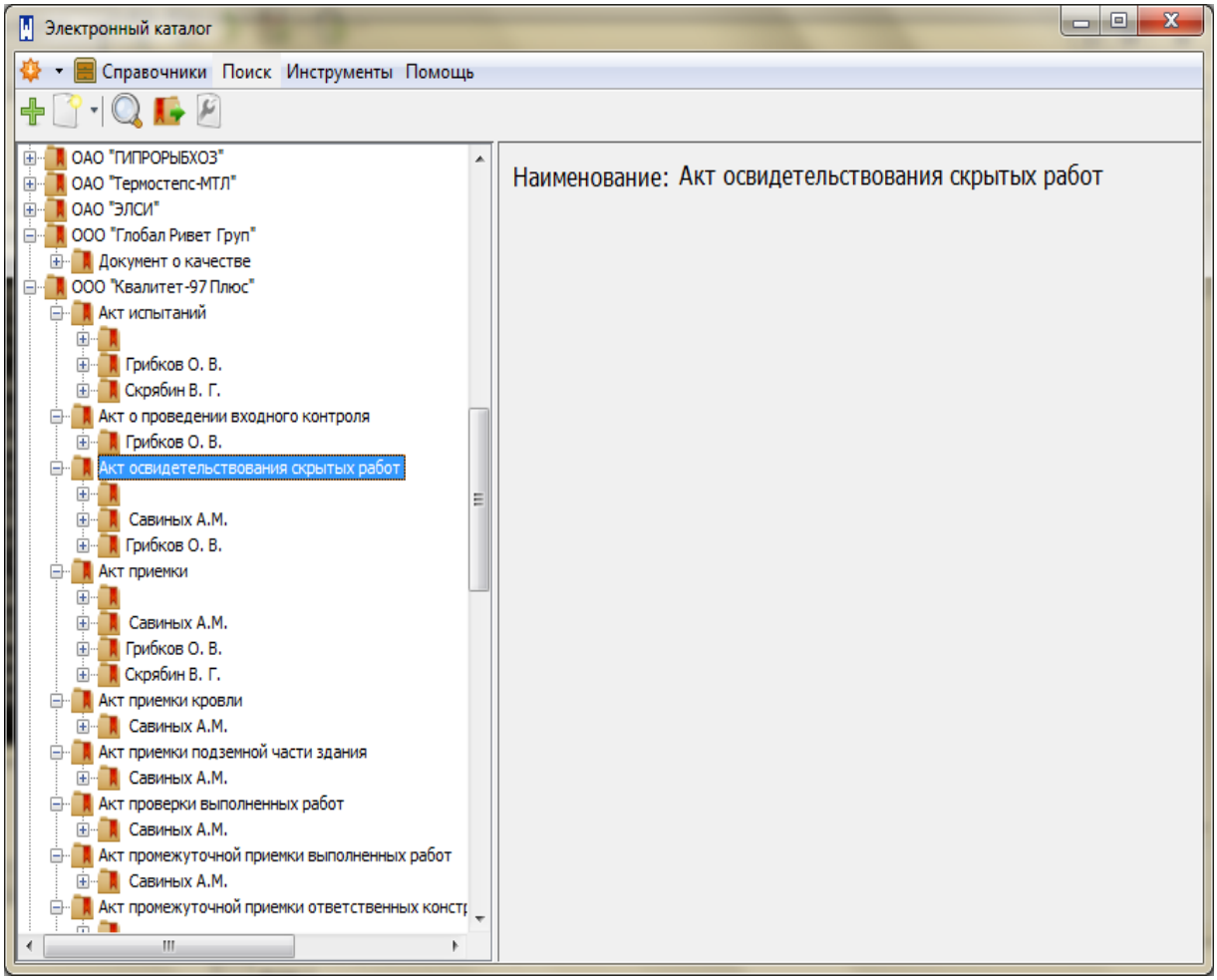


Рис. 12. Просмотр сформированного представления массива

Кроме того, путем добавления на панель инструментов нескольких кнопок и формирования различных наборов уровней иерархии в каждой, можно получить удобное переключение между различными представлениями.

Особенности технической реализации алгоритма динамической многоуровневой группировки в рамках модуля «Электронный каталог».

Техническую сложность реализации представленного алгоритма в модуле «Электронный каталог» составляет высокая степень информационной энтропии на этапе разработке программы. Пользователю системы предоставляется полная свобода в формировании дополнительных атрибутов документов, которые могут быть использованы в качестве критериев классификации. Кроме того, произвольно и количество уровней иерархии: пользователь может выбрать произвольное количество атрибутов из имеющихся. Ввиду вышеперечисленного запросы к базе данных необходимо формировать во время выполнения программы. Для реализации динамического формирования запросов рассмотрено несколько технологий:

- Использование специализированного языка запросов *HQL (Hibernate Query Language).* Данный диалект языка SQL специфицирован для *ORM* библиотеки *Hibernate.* Разработав программный код, генерирующий запросы *HQL* можно получить переносимый код. Такие запросы будут работать на любой СУБД, поддерживаемой *Hibernate.*
- Использование *JPQL (Java Persistence Query Language). JPQL* – диалект *SQL,* являющийся частью спецификации *Java Persistence Api.* Использование данного подхода, позволяет абстрагироваться не только от используемой СУБД, но и от библиотеки реализации *ORM.* Таким образом, принципиально возможным становится переход на другие средства *ORM.*
- Использование «чистого» *SQL* в диалекте используемой СУБД. Данный подход ограничивает гибкость программного кода: простой переход на другую СУБД или средство *ORM* становится невозможен без переписывания *Java* кода, генерирующего *SQL* запросы. Преимуществом данного подхода является возможность использования специфичных возможностей конкретной используемой СУБД.

Для реализации модуля «Электронный каталог» выбран путь использования «чистого» *SQL.* Выбор обусловлен несколькими факторами. Во-первых, данный подход является наименее ресурсоемким ввиду простоты тестирования и отладки кода. Во-вторых, использование встроенных функций конкретной СУБД значительно сокращает объем и уменьшает сложность генерируемого *SQL* кода. В заключение, в рамках диссертационного исследования предполагалось использование конкретной свободной СУБД – *Oracle Mysql.* Результатом принятого решения стала успешная реализация в короткие сроки и апробация алгоритма адаптивной многоуровневой группировки в режиме промышленной эксплуатации.

6. Базовые сущности частной информационной модели

Базовое ядро системы содержит модели строительных конструкций, параметров безопасности и процессов измерения их значений и собственно модель процесса контроля технического состояния. С точки зрения принадлежности к объекту эти модели ссылаются на абстрактный «Строительный объект», реализациями которого в конкретных случаях являются различные виды сооружений. Так, в рамках проекта реализованы модули для промышленного здания и высотного сооружения – дымовой трубы (рис. 13).

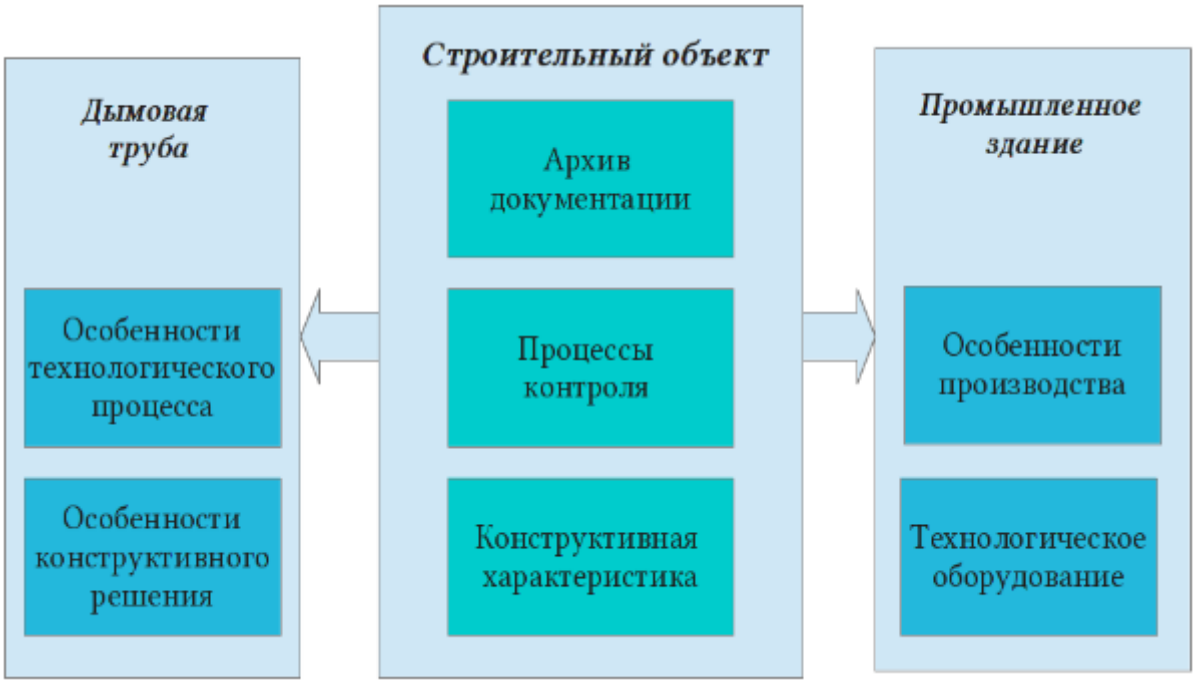


Рис. 13. Абстрактный строительный объект
и его конкретные реализации в системе

Наиболее важным с точки зрения безопасности является отражение в системе конструктивной характеристики строительного объекта. С одной стороны — важна информация о наборе типовых конструкций, использованных при строительстве. Однако при измерении параметров необходимо различать и конкретные конструкции (пусть и одного типа), расположенных в различных рядах/осях здания. Это обусловлено тем, что техническое состояние однотипных конструкций в пределах одного сооружения может различаться в силу следующих факторов: особенностей режима и условий эксплуатации, технологического процесса и повреждений, накопленных за период эксплуатации. На рис. 14 представлена диаграмма классов на языке UML, отражающая модель строительной конструкции в предлагаемой системе.

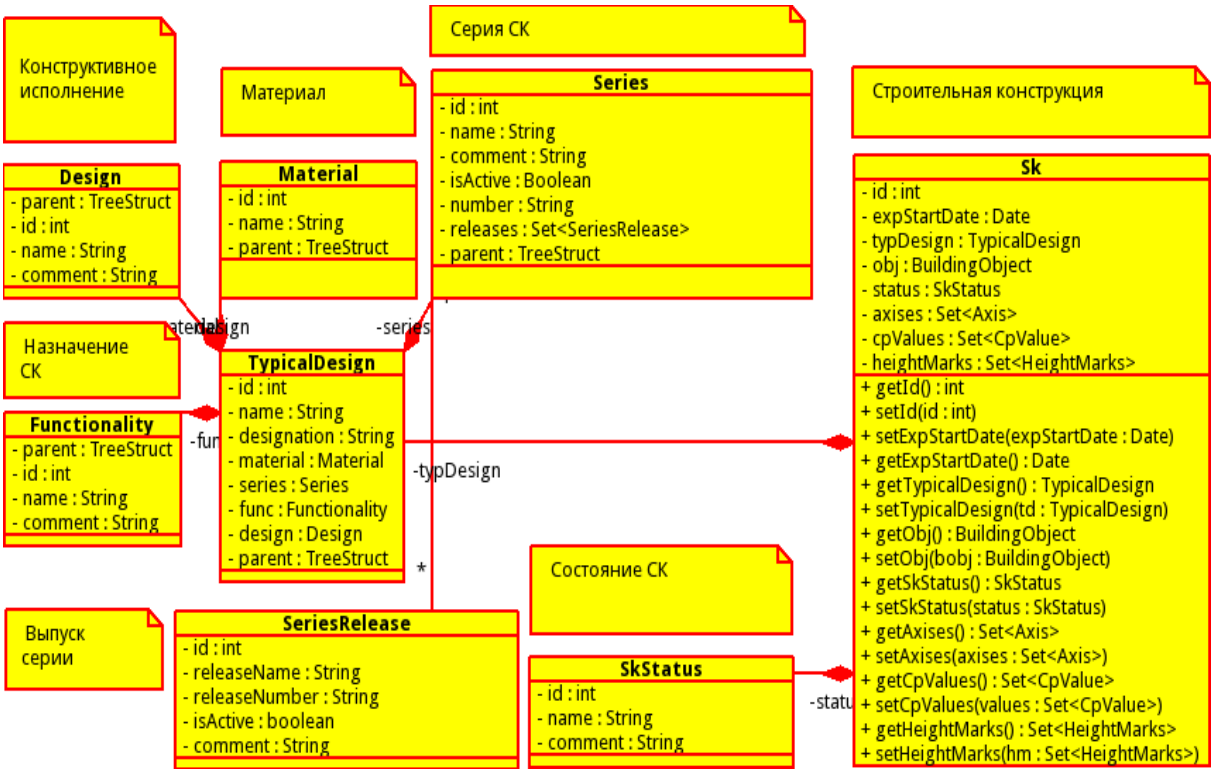


Рис. 14. Диаграмма классов на языке UML, отражающая модель строительной конструкции

Предложенная модель отражает информационную составляющую модели строительной конструкции, актуальной в контексте формируемой частной информационной модели. Так, для классификации строительных конструкций введены модели «Типовая строительная конструкция» («Typical design»), «Материал» (Material), «Конструктивное исполнение» (Design), «Функциональное назначение конструкции» (Functionality). Введение соответствующих уровней классификации необходимо, для предоставления в системе расширенных возможностей по формированию отчетов и анализу технического состояния. Например, благодаря предложенному проектному решению, в конечной системе возможно реализовать следующий вариант использования: «Получить состояние всех несущих железобетонных конструкций». Кроме того, согласно предложенной модели каждый строительный объект имеет ссылку на значение классификатора «Состояние строительной конструкции» (SkStatus), отражающее текущее состояние конструкции. Другим немаловажным аспектом модели является связь с документацией на типовые конструкции: объект «Типовая строительная конструкция» (Typical Design) содержит ссылки на справочники серий и выпусков серий типовых конструкций.

С точки зрения обеспечения безопасности эксплуатации, ключевыми данными конструкций являются параметры безопасности. Под параметром безопасности понимается значение качественного или количественного параметра конструкции, или узла сооружения (стык, шов и т.п.), изменение значения которого может повлиять на конструкционную безопасность сооружения. В рамках системы предлагается следующее проектное решение (рис. 15).

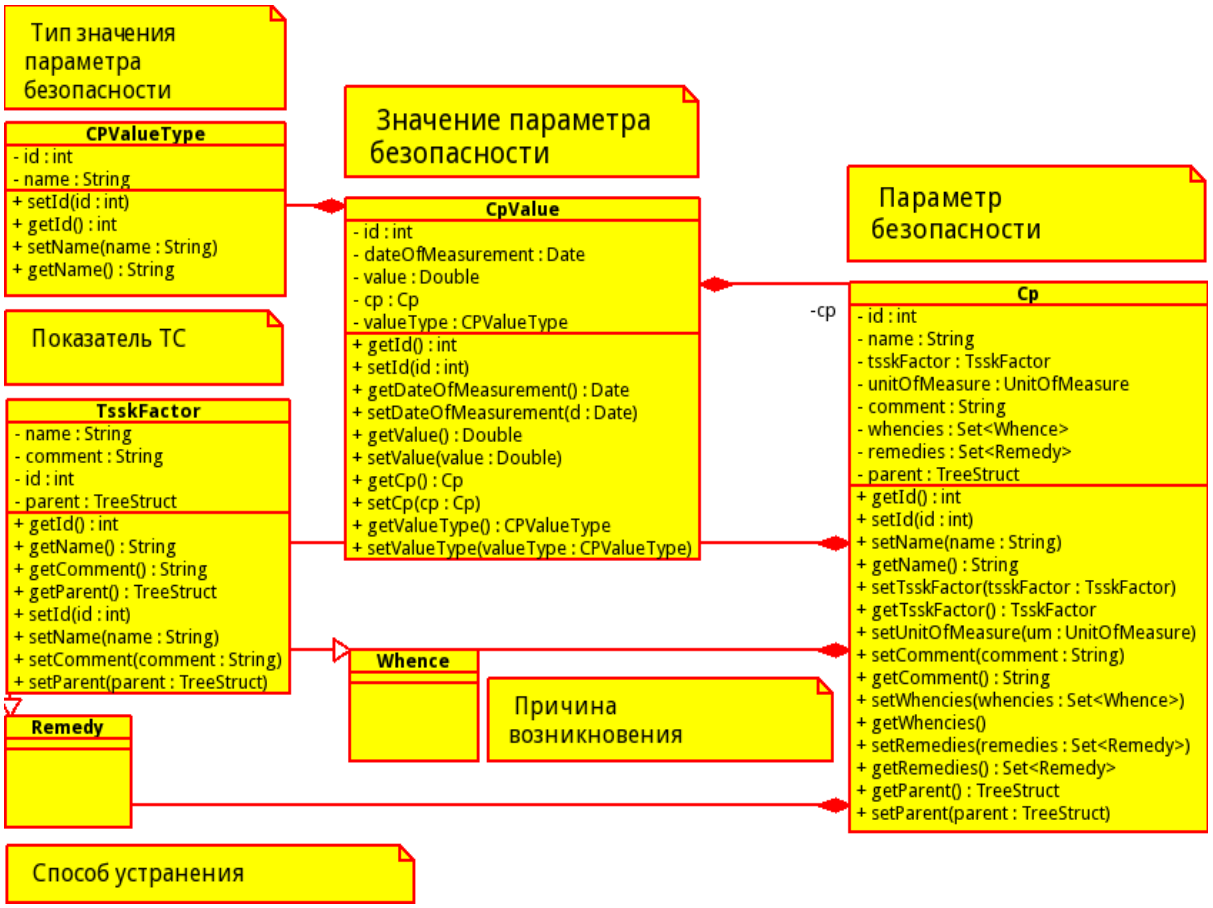


Рис. 15. Диаграмма классов на языке UML, отражающая модель параметра безопасности строительной конструкции

Сущность «Параметр безопасности» (Cp) введена для формирования справочного описания возможных типовых параметров безопасности, значения которых контролируются при исследовании технического состояния объекта. Эта сущность отражает наименование параметра, единицу измерения его значения, перечень возможных причин возникновения отклонения значения параметра и возможных мер по устранению отклонения. Кроме того, для классификации параметров безопасности, в соответствии с их природой (геометрический, физический параметры, неоднородности различного вида) сущность параметра безопасности агрегирована с сущностью «Показатель технического состояния» (TsskFactor), который детализирует описание параметра безопасности с учетом его вида. «Значение параметра безопасности» (CpValue) – особая сущность для хранения конкретного значения параметра. Для временной привязки значения предусмотрен атрибут, характеризующий дату значения. Кроме того, значения параметра безопасности классифицируются по значениям атрибута «Тип значения параметра» (CpValueType). Например, значения могут быть: фактические, проектные, расчетные или нормативные.

С точки зрения автоматизации процессов контроля параметров безопасности представляется целесообразным отражение этих процессов в информационной системе. С одной стороны, это позволит реализовать автоматизацию подготовки отчетной документации, с другой – позволит хранить в системе информацию о средствах измерения, исполнителях и др. Предлагается реализация сущностей: «Процесс

контроля технического состояния» и «Измерение параметра безопасности» (рис. 16). Кроме того, предлагается реализация сущности «Отклонение» (Defection), которая будет создаваться в системе автоматически, при выявлении недопустимого отклонения значения параметра безопасности.

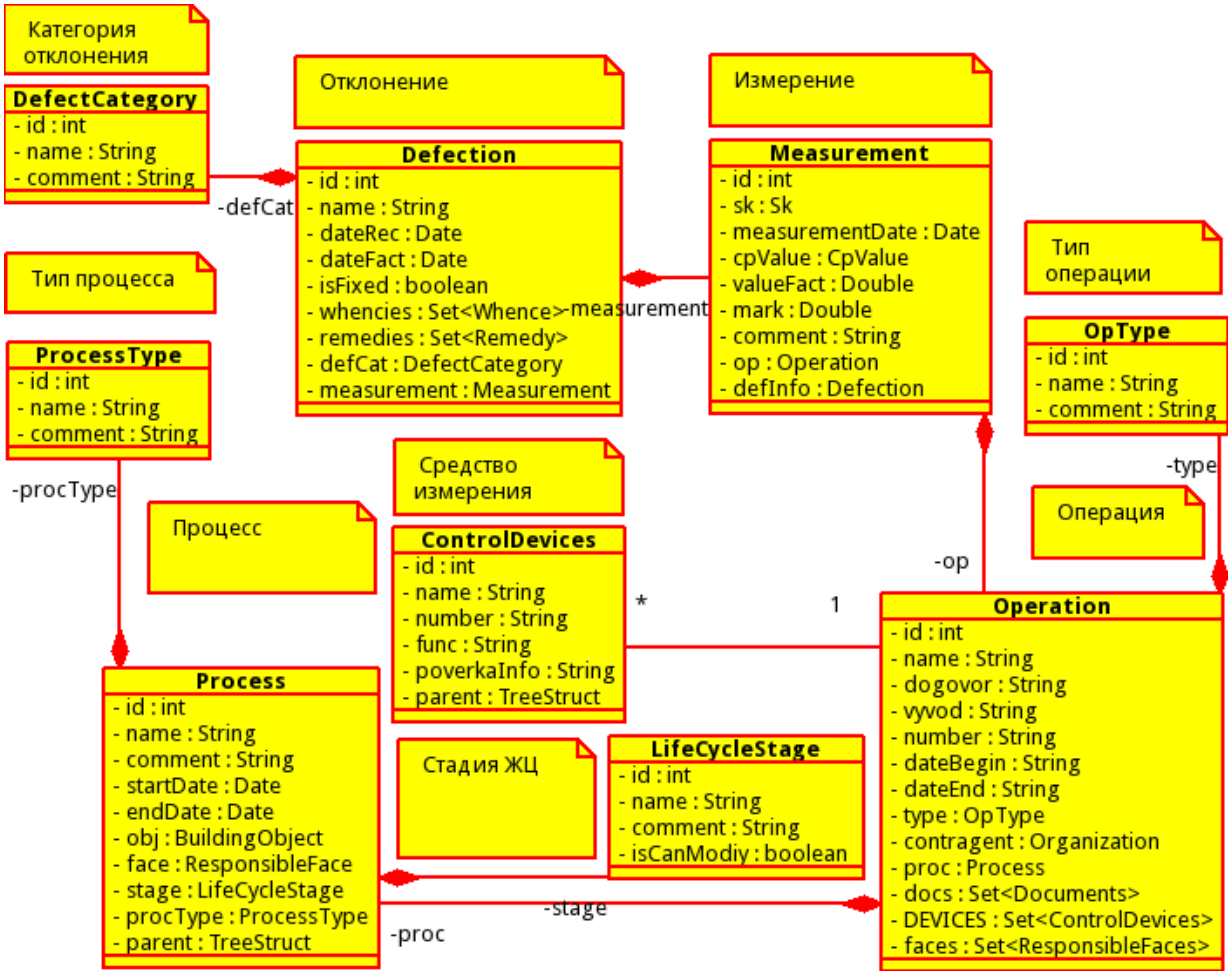


Рис. 16

Замечания

Представленные решения могут стать основой системы паспортизации. С точки зрения технологий CALS/ИПИ – представленные в разделе 3 проектные решения являются частной информационной моделью. В этой модели – любой строительный объект декомпозируется с точки зрения конструктивного исполнения и для конструкций – выносятся параметры и значения, значимые при контроле технического состояния. С одной стороны – такая модель может быть использована при разработке самостоятельных систем паспортизации (что и было сделано при разработке системы «Электронный паспорт» и «Электронный каталог»), так и при доработке существующих систем. В целом, благодаря абстрагированию от технологии разработки на данном этапе, возможно универсальное применение представленных решений: современные средства программной инженерии позволяют генерировать исходный код моделей, таблиц реляционных СУБД на основе представленных диаграмм на различных языках программирования.

Библиографический список

1. ГОСТ Р ИСО 9000-2008. Системы менеджмента качества. Основные положения и словарь [Текст]. – М.: Госстандарт России: Стандартинформ, 2008.
2. Коргин А.В. Научно-методологические основы и информационная технология автоматизации инженерных исследований при реконструкции сооружений [Текст]: дисс. д-ра техн. наук: 05.13.12 / Коргин А.В. - М., 2005. - 329 с.
3. Столяров А.Н. Информационная система экспертизы технического состояния зданий и сооружений [Текст]: дисс. канд. техн. наук: 05.13.12 / Столяров А.Н. – Нижний Новгород, 2004. - 166 с.
4. Постановление Правительства Российской Федерации от 2 июня 2001 г. № 436 «О создании и введении в действие федерального каталога продукции для федеральных государственных нужд» // СЗ РФ. 2001. № 24. Статья 2451.
5. Сорокин М.О., Хотеев С.Д. Свободные инструменты для организации работ по разработке информационных систем в строительстве / Молодежный научно-технический вестник. – М., №1, 2013 [Электронный ресурс] - Режим доступа: http://sntbul.bmstu.ru/doc/532852.html
6. ГОСТ Р 53778-2010. Здания и сооружения. Правила обследования и мониторинга технического состояния. [Текст] - введ. 25.03.2010. - М. : Федеральное агентство по техническому регулированию и метрологии : Стандартинформ, 2010. - 59 с.
7. Буч, Гради, Максимчук, Роберт А., Энгл, Майкл У., Янг, Бобби Дж., Коналлен, Джим, Хьюстон, Келли А. Объектно-ориентированный анализ и проектирование с примерами приложений: Пер. с англ. - М.: ООО «И.Д. Вильямс», 2008. - 720с.: ил. - Парал. тит. англ.
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. - СПб.: Питер, 2012. - 368 с.: ил.
9. Forman I., Forman N. Java Reflection in Action. Manning / Ira R. Forman, Nate Forman - Greenwich, 2005 - 297 p.

ОПЫТ ПРИМЕНЕНИЯ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ ЭЛЕКТРОННОГО ПАСПОРТА НА ПРОМЫШЛЕННЫХ ОБЪЕКТАХ

УДК 004.9

Сорокин М.О.
ФГБОУ ВПО «Московский государственный строительный университет», г.Москва
Алексеев О.Г.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Sorokin M.O.
Moscow State Construction University, Moscow
Alekseev O.G.
«WELD» company, Magnitogorsk, Chelyabinsk region

Аннотация
В работе описаны примеры применения информационных моделей электронного паспорта на трех существующих промышленных объектах.

Ключевые слова:
Электронный паспорт, Электронный паспорт, Информационная модель, Промышленный объект, Производственное здание, Промышленная труба, Производственный комплекс
Key words:
Electronic passport, Electronic passport, Information model, Industrial facility, Industrial building, Industrial pipe, Industrial complex

Проектная и исполнительная документация является ключевой информацией, необходимой для выполнения мероприятий, связанных с декларированием и контролем безопасности строительных объектов. Таким образом, оцифровка и долговременное хранение этой документацией является приоритетной задачей. Такая задача уже неоднократно решалась. Рассмотрим два случая.

Производственный комплекс «Первомайский хладокомбинат

Для решения вышеуказанной задачи при строительстве производственного комплекса «Первомайский хладокомбинат» был применен модуль «Электронный каталог», разработанной в рамках настоящей работы системы «Электронный паспорт». При этом сформирована база данных, состоящая из 5000 карточек, описывающих документы системы, массив оцифрованных документов составляет 16 гигабайт данных. В качестве СУБД в рамках внедрения использована база данных, распространяемая по свободной лицензии Oracle Mysql. Ниже представлены основные экраны системы.

На рис. 1 представлено стартовое окно системы, в режиме работы модуля «Электронный каталог». На данном окне представлены кнопки перехода в различные режимы работы системы:

- ***Ввод документов*** – переход в режим ввода документов. Режим предоставляет возможность формирования статичной иерархии размещения документации с помощью создания каталогов для группировки документов.
- ***Просмотр документов*** – переход в режим просмотра документов. Режим предоставляет разработаннаую в рамках работы возможность адаптивной многоуровневой группировки данных.
- ***Поиск*** – режим поиска документа в системе по наименованию.
- ***Расширенный поиск*** – режим поиска документа в системе по реквизитам документа и значениям дополнительных атрибутов.

Внешний вид пользовательского интерфейса модуля в режиме «Ввод документов» представлен на рис. 2.

Окно системы в этом режиме состоит из следующих областей: системное меню, панель инструментов, представление иерархичной структуры хранения, область просмотра атрибутов документа, область предварительного просмотра электронного образа документа. Занесение документа в систему осуществляется путем ввода данных в карточку документа, вызываемую соответствующей кнопкой на панели инструментов (рис. 3).

Карточка документа содержит поля для ввода значений реквизитов, загрузки в систему графических образов документа, интерфейс для прикрепления дополнительных атрибутов документа и ввода их значений. Кроме того, в карточке документа возможно добавление списка связанных документов.

Режим «Просмотр документа» (рис. 4) имеет схожую компоновку экрана с режимом «Ввод документа», однако предоставляет отличный набор функций.

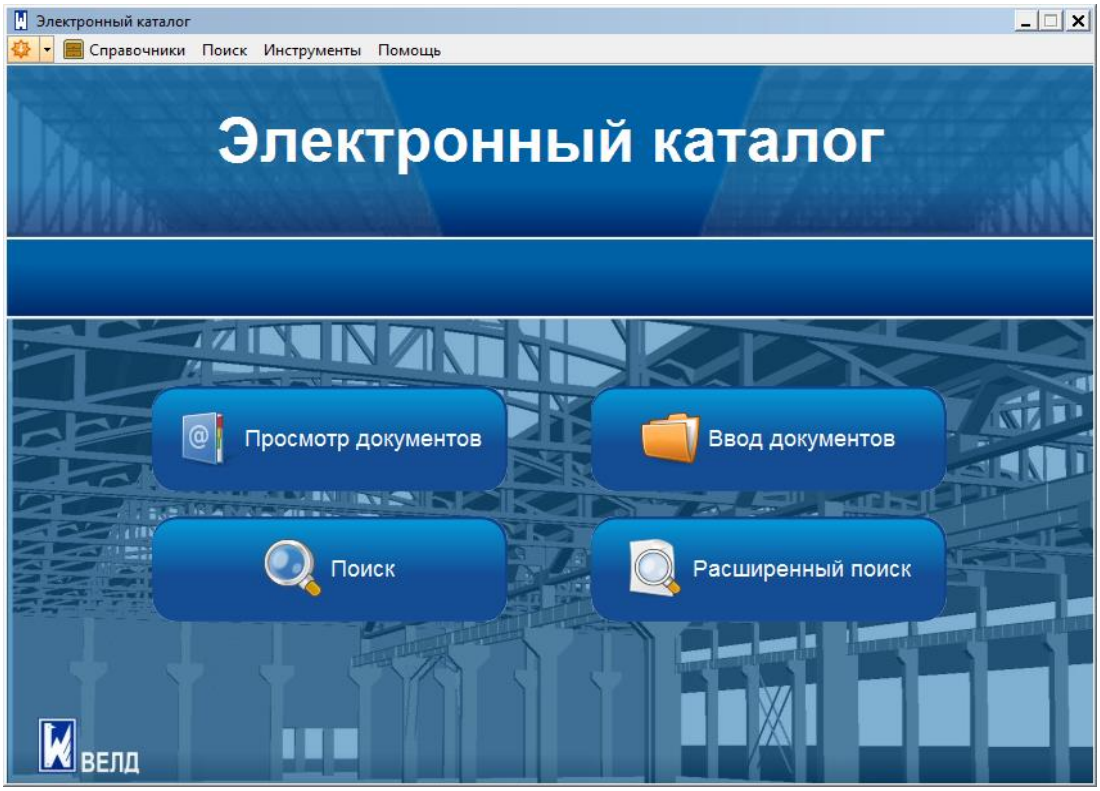


Рис. 1. Стартовое окно системы в режиме модуля «Электронный каталог»

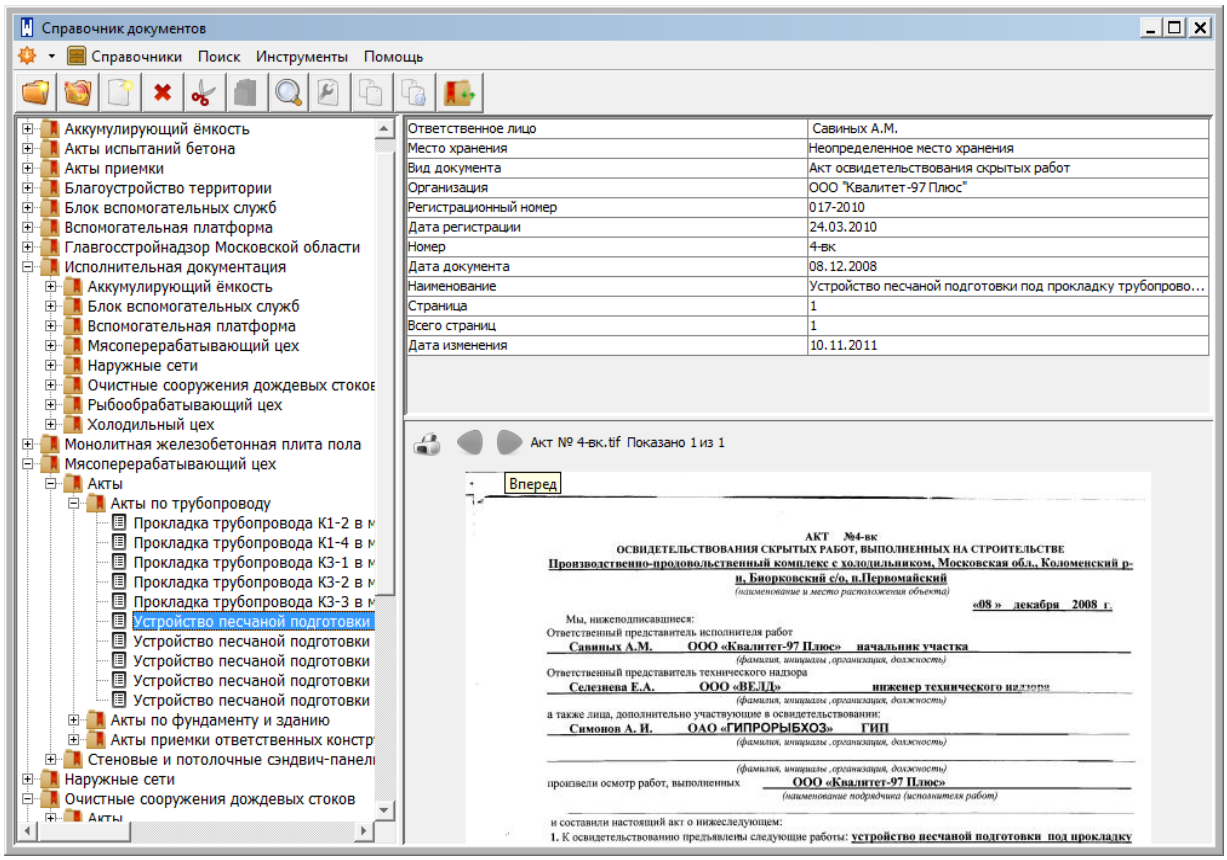


Рис. 2. Режим «Ввод документов» модуля «Электронный каталог»

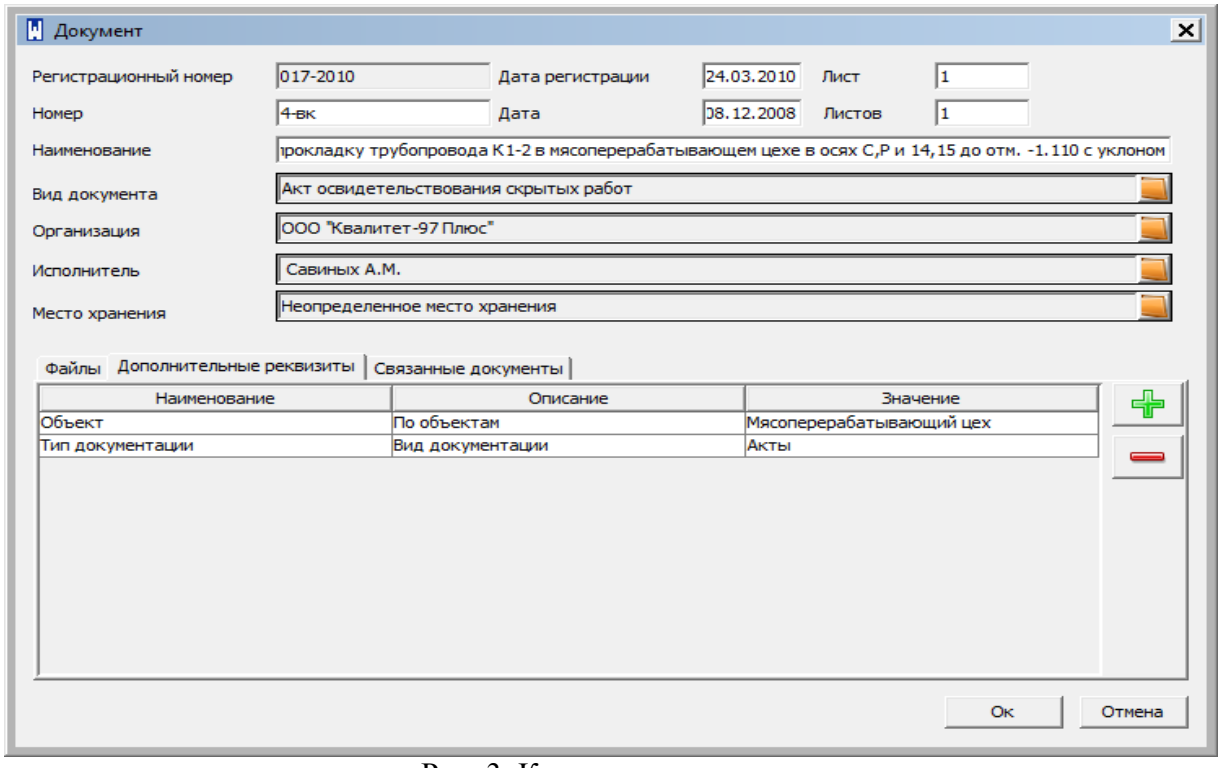


Рис. 3. Карточка документа

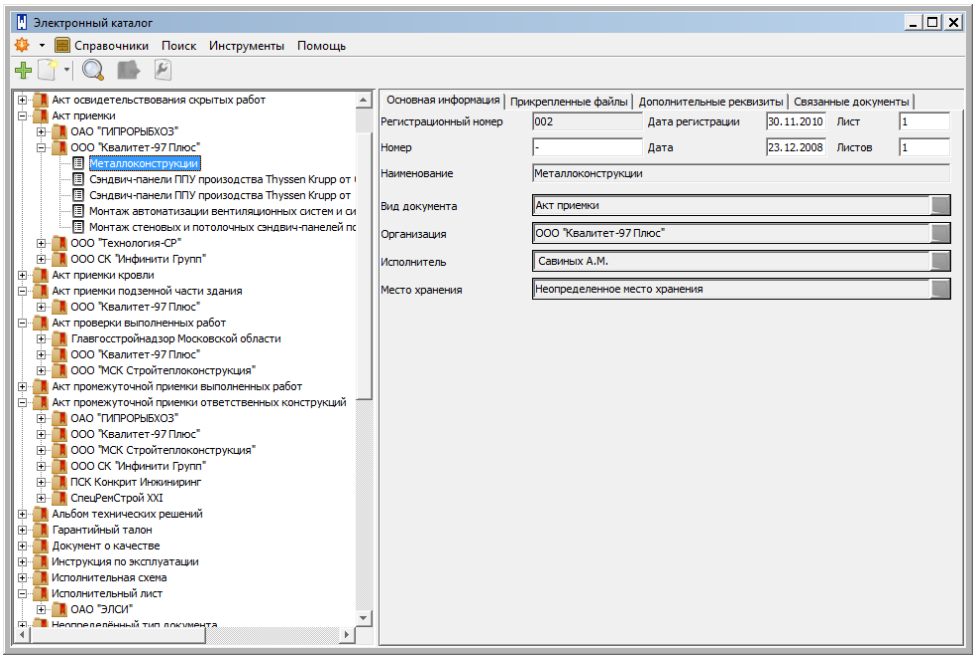


Рис. 4. Режим «Просмотр документов»

Так, на панели инструментов добавлена кнопка добавления конфигурации формирования многоуровневой адаптивной группировки. Ввиду того, что редактирование данных карточки документов не предполагается, карточка документа помещена в правую область окна. Таким образом, пользователю нет необходимости совершать дополнительные действия для открытия карточки документа для его просмотра: данные отображаются непосредственно после выбора документа в дереве.

Следующие два режима предназначены для поиска документации в системе. Так, режим «Поиск» позволяет искать документы в системе по строковому наименованию (поиск осуществляется по значениям атрибута «Наименование» в карточках документов). «Расширенный поиск» (рис. 5), позволяет сформировать более сложные поисковые запросы.

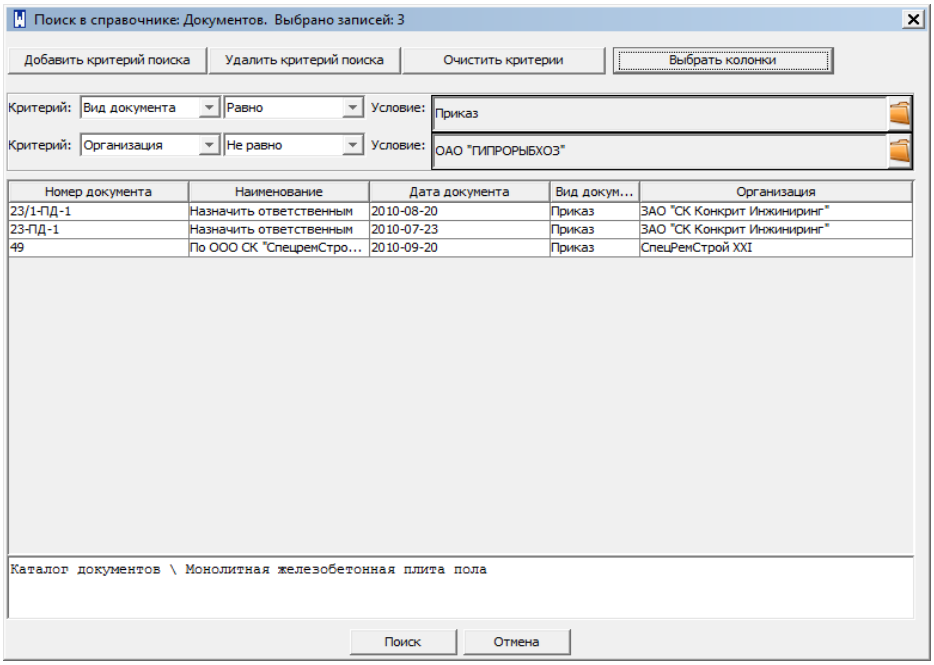


Рис. 5. Расширенный поиск

Предлагаемый пользовательский интерфейс позволяет добавить несколько поисковых условий. Каждое условие состоит из названия самого атрибута, условия и значения. Значение может быть как введено пользователем вручную, так и выбрано из соответствующего справочника. Кроме того, присутствует возможность выбрать столбцы таблицы в результатах поиска. При выборе записи в результатах поиска в нижней части окна отображается путь к документу в статичной структуре каталогов, сформированной при вводе документации в систему.

Модуль «Производственное здание»

Модуль «Производственное здание» информационной модели электронного паспорта апробирован на основании данных, полученных в результате экспертизы промышленной безопасности, выполненной для здания главного корпуса Тобольской ТЭЦ в г. Тобол силами разработчика продукта – компании «ВЕЛД». Основное внимание при выполнении работ уделено формированию частной информационной модели строительного объекта, наполнении справочника параметров безопасности и фиксировании в системе результатов измерений этих параметров.

Разработанная система позволяет хранить сведения для нескольких строительных объектов. В систему введен справочник строительных объектов, в котором непосредственно формируется информационная модель.

На рис. 6 представлен внешний вид справочника строительных объектов с одной записью: здание главного корпуса Тобольской ТЭЦ.

Введение справочника с возможностью формирования статичной иерархии путем создания каталогов требуется для возможности занесения множества строительных объектов в рамках единого предприятия или производственного комплекса. Непосредственно формирование частной информационной модели осуществляется через карточку промышленного здания (рис.7).

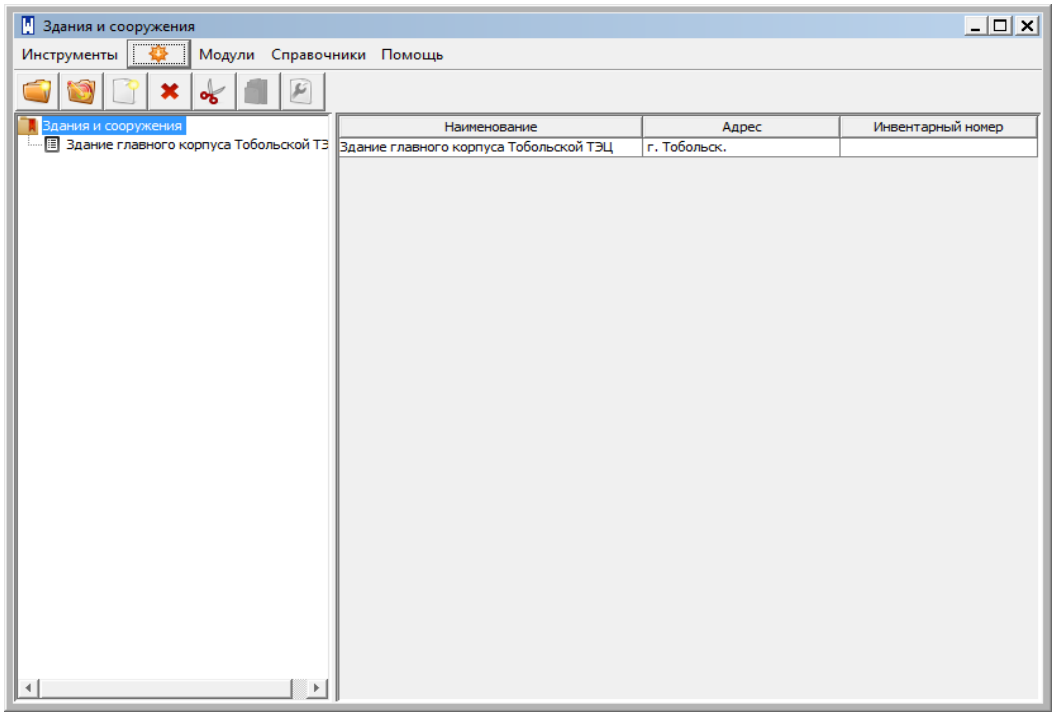


Рис. 6. Справочник строительных объектов

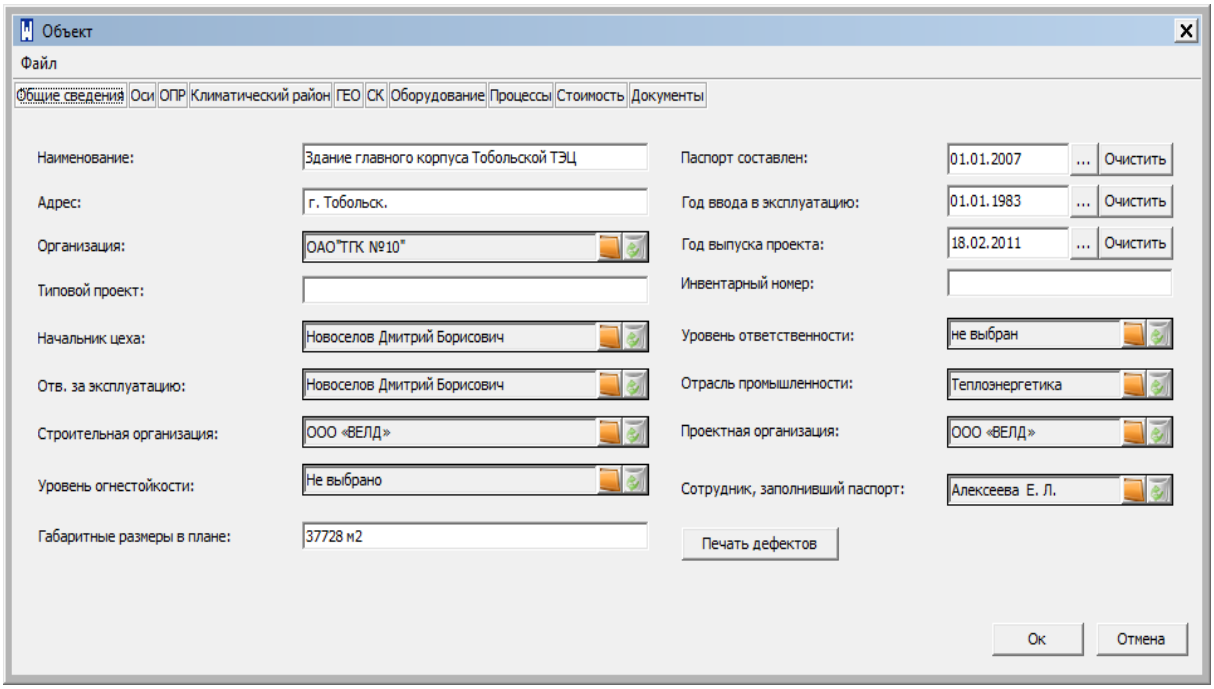


Рис. 7. Карточка промышленного здания

В карточке промышленного здания осуществляется ввод как паспортных данных объекта, включая ответственных лиц и контактную информацию службы эксплуатации, так и формирование конструктивной характеристики объекта. Кроме того, из карточки объекта осуществляется печать отчета «Дефекты и повреждения».

Конструктивная характеристика объекта и в частности формирование базовой информационной модели осуществляется на вкладках «Координационные оси» и «Строительные конструкции» (рис. 8).

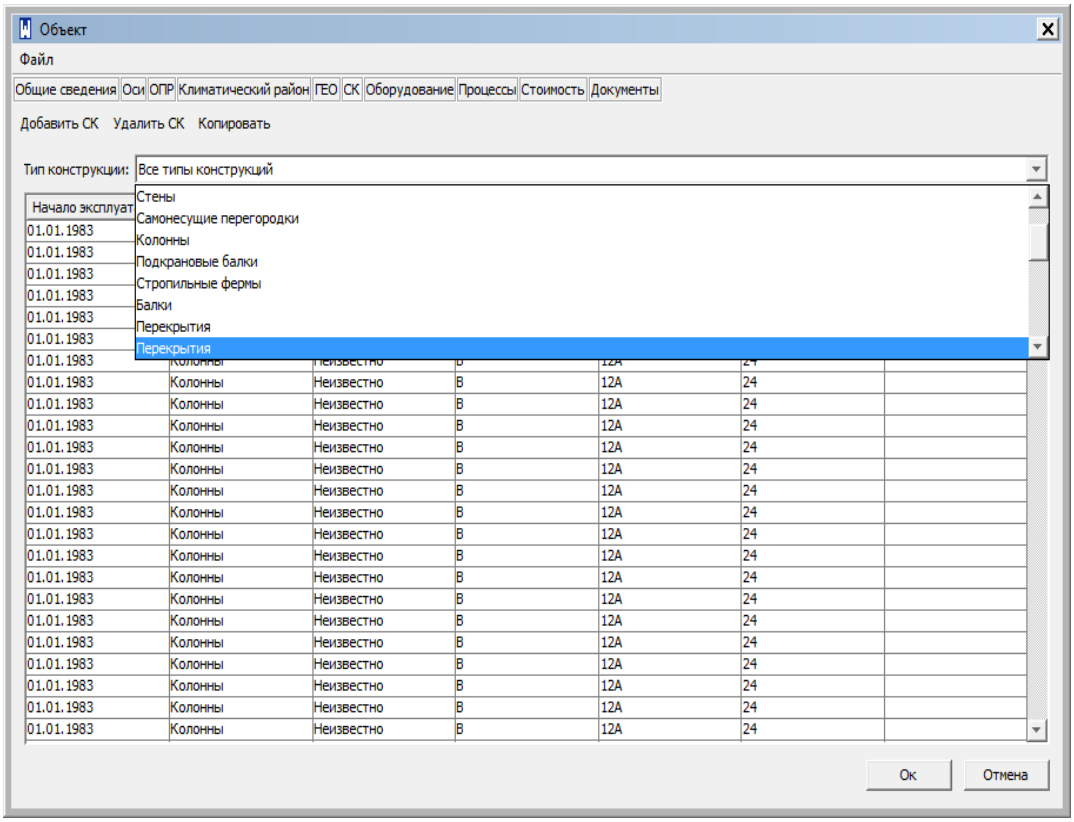


Рис. 8. Строительные конструкции

Для удобства первоначального ввода данных и работы с различными типами конструктивных элементов на данной вкладке реализована функция фильтра в таблице конструкций, а также функция копирования. Фильтр позволяет отобразить в таблице строительные конструкции одного типа (например, только колонны или балки). Функция копирования позволяет скопировать запись о строительной конструкции. Так, при первоначальном вводе возможно ввести одну конструкцию, скопировать её необходимое количество раз и затем для каждой конструкции проставить координационные оси.

Ключевой возможностью системы является учет параметров безопасности строительного объекта. Для описания параметров в системе введен общий для всех объектов справочник: «Контрольные параметры». В этом справочнике фиксируются измеряемые параметры как строительного объекта, так и окружающей среды (характеризующие климатический район и геологические условия площадки строительства) (рис. 9).

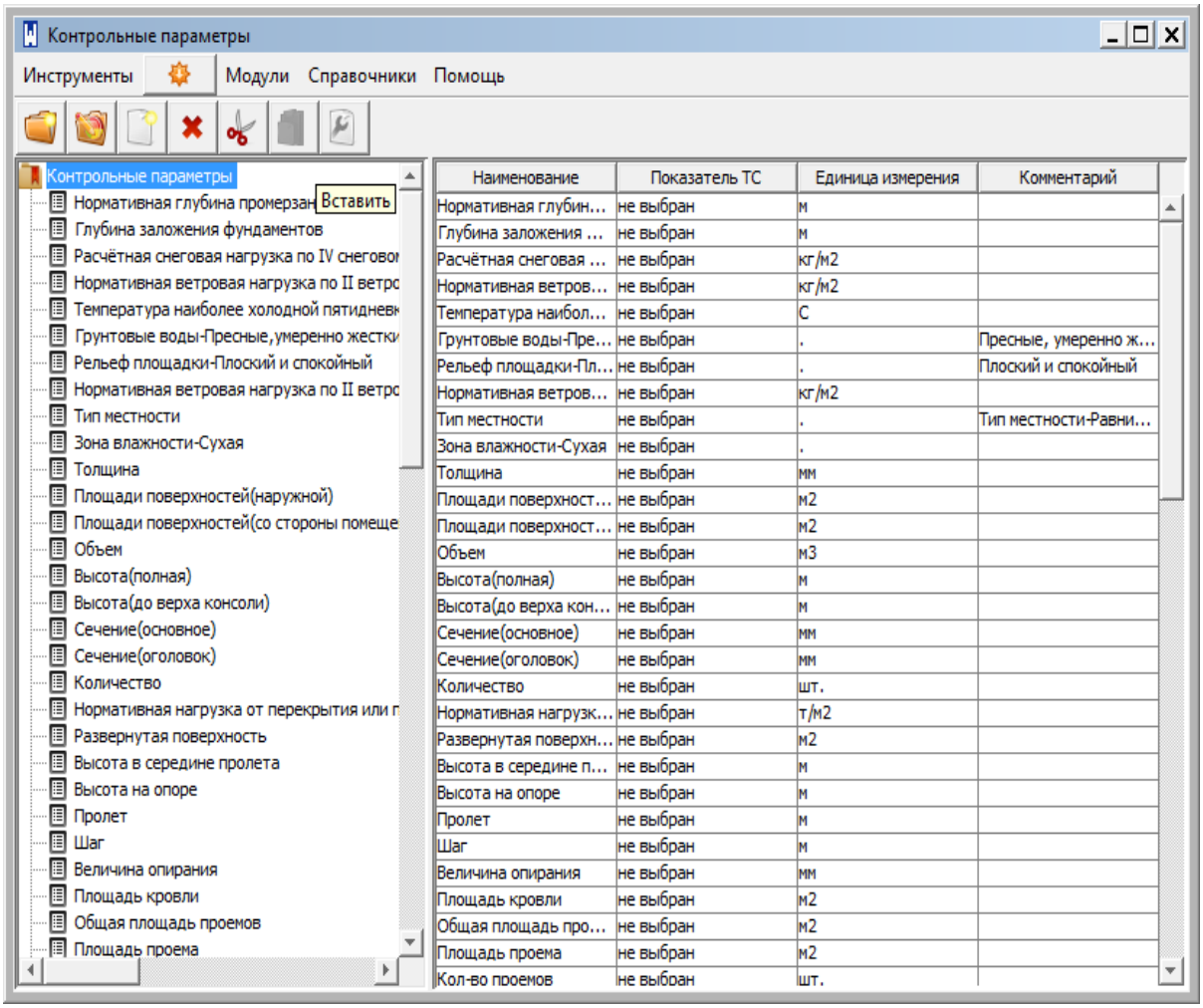


Рис. 9. Справочник контрольных параметров

Однако в справочнике параметр представлен только его описанием и подразумевает наличие единицы измерения величины значения, показатель технического состояния. Непосредственно значения фиксируются при выполнении

процесса контроля технического состояния в рамках конкретной операции. Эти процессы так же отражены в карточке объекта на соответствующей вкладке (рис. 10).

Процесс – информационная сущность системы, отражающая комплекс инженерных исследований, направленный на определение действительного состояния строительных конструкций и сооружения в целом. Отражает информацию о сроках проведения, ответственном лице и позволяет фиксировать дополнительную информацию, путем ввода текстовых данных в поле «комментарий» (рис. 11).

Операция – информационная сущность системы, отражающая конкретный вид однотипных операций в рамках процесса. Как правило, операция подразумевает сплошной или выборочный контроль определенных параметров безопасности для отдельной группы конструкций. Примером действительных операций может быть «Контроль пространственного положения конструкций» или «Контроль прочностных характеристик бетона несущих конструкций». Для ввода информации об операции в системе разработана карточка операции (рис. 12). В карточке предполагается ввод ссылок на нормативную документацию, регламентирующую выполнение измерений и расчетов, проводимых в рамках операции. Кроме того, предоставляется возможность для ввода сведений об используемых при выполнении работ средств контроля (оборудование и средства измерения). На основании зафиксированных данных и выполненных в рамках операции измерений выполняется формирование отчетов «Протокол измерений» и «Ведомость дефектов и повреждений». Ввод сведений об измерении происходит в карточке измерения (рис. 13).

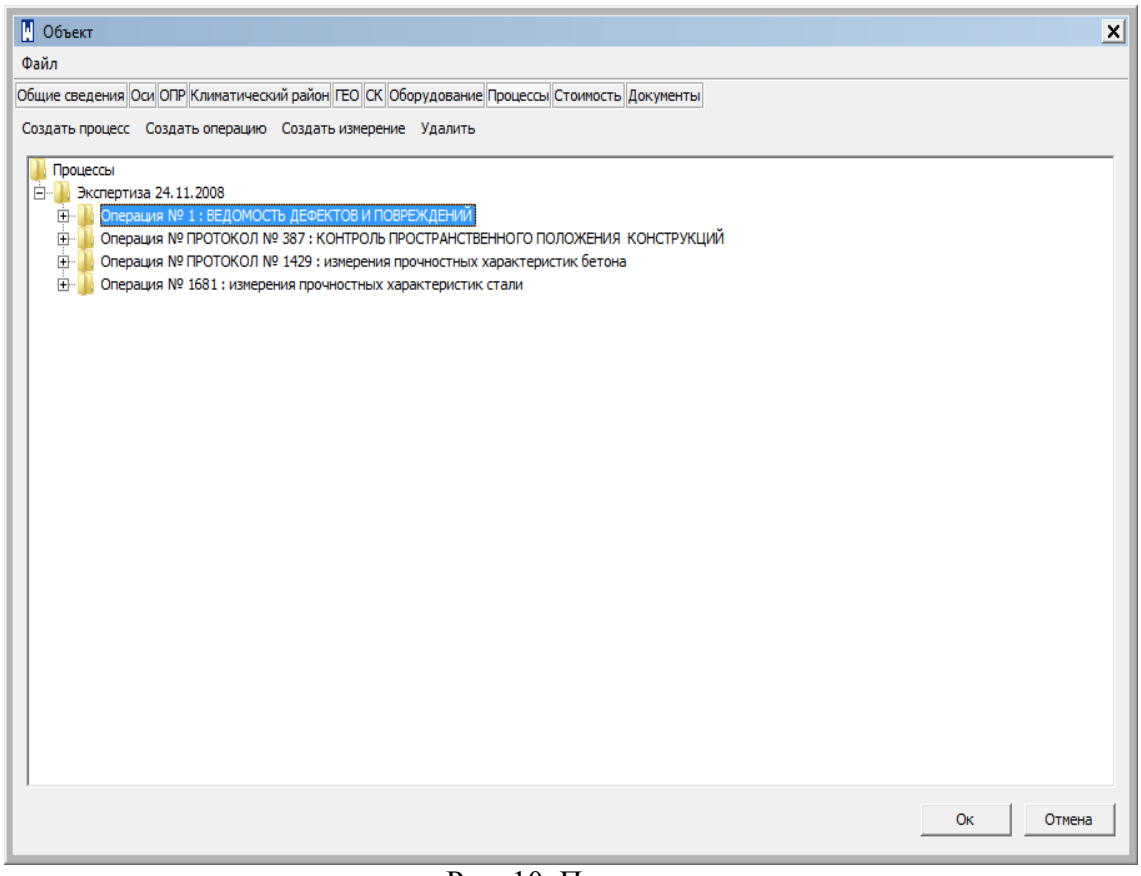


Рис. 10. Процессы

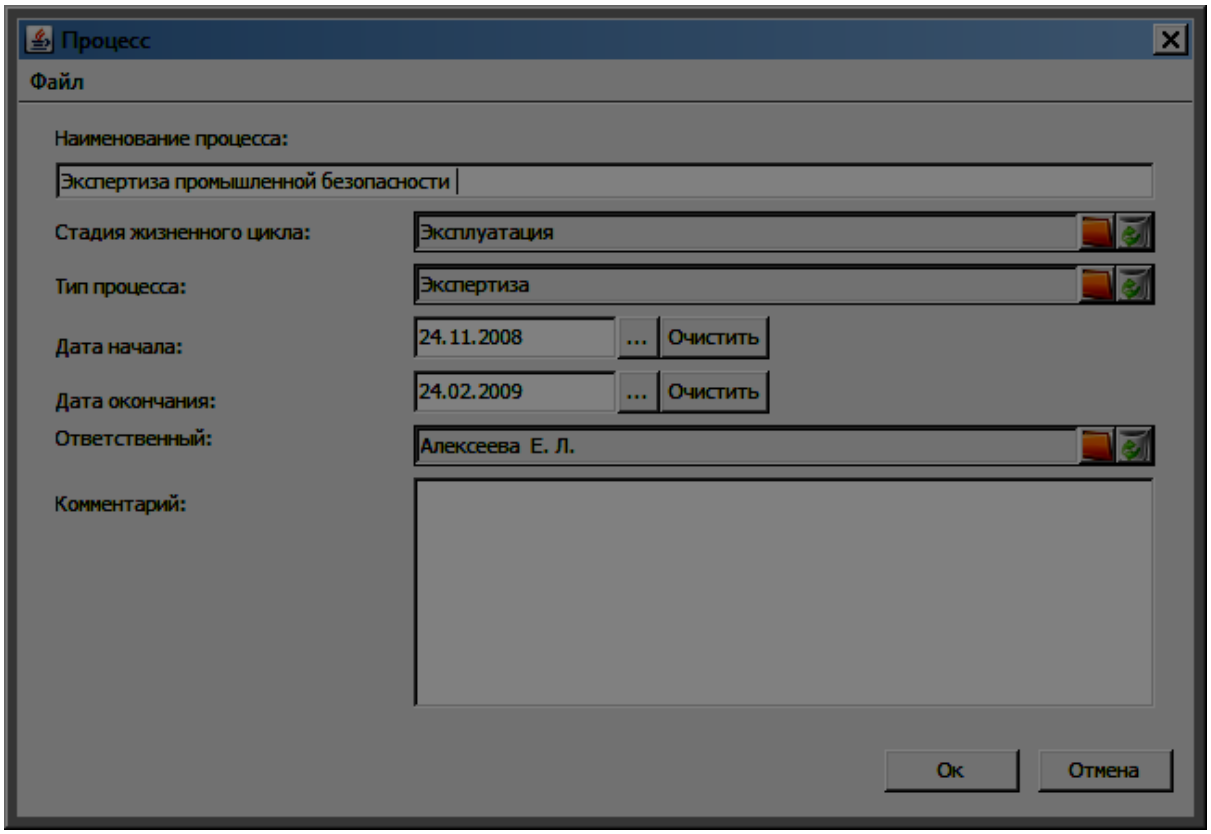


Рис. 11. Карточка процесса

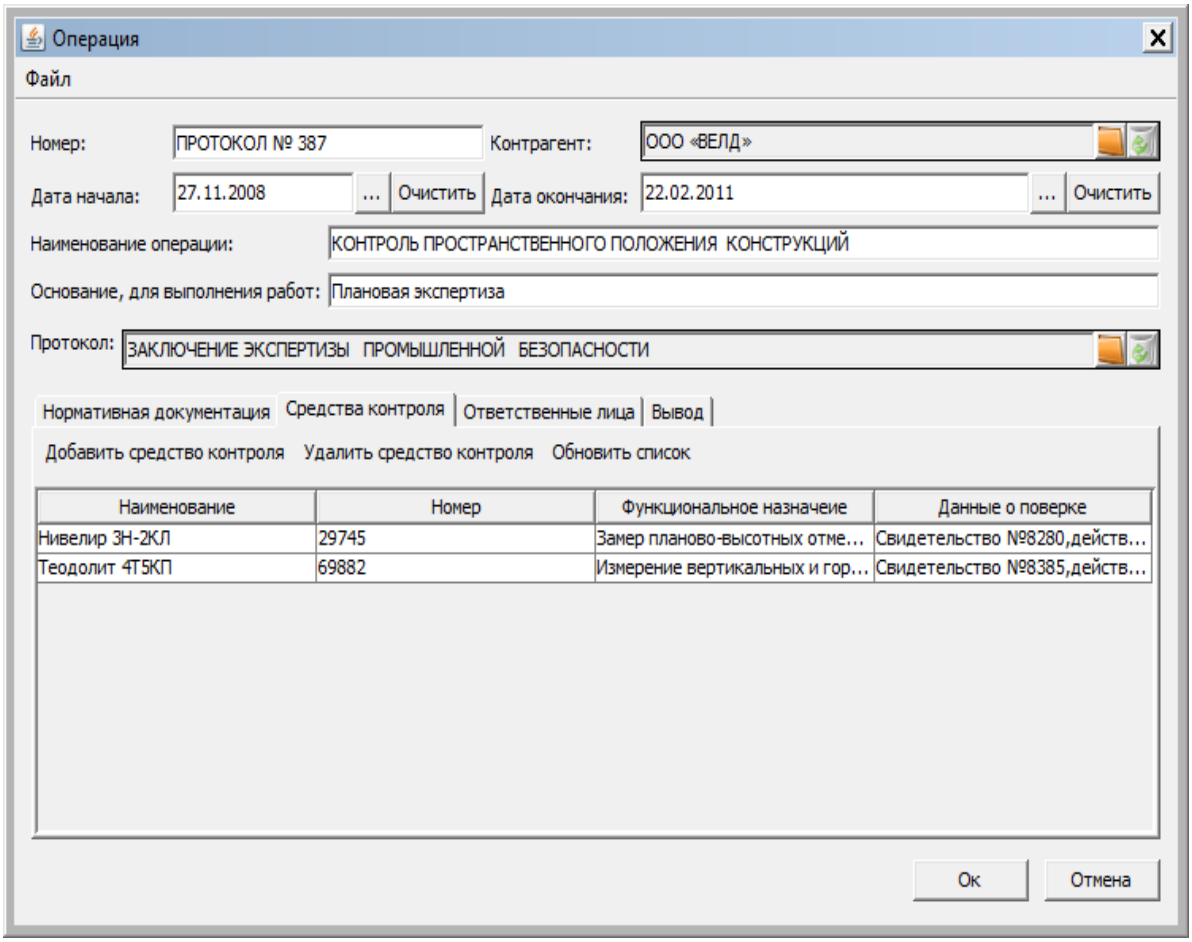


Рис. 12. Карточка операции

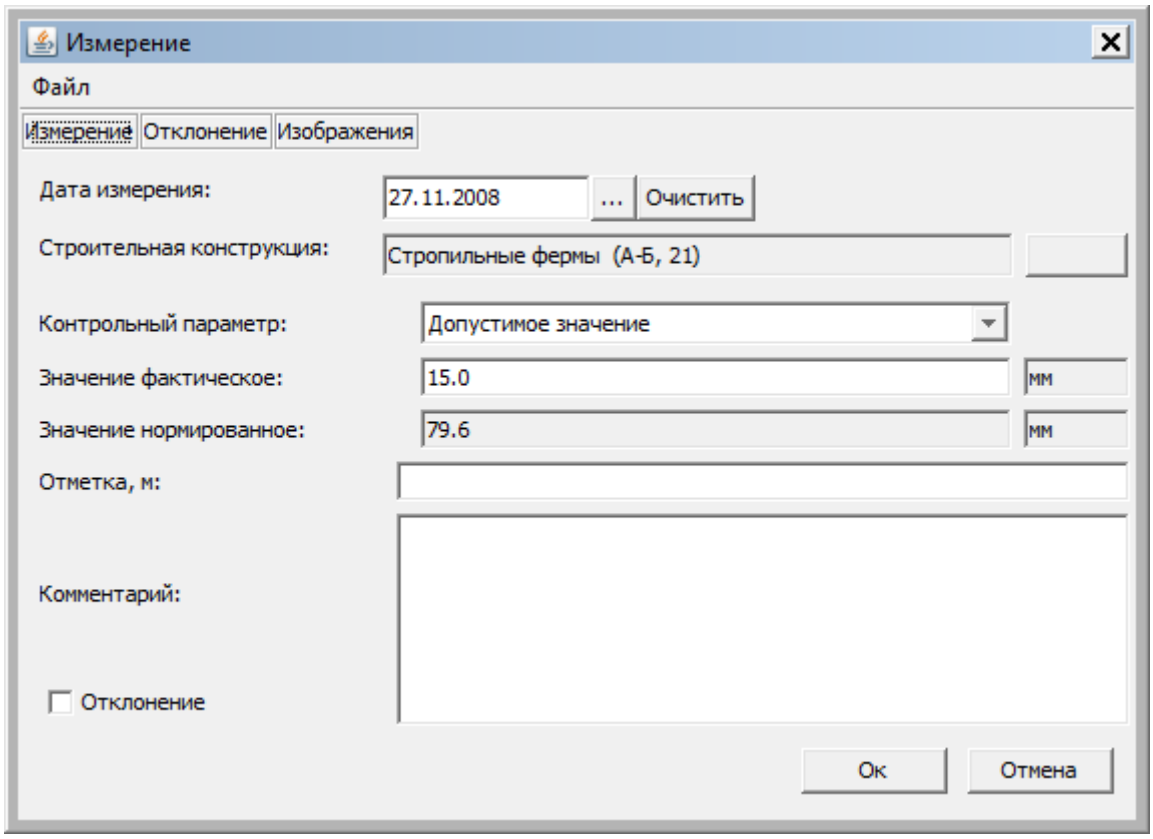


Рис. 13. Карточка измерения

В карточке измерения фиксируется конкретная операция измерения значения параметра безопасности. Причем, измеренное значение сравнивается с допустимым значением (расчетным или нормативным). Кроме того, в случае наличия недопустимого отклонения, интерфейс карточки предоставляет возможности для фиксирования отклонения и ввода дополнительной информации (рис.14). Кроме того, предоставляется возможность по выбору из справочника возможных вариантов устранения отклонения и фиксирования в системе причин возникновения отклонения. Дополнительно, предусмотрена возможность фотофиксации отклонений (рис. 15).

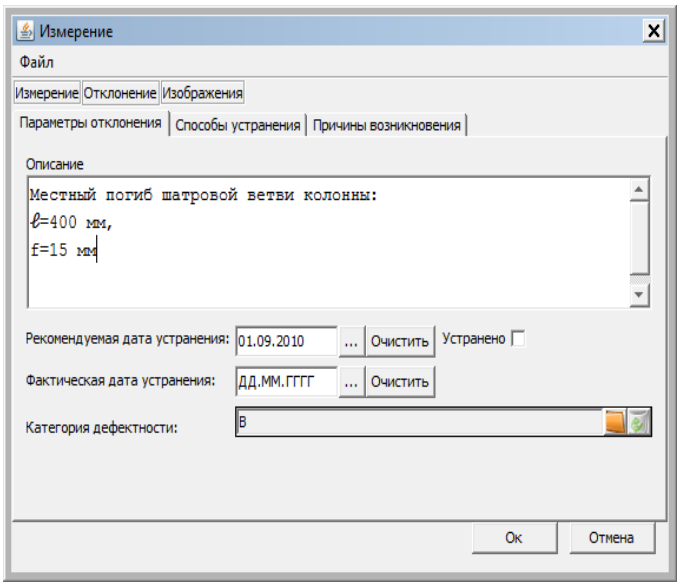


Рис. 14. Карточка измерения

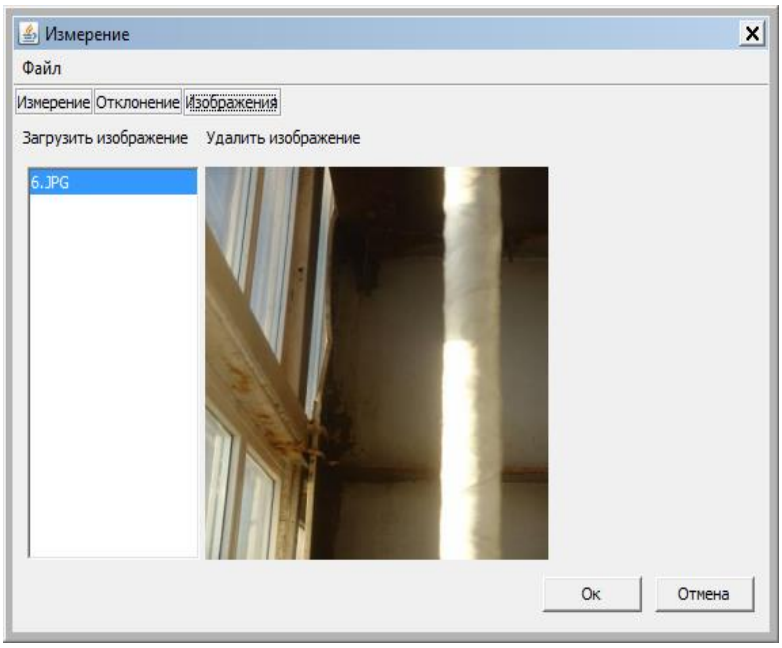


Рис. 15. Фотофиксация отклонений

Модуль «Промышленная труба»

Апробация модуля «Промышленная труба» выполнена на основании паспортных данных и данных о результатах экспертизы промышленной безопасности промышленной железобетонной трубы, предоставленных компанией «ВЕЛД». Строительный объект расположен в г. Сургут, предназначен для отвода дымовых газов от двух котлов утилизаторов и выброс их в атмосферу на высоту 96,7 м.

Работа с данными объекта осуществляется через экранную форму, представленную на рис. 16.

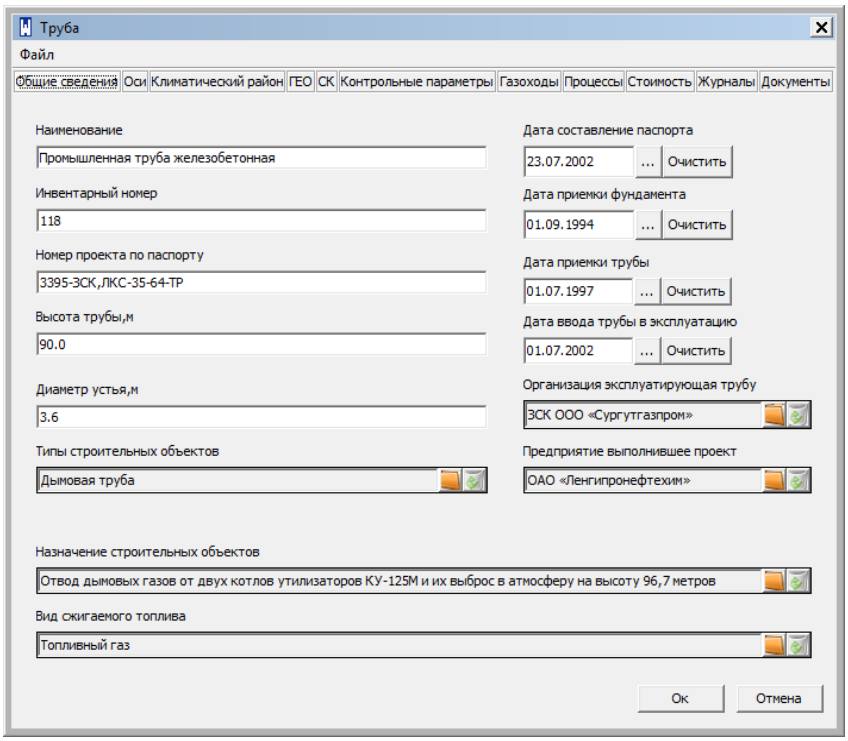


Рис.16. Карточка промышленной трубы

Вкладки «Оси», «Климатический район», «ГЕО», «СК», «Процессы» имеют аналогичное назначение и функциональность с аналогичными вкладками карточки строительного объекта, реализованной в модуле «Производственное здание». Однако, реализованы и ряд специфичных функций, характерных для данного вида строительных объектов. Например, реализована функциональность ведения следующих журналов:

- Журнал подключения агрегатов;
- Журнал контроля осадок и крена;
- Журнал выполненных работ;
- Журнал мероприятий по молниезащите.

Доступ к журналам осуществляется на соответствующей вкладке карточке объекта (рис. 17).

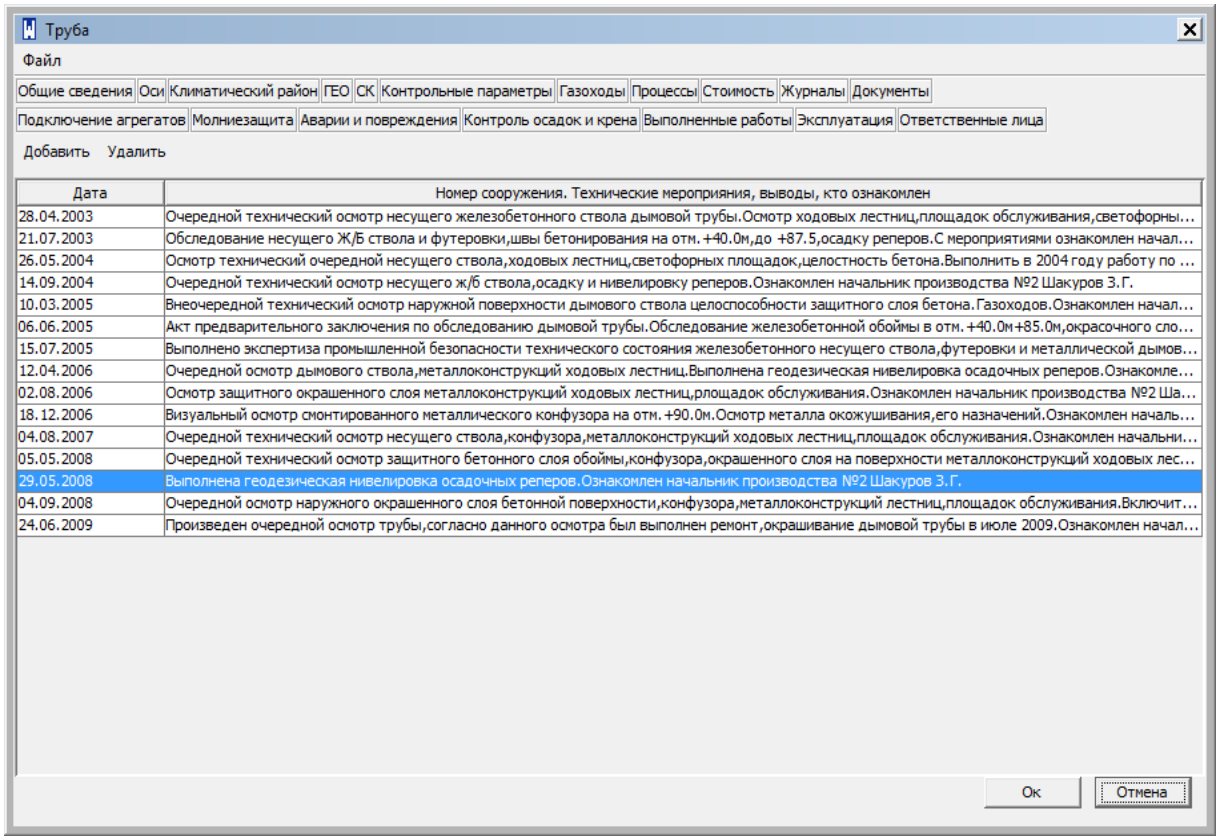


Рис. 17. Журналы

Кроме того, в рамках модуля реализован справочник нагревательного оборудования (рис. 18).

Для ввода информации о нагревательном оборудовании реализована карточка нагревательного оборудования (рис. 19).

Для нагревательного оборудования так же реализована возможность фиксирования значения параметров, с использованием общесистемного механизма.

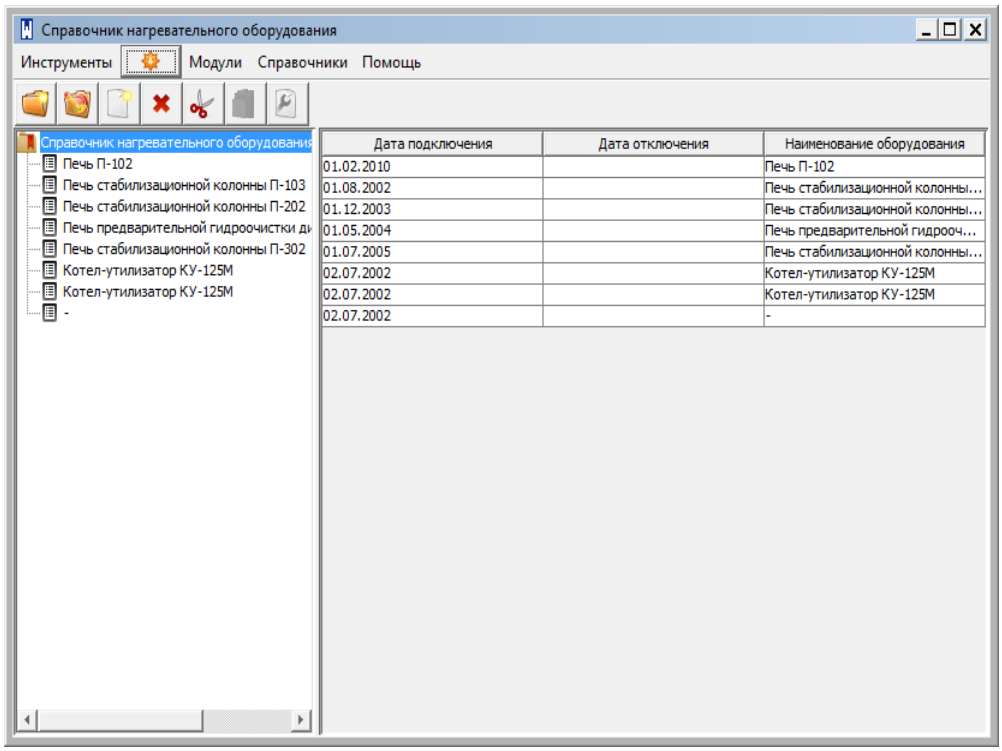


Рис. 18. Справочник нагревательного оборудования

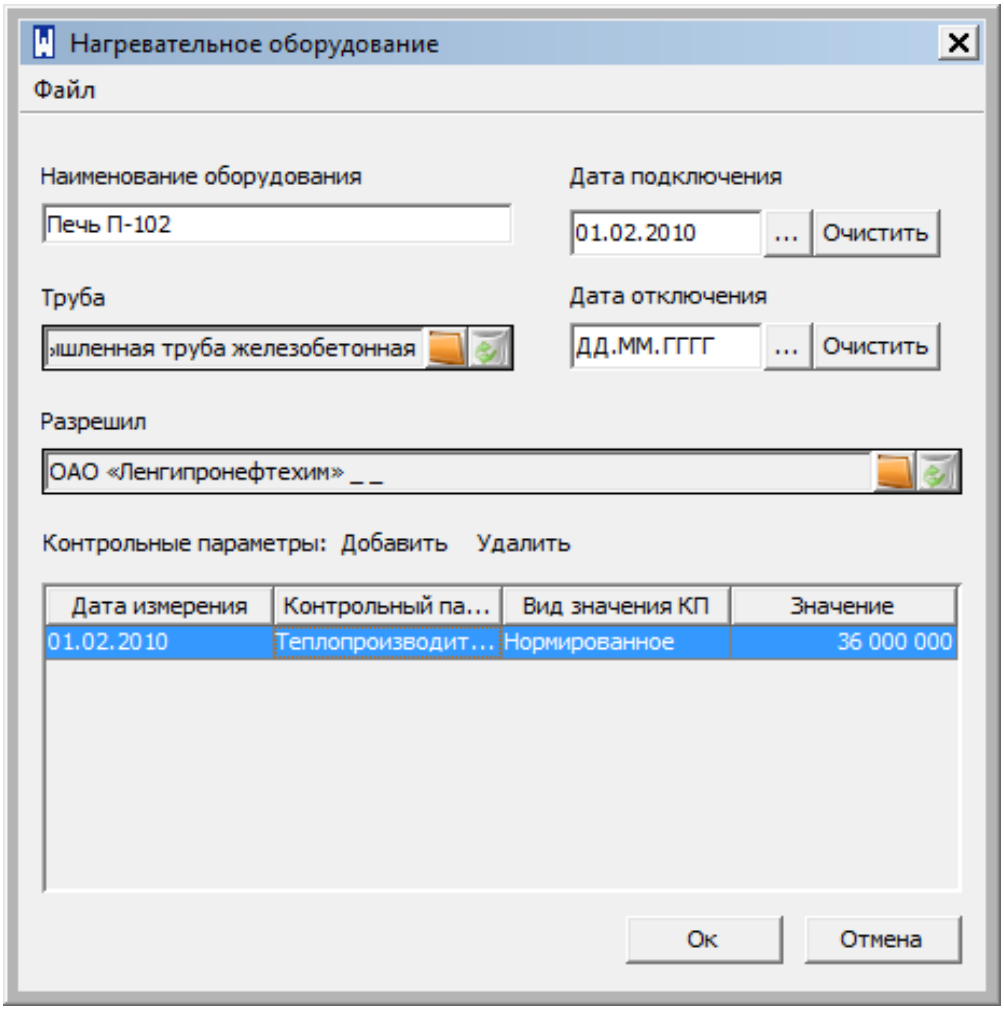


Рис. 19. Карточка нагревательного оборудования

Библиографический список

1. Ерёмин К.И. Реестр аварий зданий и сооружений 2001 - 2010 годов. / Ерёмин К.И., Махутов Н.А., Павлова Г.А., Шишкина Н.А. - Магнитогорск. Магнитогорский дом печати, 2011. - 320 с.
2. О создании технического комитета по стандартизации "Информационная поддержка жизненного цикла изделий" : Приказ от 2 июня 2001 №79 [Электронный ресурс] // Справочно-правовая система "Консультант плюс". - Режим доступа: локальный.
3. ГОСТ Р 51725.0-2001 Каталогизация продукции для федеральных государственных нужд. Комплекс нормативных документов по стандартизации. Общие положения. [Текст] - М : Госстандарт России : Изд-во стандартов, 2001

СЦЕНАРНЫЙ ПОДХОД К ОЦЕНКЕ ВЕРОЯТНОСТЕЙ АВАРИЙ НА ПЛОТИНАХ

THE SCENARIO APPROACH TO ESTIMATING THE PROBABILITY OF ACCIDENTS AT DAMS

УДК 624.042.7:519.81

Стефанишин Д.В.
Институт телекоммуникаций и глобального информационного пространства Национальной академии наук Украины, ведущий научный сотрудник, доктор технических наук
Dmytro V. Stefanyshyn
The Institute of Telecommunications and Global Information Space of the NAS of Ukraine, Principal Scientific Expert, Dr. Sc. (Tech.)

Аннотация
Анализируется опыт оценки вероятностей аварий на плотинах в рамках сценарного подхода с использованием логико-вероятностных методов.
Summary
The experience of assessing the probability of accidents at dams in the scenario approach with the use of logical and probabilistic methods has been analyzed.

Ключевые слова:
Вероятность аварии, метод деревьев отказов, метод деревьев событий, плотина, сценарный подход
Key words:
Accident probability, dam, event tree analysis method, fault tree analysis method, scenario approach

Прогнозирование аварий на плотинах является сложной системной задачей, отягощенной неопределенностью различной природы – стохастической и нестохастической, решение которой возможно только в рамках вероятностного подхода.

В последние годы нами рассматривалось несколько практических задач, связанных с оценкой вероятностей аварий на плотинах, которые эксплуатируются, строятся или проектируются в Российской Федерации и в странах бывшего СССР. Среди них – оценка вероятностей аварий на Бурейской и Мамаканской плотинах, гидросооружениях Ленинградской и Днестровской ГАЭС, Саратовского, Рогунского, Саяно-Шушенского и Нижне-Бурейского гидроузлов, Зарамагской ГЭС-1. Некоторые сведения о результатах этих исследований, о подходах, методах и моделях, которые при этом использовались, приводятся в [1-4].

Полученные результаты использовались при подготовке деклараций безопасности плотин, при обосновании решений по повышению их надежности и безопасности, при выборе вариантов гидросооружений.

В настоящей статье основное внимание уделяется развитию сценарного подхода к решению задач и использованию на его основе логико-вероятностных методов оценки вероятностей аварий на плотинах – методов деревьев отказов и деревьев событий.

Важность сценарного подхода к решению таких задач обусловливается тем, что аварии на плотинах носят системный характер, могут определяться разными природными и техногенными факторами, взаимодействием различных гидросооружений в составе гидроузлов и могут происходить по сценариям, зависящим от индивидуальных условий проектирования, строительства и эксплуатации сооружения. При расчетах вероятностей аварий на плотинах, как показывает опыт проведения таких исследований, недостаточно иметь репрезентативные выборки статистических данных об отказах и авариях на сооружениях, применять строгие математические методы, модели и современные вычислительные технологии при расчетах сооружений. При прогнозировании возможной аварии на плотине изначально приходится ориентироваться на системный анализ проблемы, что подразумевает и разработку гипотетических сценариев возможной аварии. Это связано с тем, что, во многих случаях, аварийные события на плотинах, учитывая уникальность этих сооружений и условий, в которых они проектируются, строятся и эксплуатируются, не всегда могут рассматриваться с точки зрения статистических фактов, а только как события-предположения. Сценарный подход при решении задачи вероятностного прогнозирования возможной аварии на плотине позволяет использовать синтез различных приемов и методов системного анализа: от эвристических приемов при раскрытии причинно-следственных связей до методов системно-структурного и абстрактно-логического анализа. При сценарном подходе как качество данных и математических моделей, так и обоснованность модельных сценариев аварии может определять адекватность результирующей оценки вероятности аварии на плотине, с учетом индивидуальных особенностей сооружения, его взаимодействия с другими гидросооружениями и объектами.

В результате сценарного моделирования отягощенная неопределенностью, многофакторная и, обычно, неструктурированная либо «слабо» структурированная задача прогнозирования аварии на плотине, может быть сведена к нескольким (за количеством гипотетических сценариев) «более структурированным» задачам прогнозирования, которые могут описываться более простыми математическими моделями с меньшим числом определяющих факторов и параметров.

Сценарный подход позволяет соединить возможности разных методов и моделей, используемых при оценке вероятностей отдельных аварийных событий на плотине, осуществить агрегацию моделей и синтез вероятностей, полученных для событий, инициирующих аварии либо способствующих их развитию. При оценке вероятностей отдельных аварийных событий, входящих в состав гипотетического сценария аварии, в зависимости от имеющихся данных, могут использоваться разные методы и модели: статистические; методы математической теории надежности; методы рандомизации традиционных детерминистических моделей и расчетных схем; логико-вероятностные методы (методы анализа деревьев событий, деревьев отказов) [5, 6]. При этом обобщение (агрегацию) вероятностей аварий по различным сценариям, как показывает опыт вероятностного прогнозирования, удобнее всего осуществлять логико-вероятностными методами (деревьев событий, деревьев отказов).

Метод деревьев событий

В основу метода деревьев событий по аналогии с деревом решений положен индуктивный анализ [6-8]. Дерево событий строится как бинарное граф-дерево, начиная с исходного аварийного события, способного инициировать аварийные процессы, на основе прямых логических размышлений, чтобы ответить на вопрос: «Что будет, если …?». При этом могут отображаться как ветви (сценарии) развития возможных аварийных ситуаций, вызываемых исходным аварийным событием, так и ветви, ведущие к затуханию аварийных процессов.

Разветвление дерева событий происходит в точках разделения сложного события на два несовместных события, более простых по сравнению с исходным событием. В результате на каждом этапе анализа в точках разделения ветвей формируются полные группы событий. Благодаря этому агрегация вероятностей реализации некоторых *j*-х несовместных веток (сценариев), ведущих к *k*-м формам аварии плотины в результате исходного *i*-го инициирующего аварийные процессы события, может осуществляться по формуле полной вероятности:

$$P(A_{ik}) = \sum_j P(A_{ik} \mid S_{ijk}) \cdot P(S_{ijk}), \quad (1)$$

где $P(A_{ik} \mid S_{ijk})$ – условная вероятность реализации *k*-й формы аварии для *i*-го события, инициирующего аварийные процессы за сценарием S_{ijk}; вероятность $P(A_{ik} \mid S_{ijk})$ можно определять как вероятность превышения некоторой экстремальной случайной величиной (нагрузкой, влиянием, сочетанием нагрузок и т.п.) установленной допустимой границы, задаваемой, в общем случае, случайным образом [6]; $P(S_{ijk})$ – вероятность реализации события S_{ijk} (сценария аварии), предшествующего *k*-й форме аварии на плотине.

Для простейшего сценария, реализуемого в виде сочетания нескольких *q*-х элементарных независимых аварийных событий E_{ijkq}, обусловливающих возникновение и развитие этого сценария, вероятность

$$P(S_{ijk}) = \prod_q P(E_{ijkq}). \quad (2)$$

Для сценария, реализуемого в виде сочетания некоторого исходного независимого события $E_{0,ijk}$ и нескольких последовательно зависимых друг от друга аварийных событий вероятность

$$P(S_{ijk}) = P(E_{0,ijk}) \cdot P(E_{1,ijk} \mid E_{0,ijk}) \cdot P(E_{2,ijk} \mid E_{1,ijk}) \cdot \ldots. \quad (3)$$

В формулах (2), (3) $P(E_{ijkq}), P(E_{0,ijk})$ – вероятности аварийных событий E_{ijkq}, $E_{0,ijk}$; $P(E_{1,ijk} \mid E_{0,ijk})$, $P(E_{2,ijk} \mid E_{1,ijk}) \ldots$ – вероятности перехода от события $E_{0,ijk}$ к событию $E_{1,ijk}$, от события $E_{1,ijk}$ к событию $E_{2,ijk}$ и т.д., с охватом всей цепочки аварийных событий, моделирующих наступление события (сценария) S_{ijk}.

Полная вероятность аварии в результате *i*-го аварийного события, инициирующего аварию, в случае независимости и совместности нескольких сценариев, можно оценить по формуле:

$$P(A_i) = 1 - \prod_k [1 - P(A_{ik})]. \quad (4)$$

Ниже приводится пример использования метода деревьев событий при оценке вероятности одного из гипотетических сценариев аварии (далее аварии A_1) на Саяно-Шушенской плотине. Сценарий был утвержден в качестве одного из расчетных на Техническом совещании в ОАО «Ленгидропроект в ноябре 2009 г. для подготовки Декларации безопасности гидросооружений Саяно-Шушенской ГЭС (СШГЭС) на период временной эксплуатации берегового водосброса и восстановленных гидроагрегатов ГЭС с 01.06.10 г. по 01.10.11 г. Формулировался сценарий аварии A_1 следующим образом: «В результате землетрясения левобережный склон примыкания плотины приходит в состояние предельного равновесия. Происходит потеря устойчивости скального склона в примыкании плотины и его разрушение с нарушением монолитности тела плотины выше отметки 479 м. Разрушение напорного фронта на длине 150 м (секции 6-15) с образованием прорана и волны прорыва».

Дерево событий для оценки вероятности аварии на Саяно-Шушенской плотине в результате землетрясения приводится на рис. 1.

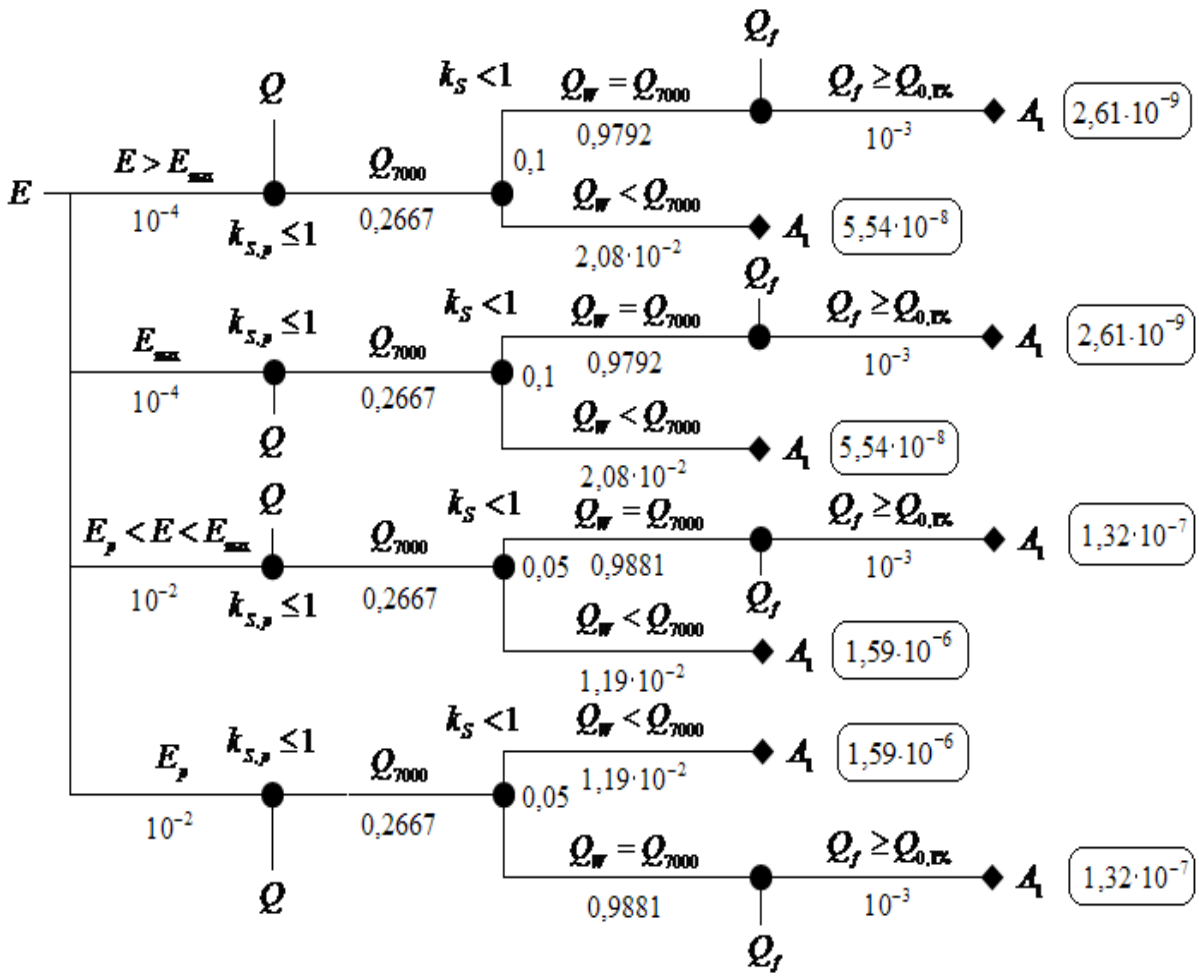


Рис. 1. Дерево событий для оценки вероятности аварии на Саяно-Шушенской плотине в результате землетрясения (сценарий A_1)

На рис. 1 приводятся следующие события, связанные с реализацией анализируемого сценария аварии:

E – землетрясение;

$E > E_{\max}$ – землетрясение, превышающее максимальное расчетное землетрясение (МРЗ); при периоде повторяемости МРЗ равном 10000 лет имеем вероятность возникновения сейсмических событий 10^{-4}, год$^{-1}$;

E_{max} – МРЗ; при периоде повторяемости МРЗ равном 10000 лет имеем вероятность возникновения сейсмического события 10^{-4}, год$^{-1}$;

$E_p < E < E_{\max}$ – землетрясение, превышающее проектное землетрясение (ПЗ), но меньшее МРЗ; при периоде повторяемости ПЗ равном 100 лет вероятность возникновения сейсмических событий 10^{-2}, год$^{-1}$;

E_p – ПЗ; при периоде повторяемости ПЗ равном 100 лет имеем вероятность возникновения сейсмического события 10^{-2}, год$^{-1}$;

$k_{s,p}$ =1 – левобережный склон примыкания плотины приходит в состояние предельного равновесия;

Q – приток в водохранилище;

Q_{7000} – уровень воды в водохранилище на момент землетрясения соответствует сбросному расходу 7000 м^3/с; при оценке вероятности события Q_{7000} учитывалось, что за период работы СШГЭС с 1979 г. по 2009 г. указанный сбросной расход превышался во время весеннего половодья 7 раз, во время дождевых паводков – 1 раз; имеем расчетную вероятность события 0,2667;

k_S <1 – потеря устойчивости склона; с запасом по риску при уровне воды в водохранилище, который соответствует сбросному расходу 7000 м^3/с, и МРЗ вероятность события k_S <1 принималась равной 0,1; то же с запасом по риску при ПЗ вероятность события k_S <1 принималась равной 0,05; такие значения вероятностей для события k_S <1 соответствуют достижению состояния предельного равновесия при расчетах устойчивости по методу предельных состояний при особом и основном сочетаниях загрузок и воздействий, соответственно;

$Q_W < Q_{7000}$ – отказ водосбросного фронта с невозможностью аварийной сработки водохранилища; оценка вероятностей события $Q_W < Q_{7000}$ осуществлялась методом деревьев отказов (см. ниже расчетные модели на рис. 2); полная вероятность $Q_W < Q_{7000}$ при МРЗ составила $2{,}08 \cdot 10^{-2}$, при ПЗ – $1{,}19 \cdot 10^{-2}$;

$Q_W = Q_{7000}$ – аварийная сработка водохранилища обеспечивается; вероятности $Q_W = Q_{7000}$ определялись как дополнение вероятностей $Q_W < Q_{7000}$ до единицы; при МРЗ имеем 0,9792, при ПЗ – 0,9881;

Q_f – паводок;

$Q_f \geq Q_{0,1\%}$ – паводок, равный расчетному паводку 0,1% вероятности превышения либо больше его; вероятность события 10^{-3}, год$^{-1}$;

A_1 – разрушение напорного фронта на длине 150 м (секции 6-15) с образованием прорана и волны прорыва в результате потери устойчивости скального склона в

примыкании плотины и его разрушение с нарушением монолитности тела плотины выше отметки 479 м.

В результате расчета модельного дерева событий (см. рис. 1) были получены вероятности развития аварии на Саяно-Шушенской плотине в результате землетрясения по различным сценариям A_1 и полная (суммарная) вероятность гипотетической аварии A_1, равная $3{,}56 \cdot 10^{-6}$, год$^{-1}$.

Метод деревьев отказов

В основу метода деревьев отказов положен дедуктивный анализ [6-8]. При этом различные события, связанные с возможной аварией на плотине, разделяются:

– на базовые (исходные) события и состояния, вероятности этих событий устанавливаются до расчета дерева отказов; на диаграмме дерева отказов изображаются окружностями с одним выходом;
– головное (вершинное, результирующее) событие-следствие дерева отказов, определяемое логикой взаимодействия всех аварийных событий и состояний, включаемых в диаграмму дерева отказов; на диаграмме головное событие изображается прямоугольником с одним «входом»;
– промежуточные события-следствия, представляющие собой результаты логического взаимодействия нескольких входных событий; вероятности событий-следствий устанавливаются в зависимости от действия логических операторов (символов-операторов) (см. ниже таблицу); на диаграмме дерева отказов промежуточные события-следствия изображаются прямоугольниками с одним «входом» и одним «выходом»;
– события-условия, при реализации которых создаются предпосылки для развития аварии; вероятности событий-условий устанавливаются до расчета дерева отказов; на диаграмме дерева отказов условия изображаются округленными прямоугольниками с боковым «входом».

Используются также символы переноса (в виде треугольников), отображающие перенесение логической связи (определенной ветви дерева отказов) с одной точки диаграммы в другую и символы для нераскрытых на момент построения дерева отказов событий (в виде ромбов) [6-8].

Среди логических операторов при построении деревьев отказов наиболее часто используются [6-8]:

"ИЛИ" (отвечает логической операции «дизъюнкция»);

"И" (отвечает логической операции «конъюнкция»);

"ИСКЛЮЧИТЕЛЬНОЕ ИЛИ" (отвечает логической операции «исключительная дизъюнкция»);

"БЛОКИРОВКА" (отвечает логической операции «импликация»).

Могут использоваться и другие логические операторы, в частности, "ЧАСТИЧНАЯ БЛОКИРОВКА", "M ИЗ N" и др. [6-8].

Логические операторы и оценка вероятностей событий-следствий методом деревьев отказов

Наименование логического оператора	Формулы для расчета вероятностей событий-следствий
"ИЛИ"	$P(A)=1-\prod_{i=1}^{n}(1-P(B_i))$ (5)
"И"	$P(A)=\prod_{i=1}^{n}P(B_i)$ (6)
"ИСКЛЮЧИТЕЛЬНОЕ ИЛИ"	$P(A)=\sum_{i=1}^{n}P(B_i)$ (7)
"БЛОКИРОВКА"	$P(A)=P(B)\cdot P(C)$ (8)
"ЧАСТИЧНАЯ БЛОКИРОВКА"	$P(A)=P(B_1)\cdot P(C)+P(B_2)\cdot(1-P(C))$ (9)
В формулах (5)÷(9) вероятности: $P(B_j)$ – исходного i-го события B; $p(C)$ – условия C; $P(A)$ – результирующего события A	

Ниже приводится пример использования метода деревьев отказов при оценке вероятности отказа водосбросного фронта СШГЭС (событие $Q_W < Q_{7000}$). Моделировалась ситуация невозможности аварийной сработки водохранилища при повреждении Саяно-Шушенской плотины в результате землетрясения (см. сценарий на рис. 1).

При оценке вероятности события $Q_W < Q_{7000}$ – «отказ водосбросного фронта и невозможность аварийной сработки водохранилища» (см. рис. 2) рассматривались следующие базовые события, которые изображались на диаграммах деревьев отказов окружностями:

1) $B_{1.1.1}$, $B_{1.1.2}$, $B_{1.1.3}$ – отказы агрегатов ГЭС по пропускной способности; вероятности событий $B_{1.1.1}$, $B_{1.1.2}$, $B_{1.1.3}$ при МРЗ принимались равными 0,003; при ПЗ – равными 0,002 – вероятностям отказа направляющего аппарата [7];

2) $B_{1.2.1} \div B_{1.2.11}$ – отказы механического оборудования эксплуатационного водосброса при маневрировании затворами; вероятности $B_{2.1} \div B_{2.11}$ при МРЗ принимались равными 10^{-3}; при ПЗ – $5 \cdot 10^{-4}$ [6];

3) $B_{1.3.1}$ – отказ механического оборудования берегового водосброса при маневрировании затвором; вероятность события $B_{3.1}$ при МРЗ принималась равной 10^{-3}; при ПЗ – $5 \cdot 10^{-4}$ [6].

Таким образом, сценарный подход позволяет осуществить синтез вероятностей аварийных событий, полученных с помощью разных методов, в том числе и методами деревьев событий и деревьев отказов. В рамках сценарного подхода можно учесть сложность иерархической структуры плотины как системы, реализовать принципы иерархии и наименьшего взаимодействия. В частности, при чрезмерной сложности иерархической структуры сооружения при решении задачи может использоваться агрегирование событий методом деревьев отказов. В свою очередь принцип наименьшего взаимодействия в системе реализуется в рамках метода деревьев событий. В результате синтеза этих двух методов в рамках сценарного подхода можно обеспечить адекватность результирующей оценки вероятности аварии на плотине с учетом ее индивидуальных особенностей, взаимодействия с другими гидросооружениями и объектами.

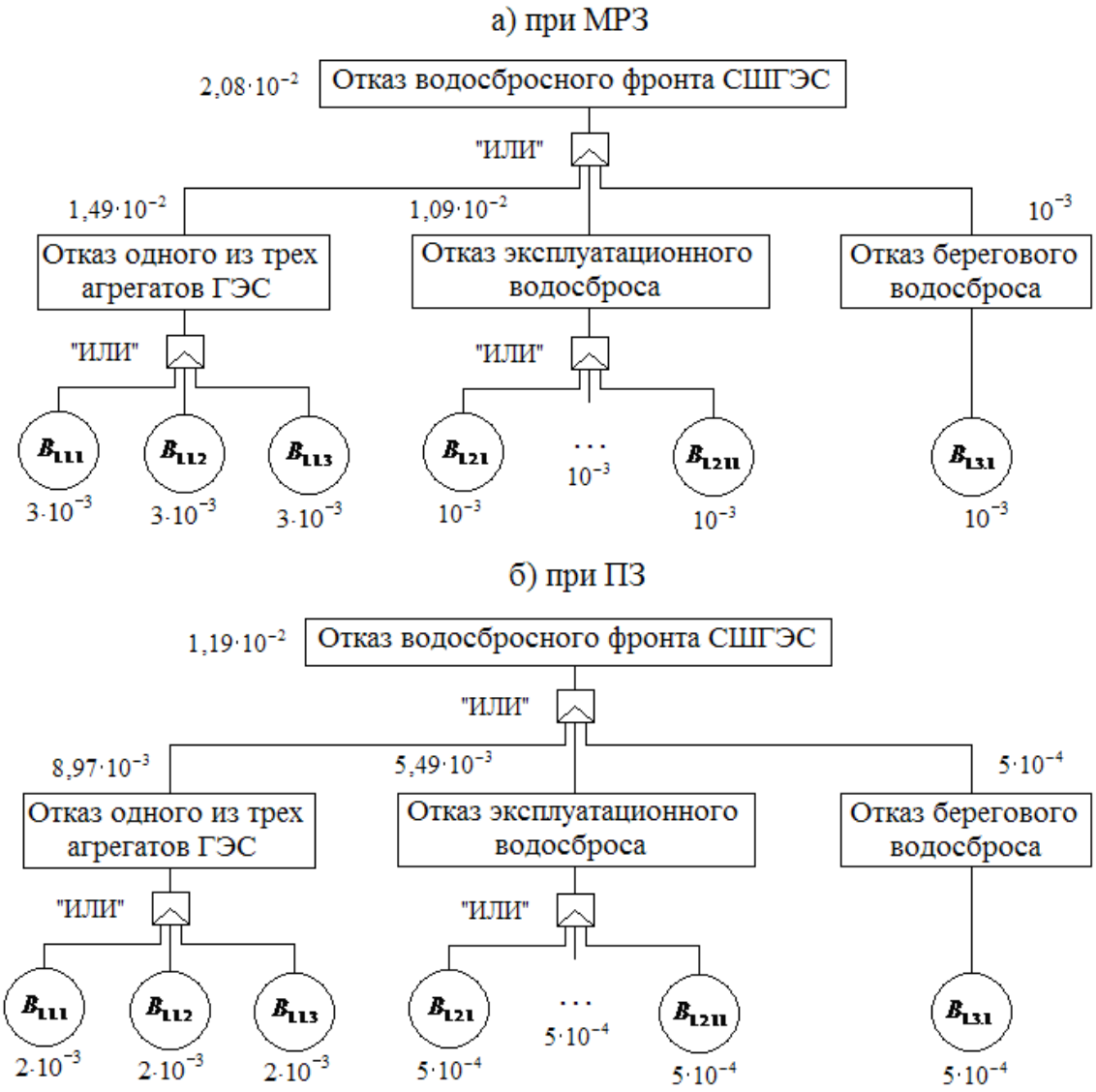


Рис. 2. Дерево отказов для оценки вероятностей события $Q_W < Q_{7000}$ – «отказ водосбросного фронта и невозможность аварийной сработки водохранилища»:
а – при МРЗ; б – при ПЗ

Библиографический список

1.Стефанишин Д.В. Некоторые результаты оценки вероятности аварии на плотине Бурейской ГЭС / Д.В. Стефанишин // Известия ВНИИГ им. Б.Е.Веденеева. 2008. Т. 251. С.10-17.

2.Stefanyshyn D.V. Assessment of accident risks to support safety of the Boureya dam / D.V. Stefanyshyn // Proc. of Int. Scientific School “Modelling and Analysis of Safety and Risk in Complex Systems”. June 24-28, 2008. Saint-Petersburg, Russia. P.P. 377-382.

3.Стефанишин Д.В. Вибрані задачі оцінки ризику та прийняття рішень за умов стохастичної невизначеності / Д.В. Стефанишин. – К.: Азимут-Україна, 2009. – 103 с.

4.Стефанишин Д.В. Вероятностное моделирование гипотетических сценариев двух нетиповых аварий на гидроэнергетических объектах при отказах автоматики / Д.В.

Стефанишин, Е.Г. Романчук: [Эл. документ] // Предотвращение аварий зданий и сооружений. – 2011. (http://pamag.ru/pressa/vmgs-heo).

5. Стефанишин Д.В. Статистические оценки живучести плотин при авариях / Д.В. Стефанишин: [Эл. документ]// Предотвращение аварий зданий и сооружений. – 2011. (http://pamag.ru/pressa/stat-ocenka-plotina).
6. Векслер А.Б. Надежность, социальная и экологическая безопасность гидротехнических объектов: оценка риска и принятие решений / А.Б. Векслер, Д.А. Ивашинцов, Д.В. Стефанишин // СПб.: ОАО «ВНИИГ им. Б.Е. Веденеева», 2002. – 589 с.
7. Kumamoto H., Henley E.J. Probabilistic Risk Assessment and Management for Engineers and Scientists / H. Kumamoto, E.J. Henley // New York. IEEE Press. 1996. – 597 p.
8. The use of risk analysis to support dam safety decisions and management. Trans. of the 20-th Int. Congress on Large Dams. Vol. 1. Q. 76. 19-22 September. Beijing-China. 2000. – 896 p.

ВОЗМОЖНОСТИ ВОССТАНОВЛЕНИЯ КУЛЬТОВЫХ ЗДАНИЙ

POSSIBILITY OF RESTORATION OF RELIGIOUS BUILDINGS

Пунгин В.Л.
Научно-исследовательское архитектурно-проектное предприятие «НИАП»
Пунгина Т.В.
Курганская государственная сельскохозяйственная академия им. Т.С. Мальцева

Аннотация
Приведены результаты обследования строительных конструкций церкви Засима и Савватия в с. Усть-Суерском Белозерского района Курганской области, предложена методика восстановления здания церкви по этапам.

Ключевые слова:
Культовые здания, Церковь Засима и Савватия, Обследование строительных конструкций, Повреждение, Восстановление
Key words:
Religious buildings, the Church Zosima and Savvaty, the Survey of construction, Damage, Restoration

Историческим наследием нашей русской культуры являются культовые здания, разбросанные по ближним и дальним рубежам российских областей. Десятилетия атеистической идеологии в государстве серьезно сказались на состоянии многих храмов и церквей, являющихся не только великолепными архитектурными объектами, но и выдающимися примерами строительного мастерства наших предков. В культовых зданиях гармонично сочетаются архитектурно-композиционные решения и виртуозные строительные приемы кирпичной кладки стен, сводов и куполов. Важной задачей на современном этапе социально-экономического развития общества является восстановление храмовых зданий, которые не утратили эксплуатационных свойств или техническое состояние которых позволяет воссоздать их конструктивные и художественные характеристики.

Крупным культовым зданием, возведенным в середине 19-го века, является церковь Засима и Савватия в с. Усть-Суерском Белозерского района Курганской области. Эта церковь двухэтажная: верхний этаж – летняя церковь, нижний – теплая церковь, эксплуатировалась зимой, как отапливаемая часть всей церкви. По сохранившимся архитектурным деталям композиционно-художественное решение церкви можно отнести к барочному стилевому направлению. По архитектурно-планировочной структуре здание церкви имеет трехчастное построение: притвор и трапезная, центральная часть (наос) и алтарь. Первый этаж перекрыт каменными сводами. В перекрытии и северной стене сохранился канал дымохода печи отопления первого этажа. Над притвором возвышается колокольня высотой более 27 м. Трапезная второго этажа перекрывалась деревянными балками, которые были выполнены из бревен диаметром более 30 см. Балки перекрывали пролет 8 м, сохранились гнезда под балки в поперечных стенах церкви. Центральная часть церкви в плане квадратная,

перекрывается куполом со световым проемом в зените. Купол завершал световой барабан с центральной главой. Высота второго этажа центральной части церкви до основания купола составляет 10 м. Алтарь перекрывается каменным сводом.

В рамках работ по восстановлению здания было выполнено обследование строительных конструкций церкви. За период десятилетий запущенности и даже варварского отношения к данному церковному зданию образовались различной степени дефекты, разрушения и утраты отдельных частей. На отдельных участках стен произошло выветривание и разрушение кирпичной кладки на глубину до 150 мм, значительное разрушение наружного декоративного штукатурного слоя. К серьезным повреждениям и утрате элементов привело отношение людей к своему культурному наследию. Были разрушены центральная глава и барабан над куполом церкви, крыша и перекрытие над трапезной, с западной части входная двухэтажная часть здания (рис. 1).



Рис. 1. Церковь Засима и Савватия в с. Усть-Суерское

В процессе обследования были выявлены трещины, носящие сквозной характер, в южной стене трапезной от карниза до верха оконного проема первого этажа, в восточной стене алтаря, в стене между центральной частью и алтарем (рис. 2, 3). В одной из шей угловой главки в кирпичной кладке выявлены продольные трещины. Наличие трещин в цокольной части стен здания церкви не выявлено. Для определения конструкции фундамента и наличия грунтовых вод были устроены шурфы до подошвы фундамента. Фундамент представляет собою кирпичную кладку толщиной 1,5 м, выполненную по толщине кирпичных стен. При вскрытии грунтовых вод обнаружено не было, грунт не насыщен влагой. Это позволило сделать вывод, что образование трещин не вызвано деформациями фундамента из-за неравномерной осадки основания.

Характер образования трещин от карниза стен здания и их расположение позволяет сделать вывод, что происхождение связано с проникновением в тело стены влаги по причине разрушения кровли.



Рис. 2. Трещина в южной стене трапезной



Рис. 3. Трещины в стене алтаря

Была выбрана методика восстановления здания церкви по этапам, первоначально тех элементов, которые в большей степени влияют на состояние стен и внутреннего пространства церкви. В первую очередь были разработаны проектные решения по восстановлению крыши над трапезной, центральная глава с барабаном, глава над колокольней (рис. 4). В процессе реконструкции воссоздаются элементы покрытия, обеспечивающие защиту внутреннего пространства и стен здания. Восстановление эксплуатационных свойств кирпичной кладки предусматривается путем инъецирования цементно-полимерных составов. Сохранение прочностной и несущей способности стен обеспечила их массивность, которая определяется толщиной кирпичной кладки 1,5 м в основании.

Характер повреждений строительных элементов здания церкви не приводит к критическому (аварийному) состоянию несущих конструкций, что обеспечивает

условие для воссоздания церкви и устранения возникших повреждений кирпичной кладки стен.

Культовые здания, обладающие массивностью стен, имеют большой запас надежности для продолжительного периода эксплуатации. Это позволяет осуществлять восстановление данных типов зданий после длительного существования в неблагоприятных условиях.

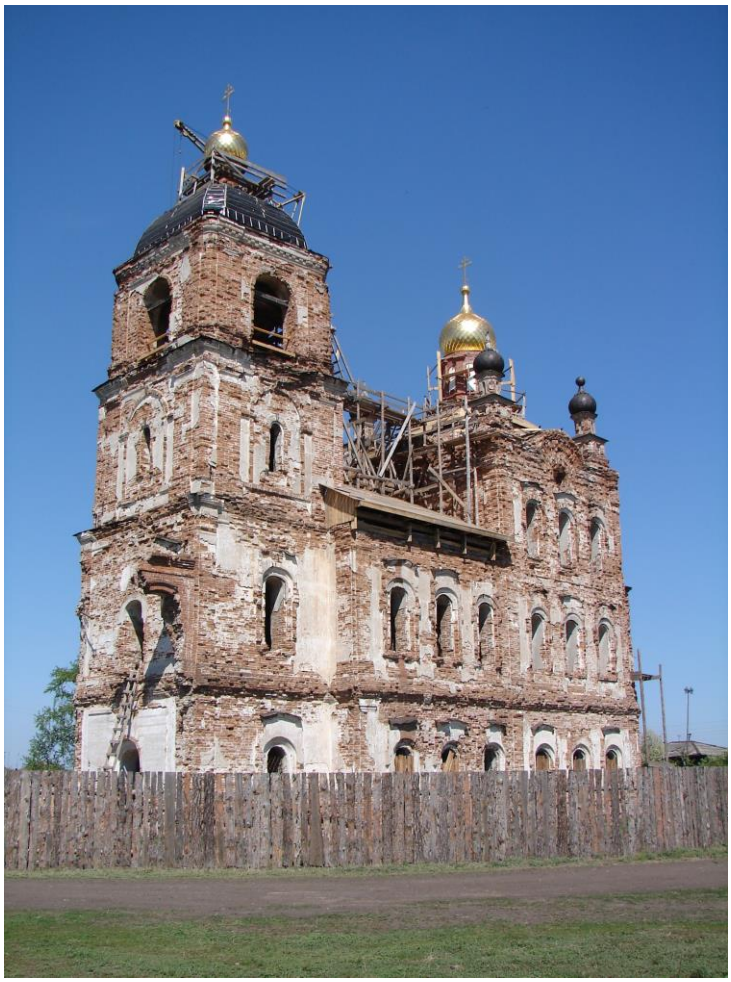


Рис. 4. Работы по восстановлению церкви Засима и Савватия в с. Усть-Суерское

Библиографический список

1.Архитектурное наследство / Отв. ред. И.А. Бондаренко. Вып. 49. – М.: КомКнига, 2008. – 344 с.
2.Архитектурное наследство/ Отв. ред. И.А. Бондаренко. Вып. 52. – М.: КомКнига, 2010. – 344 с.
3.Иванов Ю.В. Реконструкция зданий и сооружений: усиление, восстановление, ремонт: Учебное пособие. – М.: Издательство АСВ, 2009. – 312 с.
4.Практика реставрационных работ / Под ред. Ш.Е. Ратия, П.Н. Максимова. Сборник второй. – Государственное издательство литературы по строительству, архитектуре и строительным материалам. – М., 1958. – 184 с.

СБОРНО-МОНОЛИТНАЯ СОСТАВНАЯ СВАЯ

Пермяков М.Б., Веселов А.В., Токарев А.А.
ФГБОУ ВПО «Магнитогорский государственный технический университет им. Г.И. Носова», г.Магнитогорск

Permyakov M.B., Veselov A.V., Tokarev A.A.
Magnitogorsk state technical university of G.I.Nosov

Аннотация
Предлагается новая свайная конструкция для фундаментов с повышенными показателями несущей способности, а также технология возведения фундаментов из этих свай.

Ключевые слова:
Свая, Свайная конструкция, Строительство, Свайный фундамент, Здания и сооружения, Сборно-монолитная составная свая, Проектирование, Грунтовое основание
Key words:
Pile, Pile design, Construction, Pile Foundation, Buildings and structures, Precast concrete composite pile, the Design, the subgrade

За последнее время в массовом промышленном и гражданском строительстве резко возросло применение свайных фундаментов, которые, в силу ряда их достоинств, пришли на смену ленточным и столбчатым фундаментам. Для обеспечения наилучшей работы по восприятию передаваемых на них нагрузок были разработаны различные конструктивные решения свайных фундаментов и технологические процессы их выполнения. Наибольшее распространение получили фундаменты, выполняемые из призматических сборных железобетонных свай заводского изготовления длиной 6-12 метров.

Анализируя требования, предъявляемые к конструктивным параметрам призматических свай на различных стадиях их существования, начиная от изготовления конструкций и заканчивая их работой в свайных фундаментах зданий и сооружений, легко заметить крайнюю противоречивость выдвигаемых условий. С одной стороны проще и комфортнее работать с небольшими по размерам и массам конструкциями. Это объясняется тем, что увеличение длины и веса свай неизбежно влечёт за собой усложнение всех технологических процессов, снижение производительности труда и необходимость использования более мощных строительных машин и механизмов. Кроме того, в этом случае возрастает вероятность повреждения свай при забивке.

С другой стороны возникает необходимость в обеспечении, иногда очень большой, несущей способности фундаментов. Выполнение этого условия у свайных фундаментов из призматических свай в значительной степени зависит от величины силы трения грунта по боковым поверхностям свай, погруженных в грунтовое основание. Повышение несущей способности сваи обычно достигается увеличением

суммарной площади её боковых поверхностей, контактирующих с грунтом, за счет увеличения длины или размеров поперечного сечения сваи, либо одновременным увеличением этих двух параметров. Но во всех случаях увеличение площади боковых поверхностей приведет к возрастанию габаритов, объема и массы сваи. Поэтому, обладающая большой площадью боковых поверхностей, призматическая железобетонная свая всегда будет иметь значительные линейные размеры и массу.

Таким образом можно сформулировать основные требования, которым должна удовлетворять свайная конструкция:

– простое конструктивное решение;
– небольшие размеры и массы свайных элементов;
– несложная технология изготовления в условиях обычных заводов ЖБИ;
– небольшой процент армирования, использование ненапрягаемой арматуры;
– удобство производства погрузочно-разгрузочных и транспортных работ;
– несложная технология изготовления свайного фундамента с использованием обычного оборудования, применяемого для производства данного вида работ;
– высокая несущая способность свайного фундамента.

Таким образом, широко используемые в современном строительстве типовые призматические железобетонные сваи заводского изготовления не в состоянии соответствовать одновременно всем вышеупомянутым требованиям.

Для решения поставленных задач было разработано новое конструктивное решение сборно-монолитной составной сваи для фундаментов с повышенными показателями несущей способности, а также технология возведения фундаментов из этих свай.

Предлагаемая свая (рис.1) состоит из центрального монолитного железобетонного цилиндрического стержня, на котором последовательно закреплены контактирующие между собой лидирующий, промежуточные и головной продольные сборные железобетонные элементы. Каждый продольный элемент (рис.2) представляет собой единое тело, которое может быть выполнено в двух различных модификациях. В первой модификации оно представляет собой усеченный конус с двумя цилиндрами, прикрепленными к его основаниям, во второй усеченную четырехгранную пирамиду с квадратными основаниями, к которым прикреплены, соответственно, две призмы. Для упрощения восприятия информации, дальнейшее ознакомление с особенностями конструктивного решения предлагаемой составной сваи будет иллюстрироваться на примере её выполнения в первой модификации. Угол наклона образующей конусообразных частей по отношению к центральной оси у всех продольных элементов одинаковый и может задаваться в пределах от 3 до 10°. Все продольные элементы выполнены с соосными цилиндрическими каналами одинакового диаметра, соответствующего диаметру центрального монолитного цилиндрического стержня сваи.

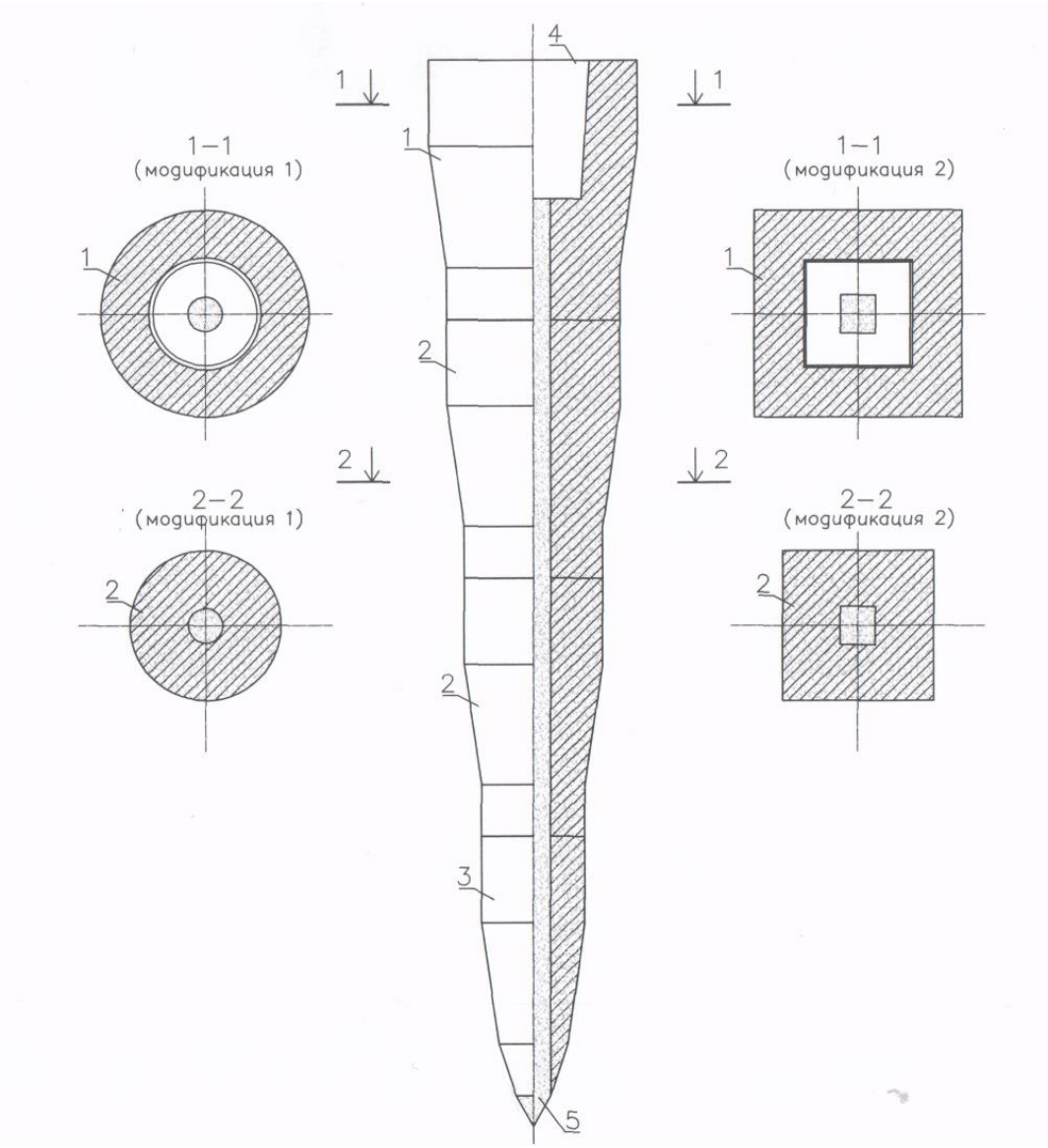
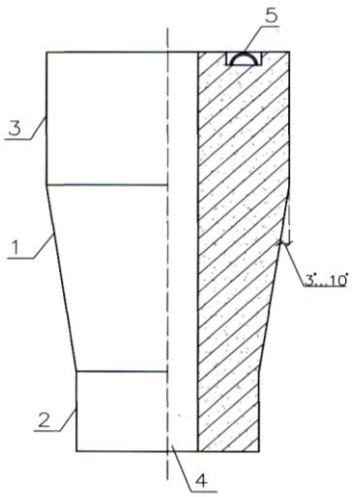


Рис. 1. Сборно-монолитная составная свая:
1 – головной элемент; 2 – промежуточный элемент; 3 – лидирующий элемент; 4 – «стакан» для установки колонны; 5 – центральный монолитный цилиндрический стержень.

Различия продольных элементов заключаются в следующем:

– концевая цилиндрическая часть лидирующего элемента, имеющая меньший диаметр, выполнена заостренной;
– в концевой цилиндрической части головного элемента, имеющей большой диаметр, может быть выполнен «стакан» для установки колонны;
– диаметры цилиндрических частей и оснований конусообразных частей каждого продольного элемента последовательно возрастают по направлению от лидирующего к головному элементу сваи. В то же время, соприкасающиеся между собой цилиндрические части соседних продольных элементов имеют одинаковые диаметры;
– высоты цилиндрических частей всех продольных элементов сваи одинаковые, а высоты конусообразных частей этих элементов последовательно уменьшаются по направлению от лидирующего к головному элементу сваи.

Рис. 2. Промежуточный сборный железобетонный элемент:
1 – фрагмент элемента в виде усеченного конуса; 2, 3 – то же, в виде цилиндров;
4 – цилиндрический канал;
5 – монтажная петля



При проектировании составной сваи предлагаемого конструктивного исполнения, изменением высоты конусообразной части любого составляющего тело сваи продольного элемента возможно регулирование его объема, что позволяет все продольные элементы сваи выполнять с одинаковым весом. Эта возможность, в свою очередь, обеспечивает получение явных преимуществ при выполнении процесса погружения свай в грунтовое основание, которых нет у типовых сборных железобетонных свай заводского изготовления.

Возведение свайного фундамента предлагаемого конструктивного исполнения производится в технологической последовательности, указанной на рис.3. Первоначально по месту погружения сваи в грунтовом основании пробуривается лидирующая скважина, диаметр которой соответствует диаметру заостренной концевой цилиндрической части лидирующего продольного элемента, а глубина на 0,5 м меньше суммарной высоты всех продольных элементов сваи. Сразу после пробуривания полость скважины заполняется тиксотропной грунтоукрепляющей композицией, например, водоцементной суспензией, которая предотвращает обрушение её стенок. При этом, под действием гидростатического давления, суспензия постепенно проникает в грунт боковых стенок и дна скважины, насыщая его и заполняя воздушные поры между грунтовыми частицами. Затем в скважину на глубину, равную высоте сваи, погружают полый перфорированный стержень, представляющий собой металлическую трубу с конусным наконечником, наружный диаметр которой несколько меньше диаметра цилиндрических каналов продольных элементов сваи (рис.4).

Длина перфорированного стержня подбирается таким образом, чтобы верхний конец погруженного в скважину стержня возвышался над поверхностью земли на высоту, превышающую высоту лидирующего продольного элемента сваи, который насаживается на стержень. После этого производится погружение забивкой лидирующего продольного элемента сваи в скважину, при достижении дна которой погружение элемента продолжается уже непосредственно в грунтовое основание до заданной проектной отметки. Погружение лидирующего и всех последующих продольных элементов осуществляется с помощью погружающего приспособления (рис.5), насаживаемого на полый перфорированный стержень сверху погружаемого элемента и передающего ему ударный импульс от свайного молота.

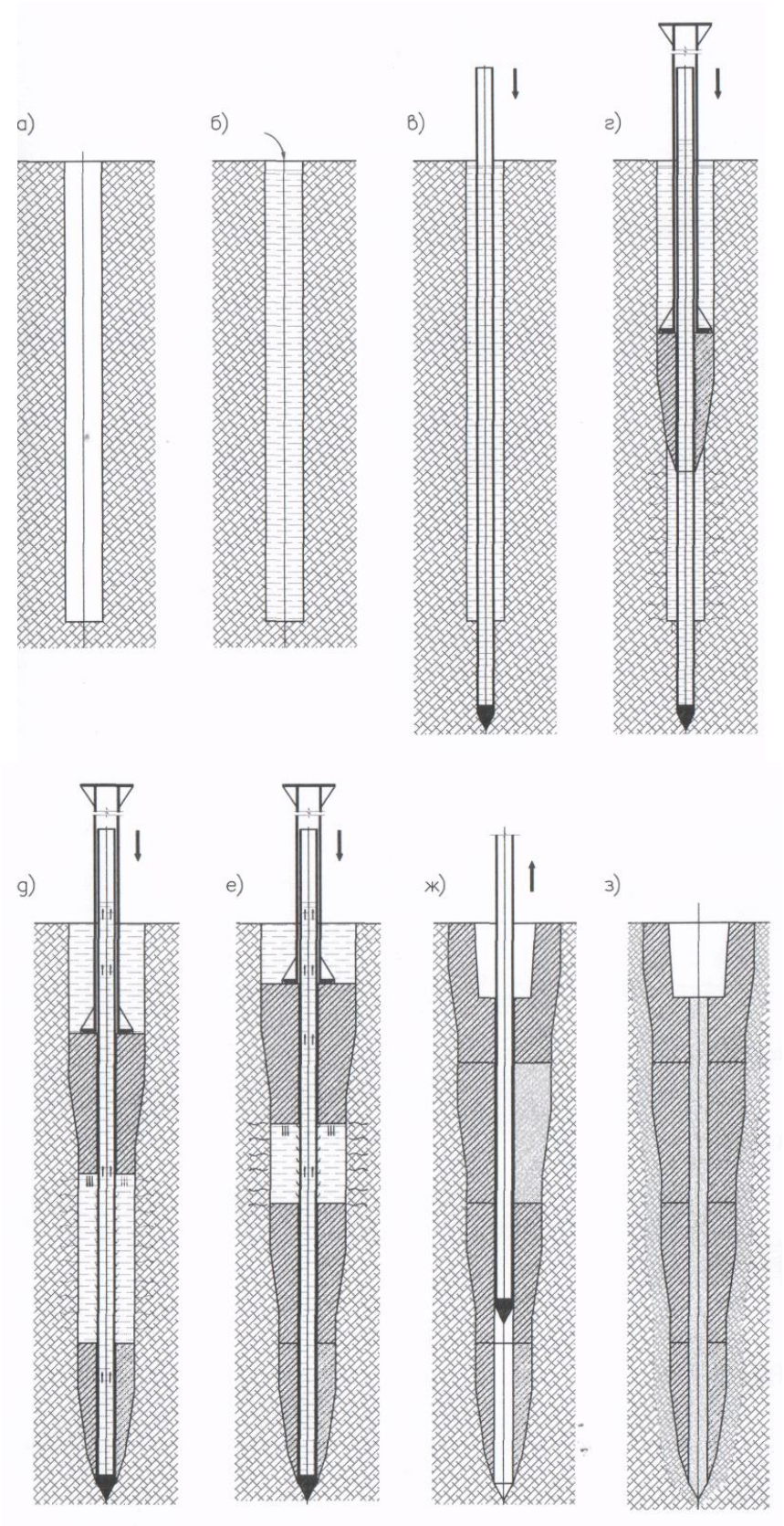


Рис. 3. Технологическая последовательность изготовления сваи:
а – бурение лидирующей скважины; б – заполнение скважины грунтоукрепляющей композицией; в – погружение в скважину полого перфорированного стержня; г – погружение свайного лидирующего элемента; д, е – то же, свайных промежуточных элементов; ж – извлечение полого перфорированного стержня; з – изготовление железобетонного монолитного стержня сваи

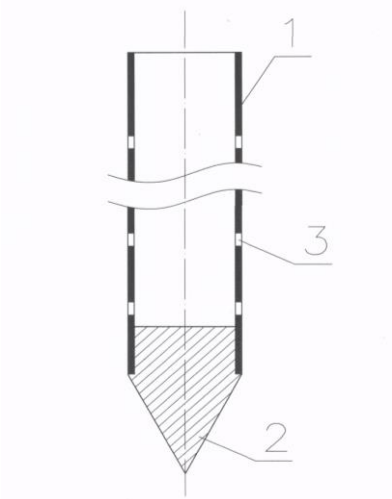


Рис. 4. Полый перфорированный стержень:
1 – стенка стержня;
2 – конусообразный наконечник;
3 – отверстия в стенке стержня

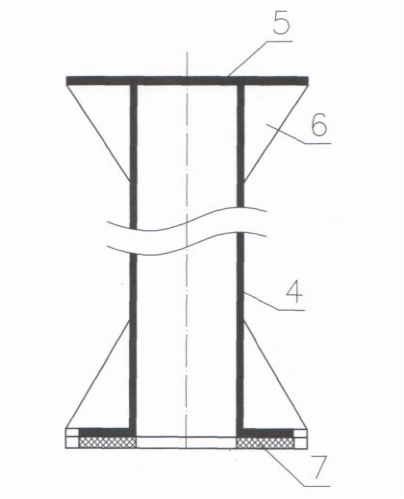


Рис. 5. Погружающее приспособление:
4 – металлическая труба; 5 – верхний фланец; 6 – усиливающие ребра;
7 – нижний фланец с амортизирующей прокладкой

В процессе погружения лидирующего продольного элемента сваи часть энергии ударных импульсов передается тиксотропной композиции, заполняющей нижерасположенное пространство скважины, создавая в ней повышенное давление (гидравлический удар). Это способствует ускоренному проникновению композиции в грунтовые стенки и дно скважины, увеличивая во всех направлениях толщину пропитанного ею слоя грунта. Вместе с тем, вследствие небольшой проникающей способности грунта, вся тиксотропная композиция, находящаяся под давлением, не успевает перейти в грунт. Часть её через сквозные отверстия в стенках полого стержня попадает в его полость, заполняет её, а затем вытекает через отверстия, расположенные выше погружаемого свайного элемента, в пространство скважины над ним.

После достижения лидирующим продольным элементом проектной отметки, процесс погружения прекращается, погружающее приспособление из скважины извлекается, на перфорированный стержень насаживается следующий по порядку промежуточный продольный элемент и цикл погружения повторяется. Так продолжается до тех пор, пока все продольные элементы не займут свои места в свайной конструкции. Затем перфорированный стержень из осевого цилиндрического канала в теле сваи извлекается, а в него опускается арматурный каркас, после установки которого полость канала заполняется мелкозернистым бетоном. Изготовление сваи можно считать практически законченным, когда пропитанная композицией грунтовая масса вокруг тела сваи затвердеет, а уложенный бетон наберет заданную прочность.

Предлагаемая сборно-монолитная составная свая по сравнению с типовой призматической железобетонной сваей заводского изготовления при одинаковой массе конструкций обладает повышенной несущей способностью. Это достигается за счет:

– высокой степени уплотнения грунта по боковым поверхностям сваи при неоднократном прохождении погружаемых продольных свайных элементов с постоянно возрастающими диаметрами стволов;

– большой площади наклонных поверхностей конусообразных частей продольных элементов, составляющих тело сваи;
– пропитки тиксотропной грунтоукрепляющей композицией грунтового пространства вокруг тела сваи, которое, после его затвердения, образует единый прочный монолит со свайной конструкцией.

Кроме этого, новое конструктивное решение предлагаемой сваи обеспечивает получение многих дополнительных преимуществ перед типовой призматической железобетонной сваей. Например, по сравнению с трудоемким процессом изготовления на заводе ЖБИ длинномерной типовой сваи, выполняемой, как правило, с предварительно напрягаемой арматурой, процесс изготовления без предварительного напряжения коротких и легких сборных элементов составной сваи, технологически является гораздо более простым.

При проведении работ по погрузке и разгрузке свай, в теле перемещаемой в горизонтальном положении длинномерной типовой железобетонной сваи возникают ненужные изгибающие и растягивающие напряжения, которые могут стать причиной её поломки даже при выполнении предварительного напряжения арматуры. В аналогичных условиях перемещение сборных элементов составной сваи технологически предусмотрено только в вертикальном положении, что, в сочетании с небольшими размерами и массой элементов, будет являться гарантией их целостности.

Погружение в грунтовое основание длинномерных массивных типовых железобетонных свай возможно только при использовании свайных молотов, обладающих большой энергией удара. Для преодоления возрастающих в процессе забивки сваи сил сопротивления погружению, необходимо постоянно наращивать величину ударного усилия, что сопровождается увеличением сжимающих напряжений в головной части сваи. Зачастую величины напряжений превышают допустимые значения и это приводит к разрушению бетона головных частей забиваемых свай и их недопогружению до проектной отметки. В таких случаях, как правило, рядом с поврежденной сваей приходится дополнительно забивать дублирующую сваю, что в конечном итоге увеличивает стоимость и продолжительность строительства.

В тоже время, при последовательном погружении небольших легких свайных элементов в заранее пробуренную в грунтовом основании лидирующую скважину, диаметр которой незначительно меньше диаметра погружаемого элемента, а боковые стенки пропитаны жидкой тиксотропной композицией, выполняющей роль смазки, требуется минимальная энергия. В этом случае значения сжимающих напряжений в бетоне конструкций не достигают приводящих к разрушению критических величин и повреждения погружаемых элементов не происходит.

И наконец, конструктивное исполнение предлагаемой составной сваи позволяет в достаточно широком диапазоне регулировать показатели её несущей способности простым изменением количества одинаковых по массе погружаемых продольных элементов. При этом, выполнение различных по своим размерам, массе и несущей способности свайных конструкций возможно производить по приведенной раннее технологии с использованием любых свайных молотов, технические параметры которых достаточны для погружения отдельного продольного элемента составной сваи.

Вывод: по сравнению с типовой призматической сборной железобетонной сваей предлагаемая составная свая при одинаковой массе конструкций обладает следующими преимуществами:

– на стадии изготовления конструктивных элементов – большей технологичностью, меньшим процентом армирования, отсутствием необходимости выполнения предварительного напряжения арматуры;
– на стадии возведения фундамента – возможностью выполнения работ по изготовлению свайных конструкций больших размеров и масс с использованием маломощного сваебойного оборудования;
– на стадии эксплуатации сооружения – обеспечение повышенной несущей способности свайного фундамента.

Библиографический список

1. ГОСТ 19804.2-79*. Сваи забивные железобетонные цельные сплошного квадратного сечения с поперечным армированием ствола с напрягаемой арматурой

НАШИ АВТОРЫ

Алексеев О.Г.
Заместитель директора управления информационных технологий ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Веселов А.В.
Доцент кафедры строительного производства и автомобильных дорог архитектурно-строительного факультета, кандидат технических наук

Гаврилкович В.И.
ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Макшанов Е.О.
Инженер управления информационных технологий ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Пермяков М.Б.
Декан архитектурно-строительного факультета, кандидат технических наук

Пунгин В.Л.
Директор научно-исследовательского архитектурно-проектного предприятия «НИАП», кандидат архитектуры, доцент

Пунгина Т.В.
Старший преподаватель кафедры архитектуры и графики Курганской государственной сельскохозяйственной академии им. Т.С. Мальцева

Сорокин М.О.
Аспирант ФГБОУ ВПО «Московский государственный строительный университет», г.Москва

Стефанишин Д.В.
Институт телекоммуникаций и глобального информационного пространства Национальной академии наук Украины, ведущий научный сотрудник, доктор технических наук

Ткаченко М.С.
Инженер управления информационных технологий ООО «ВЕЛД», г.Магнитогорск, Челябинская область

Токарев А.А.
Аспирант кафедры строительного производства и автомобильных дорог архитектурно-строительного факультета

Шишкина Н.А.
Инженер управления информационных технологий ООО «ВЕЛД», г.Магнитогорск, Челябинская область