УДК 004.415.2
DOI 10.22213/2413-1172-2017-2-143-149

В. Г. Тарасов, кандидат технических наук, профессор, ИжГТУ имени М. Т. Калашникова
А. О. Трифонов, аспирант, ИжГТУ имени М. Т. Калашникова

МЕТОДИКА ДЕКЛАРАТИВНОГО ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ: АНАЛИЗ ТРЕБОВАНИЙ К СИСТЕМЕ

Введение

Широкое распространение объектно ориентированных языков повлекло появление новых подходов к анализу и проектированию программного обеспечения. В период с 1989 по 1994 г. число методов проектирования возросло с десяти более чем до пятидесяти. Тем не менее проблема выбора метода и его соответствие требованиям конкретной задачи привели к появлению нового поколения методов проектирования. Центральными из них стали метод Гради Буча, OOSE (Object-oriented soft ware engineering), разработанный Иваром Якобсоном, и OMT (Object modeling technique), разработанный Джеймсом Рамбо. Финальным результатом является язык UML (Unified Modeling Language), представляющий объединение перечисленных методов. В 1997 г. UML признан стандартом процесса проектирования программного обеспечения, применяемым в визуализации, спецификации, конструировании и документировании артефактов программных систем [1].

Вызовы современной индустрии информационных технологий

Постоянное возрастание сложности проектируемых систем приводит к усложнению процесса проектирования. Это логически сложная, трудоемкая задача, требующая высокой квалификации специалистов и глубокого понимания нотации и семантики UML [2]. Сегодня на первый план выходят следующие характеристики программных продуктов [3]:

- поддержка высокой скорости изменения бизнес-процессов;
- качество программного продукта;
- степень документированности (как с точки зрения пользователей, так и ИТ-специалистов и разработчиков);
- легкость сопровождения;
- возможность расширения функционала в соответствии с запросами пользователей.

Следующие факторы обостряют проблемы процесса проектирования программного обеспечения:

- быстрое развитие индустрии и усложнение проектируемых систем;
- инертность компаний и разрабатываемых продуктов;
- стремление получить больше отдачи от инвестиций, оптимизация бизнес-процесса разработки ПО;
- проблемы интеграции различных этапов проектирования.

Современные методы проектирования программного обеспечения

На сегодняшний день одним из наиболее популярных методов проектирования программного обеспечения является предметно ориентированный подход (Domain Driven Development), основной идеей которого является выделение релевантной модели предметной области и ее дистилляция [4]. Архитектура проектируемой системы основана на агрегатах, неделимых сущностях, модели предметной области. Данный подход представляет итеративную модель проектирования и разработки ПО, а также набор лучших практик, хорошо себя зарекомендовавших в проектах, которым присуща сложная организация модели предметной области. Однако данный метод имеет существенные недостатки [5]:

- сложность сопровождения;
- сложность внесения изменений;
- ресурсоемкий и экономически дорогой подход проектирования;
- высокий порог вхождения в роль проектировщика.

Современные методы проектирования также направлены на решение проблемы масштабируемости информационных систем, выделяемой в отдельный класс задач. Наиболее популярными решениями являются событийно ориентированные и микросервисные архитектуры [6].

Таким образом, существующие методы проектирования программного обеспечения не решают весь спектр актуальных проблем и являются набором лучших практик, предназначенных для узкого круга задач. Проектирование программного обеспечения – сложный, ресурсоемкий и дорогой вид деятельности, имеющий высокий порог вхождения в роль проектировщика. На сегодняшний день остро стоит вопрос о необходимости создания нового метода, учитывающего особенности и требования современных процессов проектирования.

Методика декларативного процесса проектирования

Представим процесс проектирования в виде последовательности этапов: определение требований к системе, их анализ, проектирование и разработка. Каждый этап включает в себя построение набора диаграмм, наглядно представляющих архитектуру проектируемой системы с определенной точки зрения. Диаграммы сопровождаются спецификациями, содержащими более подробную информацию. Стрелки, связывающие диаграммы между собой, образуют декларативный переход из одного состояния проектирования в другое. Декларативный переход решает схожую задачу с шаблонами проектирования в императивных языках; он представляет набор правил и решений задач, часто возникающих при проектировании архитектуры информационной системы. Именно эти правила декларативного перехода являются характерной чертой данной методики [7]. Стоит отметить, что существует возможность обратного перехода на предыдущий этап, так как итеративность является неотъемлемой характеристикой процесса проектирования [8]. Предлагаемая общая схема декларативного процесса проектирования представлена на рис. 1.

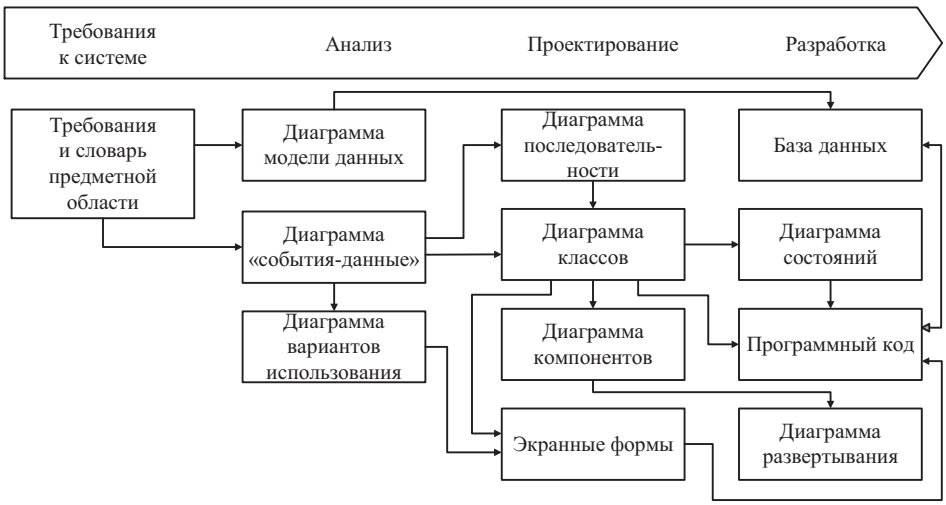

Рис. 1. Общая схема декларативного процесса проектирования

Рассматриваемые этапы решают следующие задачи:

• определение требований к системе:

– выделение процессов, происходящих в системе;

– создание словаря предметной области.

• анализ:

– определение модели предметной области, сущностей, их атрибутов и ассоциаций;

– описание процессов в формальном виде, определение входных и выходных данных и событий, описание основных и альтернативных сценариев выполнения;

– определение зависимостей между процессами.

• проектирование:

– определение архитектуры проектируемой системы;

– описание внутреннего устройства процессов и взаимодействия между ними;

– создание компонентов системы;

– выделение основных классов и их методов;

– определение макетов экранных форм.

• разработка:

– создание схемы базы данных;

– реализация спроектированных классов их методов;

– реализация методов развертывания и доставки системы.

Данная статья рассматривает этапы «Требования к системе» и «Анализ» на примере системы проведения онлайн-соревнований по спортивному программированию BACS [9]. Первый этап начинается с создания текста документа требований к системе (Product Requirements Document – PRD). Приведем пример такого документа.

Проектируемая система «**BACS**» предназначена для проведения онлайн соревнований по спортивному программированию. **Гость** должен пройти регистрацию, чтобы стать **участником**. Для доступа к сайту **участник** должен авторизоваться. В случае если **участник** забыл **пароль**, он может пройти процедуру восстановления пароля. **Участник** получает список активных **контестов**. Каждый **контест** состоит из списка **задач**. **Участник** регистрируется в контесте, предварительно выбрав его из списка активных контестов. После этого участник участвует в контесте, выбрав контест из списка моих контестов. Во время участия в контесте **участник** имеет возможность:

• получить список задач;

• отправлять решения на онлайн-проверку;

• посмотреть рейтинг участников. **Рейтинг участников** вычисляется при каждом обращении.

По результатам анализа документа требований к системе определяется словарь предметной области, включающий сущности модели предметной области и процессы, происходящие в проектируемой систе-

ме. Стоит отметить, что текст документа требований к системе может уточняться и обновляться как по мере уточнения словаря, так и во время процесса проектирования. Это естественный процесс, предполагающий итеративность на любом этапе.

Под сущностями понимаются типы объектов предметной области. Помимо типа сущность имеет список первостепенных атрибутов, фиксируемых в словаре. Существует два типа сущностей:

- активные – инициируют взаимодействие с другими объектами;
- пассивные – являются пассивными участниками взаимодействия сущностей.

Словарь рассматриваемой предметной области может иметь следующий вид.

Активные сущности:

- **Гость** – пользователь, не зарегистрированный в системе.
- **Участник** – пользователь, зарегистрированный в системе. Атрибуты: логин, пароль, ответ на секретный вопрос и др.

Пассивные сущности:

- **BACS** – проектируемая система.
- **Контест** – онлайн-соревнование по спортивному программированию. Имеет фиксированные сроки проведения. Атрибуты: название, дата начала, дата окончания.
- **Задача** – элемент контеста, состоящий из условия, ограничений, входных и выходных данных. Атрибуты: название, условия, автор, ограничения.
- **Посылка** – исходный код программы, решающей поставленную задачу на определенном языке программирования. Атрибуты: язык решения, текст решения, результат проверки, время посылки, номер задачи, отправитель.
- **Рейтинг участников** – табличный отчет, колонки которого содержат список задач, строки – список участников. Значение ячейки в виде знака «+» показывает, что участник решил задачу. При неудачной проверке решения отображается знак «–» и количество неудачных попыток. Отчет вычисляется на основе списка задач, списка участников и списка отправленных решений. Атрибуты: список задач, список посылок, список участников, контест.

Под процессами понимаются последовательности операций (действий, активностей), направленных на достижение определенной цели.

Процессы:

- *авторизовать участника* – операция, инициируемая участником, проверяющая его логин и пароль и допускающая к возможностям системы;
- *зарегистрировать участника* – операция, инициируемая гостем, создающая учетную запись участника в системе;
- *восстановить пароль участника* – операция, инициируемая участником, восстанавливающая забытый пароль;
- *получить список активных контестов* – операция, инициируемая участником, запрашивающая список активных контестов;
- *получить список моих контестов* – операция, инициируемая участником, запрашивающая список контестов, в которых он зарегистрирован;
- *зарегистрировать участника в контесте* – операция, инициируемая участником, регистрирующая его в контесте;
- *выбрать контест* – операция, инициируемая участником, выбирающая контест для участия;
- *участвовать в контесте* – операция, инициируемая участником, предоставляющая доступ к контесту;
- *получить список задач* – операция, инициируемая участником, запрашивающая список задач;
- *отправить решение* – операция, инициируемая участником, передающая решение на онлайн-проверку и возвращающая результат проверки;
- *просмотр рейтинга участников* – операция, инициируемая участником, запрашивающая рейтинг участников.

Название процесса должно кратко характеризовать совершаемое действие, а именно: название должно быть глагольным словосочетанием – глагол, который связан с именем существительным в косвенном падеже, иллюстрирующим такой характер подчинительной связи, как управление [10].

Определенные требования к системе и словарь предметной области являются результатом первого этапа «Требования к системе». Следующий этап «Анализ» направлен на определение модели предметной области и описание процессов и их зависимостей.

Анализ модели предметной области направлен на определение атрибутов сущностей и отношений. На рис. 2 изображена диаграмма модели данных рассматриваемой предметной области. На диаграмме изображены сущности и их атрибуты, определенные в словаре предметной области. Важным параметром является отношение между сущностями, определяемое исходя из контекста предметной области, который не всегда может быть подробно описан в требованиях к системе. Для этого проектировщику необходимо изучить предметную область решаемой задачи и выявить возможные и достаточные отношения сущностей [11]. Модель данных предметной области является первым формализованным результатом этапа «Анализ». Определенные сущности, их атрибуты и отношения являются основной архитектуры проектируемой информационной системы.

На этом моменте завершается анализ и проектирование модели данных; дальнейшие действия направлены на анализ выделенных процессов.

Преобразуем требования к системе из произвольного текста в более формальный вид записи. Для этого предлагается табличный вид записи и эквивалентное графическое описание «события-данные». Такой вид записи уточняет информацию о взаимодействии процессов, рассматривая их внешние характеристики. В отличие от формата записи вариантов использования или пользовательских историй [12] такой формат позволяет опреде-

лить процесс как пересечение потока событий и потока данных, рассматривая события и данные вместе, а не раздельно.

Таблица содержит описание процесса, содержащего определение действующего лица, основного потока – успешного сценария выполнения, и возможных альтернативных потоков – исключительных сценариев, определяющих нестандартное выполнение процесса. Описание альтернативных потоков является важным этапом анализа требований, определяющим поведение системы в исключительных ситуациях. Любой процесс, помимо основного потока, должен иметь хотя бы один альтернативный поток.

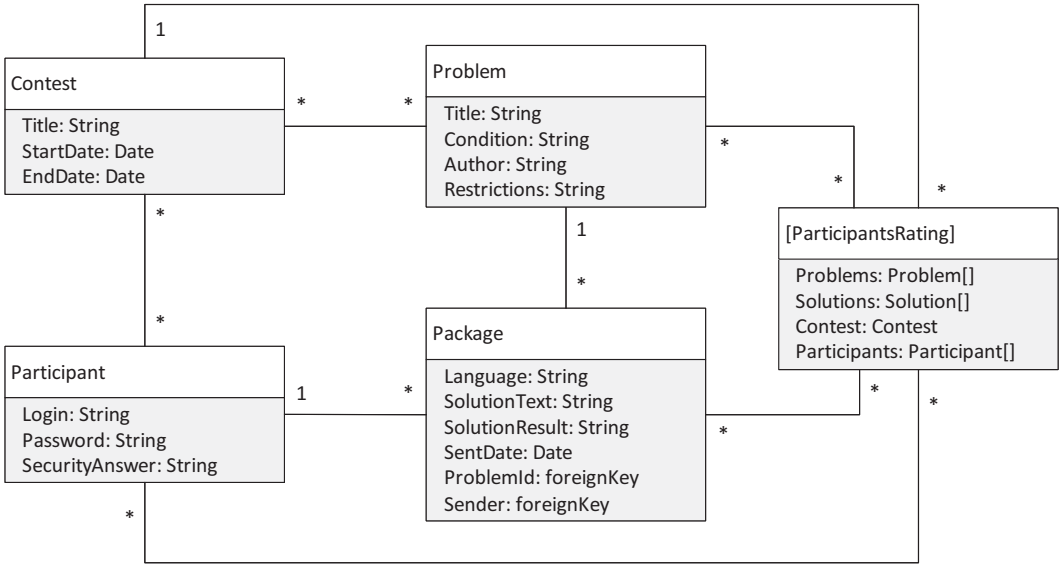

Рис. 2. Диаграмма модели данных

Основной поток содержит следующие атрибуты:

- название – идентификатор потока;
- инициатор – объект, инициирующий начало выполнения процесса; существуют внешние (визуальные) и внутренние (невизуальные) инициаторы;
- условия – необходимые требования для выполнения процесса;
- входные данные – данные, передаваемые на вход процессу;
- выходные данные – данные, генерируемые процессом после выполнения;
- выходные события – события, генерируемые процессом после выполнения.

Альтернативный поток содержит следующие атрибуты:

- название – идентификатор альтернативного потока;
- инициатор – объект, инициирующий начало выполнения процесса;
- процесс – инициируемый процесс.

Еще одним новым элементом предлагаемого формата записи процессов является концепция контейнера. Контейнер – объект, представляющий определенный формат хранения, отображения и обработки данных. Предполагается, что все инициаторы, события и данные могут быть описаны соответствующим типом контейнера. Таким образом, все атрибуты процесса описываются и используются единообразным способом.

В табличной записи тип контейнера выделен курсивом. Выделим следующие типы контейнеров:

- визуальные (пользовательские) – графические компоненты, взаимодействующие с пользователем;
- невизуальные (системные) – любые хранилища данных.

Пример табличного описания процессов изображен в табл. 1 и 2.

Таблица 1. **Процесс «Авторизовать участника»**

Атрибут	Значение
Действующее лицо	**Участник**
Основной поток	
Название	Авторизовать участника
Инициатор	Начать процесс авторизации – *визуальный инициатор*
Условия	Участник зарегистрирован (*сущность* Participant – определена)
Входные данные	• логин – *строка*; • пароль – *строка*
Выходные данные	Участник авторизован – *переменная сессии*
Альтернативный поток 1	
Название	Неверный логин/пароль
Инициатор	*Визуальный инициатор*
Процесс	Восстановить пароль участника, Авторизовать участника
Альтернативный поток 2	
Название	Ошибка процесса авторизации
Инициатор	*Невизуальный инициатор*
Процесс	Сообщить об ошибке

Процесс «Авторизовать участника» предназначен для действующего лица **Участник**. Основной поток

инициализируется визуальным инициатором, т. е. элементом пользовательского интерфейса. Для начала процесса требуется выполнение условия «участник зарегистрирован», т. е. система должна содержать учетную запись участника. Входными данными процесса являются логин и пароль, на выходе процесс генерирует переменную сессии «участник авторизован». Таблица описывает альтернативный поток «неверный логин/пароль», выполняемый в случае неудачной попытки авторизации пользователя. Первый альтернативный поток имеет 2 ссылки на процессы «Восстановить пароль участника» и «Авторизовать участника». Это значит, что при выполнении альтернативного потока пользователю будет предложен выбор соответствующего действия. Второй альтернативный поток является типичным для всех процессов, в случае ошибки пользователь перенаправляется на процесс «Сообщить об ошибке», который показывает соответствующее сообщение пользователю, например, внутренняя ошибка сервера или временная недоступность ресурса.

Таблица 2. **Процесс «Получить список активных контестов»**

Атрибут	Значение
Действующее лицо	**Участник**
Основной поток	
Название	Получить список активных контестов
Инициатор	Начать процесс получения списка активных контестов – *визуальный инициатор*
Условия	Участник авторизован (*переменная сессии* участник авторизован – определена)
Входные данные	
Выходные данные	Список активных контестов – *таблица* (*название, дата начала, дата окончания*)
Альтернативный поток 1	
Название	Ошибка процесса получения списка активных контестов
Инициатор	*Невизуальный инициатор*
Процесс	Сообщить об ошибке

Процесс «Получить список активных контестов» предназначен для действующего лица **Участник**. Основной поток инициализируется визуальным инициатором. Для начала процесса требуется выполнение условия «участник авторизован», т. е. должна существовать соответствующая переменная сессии. Процесс не имеет входных данных, а на выходе возвращает таблицу активных контестов. Процесс также имеет альтернативный поток, инициирующий процесс «Сообщить об ошибке».

Представим графический вид записи описания процессов, соответствующий табличному варианту записи. Графический формат, представленный на рис. 3, 4, необходим для более наглядного представления табличной информации. Процесс, обозначенный прямоугольником, является черным ящиком, вбирающим горизонтальный поток данных и вертикальный поток событий. Данная схема имеет следующие терминаторы – начальные и конечные точки выполнения процесса:

- C – условия;
- V – визуальный инициатор;
- S – невизуальный инициатор;
- X – успешное завершение;
- E – успешное завершение и вызов другого процесса;
- A – альтернативное завершение и вызов другого процесса.

Важнейшим результатом является установление связей между процессами путем выявления зависимостей на основе значений свойств «условия», «входные данные», «выходные данные» в основных потоках и «процесс» в альтернативных потоках диаграмм «события-данные».

Воспользуемся для этого диаграммой вариантов использования, представленной на рис. 5. Такая зависимость процессов является руководством для организации пользовательского интерфейса и связи экранных форм между собой.

Диаграмма вариантов использования строится по следующим правилам:

- каждому типу активной сущности соответствует свой актер на диаграмме;
- всем заявленным процессам соответствует свой вариант использования;
- актер имеет связи (ассоциации) – соединения с корневыми вариантами использования, не имеющими зависимостей от других процессов;
- если процесс A зависит от выходных данных или события B, то вариант использования A имеет связь include с вариантом использования B;
- если процесс A вызывает процесс C в альтернативном потоке, то вариант использования A имеет связь extend с вариантом использования C;
- если процесс A вызывает процесс D в основном потоке, например, в виде контейнера «ссылка», то вариант использования A имеет связь generalization с вариантом использования D.

Каждый вариант использования обозначается овалом и имеет связи включения, расширения и обобщения с другими вариантами использования. Стрелки обозначают тип зависимости между вариантами использования, include – обязательное выполнение включаемого варианта использования, extend – дополнение расширяемого варианта использования, generalization – обобщение частных вариантов использования.

На этапе «Требования к системе» был определен словарь предметной области, описывающий типы сущностей и процессы, происходящие в системе. Результатом этапа «Анализ» является модель предметной области, содержащая атрибуты и отношения сущностей, формальное определение внешних характеристик процессов, описанное в табличном виде и в виде диаграмм «события-данные». Кроме того, результатом анализа является определение зависимости процессов друг от друга и ее графическое представление в виде диаграммы вариантов использования. Полученные описания и диаграммы являются основой этапа «Проектирование».

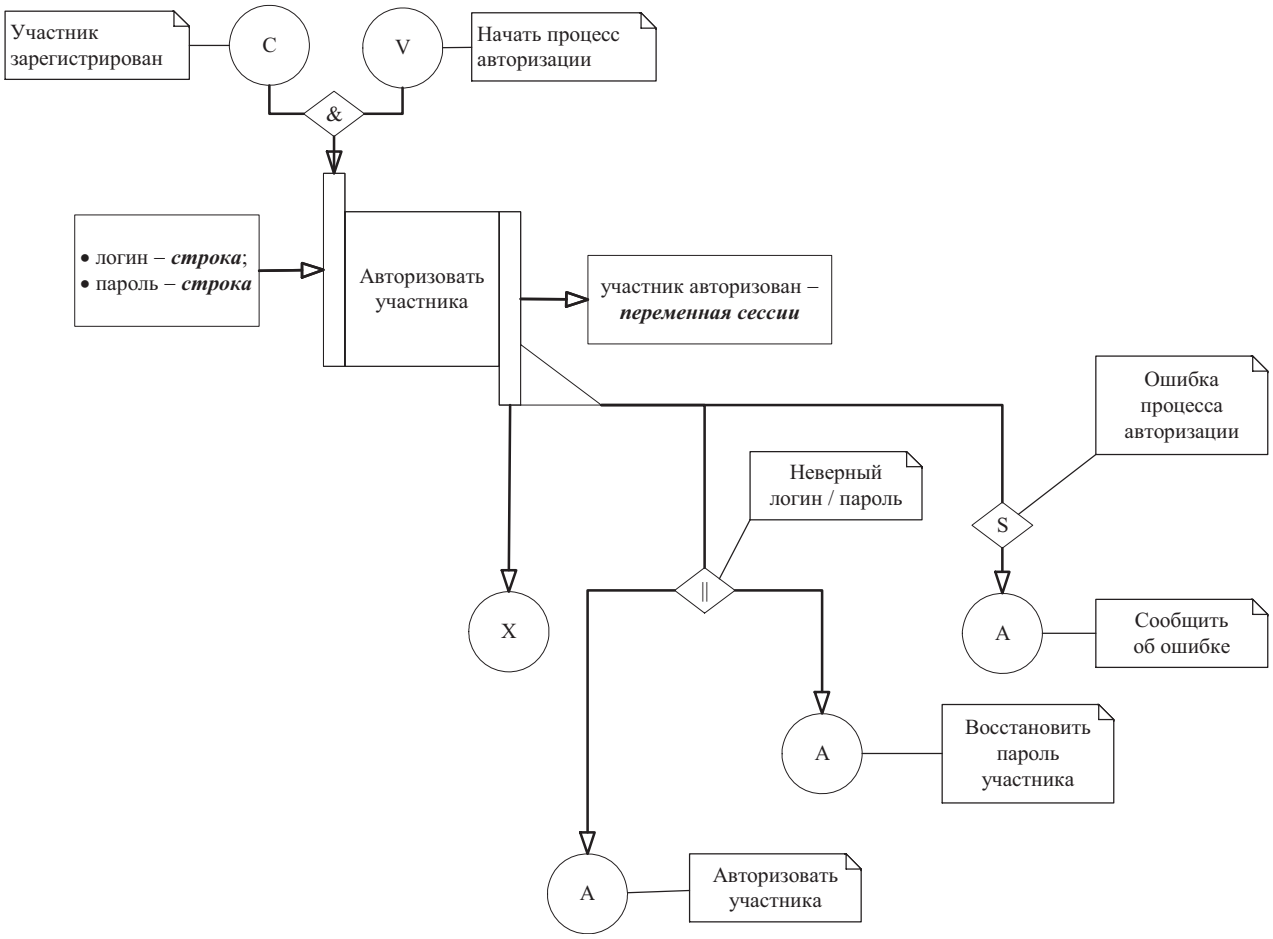

Рис. 3. Диаграмма процесса «Авторизовать участника»

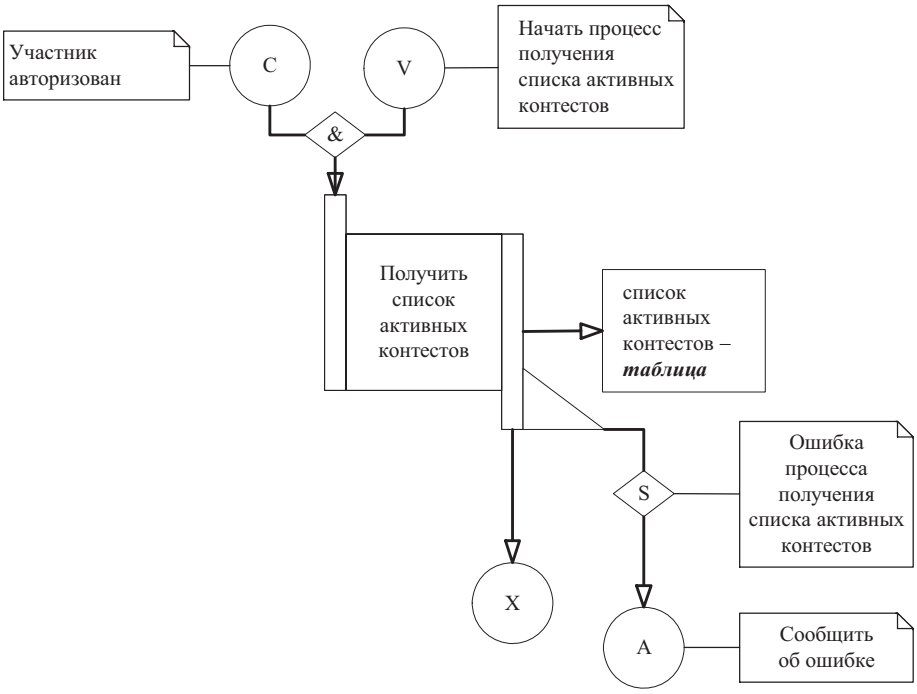

Рис. 4. Диаграмма процесса «Получить список активных контестов»

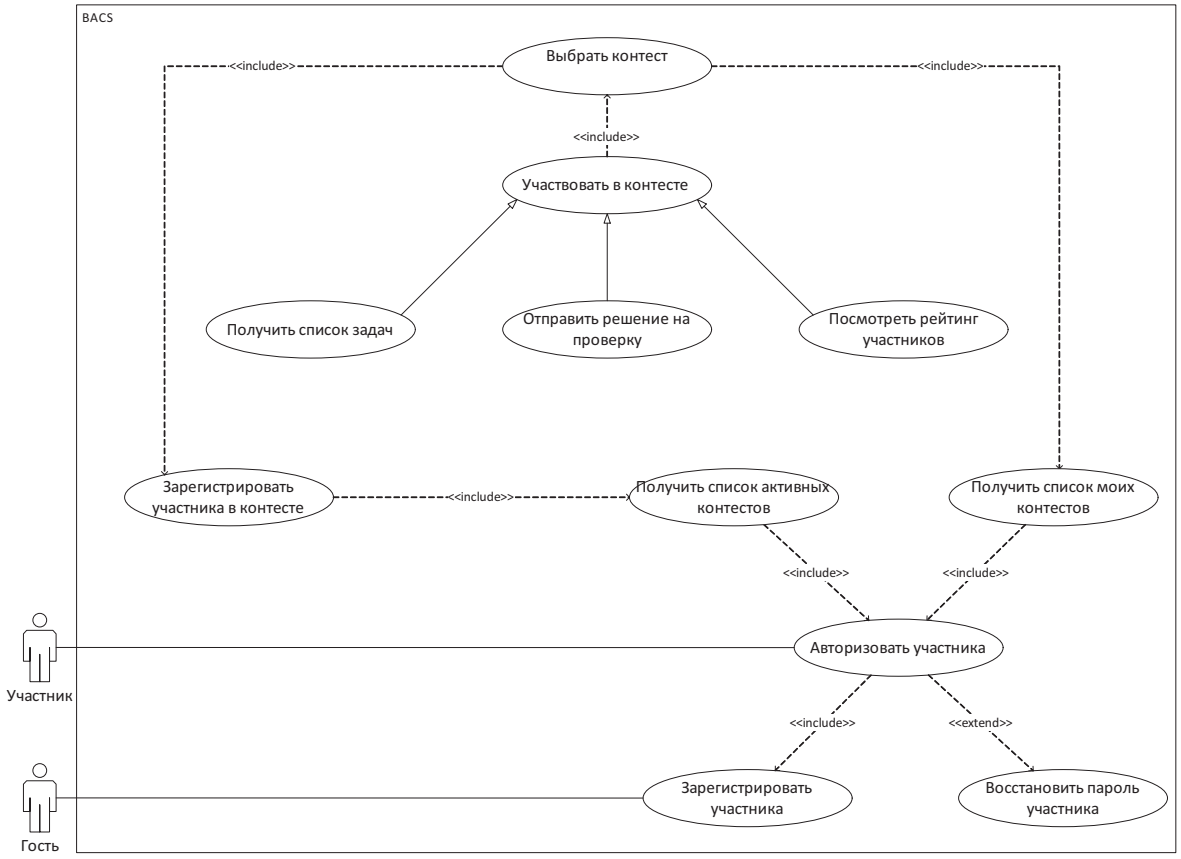

Рис. 5. Диаграмма вариантов использования BACS

Заключение

Рассмотрены актуальные проблемы современного процесса проектирования программного обеспечения. Представлена методика декларативного проектирования. Данная методика делает акцент на формальных правилах декларативного перехода между различными состояниями процесса проектирования. Рассмотрен пример определения требований к проектируемой системе и их анализа с использованием предлагаемой методики.

Использование данной методики способно снизить порог вхождения новых специалистов в роль проектировщика информационных систем, а также уменьшить время, затрачиваемое проектировщиком на понимание и изучение архитектуры существующих информационных систем.

Формальные правила декларативного перехода могут быть автоматизированы в рамках CASE-средства, реализующего данную методику, применение которого может существенно снизить затраты на процессы проектирования, поддержки и сопровождения.

Библиографические ссылки

1. *Буч Г.*, *Рамбо Д.*, *Якобсон А.* Язык UML. Руководство пользователя : пер. с англ. – М. : ДМК, 2001. – 432 с.

2. *Trifonov A. O.*, *Tarasov V. G.* Software design issues and declarative approach // Молодые ученые – ускорению научно-технического прогресса в XXI веке : [Электронное научное издание] : сборник тр. IV Всерос. науч.-техн. конф. аспирантов, магистрантов и молодых ученых с междунар. участием (Ижевск, 20-21 апреля 2016 года) / ИжГТУ имени М. Т. Калашникова / ИжГТУ имени М. Т. Калашникова, 2016. – 1044 с. (33 Мб).

3. *Иван Пономарев*. Case-средства: в борьбе со сложностью мира [Электронный ресурс] // PCWeek. – 2004. – № 18. – URL: https://www.pcweek.ru/idea/article/detail.php?ID=67597 (дата обращения: 11.10.2016).

4. *Эванс Эрик*. Предметно ориентированное проектирование (DDD): структуризация сложных программных систем : пер. с англ. – М. : И. Д. Вильямс, 2011. – 448 с.

5. Там же.

6. *Stephen Ferg*. Event-Driven programming: Introduction, Tutorial, History (2006-02-08) [Электронный ресурс]. – URL: http://Tutorial_EventDrivenProgramming.sourceforge.net (дата обращения: 07.02.2017).

7. *Trifonov A. O.*, *Tarasov V. G.* Указ. соч.

8. *Frederick P. Brooks*. The design of design: essays from a computer scientist. – Reading, MA : Addison-Wesley, 2010. – 437 с.

9. *Гатин Р. М.*, *Тарасов В. Г.*, *Юминов А. В.* Система автоматизированной проверки задач для школьных олимпиад. Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н. Н. Войта. – Ульяновск :УлГТУ, 2011.– 416 с.

10. *Ладыженская Т. А.*, *Баранов М. Т.*, *Тростенцова Л. А.* Русский язык : учебник для 5-го класса общеобразовательных школ. – Ч. 1. – М. : Просвещение, 2012. – 192 с.

11. *Эванс Эрик*. Указ. соч.

12. *Алистер Коберн*. Современные методы описания функциональных требований к системам : пер. с англ. – М. : Лори, 2002. – 288 с.

Получено 24.04.2017