УДК 621.397:004.738

П.И. Соснин, С.С. Шумилов, А.Е. Ивасев

АРХИТЕКТУРНЫЙ ПОДХОД К ПРЕЦЕДЕНТНО-ОРИЕНТИРОВАННОМУ РЕШЕНИЮ ЗАДАЧ В РАЗРАБОТКЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ[1]

Соснин Петр Иванович, *доктор технических наук, профессор, окончил радиотехнический факультет Ульяновского политехнического института. Заведующий кафедрой «Вычислительная техника» Ульяновского государственного технического университета. Область научных интересов – прикладные интеллектуальные системы. [e-mail: sosnin@ulstu.ru].*

Шумилов Сергей Сергеевич, *окончил радиотехнический факультет УлГТУ, главный конструктор ФНПЦ АО «НПО «Марс». Область научных интересов – системы автоматизированного проектирования. [e-mail: mars@mv.ru].*

Ивасев Александр Евгеньевич, *окончил факультет информационных систем и технологий УлГТУ, аспирант УлГТУ. Область научных интересов – системы автоматизированного проектирования. [e-mail: nevskei@yandex.ru].*

Аннотация

В статье представлен подход к архитектурному моделированию, ориентированный на его применение в разработке функциональных составляющих автоматизированной системы (АС), с каждой из которых связано решение определённой проектной задачи. Для обеспечения независимости применений подхода от предметной области АС его реализация связана с созданием подсистемы «Задача», расширяющей потенциал инструментально-моделирующей среды OwnWIQA, обслуживающей решение проектных задач на рабочем месте члена группы проектировщиков АС. К специфике инструментария относится формирование по ходу решения задачи её модели повторного использования (модели прецедента), которая включается в Базу Опыта, используемую в процессе проектирования определённой АС или их семейства. Подсистема «Задача» разработана в среде OwnWIQA как её приложение с использованием встроенного в этот инструментарий исполняемого псевдокодового языка. Построенная версия подсистемы «Задача» интерпретируется авторами как прототип, перепрограммирование которого будет проведено на языке C#.

Ключевые слова: архитектурное моделирование, модель прецедента, проектная задача, прототип.

AN ARCHITECTURAL APPROACH TO THE PRECEDENT-ORIENTED PROBLEM SOLVING IN THE DESIGNING OF AUTOMATED SYSTEMS

Petr Ivanovich Sosnin, *Doctor of Science in Engineering, Professor; graduated from the Faculty of Radioengineering of Ulyanovsk Polytechnic Institute; Head of the Department of Computer Engineering at Ulyanovsk State Technical University; his scientific interests are in the field of applied intelligent systems. e-mail: sosnin@ulstu.ru.*

1 Работа выполнена при поддержке РФФИ (гранты № 18-07-00989а, № 18-47-730016р-а, № 18-47-732012 р_мк и Министерства науки и высшего образования РФ, Госзадание № 2.1534.2017/4.6).

Sergei Sergeevich Shumilov, *graduated from the Faculty of Radioengineering of Ulyanovsk State Technical University; Chief Designer at FRPC JSC 'RPA 'Mars'; his scientific interests are in the field of computer-aided design systems. e-mail: mars@mv.ru.*

Aleksandr Evgenievich Ivasev, *graduated from the Faculty of Information Systems and Technologies of UlSTU; Postgraduate Student at UlSTU; his scientific interests are in the field of computer-aided design systems. e-mail: nevskei@yandex.ru.*

Abstract

The article presents an approach to architectural modeling focused on its use for the development of functional components of an automated system (AS), each of which is associated with the solution of a specific design task. To ensure the independence of the application of the approach from the domain of the AS, its implementation is associated with the creation of the subsystem TASK, which expands the potential of the instrumentally modeling environment OwnWIQA, which serves to solve design tasks in the workplace of a member of the design team. The specificity of the toolkit includes the formation for the solved task its reusable model (model of precedent), which is included in the Experience Base used in the design process of a certain AS or their family. The subsystem TASK was developed in the OwnWIQA as its application using the executable pseudo code language embedded into this toolkit. The constructed version of the subsystem TASK is interpreted by the authors as a prototype, the reprogramming of which will be conducted in C #.

Key words: architectural modeling, precedent model, design task, prototype.

Введение

Профессионально зрелая разработка современной автоматизированной системы (АС) немыслима без обязательного построения и оперативного использования её архитектурного представления (Architecture Description, AD), которое считается «принципиальной» версией АС, востребованной на всех этапах её жизненного цикла. Что особенно важно, эта версия является первым (самым ранним) представлением АС, которое может быть проверено (испытано) как целостная система. Более того, в этой версии АС представлены самые существенные требования и взаимоотношения, гарантирующие, что интересы (concerns, $\{C_i\}$) заинтересованных сторон (stakeholders, стейкхолдеров) в проекте АС учтены, причём представлены в формах, способствующих конструктивному пониманию их стейкхолдерами.

Во всех последующих версиях детализации АС требования, интегрированные в AD, должны соблюдаться в обязательном порядке, сохраняя тем самым вложенную в AD целостность, которая может изменяться по ходу разработки АС, но только если для этого имеются очень серьезные основания. Тем самым, артефакт AD в его текущем состоянии фиксирует те взаимоувязанные высокоуровневые требования и соответствующие им решения, вносить изменения в которые на нижних уровнях детализации запрещено, поскольку возможные изменения в AD являются слишком дорогостоящими.

Высокая значимость артефактов типа AD нашла своё выражение в нормативах, интегрирующих опыт архитектурного моделирования систем с программным обеспечением. Среди таких нормативов особое место занимает стандарт IEEE 1497:2000 и его расширение ISO/IEC/IEEE 42010:2011 (русифицированная версия которого введена для применения в 2016 году как ГОСТ Р 57100). В основу стандартов положено формирование системы $S(\{V_j\})$ «архитектурных видов» (Architecture Views, $\{V_j\}$), исходя из «точек зрения» (Viewpoints, $\{VP_k\}$), оговоренных в техническом задании на проект. Именно описание системы $S(\{V_j\})$, включающее все спецификации, позволяющие материализовать виды в разрабатываемой АС, и есть артефакт AD, в котором виды $\{V_j\}$ раскрывают совокупность интересов $\{C_i\}$, учитываемых в проекте. Отметим, что за каждым архитектурным видом стоит его графическое (в типичном случае block-and-line) изображение с необходимым описанием и спецификациями.

В разработке конкретной АС ответственность за построения артефакта лежит на архитекторе, обладающем необходимыми компетенциями и использующем специализированные средства, обеспечивающие выполнение роли «архитектор». В статье предлагается версия архитектурного моделирования, ориентированная на её применение проектировщиком в решении проектных задач.

С предлагаемой версией моделирования связывается архитектурный подход, нацеленный на использование полезных «архитектурных видов» в процессе проектирования с «точки зрения решения проектной задачи». Реализация архитектурного подхода выявила потребность в разработке специализированных средств поддержки активности проектировщиков, прототипирование которых проведено в инструментально-моделирующей среде OwnWIQA, ориентированной на поддержку концептуальной активности проектировщика в работе с задачами на его рабочем месте [2]. Инструментарий OwnWIQA настроен на объединение его экземпляров в комплексы в корпоративной сети разработки АС для обслуживания работ в группе проектировщиков.

Вопросно-ответный подход к решению проектных задач

В течение многих лет на кафедре «Вычислительная техника» Ульяновского государственного технического университета исследовались процессы концептуального проектирования АС в условиях оперативного взаимодействия проектировщиков с опытом и его моделями. Этот этап был выбран потому, что он непосредственно предполагает работу с семантическими (концептуальными) объектами, система которых материализуется в форме разрабатываемого концептуального проекта АС. Кроме того, на практике этот этап обычно является источником дорогостоящих семантических ошибок, обусловленных проблемами персонального и коллективного понимания, или, другими словами, этот этап подвержен негативному влиянию человеческих факторов, особенно когда проектировщики в проекте сталкиваются с новыми задачами [3].

По ходу исследований была разработана инструментально-моделирующая среда WIQA, постоянному совершенствованию которой способствовала её архитектура. В основу данной архитектуры была положена система архитектурных видов на (концептуальный) проект АС и процесс проектирования семейства родственных АС. Центральное место в системе видов выделено задачной структуризации проекта и вопросно-ответному взаимодействию с опытом в процессе решения задачи.

Особое место в систематизации архитектурных видов на проект занимает построение их экземпляров для конкретных проектов, для чего используются инструментально обеспеченные способы отображения основных сущностей операционного пространства проектирования на семантическую память вопросно-ответного типа. В результате отображений для проекта формируется соответствующее ему концептуальное пространство, в котором осуществляются построение и интеграция экземпляров видов и их использование. Особенности концептуального пространства проекта в сетевой версии инструментария NetWIQA и его персонифицированной версии OwnWIQA детально раскрыты в публикациях [1, 2].

К настоящему моменту времени потенциал версий инструментария WIQA обеспечивает ряд представлений задач, используемых в активностях, приведённых на рисунке 1. Для этих активностей общим является формирование, обработка и преобразование объектов, которые имеют вид вопросно-ответных сетей (QA-сетей), локализованных в концептуальном пространстве проекта АС.

Для повышения эффективности работ с такими объектами принято решение о реализации инструментария

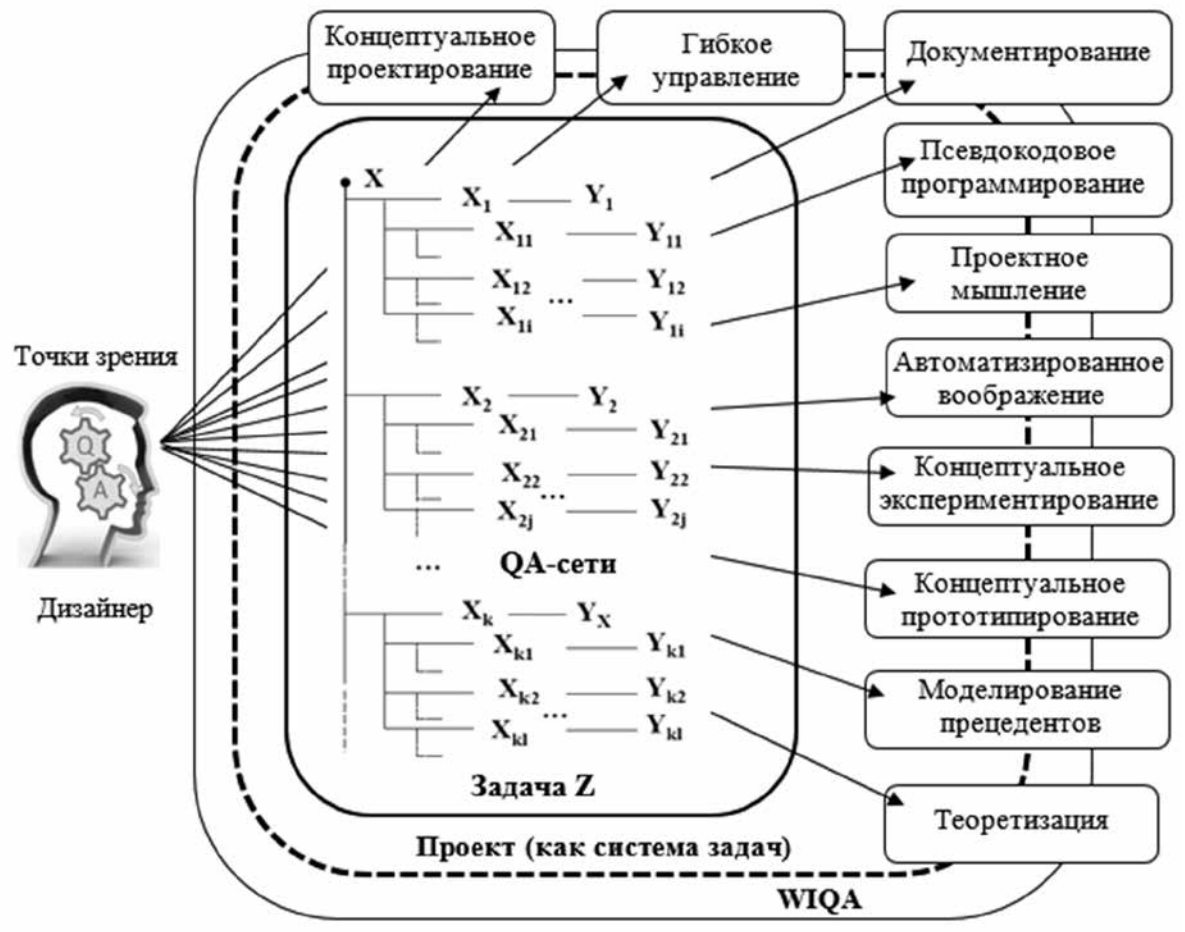

Рис. 1. Освоенные активности концептуального проектирования

работы с системой архитектурных видов S({Vj}) в единой подсистеме с реализацией очередного шага в развитии версии OwnWIQA – создании её подсистемы «Задача».

На схеме, отражающей (потенциальную) работу проектировщика с задачей, специально используется термин «точка зрения», для которого в стандарте ISO/IEC/IEEE 42010:2011 введено следующее определение – это *«набор образцов, шаблонов и соглашений для построения одного типа «архитектурных видов». В нем определяются заинтересованные стороны, чьи проблемы отражены в точке зрения, а также руководства, принципы и шаблонные модели для построения соответствующих видов»* [4].

Именно с системой «точек зрения» на «задачу» в статье и предлагается связать, во-первых, интегральную совокупность типовых диаграммных видов (шаблонов) «задачи», а во-вторых, встроить подсистему инструментальных средств «Задача» в среду OwnWIQA для эффективной работы с проектными задачами в концептуальном пространстве проекта любой разрабатываемой системы.

Сущность архитектурного подхода к решению проектных задач

В стандарте ISO/IEC/IEEE 42010:2011 предусмотрено, что в общем случае «интерес» может быть архитектурно раскрыт связной группой полезных «точек зрения» и соответствующих им видов.

Например, такой конструкт как «архитектурное решение» авторы публикации [5] рекомендуют рассматривать, используя следующие точки зрения:

- «Детали принятия решений»;
- «Связь с требованиями»;
- «Хронология принятия решений»;
- «Участие **заинтересованных сторон** решения».

Более того, эти же авторы в следующей публикации [6] предложили расширить приведённый набор, включив в него точку зрения «Силовые воздействия на архитектурное решение».

Ещё один приём приведён в публикации [7], где предлагается связать с «точкой зрения на контекст» разрабатываемой АС, а вернее с представлением среды (пространства) её разработки и эксплуатации, следующий набор интересов, а значит и соответствующих им архитектурных видов:

- **Границы** АС: какие составляющие находятся в диапазоне управления системой (системная область), а какие – нет (область контекста)? Где граница между АС и ее контекстом, и какие взаимодействия между системой и ее контекстом пересекают эту границу?
- **Пользователи** АС: кто является пользователями системы? Каковы их типы, роли и характеристики? Как и где они получают доступ и используют АС?
- **Внешние зависимости**: какие внешние службы и/или приложения имеют отношение к АС, включая их свойства и поставщиков?
- **Среда выполнения:** какова ожидаемая или желаемая среда технического исполнения, в которой АС будет работать?
- **Воздействие заинтересованных сторон**: какие заинтересованные стороны, включая организации и их ресурсы, влияют на АС и каким образом? Какое влияние оказывает АС на организации и заинтересованные стороны?

Ещё одной возможностью учёта и материализации «интересов» является их детализация, распределение по совокупности видов и интеграция набора (распределённых) составляющих интересов в рамках «архитектурных видов». С такой возможностью связывают «аспектно-ориентированное» представление и материализацию «интересов» [8].

В нашей версии архитектурного подхода к прецедентно-ориентированному решению проектной задачи, их групп и систем принято решение использовать все приведённые возможности учёта, спецификации и овеществления «интересов» и «архитектурных видов», выбранных для разработки подсистемы инструментальных средств «Задача». Отметим, что если задачи проекта любой АС будут решаться в среде OwnWIQA, то «интересы» и «архитектурные виды», вложенные в подсистему «Задача», будут учитываться и овеществляться в каждом экземпляре задачи. А значит, они будут наследоваться создаваемой АС, во-первых, на концептуальном этапе её разработки, а во-вторых, в повторных применениях решений. Тем самым, версия архитектурного подхода, представляемая в статье, создаётся для решения проектных задач АС, но его инструментальная материализация используется авторами для построения подсистемы «Задача» инструментария OwnWIQA.

Распределение задачи в концептуальном пространстве проекта АС

Совокупность архитектурных видов, составляющих архитектуру разрабатываемой АС, требуется согласовать так, чтобы они образовывали целостность, соответствующую теории её проекта [9]. Так как теория проекта и архитектура АС строятся параллельно по ходу проектирования, то они взаимодополняют друг друга, причём, каждый вид является, во-первых, (корректным) приложением теории, а во-вторых, подсказывает шаги её развития, обеспечивая интерпретацию и способствуя архитектурному пониманию.

По этой причине с подсистемой «Задача» связана совокупность архитектурных решений в условиях, представленных на рисунке 2, на котором за каждым блоком и его именем стоит определённый «интерес», выбранный авторами статьи для его овеществления в процессе решения. Например, вид, стоящий за блоком «Теоретизация», указывает, что в состав средств инструментария OwnWIQA следует встроить совокупность средств «Теоретизация», обслуживающую формирование и использование теории проекта разрабатываемой АС. Более того, этот вид должен быть согласован с остальными видами архитектуры OwnWIQA, за каждым из которых

стоят полезные артефакты и активности.

На рисунке 2 отсутствуют связи между блоками, но для блоков используется два цвета («белый» и «серый»), чтобы отделить основную линию активности проектировщика (белый цвет) от сопровождающих действий (серый цвет).

Основная линия активности – это «Профессиональная деятельность», по ходу которой проектировщик решает задачи, используя и развивая свой опыт. Модели данного опыта откладываются и размещаются в полезном артефакте, в персонифицированной «Базе Опыта». Каждый блок основной линии выводит на определённые интересы или их совокупности и соответствующие им архитектурные виды.

В действиях основной линии оперативно используется автоматизированная поддержка (блоки, помеченные серым цветом), с составляющими которой также связаны архитектурные виды и их материализация в инструментальной среде OwnWIQA.

В намеченной версии комплектации совокупности видов на задачу и их интеграции в единое целое используется следующий набор установок:

1. В интегральной совокупности типовых видов должны найти своё место типовые представления задачи, которые используются в уже освоенных активностях, отмеченных на рисунке 1.

2. Каждый дополнительный типовой вид на задачу должен быть настроен на его использование в семантической памяти среды OwnWIQA.

3. Образцы типовых видов на задачу и их интегральная совокупность должны размещаться в концептуальном пространстве [10] соответствующего проекта.

Исходя из этих установок, выбрана совокупность архитектурных видов, ряд названий которых на рисунке 3 подсказывает, за что они отвечают в процессе концептуального решения проектной задачи. Виды с этими названиями

Рис. 2. Структура OwnWIQA с позиций решения проектной задачи

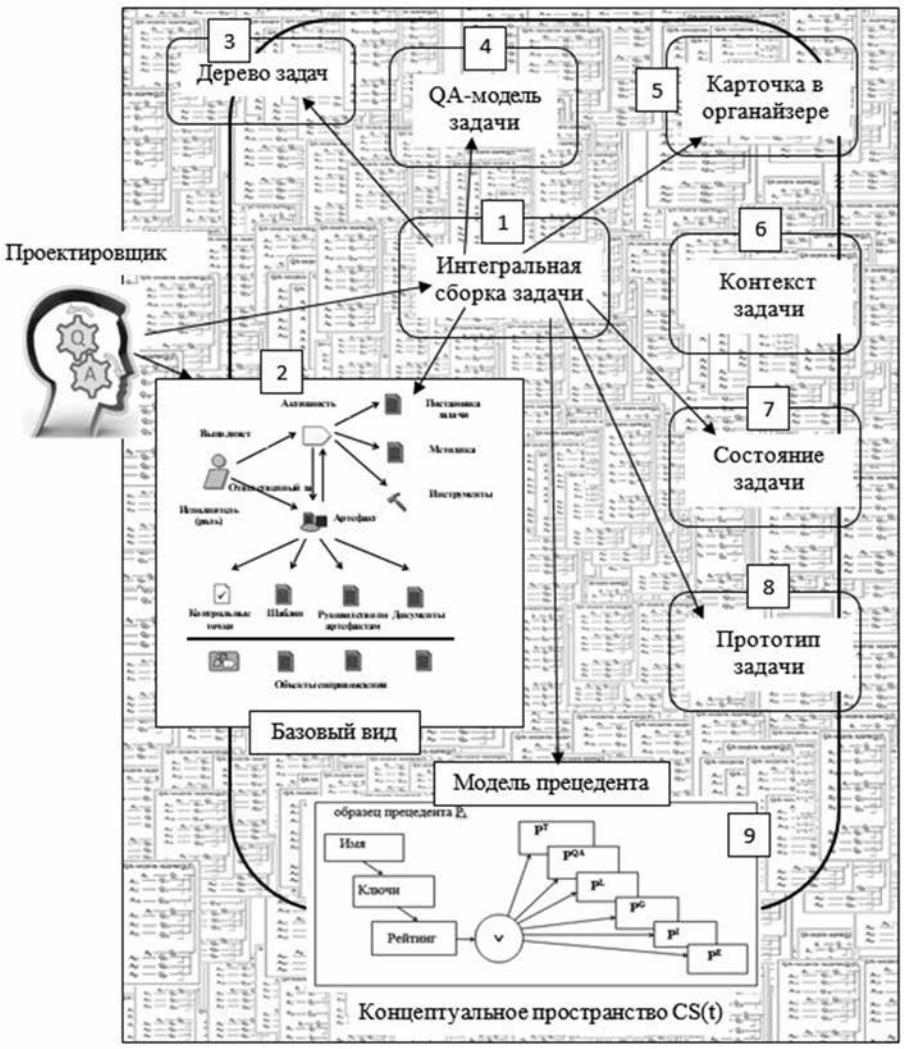

Рис. 3. Система архитектурных видов подсистемы «Задача»

выводят на информацию о задаче, содержащуюся в описании инструментария OwnWIQA [2]. По этой причине ниже представлены только решения по архитектурным видам с номерами 1, 2 и 9.

Отметим, что рисунок 3 отражает только проекцию подсистемы «Задача» на визуально доступные модели проектной задачи, с которыми проектировщик взаимодействует в процессе её решения, применяя активности и другие артефакты схемы, представленной на рисунке 2. Так, например, в рамках системы архитектурных видов подсистемы «Задача» формируются очередные модели прецедентов, которые после завершения решения размещаются в Базе Прецедентов.

Прототипирование архитектурных решений

Очень важной составляющей инструментария OwnWIQA является встроенный в него комплекс средств псевдокодового программирования, предназначенных для двух целей:

- средства позволяют пользователям (а значит и создателям) OwnWIQA расширить функциональный потенциал инструментария;
- с помощью этих средств можно строить несложные приложения или прототипы приложений, программируя подходящую оболочку над комплексом OwnWIQA.

Эти возможности помогли авторам статьи разработать прототипную версию подсистемы «Задача», связывающую прототипы для архитектурных видов, приведённых на рисунке 3. Что стоит за псевдокодовым прототипированием, продемонстрируем для архитектурных видов 1, 8 и 9. Начнём с интегрального вида (вида с номером 9), визуальная (интерфейсная) форма которого приведена на рисунке 4.

Взаимодействие проектировщика с интегральной формой позволяет выбрать то представление задачи, которое оказалось ему необходимо. По сути дела, эта форма имеет вид меню, позиции которого помечены иконками с их названиями. Специфика в том, что меню «нарисовано» в специализированном графическом редакторе, а с иконками связаны индексные ссылки, обеспечивающие переход к выбранному представлению задачи. Индексация является одной из команд редактора, требующей ввести, например, адресную ссылку на другой рисованный интерфейс, или активизировать команду перехода к приложению по имени файла и его расширению, или активизировать процедуру или функцию, запрограммированную на псевдокоде или другом языке программирования, но включённую в специализированную библиотеку.

Предположим, что активизирован переход к базовому представлению задачи. В результате такого выбора проектировщику откроется рисованная интерфейсная форма, представленная на рисунке 5.

Картина базового вида специально нарисована по образцу визуального представления задачи в среде Rational Unified Process (технология проектирования систем с программным обеспечением, разработанная корпорацией IBM [11]). В этой форме, кроме уже отмеченных возможностей индексации её элементов, применяется кнопка навигации «Назад» и поле ввода имени задачи, конкретное значение которого переводит типовой образец архитектурного вида в экземпляр задачи и включение задачи в рабочий каталог задач, зарегистрированных в Органайзере. Для конкретного

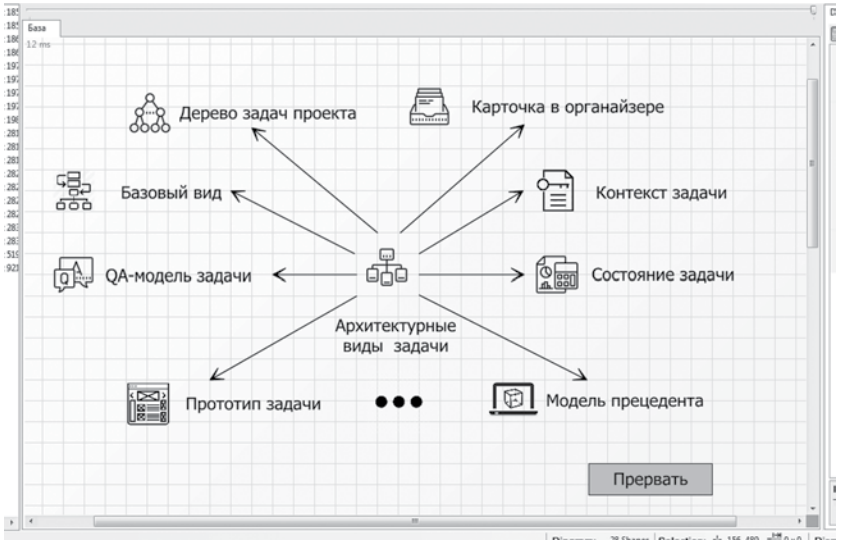

Рис. 4. Интегральный вид на подсистему «Задача»

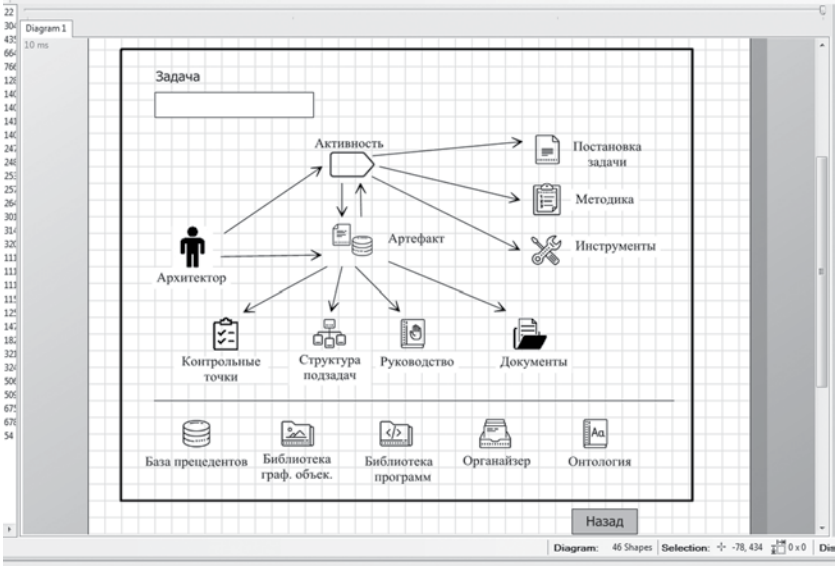

Рис. 5. Базовый архитектурный вид решаемой задачи

экземпляра задачи её базовый вид можно легко модифицировать, если это окажется целесообразным.

Для рассматриваемого архитектурного вида следует отметить, что ряд артефактов, на которые проектировщика выводит его интерфейсная форма, формируются по ходу решения задачи и доступны при переходе к ним в их текущем состоянии (например, постановка задачи, методика, артефакты).

Все элементы интерфейсной формы базового вида также индексированы. К особенностям этого вида относится то, что с ним связана группа программных агентов, каждый из которых обеспечивает идентичность декларативных составляющих задачи, используемых в других видах (например, идентичность текущей версии постановки задачи в базовом виде и в соответствующей модели прецедента, строящейся по ходу решения задачи).

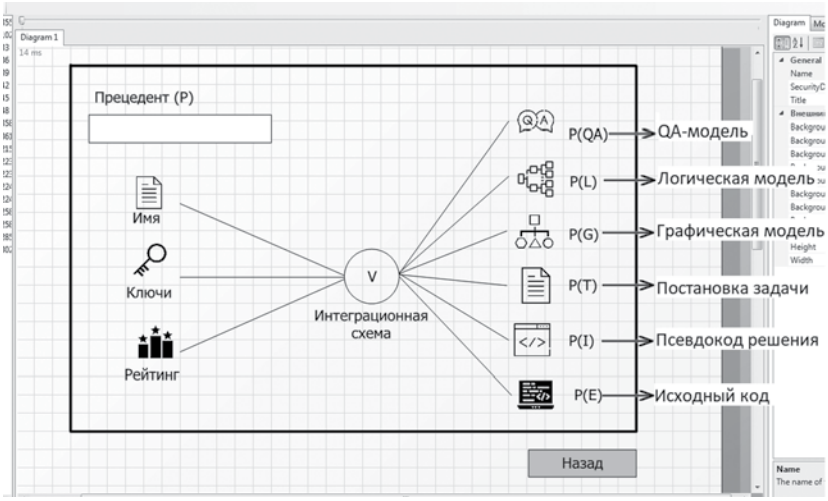

Рис. 6. Визуализация модели прецедента

Структура архитектурного вида «модели прецедента» соответствует рисованной интерфейсной форме, представленной на рисунке 6, на котором для ряда иконок скриншота введены их названия, подсказывающие переходы к составляющим, интегрированным в модели прецедента.

Структура модели и детали её составляющих с учётом их оперативного формирования, подробно рассмотрены в публикациях [12] и [13]. И для этого архитектурного вида актуально обеспечение идентичности содержания и структуры доступных и обрабатываемых концептуальных объектов, представляемых QA-сетями.

Для обеспечения идентичности реализованы механизмы копирования объектов (с сетевой структурой) в специализированную буферную память для последующей загрузки из этой памяти по месту назначения. В реализации механизмов копирования и загрузки используются операции экспорта и импорта QA-сетей с их промежуточным представлением в формах XML-файлов. В буферную память можно скопировать группу объектов, что привело к разработке псевдокодовой утилиты, обслуживающей управляемую выборку объектов из буфера для их загрузки по месту назначения. Псевдокод содержательной части утилиты имеет следующий вид:

Код приведён для того, чтобы показать его синтаксис, допускающий использование процедур и функций, которые могут быть запрограммированы как на псевдокоде, так и на других языках. В приведённом фрагменте функции запрограммированы на языке C# и выбираются из библиотеки, включённой в OwnWIQA.

```
1.3. BUF
  1.3.1. &mynewproject& := QA_GetAreaId("Буферный проект");
  1.3.2. IF &mynewproject& < 0 THEN BEGIN
    1.3.2.1. &mynewproject& := QA_CreateArea("Буферный проект");
    1.3.2.2. END
  1.3.3. QA_DeleteNode(&mynewproject&, 0);
  1.3.4. &myproject& := QA_GetAreaId("Диаграмма");
  1.3.5. &qaid& := QA_FindQAItem (&myproject&, "Первая задача") ;
  1.3.6. QA_CopyQATree(&myproject&, &qaid&, &mynewproject&, 0);
  1.3.7. &qanewid& := QA_FindQAItem (&myproject&, "Первая задача");
  1.3.8. QA_SetQAText(&mynewproject&, &qanewid&, "QA-модель задачи");
  1.3.9. FINISH
```

Как было отмечено выше, архитектурное представление подсистемы «Задача» нацелено на его использование как образца для применения предлагаемой версии архитектурного подхода к разработке функциональных подсистем, с каждой их которых связывается соответствующая проектная задача. Для такого применения больше подходит прототипная версия подсистемы «Задача», а не её версия на языке C#, которую авторы статьи запланировали разработать для включения в исходные коды OwnWIQA, компоненты которой запрограммированы на языке C#.

Для того чтобы продемонстрировать применение прототипных решений, разработанных для подсистемы «Задача», на рисунке 7 приведён прототип базового вида для подсистемы онтологического сопровождения, представленной блоком на рисунке 2.

Для этого вида поясним только то, что он настроен на логико-лингвистический анализ текстовых единиц, которые имеют целостное смысловое содержание (то есть дискурсов). Такой анализ приходится осуществлять для составляющих постановок проектных задач, а также составляющих их вопросно-ответных моделей. Анализ осуществляет контроль за корректным использованием языка проекта, понятийное ядро которого составляет онтология проекта [14].

Отметим также, что для визуализации вида используется рисованная интерфейсная форма, построение которой сводится к выбору подходящих иконок и их имён, их размещению и связыванию в поле графического редактора, а также к индексации элементов формы, выводящей на остальные виды задачи онтологического сопровождения.

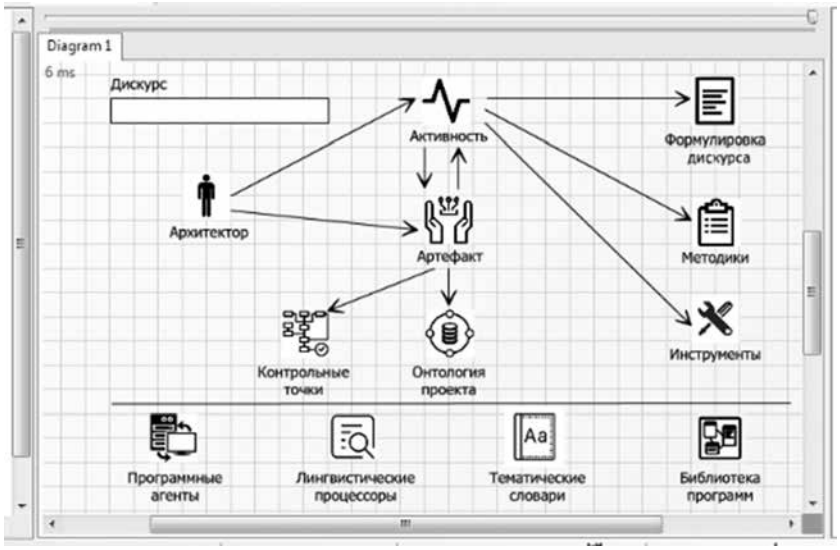

Рис. 7. Базовый архитектурный вид онтологического сопровождения решения задачи

Заключение

Статья представляет очередной шаг в развитии потенциала инструментария WIQA, обеспечивающего решение проектных задач. В реализации этого шага совершенствование инструментария интерпретируется как проект, осуществляемый с помощью средств, уже встроенных в комплекс WIQA, среди которых принципиальное место занимает псевдокодовое программирование.

В основу развития положено архитектурное моделирование, нацеленное на интеграцию в интерактивную сеть ранее освоенных представлений задачи и дополнительных её представлений. К специфике интеграции относится проверка намеченных проектных решений на исполняемых прототипах, в реализации которых использована семантизированная графика, обслуживающая не только визуализацию представлений задачи и переходы между ними, но и понимание ситуаций, а также понимание причинно-следственных переходов между состояниями процесса решения.

В принципе, архитектурно-прототипную версию интеграции представлений задачи можно довести до рабочего состояния, связав с этой версией псевдокодовую реализацию подсистемы «Задача». Но в перспективе намечено повышение эффективности этой подсистемы за счёт многоагентной поддержки актуальности данных, которые обязаны быть идентичными в разных представлениях решаемой задачи.

СПИСОК ЛИТЕРАТУРЫ

1. Маклаев В.А., Соснин П.И. Создание и использование автоматизированной базы опыта проектной организации. – Ульяновск : УлГТУ, 2012. – 360 с.

2. Sosnin P. Experience-Based Human-Computer Interactions: Emerging Research and Opportunities. IG-Global, 2017. 294 p.

3. Clarke P., O'Connor R.V. The situational factors that affect the software development process: Towards a comprehensive reference framework // Journal of Information Software and Technology. 2012. Vol. 54 (5). pp. 433–447.

4. ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description. 2011. pp. 1–46.

5. Modeling Context with an Architecture Viewpoint / A. Bedjeti, P. Lago, G.A. Lewis, R.D.D. Boer, R. Hilliard // IEEE International Conference on Software Architecture (ICSA), 2017. pp. 117–120.

6. Heesch U. van, Avgeriou P., Hilliard R. Forces on Architecture Decisions – A Viewpoint. // Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, IEEE Computer Society, Washington, DC, USA, 2012. pp. 101–110.

7. Heesch U. van, Avgeriou P., Hilliard R. A documentation framework for architecture decisions. // Journal Syst. Softw. 2012. vol. 85, 4, pp. 795–820.

8. Hilliard R. Using Aspects in Architectural Description // Lecture Notes in Computer Science, 2007. vol. 4765. pp. 65–68,

9. Sosnin P. Substantially evolutionary theorizing in designing software-intensive systems. // Information (Switzerland), 2018. 9 (4).

10. Sosnin P. Question-answer nets as a valuable source of information in designing the software intensive system. // 25th Telecommunications Forum, TELFOR 2017 – Proceedings, 2017. pp. 1–4.

11. IBM Rational Unified Process. – URL: http://www-01.ibm.com/software/rational/rup/.

12. Sosnin P. Conceptual Experiments in Automated Designing // Projective Processes and Neuroscience in Art and Design / Eds.: Rachel Zuanon. IG-Global, 2016. pp. 155–181.

13. Sosnin P. Precedent-oriented approach to conceptually experimental activity in designing the software intensive systems // International Journal of Ambient Computing and Intelligence, 2016. Vol. 7 (1). pp. 69–93.

14. Sosnin P., Pushkareva A. Ontological controlling the lexical items in conceptual solution of project tasks. // Lecture Notes in Computer, 2017. Vol. 10409. pp. 31–46.

REFERENCES

1. Maklaev V.A., Sosnin P.I. *Sozdanie i ispolzovanie avtomatizirovannoi basy opyta proektnoi organizatsii* [Creation and Application of Automated Experience Factory of Design Company]. Ulyanovsk, UlSTU Publ., 2012. 360 p.

2. Sosnin P. *Experience-Based Human-Computer Interactions: Emerging Research and Opportunities,* IG-Global Publ., 2017. 294 p.

3. Clarke P., O'Connor R.V. The Situational Factors that Affect the Software Development Process: Towards a Comprehensive Reference Framework. *Journal of Information Software and Technology,* 2012, vol. 54 (5), pp. 433–447.

4. ISO/IEC/IEEE 42010:2011 Systems and Software Engineering – Architecture Description. 2011. pp. 1–46.

5. Bedjeti A., P. Lago, G.A. Lewis, R.D.D. Boer, R. Hilliard. Modeling Context with an Architecture Viewpoint. *IEEE International Conference on Software Architecture (ICSA),* 2017, pp. 117–120.

6. Heesch U. van, Avgeriou P., Hilliard R. Forces on Architecture Decisions – A Viewpoint. In *Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture,* IEEE Computer Society, Washington, DC, USA, 2012. pp. 101–110.

7. Heesch U. van, Avgeriou P., Hilliard R. A Documentation Framework for Architecture Decisions. *Journal. Syst. Softw.,* 2012, vol. 85 (4), pp. 795–820.

8. Hilliard R. Using Aspects in Architectural Description. *Lecture Notes in Computer Science,* 2007, vol. 4765, pp. 65–68.

9. Sosnin P. Substantially Evolutionary Theorizing in Designing Software-Intensive Systems. *Information* (Switzerland), 2018, 9 (4).

10. Sosnin P. Question-Answer Nets as a Valuable Source of Information in Designing the Software Intensive System. *Proc. 25th Telecommunications Forum, TELFOR 2017.* 2017, pp. 1–4.

11. IBM Rational Unified Process. Available at: http://www-01.ibm.com/software/rational/rup/.

12. Sosnin P. *Conceptual Experiments in Automated Designing. Projective Processes and Neuroscience in Art and Design, Eds.: Rachel Zuanon.* IG-Global Publ., 2016. pp. 155–181.

13. Sosnin P. Precedent-Oriented Approach to Conceptually Experimental Activity in Designing the Software Intensive Systems. *International Journal of Ambient Computing and Intelligence,* 2016. vol. 7 (1), pp. 69–93.

14. Sosnin P., Pushkareva A. Ontological Controlling the Lexical Items in Conceptual Solution of Project Tasks. *Lecture Notes in Computer,* 2017, vol. 10409, pp. 31–46.