УДК 004.415.2

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ СПЕЦИФИКАЦИИ КОНЦЕПТУАЛИЗАЦИЙ В ПРОЕКТИРОВАНИИ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

П.И. Соснин[1], В.А. Маклаев[2]

[1] Ульяновский государственный технический университет
sosnin@ulstu.ru
[2] ОАО ФНПЦ «Научно-производственное объединение «Марс»
mars@mv.ru

Аннотация

Совершенствование потоков работ, обеспечивающих спецификации концептуализаций, способствует повышению степени успешности проектирования современных автоматизированных систем. В статье предлагаются новые формы и средства спецификации, в основу которых положено оперативное построение и использование онтологии проекта в процессах его пошаговой детализации, вопросно-ответного анализа проектных задач и их моделирования как прецедентов.

***Ключевые слова**: автоматизированные системы, вопросно-ответное моделирование, проектный прецедент, псевдо-кодовое программирование, спецификация концептуализаций, онтология проекта.*

Введение

Наиболее существенным индикатором зрелости инженерной деятельности определённого вида является реально наблюдаемая степень успешности её результатов, которую целесообразно определять статистически. В этом плане говорить о достаточной зрелости «*инженерии разработок систем, интенсивно использующих программное обеспечение* (Software Intensive Systems, SIS)» не приходится, поскольку степень успешности такой деятельности до сих пор чрезвычайно низка (около 35 %) [1], что приводит к безвозвратным ежегодным потерям в сотни миллиардов долларов.

Сам факт такого объёма финансовых потерь, независимо от подходов и способов оценок успешности, требует изменения существующего положения дел, основная причина которого *негативные проявления человеческого фактора* на ранних этапах проектирования SIS [2]. Перелома в положении дел разумно ожидать от достижения достаточной зрелости «*инженерии интеллектуальной активности проектировщиков в концептуальном проектировании SIS*». Другими словами, перелом не возможен без комплексной автоматизации концептуальной деятельности проектировщиков SIS.

Необходимость *инженерии концептуальной деятельности* осознана достаточно давно. Так, например, в практике разработок SIS такое осознание привело к созданию «*инженерии требований* (requirement engineering)», которая активно развивается последние 20 лет [3]. Более того, в настоящее время в практику *программной инженерии* активно внедряются методы и средства «*инженерии знаний* (knowledge engineering)» [4] и «*инженерии онтологий* (ontology engineering)» [5]. Каждая из названных инженерий вводит элементы автоматизации в концептуальную активность проектировщиков. Их согласованная интеграция и развитие должны привести к зрелой *инженерии разработок SIS*.

В статье представляется комплекс средств, предназначенный для спецификации концептуализаций, порождаемых коллективом проектировщиков в концептуальном проектирова-

нии SIS. Отличия предлагаемых средств от известных инструментальных решений определяют *вопросно-ответный подход* к коллективной концептуальной деятельности и её структуризация, в основу которой положены *прецеденты*, ориентированные на активное участие в их исполнении как разработчиков SIS, так и их пользователей. Особое место в статье отводится средствам (без деталей процессов) оперативного формирования и использования онтологий проектов SIS.

В завершение введения отметим, что в российской нормативной базе класс «Автоматизированных Систем (АС)» является родственным классу SIS и всё отмеченное выше для класса SIS справедливо и для класса АС. По этой причине все последующие предложения и рассуждения будут относиться к классу автоматизированных систем.

1 Исходные предпосылки

Особенности авторского подхода к спецификации концептуализаций обусловлены следующими исходными предпосылками:

- Необходимо учитывать и специфицировать *те и только те концептуализации*, которые находят *явное представление* в концептуальном проекте разрабатываемой $АС_i$. Такая установка предохраняет от бесконтрольного использования лексики в нормативных проектных документах и в рабочих текстах, регистрирующих оперативные рассуждения проектировщиков для индивидуального и общего пользования. Бесконтрольность употребления лексических единиц в процессе проектирования считается опаснейшим (потенциальным) источником концептуальных ошибок.
- *Спецификации* концептуализаций должны разрабатываться как специализированная автоматизированная система $АС^К_i$, представляющая разрабатываемую $АС_i$ концептуально и предназначенная (как концептуальный проект в его текущем состоянии), в первую очередь, для обеспечение *адекватного понимания их содержания* проектировщиками и другими лицами, заинтересованными в разработке АС.
- Основным *системообразующим ядром* системы спецификаций $АС^К_i$ должна быть *онтология проекта*, которая создаётся в процессе специфицирования концептуализаций как *специализированная автоматизированная система* $АС^О_i$, предназначенная для представления концептуального базиса (концептуальных конструктов), на основе которых строится система $АС^К_i$. Разработку $АС_i$ должна сопровождать *её и только её онтология проекта* $АС^О_i$, в построении которой целесообразно использовать концептуальные конструкты из любых полезных источников.
- *Системообразующей основой* системы концептуальных конструктов $АС^О_i$ должна быть *система понятий*, сопровождающих разработку концептуального проекта $АС_i$. Онтология проекта $АС^О_i$ должна быть материализована в виде, который обеспечивает потенциальное использование её конструктов в разработках родственных $\{АС_j\}$.
- Спецификации должны быть материализованы в *интерактивной форме*, способствующей *различным группам их пользователей* взаимодействовать со спецификациями так, чтобы адекватным образом понять их концептуальное содержание на любом этапе жизненного цикла $АС_i$.
- Адекватность понимания спецификаций, представляющих концептуальные конструкты с *алгоритмическим содержанием*, должна достигаться, при необходимости, с помощью *имитационных экспериментов* с прототипами. *Проектировщик* в таких экспериментах должен выполнять роль «*процессора*».
- *Форма материализации спецификаций* должна быть такой, чтобы их было легко *корректировать*, если в них обнаружены ошибки или элементы, затрудняющие их понимание.

- Содержание, вложенное в спецификации, должно быть *достаточным для успешного построения* всех составляющих разрабатываемой АС.

Содержание предпосылок было использовано как основной источник требований при разработке комплекса средств, обеспечивающего построение и использование онтологии проекта разрабатываемой АС. Этот комплекс создан в виде совокупности специализированных расширений (plug-ins) инструментальной среды WIQA (Working In Questions and Answers), предназначенной для концептуального проектирования АС коллективом разработчиков.

2 Инструментальная среда концептуального проектирования

Инструментальная среда WIQA изначально разрабатывалась для моделирования совокупностей типовых и предметных проектных задач, которые приходится решать разработчикам АС. Как источник типовых задач были использованы типовые задачи потоков работ концептуального проектирования в мастер-методологии Rational Unified Process (RUP) [6]. По предназначению и содержанию WIQA разрабатывалась как специализированная $АС^{Т}$ технологического типа.

При использовании инструментария WIQA в концептуальном проектировании конкретной $АС_i$ с ней связывалась исходная (корневая) предметная задача Z^*_i разработки $АС^{К}_i$, для построения решения которой применялся метод пошаговой детализации. Пошаговая детализация была ориентирована на порождение подчинённых задач и активное использование в процессе поиска решения вопросно-ответных рассуждений проектировщиков и типовых (технологических, служебных) проектных задач. Более того, любая задача проектирования (корневая, предметные подчинённые и типовые подчинённые) в процессе решения представлялись вопросно-ответными моделями (Question-Answer models, QA-models, QA-моделями), каждая из которых содержала текущее состояние вопросно-ответных рассуждений (QA-рассуждений), использованных в построении решения соответствующей задачи. Именно из QA-рассуждений проектировщиками извлекались очередные подчинённые задачи пошаговой детализации. Обобщённая схема такой версии концептуального проектирования приведена на рисунке 1.

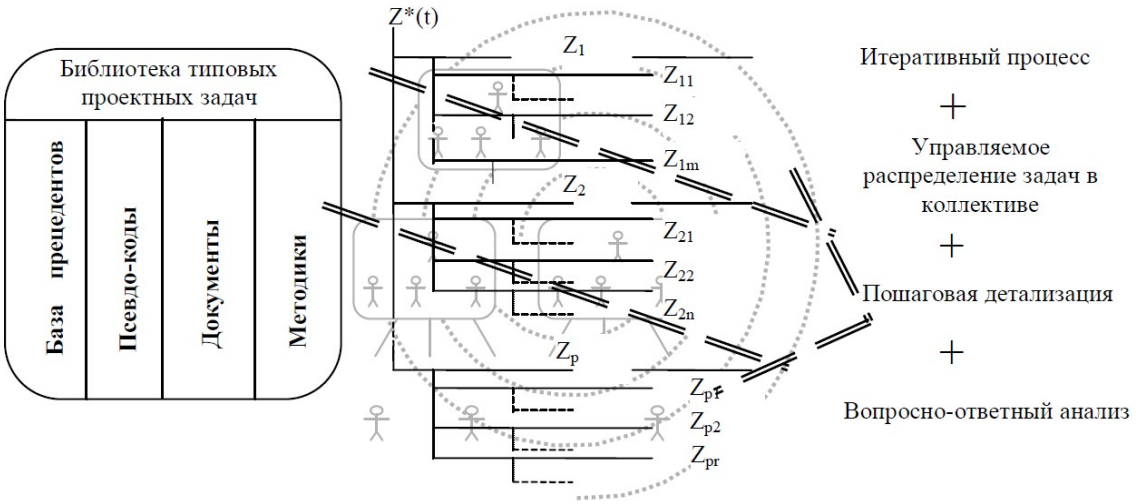

Рисунок 1 - Пошаговая детализация в концептуальном решении задач проектирования

Одной из важнейших особенностей представленной версии концептуального проектирования является потенциальное применение всех средств инструментария WIQA к любой задаче Z_i дерева $T(\{Z_i\})$ проектных задач $\{Z_i\}$, если в этом будет необходимость. В таком применении задача Z_i представляется её QA-моделью $QA(Z_i)$, обобщённая структура видов которой приведена на рисунке 2.

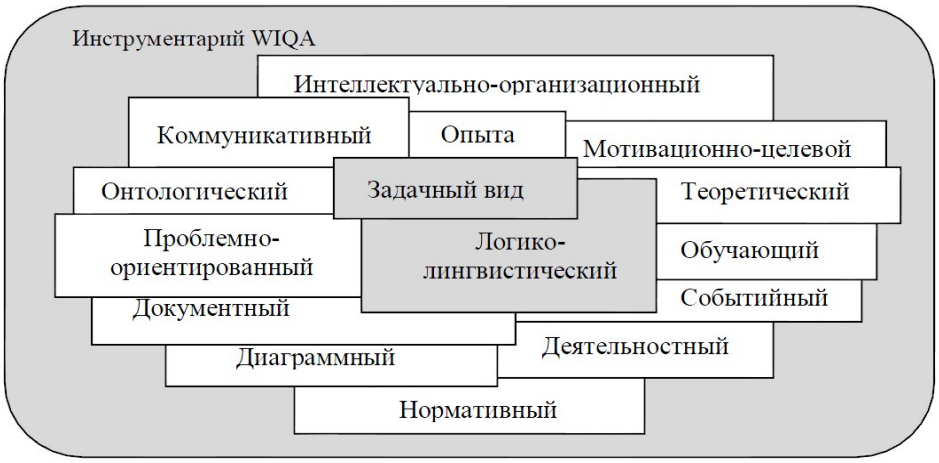

Рисунок 2 - Система видов QA-модели задачи

Все виды интерактивно доступны с любого рабочего места группы проектировщиков (разумеется, в рамках полномочий). Каждый вид содержит декларативную часть (артефакты вида) и процедурную часть (средства формирования и использования артефактов), которые подробно раскрыты в [7]. Центральное место в QA-модели занимают задачный и логико-лингвистический виды, представляющие систему постановок задач и вопросно-ответную структуризацию их решений с поясняющими диаграммными схемами и иллюстрациями других типов (если необходимо или полезно). Интерфейсная форма доступа для задачного вида и логико-лингвистического вида (в форме протокола QA-рассуждений) представлена на рисунке 3.

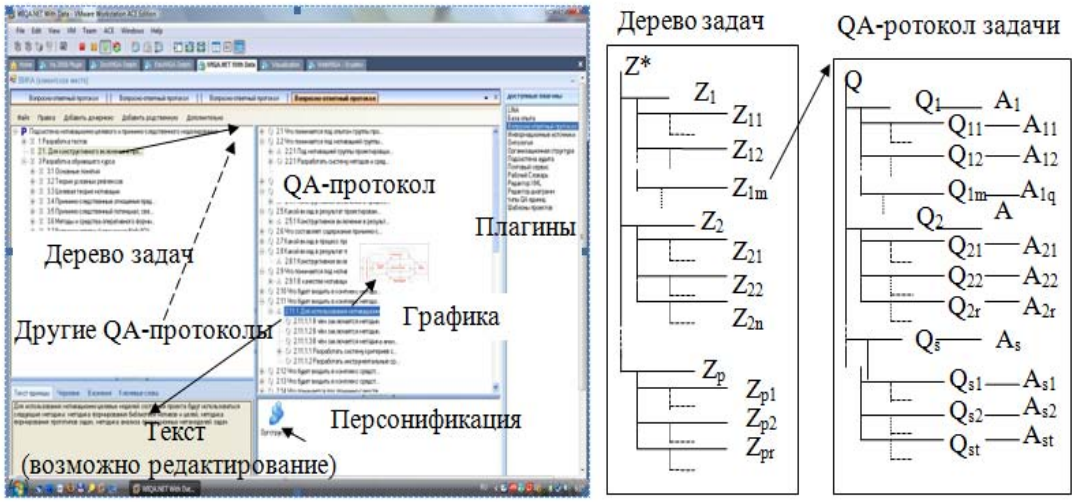

Рисунок 3 - Структура и представление задач и их QA-моделей

Более детально, любое образование любого из Z-, Q- или A-типов представляет собой интерактивный объект, для обозначения которого введём символику Ob_QA.I, где QA.I – индексное имя задачи, вопроса или ответа, специфицирующее тип и место объекта в иерар-

хии объектов (отметим, что задача тоже вопрос, но «задачного типа», ответом на который является решение задачи).

Свойства любого из объектов {Ob_QA.I} открываются в интерактивном взаимодействии с проектировщиком. Та часть любого QA-объекта, которая визуально доступна на интерфейсной форме рисунка 3, выступает в роли «посредника» между объектом и проектировщиком, открывающим ему доступ к любому из свойств объекта или любой их группе, если в этом появляется необходимость. Затребованные свойства доступны через команды интерфейса и плагины. Любой «посредник» открывается перед проектировщиком через его текстовое описание (постановка задачи, формулировка вопроса или ответа) и индексное имя, представляющее тип (или подтип) объекта и его «место» в иерархии дерева задач или QA-протокола. Индексное имя полезно интерпретировать и использовать как «адрес» объекта в их системе (в вопросно-ответной базе, QA-базе.)

В совокупности свойств любого QA-объекта, например Ob_Z.I, Ob_Q.J или Ob_A.K, различаются:

- базовые свойства, каждое из которых представляется с помощью соответствующего атрибута QA-базы;
- дополнительные свойства (дополнительные атрибуты), которые вводятся адресно, то есть приписываются конкретному QA-объекту или их определённой группе с помощью средств объектно-реляционных преобразований (специальное расширение в инструментарии WIQA).

Потенциал QA-моделирования, вложенный в инструментарий WIQA, достаточен для представления концептуализаций любых типов (тексты, таблицы, диаграммы, диаграммные схемы и композиции отмеченных форм), а также любого содержания, которые используются в проектировании AC_i. Всё связанное с использованием языка проекта, интерактивно доступно с рабочих мест проектировщиков, участвующих в разработке AC_i.

Особое место в осуществлении концептуализаций занимает текстовая информация, используемая в каждом Ob_QA.I и являющаяся его составной частью, для обозначения которой введём следующую символику $L.I(t)= P^L(Ob\_QA.I(t))$, где Pr^L – проекция объекта на использование языка проекта AC_i, t – момент времени жизненного цикла объекта Ob_QA.I(t). В обозначении учтено, что каждый объект $Ob^T\_QA.I(t)$ является динамическим конструктом, который формируется (порождается) по ходу проектирования AC_i. Часть L.I(t) соответствующего QA-объекта также является динамическим конструктом.

В общем случае проекция L.I(t) включает следующие составляющие:

- $T.I(t, t_n)$ – текущее описание объекта, зарегистрированное в момент времени t_n и доступное проектировщику непосредственно через интерфейсную форму, представленную на рисунке 3;
- $T.I(t_{n-1})$ – все предыдущие версии описания объекта, изменения которых раскрывают динамику формирования описания;
- $DC(t, t_n)$ – черновик (draft copy), раскрывающий «черновые записи», включая информационные источники, на основе которых сформировано состояние $T.I(t, t_n)$;
- другие употребления языка проекта, которые раскрываются при использовании плагинов, применимых для объекта Ob_QA.I;
- для объекта Ob_QA.I, в составе которых имеются подчинённые, проекция L каждого из подчинённых включается в L.I(t), но её содержимое становится доступным при выборе этого пождчинённого объекта.

При порождении (построении) концептуализаций среди названных составляющих особое место занимает система $S(\{T.I(t, t_n)\})$, объединяющая (как это показано на рисунке 3) текущие описания всех объектов, сформированных и используемых в процессе проектирования

AC_i в текущий момент времени. Именно лексика этой системы и является лексикой языка проекта, употребления которой должно быть обязательно доведено до *явных референций* на то, что найдёт материальное воплощение в AC_i. Именно из составляющих системы $S(\{T.I(t, t_n)\})$ и их содержания должна разрабатываться *онтология проекта*.

3 Процесс концептуализации

Жизненный цикл любой AC_i начинается с осознания определённой связной совокупности потребностей, выраженной и зарегистрированной в текстовой форме (пусть текстом T_0) на естественном или естественно-профессиональном языке. Если реакция на выделенную совокупность потребностей приведёт к решению о разработке AC_i, то с этого текста T_0 начнётся процесс концептуализации её проекта.

Если процесс начинается в среде WIQA, то текст T_0 оформляется как исходная постановка задачи $Z^*(AC_i)$, регистрируемая в дереве задач проекта AC_i в виде текста $T^*I(t_0, t_0)$, форма и содержание которого должна удовлетворять определённым требованиям [8]. В этом плане текст $T^*I(t_0, t_0)$ является исходной (первой) концептуализацией проекта AC_i. По ходу разработки текст $T^*I(t, t_0)$ без изменения формы корректируется и уточняется, его лексика приводится к языку проекта, а содержание специфицируется. Так как задаче Z^* подчинены все остальные задачи проектирования AC_i, то спецификацией этой концептуализации логично считать концептуальный проект AC_i.

Выше было отмечено, что проектирование осуществляется в формах пошаговой детализации, в которой к любой задаче дерева задач, формируемого по ходу проектирования (по ходу пошаговой детализации), применимы все необходимые средства инструментария WIQA. В результате строится QA-модель задачи, включающая концептуализации и их спецификации, вложенные в соответствующие виды модели (рисунок 2). Формы концептуализации задачного и логико-лингвистического видов представлены выше. Остальные формы раскрываются, заполняются концептуализациями, специфицируются и интерактивно доступны с помощью специализированного набора плагинов, встроенных в WIQA.

Общим для всех форм концептуализации является то, что они ориентированы на единообразную интерпретацию любой задачи дерева задач – за *задачей стоит проектный прецедент*, который либо применяется в той «точке» дерева задач, где он расположен, либо строится для повторных применений в будущем, например, пользователем проектируемой AC_i. Прецедент – это *активность* человека или группы лиц, связанная с действием или решением или поведением, осуществлённым в прошлом, которая полезна как *образец для повторных использований* и/или *оправдания повторных действий* по такому образцу [9].

Любой прецедент для обеспечения его повторных применений должен быть подготовлен. Другими словами для повторного осуществления прецедента должна быть создана его модель (схема), применение которой позволит исполнителю или исполнителям повторить закодированные в модели прецедента действия. Учитывая то, что прецеденты лежат в основе человеческого опыта, в который их модели включаются подобно условным рефлексам, а вернее интеллектуально обработанным условным рефлексам, инструментарий WIQA настроен на типовую модель прецедента [10] с обобщённой (интегральной) схемой, представленной на рисунке 4.

В набор практически полезных моделей прецедента, порождаемых в процессе его разработки, включены:

- *текстовая модель* P^T, представляющая постановку задачи $Z(P_i)$, в результате решения которой создан образец прецедента (как определённый результат интеллектуального освоения реального прецедента);

- *логическая модель* P^{L}, конкретизирующая типовую логическую модель (представлена на рисунке 4) в виде формулы логики предикатов, записанной на языке постановки задачи P^{T};
- *графическая модель прецедента* P^{G}, представляющая его обобщённо с использованием «*block and line*» средств (например, диаграммы активности на языке *UML*);
- *вопросно-ответная модель* P^{QA}, соответствующая задаче $Z(P_i)$;
- *модель* P^{I}, представляющая вложенное в прецедент поведение *исходным кодом* его программы;
- *модель* P^{E}, выводящая на *исполняемый код* программы, кодирующей образец прецедента;
- *интегральная* (схематическая) *модель* прецедента P^{S} в виде его схемы, объединяющей все специализированные модели прецедента в единое целое.

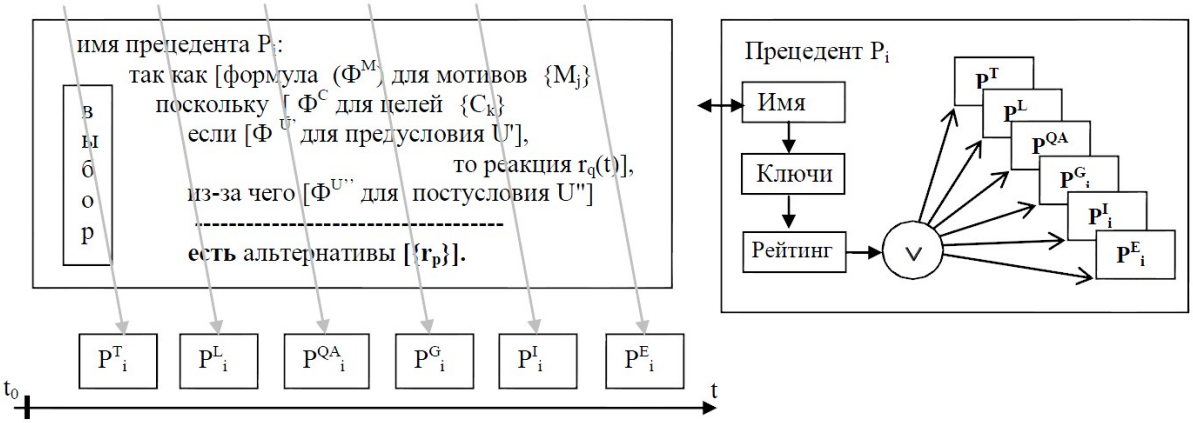

Рисунок 4 - Модель прецедента (жизненный цикл и интегральная схема)

Представленная типовая модель прецедента (совокупность специализированных моделей и их объединение) используется для представления как прецедентов разрабатываемой AC_i, так и для представления прецедентов технологии, включая, например, прецеденты документирования и использования стандартов. Другими словами, прецедентный подход систематически применяется для задач формирования и использования всех видов, включённых в QA-модель задачи (рисунок 2).

Система видов QA-модели и типовая модель прецедента специфицированы так (содержат совокупность декларативных и процедурных форм), чтобы с их помощью можно было специфицировать концептуализации любых типов и любого содержания, которые было решено включить в концептуальный проект AC_i. С их помощью можно выразить и специфицировать концептуализации функциональных и нефункциональных требований, ограничений и принятых решений с декларативным и алгоритмическим содержанием.

4 Средства формирования и использования онтологий проектирования

В концептуализации AC_i особое место занимает лексика $L^{Л}(t)$ языка проекта $L^{П}(t)$, для спецификации которой используется онтология проекта $AC^{O}{}_i(t)$. С инструментальной средой WIQA как специализированной AC^{T} также связана лексика её языка L^{WIQA} и онтология $WIQA^{O}(t)$, развиваемая за счёт расширений (WIQA открытая система инструментальных средств). Так, что проектировщикам в их оперативной работе приходится использовать два связанных языка $L^{П}(t)$ и L^{WIQA}, согласованных с онтологиями $AC^{O}{}_i(t)$ и $WIQA^{O}(t)$. На проектировщиках лежит ответственность только за формирование языка $L^{П}(t)$ и онтологии $AC^{O}{}_i(t)$. Вся информация, необходимая и достаточная для таких построений, а вернее для построения текущих состояний $L^{П}(t)$ и $AC^{O}{}_i(t)$, присутствует в системе $S(\{T.I(t, t_n)\})$.

В общем случае построения $L^{П}(t)$ и AC^{O}_{i} (t) начинет осуществляться с «пустых» состояний, но в любом случае и в любом и в любом их состоянии в основу развития языка и онтологии в WIQA положен предикативный анализ простых предложений, извлекаемых из текстов системы S({T.I(t, t_n)}). Обобщённая схема формирования онтологии AC^{O}_{i} (t) приведена на рисунке 5.

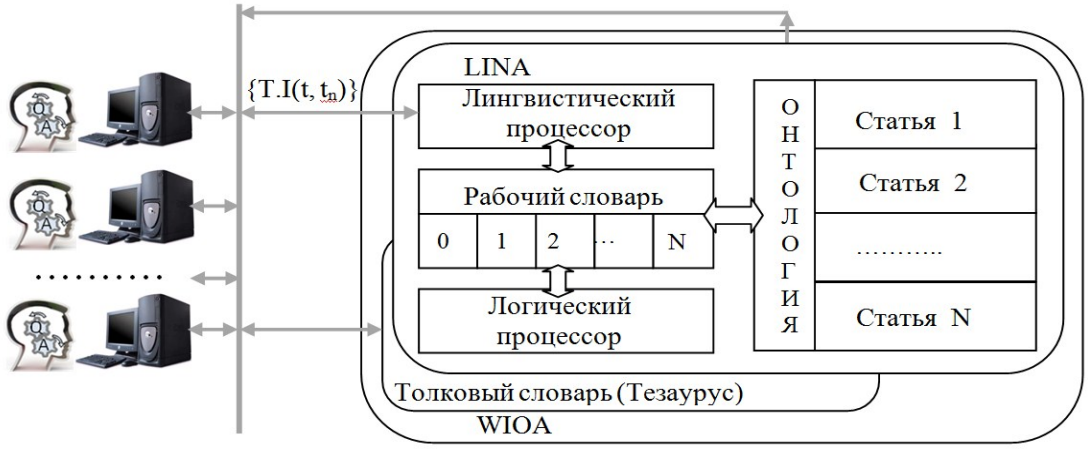

Рисунок 5 - Обобщённая схема формирования онтологии проекта

Средства предикативной обработки текстовой информации, вложенной в поток {T.I(t, t_n)}, доступны как «Лингвистический процессор» с каждого рабочего места проектировщиков, разрабатывающих AC_i. Эти средства объединены с другими (Рабочий словарь, Логический процессор, Онтология), представленными на рисунке 5, в единый комплекс LINA (Linguistic In Nominative Activity), обслуживающий логико-лингвистические потоки работ с языком проекта и его употреблениями. Комплекс LINA обеспечивает формирование языка проекта $L^{П}(t)$ и его материализацию в виде онтологии AC^{O}_{i} (t) в процессах контроля текстов постановок проектных задач и текстов формулировок требований, ограничений и проектных решений. Другими словами, существенные действия по построениям $L^{П}(t)$ и AC^{O}_{i} (t) включены в процессы контроля тех текстов в процессе проектирования и проекте, которые в обязательном порядке должны быть проверены на их *правильность*.

Когда (в определённых приложениях) вводится оценка «правильность», предполагается, что её значения можно проверить или явно или неявно. В рассматриваемом случае «правильность» введена для оценивания «понимания» теми лицами (субъектами, обозначим {Sb_i}), которые используют или будут использовать постановки задач и формулировки требований, ограничений и проектных решений. Для успешности коллективного проектирования «правильность» должна быть проверяемой. Один из важнейших типов проверок на «правильность» должен быть связан с «достаточной степенью единообразного понимания», когда при взаимодействии любого Sb_i с выбранной единицей T.I(t, t_n) у него отсутствуют весомые причины для *исправления* T.I(t, t_n) или составляющих этой единицы. Более того, если такие причины имеются, то Sb_i обязан объяснить *что*, *как* и *почему* должно быть *исправлено*. Разумеется, для того, чтобы исправлять, то, что исправляется, должно быть написано.

К числу важнейших проверок на «правильность» относится и связь проверяемой единицы T.I(t, t_n) «с её материальным представлением в проектируемой АС». Специфицируются только те концептуализации, которые обязательно найдут своё материальное воплощение в созданной АС. Более того, в такие спецификации должна вкладываться информация о референции, например, в форме адресных указаний на (все) материализации специфицируемой концептуализации. Разумеется, в спецификации должны присутствовать и указания на кон-

цептуализируемые (моделируемые) референты (по крайней мере, в виде примера или примеров).

Таким образом, для построения онтологии проекта необходимы средства, обеспечивающие контроль на *правильность* текстовых описаний, в первую очередь, в представленном выше смысле. В комплексе средств LINA контроль за *правильностью* лексики текста $T.I(t, t_n)$ обслуживает лингвистический процессор. Правильность лексики контролируется на уровне простых предложений в контексте их употребления. Поэтому на лингвистический процессор ложится задача анализа текста $T.I(t, t_n)$, для решения которой используется его автоматизированный перевод на язык логики предикатов простых предложений. Проверяемые семантические составляющие простого предложения $ПП_{ijk}$, где i - индекс текста, j - индекс (номер) сложного приложения в тексте, k - индекс простого предложения в сложном, представлены на рисунке 6.

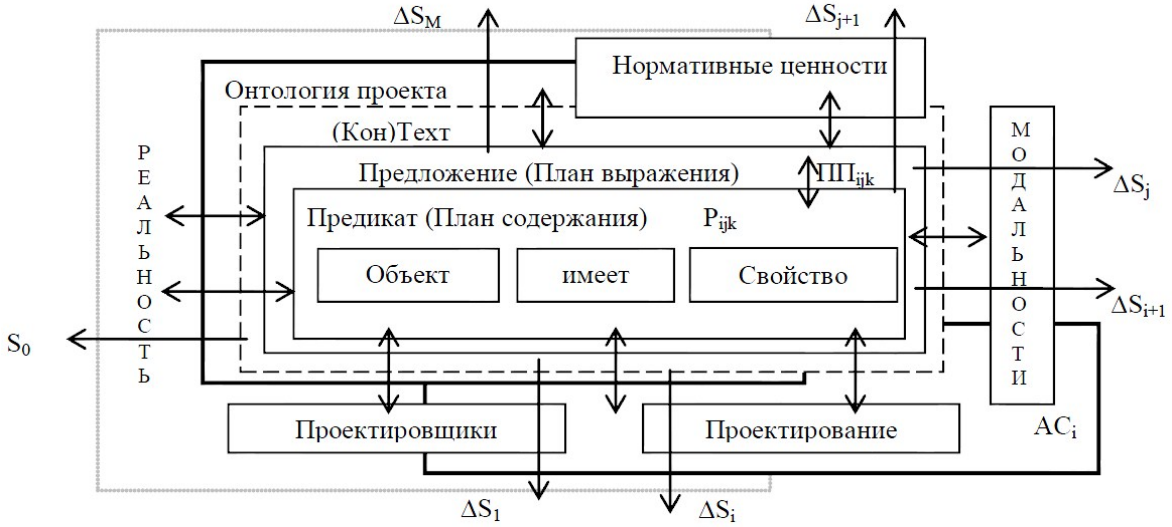

Рисунок 6 - Структура семантики простого предложения $ПП_{ijk}$

Семантическая модель $S(ПП_{ijk})$, используемая для спецификации его концептуального содержания, определена как аддитивная совокупность составляющих

$S(ПП_{ijk}) = S_0(ПП_{ijk}) + \sum \Delta S_m(ПП_{ijk})$,

в число которых включены:

$S_0(ПП_{ijk})$ – предикатная модель PP_{ijk}, прошедшая проверку на соответствие онтологии;

$\Delta S_1(ПП_{ijk})$ – синтаксемные характеристики [2] предложения $ПП_{ijk}$;

$\Delta S_2(ПП_{ijk})$ – вероятностные характеристики $ПП_{ijk}$;

$\Delta S_3(ПП_{ijk})$ – характеристики нечёткости свойства или отношения, названного в $ПП_{ijk}$;

$\Delta S_4(ПП_{ijk})$ – причина управляющего воздействия на процесс проектирования;

$\Delta S_5(ПП_{ijk})$ – вариант интерпретации проектировщиком содержания $ПП_{ijk}$.

Реальность семантических анализа $ПП_{ijk}$ и проверок такова, что «вычисления» семантических составляющих могут быть прерваны из-за различных объективных причин (новое понятие, непонимание, неполнота знаний, преждевременность предикации, ...). Часть из этих причин приводит к вопросам, на которые следует найти ответ или построить его (развитие онтологии, управление процессом проектирования).

Контроль за *правильностью* текстов и их фрагментов типов «сложное предложение» или «группа сложных предложений» (например, абзац) обслуживается с помощью «Логического процессора», с помощью которого проектировщик решает задачу синтеза формулы логики предикатов для текста $T.I(t, t_n)$ или его фрагмента. Для контроля за *правильностью* формулы используется автоматизированное построение её диаграммного (block and line) представле-

ния. Для сборки простых предикатов в формулы используется библиотека шаблонов «сложных предложений, состоящих из двух простых» для русского языка. Формулы специфицируют концептуальное содержание текстов и их фрагментов. Построение каждой конкретной формулы осуществляется проектировщиком в операционной обстановке с использованием средств визуального редактирования.

Принципиальным типом формул логики предикатов для текстовых единиц является логическая модель P^L проектного прецедента P, построенная на основе его текстовой модели P^T. Одним из применений формулы P^L является её использование для построений формулы P^A доступа к прецеденту. В простейшем виде формула P^A может представлять собой И-связку «k_1& k_2&….&k_N» ключей {k_n} доступа. Именно такой вид логического доступа (обозначим его $f^1(\{k_n\})$) используется в комплексе WIQA для «грубого» доступа к прецедентам. Для повышения степени избирательности в комплексе WIQA предусмотрено построение формул доступа типа $f^2(\{k_n\})$, в которых ограничения на использование логических операций сняты.

В осуществлении потоков работ, исполняемых с помощью лингвистического и логического процессоров, используются компоненты «Рабочий словарь», «Онтология» и «Толковый словарь», который не включён в комплекс LINA. Компонента «Рабочий словарь», выполняет функции «мягкой» версии онтологии, аккумулирующей всю извлечённую на текущий момент времени онтологическую информацию, для её распределения по статьям «Онтологии».

Каждая статья «Онтологии» включает информацию, проверенную в процессах контроля правильности и используемую в очередных процедурах контроля правильности. Более того, компонента «Онтология» содержит средства, обеспечивающие выполнение потоков работ для систематизации статей онтологии. В «Онтологии» поддерживается систематизация по следующему набору отношений «часть-целое», «род-вид», «ассоциация».

Компонента «Толковый словарь» включена в состав инструментария WIQA для исполнения традиционной нагрузки, возлагаемой на тезаурусы. В толковый словарь включают лексику и её определения (разных видов), к которой не предъявляют таких же жестких требований, как к лексике языка проекта $L^{\Pi}(t)$. В частности. в этот словарь вынесена лексика языка L^{WIQA}, ориентированная на её использование проектировщиками AC_i.

5 Средства псевдо-кодовой спецификации онтологических конструктов с алгоритмическим содержанием

Основу лексики языка проекта составляют лексические единицы, которые используются в спецификациях проектных прецедентов, а вернее, используются в представлении их моделей, в состав которых входят модели P^I и P^E. Для представления этих моделей в концептцуальном проектировании, то есть для их концептуального представления и специфицирования, в состав комплекса WIQA включены средства псевдо-кодового программирования. В каждой псевдо-кодовой программе специфицирован определённый объём использования естественного (или естественно-профессионального) языка в его алгоритмическом употреблении.

Отметим, что в типологии определений понятий выделяют, например, операциональные определения (процедуры измерения физических величин и других понятий), генетические определения (процедуры порождения определяемых объектов) и алгоритмические определения (алгоритмические схемы определяемых объектов).

Без использования понятий, содержание которых раскрывают определения отмеченных типов, невозможно создавать модели проектных прецедентов, так как:

- в проверках условий доступа нельзя полагаться на умозрительное оценивание того, что стоит за понятиями, входящими в ключи доступа $\{k_n\}$ и логические формулы $f^1(\{k_n\})$ и $f^2(\{k_n\})$, а следует использовать конструктивные вычисления их значений;
- реакции $r_q(t)$, кодируемые в моделях прецедентов, в общем случае состоят из совокупности действий $S(\{d_{pq}(t)\})$, связанных условиями их выполнения (например, последовательное выполнение действий, циклические выполнения, условные версии продолжения активности).

Особо важным является и то, что исполнение действий $\{d_{pq}(t)\}$ реакции $r_q(t)$, предусмотренных их описанием в модели прецедента, может осуществляться человеком или группой лиц, использующих инструменты (например, компьютеры) или нет. А значит, алгоритмическое представление $S(\{d_{pq}(t)\})$ для его исполнения в человеко-компьютерной среде (возможно с инструментами других типов) возможно, но оно должно ориентироваться на его исполнение связанной совокупностью «процессоров», включающих в самом простом случае человека, выполняющего роль «процессора» («интеллектуального процессора», I-процессора), и компьютерного процессора (К-процессора).

В предлагаемом подходе к специфицированию концептуализаций любой проектный прецедент представляется в виде нормативной модели (рисунок 1), визуальные формы которой ориентированы на взаимодействия с проектировщиками, выполняющими функции I-процессоров, использующих в таких взаимодействиях К-процессоры. Модели проектных прецедентов и их составляющих включаются в онтологию проекта в обязательном порядке. Именно по названным причинам в состав средств комплекса WIQA для представления алгоритмического содержания прецедентов введены средства псевдо-кодового программирования.

Комплекс средств псевдо-кодового программирования включает:

1) Язык псевдо-кодового программирования «WIQA», конструкты которого ориентированы на создание псевдо-кодовых программ, согласованно исполняемых в инструментальной среде WIQA совокупностью I-процессоров и К-процессоров, используемых в процессах концептуального проектирования АС. В выборе названия языка (псевдо-кодовый алгоритмический язык «WIQA») было учтено, что этот язык является встроенным алгоритмическим языком инструментария WIQA (инструментария, обслуживающего процессы концептуального решения задач). Кроме уже отмеченной специфики, язык WIQA позволяет: приписывать необходимую семантику традиционным типам данных; разрабатывать объектно-ориентированные псевдо-кодовые программы; разрабатывать прототипы программных решений, учитывающих обращения к базам данных. Так как имеется возможность настройки лексики языка «WIQA» на проект конкретной AC_i, то можно считать, что язык WIQA входит в состав языка L^{WIQA} как его составная часть, формализующая естественно-профессиональный язык (проекта) в его алгоритмическом употреблении.
2) Инструментальную среду псевдо-кодового программирования в следующем составе:

- Редактор псевдо-кодовых программ, обеспечивающий построение исходного кода программ на языке WIQA и его автоматическое преобразование в форму QA-протокола программируемой задачи. Адресация в протоколе (вопросно-ответная индексация операторов программы) может быть привязана проектировщиком к точке загрузки в дерево задач или к хранению программы в библиотеке псевдо-кодовых программ. Для объявления переменных используется вызов «Мастера объявлений», настроенный на традиционные типы данных, но позволяющий расширять атрибутику объявления проектировщиком (например, вводить атрибутику, определяющую семантические характеристики переменной).
- Компонента «Дополнительная атрибутика», предназначенная для объектно-реляционного преобразования QA-объектов, в которые вложены операторы псевдо-кодовой программы. Именно за счёт функционала этой компоненты проектировщик (выполняющий роль про-

граммиста на языке WIQA) может приписать любому QA-объекту дополнительные атрибуты, в частности атрибуты типа для специфицируемой переменной.

- Интерпретатор псевдо-кодовых программ, предоставляющий возможность I-процессору исполнять программу шаг за шагом, контролируя оперативные результаты исполнения операторов. Интерпретатор позволяет проектировщику выделить группу операторов программы и автоматически их выполнить. Исполнение программы (под контролем «Системы прерываний» инструментария WIQA) может быть прервано на любом шаге с возможностью возврата в «точку прерывания».
- Компилятор псевдо-кодовых программ, обеспечивающий выполнение программ K-процессором в соответствии с алгоритмов, вложенным в программу. К такому режиму исполнения определённой программы логично переходить в условиях, когда её исполнение в режиме интерпретации I-процессором доведено до автоматизма (многократное исполнение доведено до навыка).
- Компонента «Графический редактор диаграммных схем с интерактивностью», предоставляющая возможно «зарисовывать» интерфейсы для интерактивного связывания совокупностей псевдо-кодовых программ с возможностью передачи между ними данных. Редактор позволяет включать в комплексы программ такого рода необходимые exe-коды, если они есть и необходимы, а также вызов для демонстрации файлов, например doc-файлов.

Представленные возможности псевдо-кодовой спецификации концептуализаций полезны не только для представления алгоритмической семантики сложных концептов в онтологии проекта. С их помощью можно создавать исполняемы прототипы проектных решений, допускающие их имитационное моделирование, например, для проверки их *правильности*.

В завершение пункта и не более, чем с демонстрационными целями, представим одну из псевдо-кодовых процедур программы, обслуживающей диагностику и ремонт приборного блока:

Q 1.12 *PROCEDURE &Отказ_ВМС&*
Q 1.12.1 *ENDPROC &Отказ_ВМС&*
Q 1.12.2 *CALL &Отключить_прибор&*
Q 1.12.3 *CALL &Заменить_ВМС&*
Q 1.12.4 *CALL &Включить_прибор&*
Q 1.12.5 *INPUT &indSETVMS&*
Введите 1, если индикатор СЕТЬ на лицевой панели ВМС светится, 0, если не светится.
Q 1.12.6 *INPUT &indTGVMS&*
Введите 1, если индикатор ТГ на лицевой панели ВМС светится, 0, если не светится.
Q 1.12.7 *INPUT &indFACEVMS&*
Введите 1, если есть изображение на лицевой панели ВМС, 0, если нет.
Q 1.12.8 *IF (&indSETVMS& == 1) && (&indTGVMS& == 1) && (&indFACEVMS& == 0) THEN GOTO &UVS01FAIL&*
Q 1.12.9 *LABEL &UVS01FAIL&*
Q 1.12.9.1 *Неисправность: Неисправность УВС.01*
Q 1.12.9.2 *CALL &Отключить_прибор&*
Q 1.12.9.3 *CALL &Заменить_УВС_01&*
Q 1.12.10 *Неисправность: Неисправность ВМС*
Q 1.12.11 *CALL &Отключить_прибор&*
Q 1.12.12 *RETURN*

В представленной процедуре, и в проверке условий и в действиях, основная нагрузка ложится на человека (оператора, инженера в роли I-процессора), ответственного за надёжную работу блока, в составе определённой АС. Действия процедуры считываются I-процессором с экрана монитора и выполняются шаг за шагом (под контролем интерпретатора) в соответствии со складывающейся обстановкой в диагностике блока.

Заключение

В статье представлен комплекс средств спецификации концептуализаций, порождаемых и используемых в процессах проектирования автоматизированных систем. Основное внимание сосредоточено на средствах, обслуживающих концептуальный этап проектирования, на котором концептуальные ошибки наиболее опасны и дорогостоящи. Детали использования средств из интересов статьи (в основном из-за её объёма) исключены.

Для спецификации концептуализаций, порождаемых в процессе пошаговой детализации проектных задач, разработана система интерактивных вопросно-ответных форм, заполняемых информационным содержанием – дерево проектных задач, вопросно-ответная модель задачи, вопросно-ответное представление проектного решения. Особое место в спецификациях отведено типовой модели прецедентов, поскольку основным предназначением спецификаций концептуализаций является их повторное многократное использование как в процессе разработки АС, так и в процессах её эксплуатации. К числу важнейших задач повторного использования концептуализаций отнесены задачи их единообразного понимания членами группы разработчиков и другими лицами, заинтересованными в создании АС и участвующими в её эксплуатации.

Центральное место в спецификациях концептуализаций отведено онтологии проекта, в статьях которой регистрируются базовые концепты (определения лексики проекта) и другие концептуальные конструкты (состоящие из связанной совокупности базовых концептов), материализованные в разрабатываемой (разработанной) АС. Средства формирования базовых концептов онтологии ориентированы на предикативный анализ *правильности* употребления лексики в текстах постановок задач, формулировок проектных решений и ограничений. В процессе анализа используются предикатные формы простых предложений. Для формирования составных концептуальных конструкций разработаны средства синтеза их формул в логике предикатов, для построения которых используются предикатные модели сложных предложений, состоящих из двух простых. Для составных концептуальных конструкций предусмотрено их разбиение на две (связанные ссылкой) части, одна из которых включается в онтологию, а другая либо в библиотеку, либо в базу прецедентов.

Для формирования концептуальных конструктов с алгоритмическим содержанием разработан комплекс средств их псевдо-кодового программирования на специализированном языке «WIQA», встроенном (как и все предлагаемые средства) в инструментальный комплекс WIQA. Специфику языка определяет его настройка на формы спецификации, представленные выше. Потенциал языка «WIQA» достаточен для создания имитационных моделей проектных решений в объектно-ориентированном стиле с использованием доступа к моделям баз данных или их фрагментов.

Список источников

[1] Reports of the Standish Group. http:// www.standishgroup. com (дата обращения: 03.03. 2012).

[2] Software Intensive systems in the future. Final Report//ITEA 2 Symposium. http://symposium.itea2.org/ symposium2006/ main/publications/ TNO_IDATE_study_ ITEA_SIS_ in_the_future_Final_Report.pdf (дата обращения: 15.03. 2012).

[3] Westfall L. Software Requirements Engineering: What, Why, Who, When, and How. http://www.westfallteam.com/Papers/ The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf. (дата обращения: 20.03. 2012).

[4] Kendal, S.L., Creen, M. An introduction to knowledge engineering. - London: Springer, 2007. – 287 p

[5] Falquet G., Métral C., Teller J., Ch. Tweed Ch. Ontologies in Urban Development Projects (Advanced Information and Knowledge Processing). v. VIII, – Springer-Verlag, 2011. – 241 p.

[6] Кролл П., Крачтен Ф. Rational Unified Process – это легко. Руководство по RUP для практиков. [Текст]. – М.: Изд. Дом Кудиц-Образ, 2004. – 432 с.

[7] Соснин П.И. Вопросно-ответное моделирование в разработке автоматизированных систем. [Текст]. / П.И. Соснин – Ульяновск:УлГТУ,2007. – 333 с.

[8] Соснин П.И. Вопросно-ответное программирование человеко-компьютерной деятельности. [Текст]. / П.И. Соснин – Ульяновск:УлГТУ,2010. – 240 с.

[9] Precedent. URL: http://dictionary.reference.com/browse/precedent. (дата обращения: 11.03. 2012).

[10] Sosnin P. Question-Answer Shell for Personal Expert System..// Chapter in the book “Expert Systems for Human, Materials and Automation.” Published by Intech, 2011. – pp. 51-74.

Сведения об авторах

Соснин Пётр Иванович, 1945 г. рождения, д.т.н., профессор, заведующий кафедрой "Вычислительная техника" Ульяновского государственного технического университета. Член Международной академии информатизации, Российской и Европейской Ассоциаций искусственного интеллекта, член IEEE и Computer Society, член международного общества WSEAS, председатель Ульяновского отделения Российской ассоциации искусственного интеллекта, эксперт Министерства промышленности, науки и технологий, рецензент "Journal of Intelligent and Fuzzy System". Автор более чем 290 публикаций, в том числе в 9 монографий и 6 учебных пособий.

Peter Ivanovich Sosnin, 1945, doctor of technical sciences, professor, head of the department "Computer Science", Ulyanovsk State Technical University. Member of the International Academy of Informatisation, a member of the Russian and the European Associations of Artificial Intelligence, a member of IEEE and Computer Society, a member of international society WSEAS, chairman of the Ulyanovsk Branch of the Russian Association of Artificial Intelligence, Expert of Ministry of industry, Science and Technology, a re-viewer of "Journal of Intelligent and Fuzzy System". The research results are presented in more than 290 publications, including 9 monographs and 6 textbooks.

Маклаев Владимир Анатольевич, генеральный директор Федерального научно-производственного центра Открытое акционерное общество "Научно-производственное объединение "Марс", к.т.н.

Vladimir Anatolievich Maklaev, candidate of technical sciences , Managing Director, Federal Research-and-Production Center Open Joint-Stock Company "Research-and-Production Association "Mars"