OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ПРЕЦЕДЕНТНО-ОРИЕНТИРОВАННАЯ БАЗА ОПЫТА ПРОЕКТНОЙ ОРГАНИЗАЦИИ

Маклаев В.А.[*] Соснин П.И.[**]

[*] *Федеральный Научно-Производственный Центр ОАО «НПО «МАРС», г. Ульяновск, Россия*

mars@mv.ru

[**] *Ульяновский государственный технический университет, г. Ульяновск, Россия*

sosnin@ulstu.ru

В статье представлена модель коллективного опыта проектной организации, разрабатывающей семейство автоматизированных систем. Специфику модели определяет репозиторий активов, для формального представления которых используется продукционная и вопросно-ответная логики. Требования и спецификации, вложенные в реализацию модели, согласованы с базовыми стандартами на разработку автоматизированных систем, интенсивно использующих программное обеспечение.

Ключевые слова: база опыта, вопросно-ответный, прецедент, псевдопрограммирование.

ВВЕДЕНИЕ

На настоящий момент времени наименее освоенным типом инженерной деятельности считают «разработку систем, интенсивно использующих программное обеспечение (Software Intensive Systems, SIS)» [Software, 2011]. В качестве основания для такого утверждения принято приводить чрезвычайно низкую степень успешности (около 30%), которая с 1994 года (каждые два года) регистрируется Корпорацией Standish Group [Reports, 2011] в США и многократно подтверждалась и подтверждается другими исследователями [Charette, 2005].

За проблемой успешности стоят ежегодные потери в сотни миллиардов долларов, что, разумеется, требует анализировать причины негативов и средства их устранения или снижения, а также искать то новое, ещё не применявшееся в разработках SIS, что будет способствовать повышению успеха такой деятельности. В попытках решить проблему уже созданы, проверены и освоены разнообразные полезные средства, к числу которых, например, относится инженерный опыт, представленный в стандартах.

Можно называть и перечислять другие уже предложенные средства, но статистика разработок SIS продолжает подтверждать, что проблема успешности сохраняется и её основной причиной являются негативные проявления человеческого фактора в условиях коллективной интеллектуальной деятельности.

По глубокому убеждению авторов, одним из принципиальных направлений повышения успешности разработок систем класса SIS, в который входят автоматизированные системы (АС), является **моделирование коллективного опыта**, которое должно сохранять **сущность индивидуального опыта** членов коллектива. В статье представляется текущее состояние разработки комплекса средств «База опыта проектной организации» (в дальнейшем БАЗА ОПЫТА), специфику которого определяют ориентация на прецеденты, их вопросно-ответное моделирование и псевдопрограммирование в условиях создания организацией семейств АС. Комплекс создаётся как открытая информационная система, развитие которой согласовано с управляемым (запланированным) повышением профессональной зрелости производственных процессов и коллектива проектировщиков.

1. Исходные предпосылки

В проектных организациях, разрабатывающих семейства **АС**, для достижения успеха открывается возможность для совершенствования не только инвариантных составляющих проектной деятельности, но и специализированных составляющих, обусловленных спецификой семейства. На первый план выходит «**повторное использование** (**reuse**)», за осуществление которого

отвечает **уникальный опыт коллектива проектной организации**.

Опыт индивида – это **природно-искусственный феномен**, проявляющий своё существование через действия человека по их успешным образцам в прошлом. За этим феноменом стоит интеллектуальная обработка «условных рефлексов» (как определённого рода экспериментов, experience) для их управляемых повторных применений в будущем. Такое природное образование является динамическим и локализовано в мозговых структурах индивида.

Опыт индивида неотделим от моделей опыта, которые он освоил как член человеческого общества. Именно в этом смысле феномен опыта содержит искусственную составляющую. Модели опыта осваиваются индивидом в процессах взаимодействия с теми их источниками, с которыми он сталкивался в своей жизни. В рассматриваемом случае в число таких источников входит согласованная коллективная профессиональная деятельность членов проектной организации.

Уникальный опыт коллектива проектной организации представляет собой интегрированное образование, включающее индивидуальный опыт членов коллектива и ту совокупность моделей опыта, которая используется в разработке семейства АС, включая модели опыта созданные в такой работе. Легко согласиться с тем, что успешность проектной организации принципиальным образом зависит от (эффективности) интеграции опыта, существующей в организации.

Другими словами, одной из наиболее важнейших составляющих организации коллектива является **связывание индивидуального опыта в коллективный опыт**, для осуществления которого используют **модели опыта** в разных версиях **(язык** предметной области семейства продуктов, **инструментарий** проектирования, **онтология** проектов, **концептуальные схемы**, **руководства** разных типов, **схемы потоков рабо**т и другие модели).

В число основных требований к разрабатываемой БАЗЕ ОПЫТА была включена следующая установка: **модель коллективного опыта**, связывающая в единое целое опыт членов коллектива, должна **представлять и интегрировать единицы опыта подобно** тому, как такие единицы представляются и интегрируются в **опыте индивида**.

Если **представление и интеграция единиц индивидуального и коллективного опыта подобны**, то (в рамках подобия) для индивида его **взаимодействие**, как с моделью своего опыта, так и с моделями опыта других индивидов (в рассматриваемом случае членов коллектива), будет осуществляться по **единым схемам**. В этом случае модель коллективного опыта образует **расширение модели индивидуального опыта**. Вторая установка, положенная в основу моделирования опыта, связана с **сохранением самого существенного в представлении единиц опыта** как индивидуального, так и коллективного.

В основе опыта лежат прецеденты, с каждым из которых связан интеллектуально обработанный объём определённой активности, моделирующий определённую единицу опыта. Моделирование опыта в мозговых структурах человека является прецедентно-ориентированным

Любой **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.

Обобщением вышесказанного с учётом первой установки является следующее требование к интеграции опыта: в **основу модели коллективного опыта**, материализованной в компьютерной среде, должны быть положены **модели прецедентов**, согласованные с моделями прецедентов в мозговых структурах человека.

Прецеденты и их модели обеспечивают повторное реагирование по успешным образцам [Precedent, 2011], что имеет принципиальное значение для проектирования, особенно при создании семейств продуктов (в рассматриваемом случае **семейств АС**).

В моделировании коллективного опыта, учитывающей специфику семейств **АС**, авторы ориентировались на стандарт FSPLP (V5.0), разработанный в Институте Программной Инженерии (Software Engineer Institute, **SEI**) Университета Карнеги-Меллон. В этом стандарте для выделения единиц повторного использования используется понятие «**актива**», референты которого могут иметь различную природу.

Особо важным в стандарте является то, что в нём называются и растолковываются те **активности, которые способствуют повышению успеха в создании линеек программных продуктов**. Эти активности, применяя аналогии и адаптацию, можно использовать и для повышения успешности в разработках семейств **АС**.

2. Средства моделирования опыта

Ориентируясь на прецеденты, к повторному использованию можно подготовить любые активности, полезные в разработках **АС**. Такая подготовка требует определиться с **моделями коллективной работы** на любом этапе жизненного цикла **АС**. В современных технологиях разработок SIS и АС коллективная работа обычно представляется в форме **потоков работ**. Наиболее полно и последовательно потоки работ применяются в технологии Rational Unified Process (**RUP**) [Kroll, 2003].

В основу комплекса средств формирования и использования модели коллективного опыта положена специализированная система потоков работ «**Взаимодействие с опытом**», исполнение каждого из которых в инструментально моделирующей среде WIQA (Working In Questions and Answers) [Sosnin, 2009]. Специфику инструментария WIQA определяет его ориентация на конструктивное использование в процессах решение задач рассуждений вопросно-ответного типа (другими словами, ориентация на диалоговые механизмы решения задач). Можно считать, что WIQA инструментально обеспечивает построение концептуального решения задачи в виде его разложения в базисе вопросов и ответов.

Выбор вопросно-ответного базиса для реализации потоков работ «**Взаимодействие с опытом**» обусловлен тем, что **сознание** человека является **диалогичным по своей природе**, поскольку оно **специализируется на решениях задач доступа к моделям прецедентов**, локализованным в его мозговых структурах. А в соответствии с первой установкой взаимодействия с моделями коллективного опыта должны сохранять самое существенное из взаимодействия индивида с освоенным им опытом.

Компонентная структура базового комплекта инструментария WIQA обобщённо представлена на рис. 1, на котором (без их распределения между серверной и клиентской частями) отражены основные функциональности.

Рисунок 1 – Компонентная структура базового комплекта

Для создания базы опыта (в зависимости от структуры проектной организации, текущего фронта разработок и существующей корпоративной сети) необходимо развернуть, настроить и связать в единое целое определённое число базовых комплектов. Каждый развёрнутый комплект будет выполнять функции клиент-серверного приложения, обеспечивающего работу определённой группы членов проектной организации. Один из комплектов должен выполнять функцию репозитория базы опыта. Информационная связность комплектов в единое целое обеспечивается за счёт механизмов репликации.

Базовыми структурами данных репликации являются «дерево задач» и «вопросно-ответная модель задачи», специфика и связность которых отражены на рис. 2.

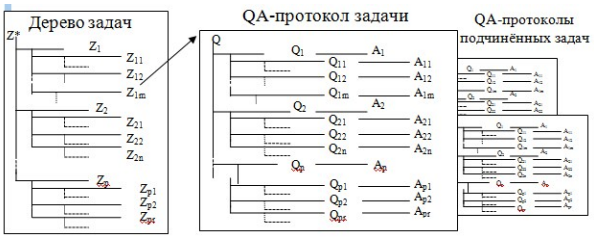


Рисунок 2 – Вопросно-ответные структуры данных

Специфику представленных структур определяет атрибутика объектов моделирования, называемых «задачами», «вопросами» и «ответами», которые проявляют себя в процессах формирования, структуризации, систематизации и использования индивидуального и коллективного опыта. За каждой «задачей» стоит выявление соответствующего ей «прецедента» и его интеллектуальное освоение, включающее решение задачи и завершающееся введением «модели прецедента» в «модель опыта», в рассматриваемом случае в базу опыта. Любая «задача» понимается как тип вопроса, ответ на который связан с решением задачи.

За любым «вопросом» стоит обращение к индивидуальному или коллективному опыту, активизированное, в рассматриваемом в статье случае, ситуациями проектирования, затребовавшими обращение к доступному опыту. В этом плане «вопрос» - это природно-искусственный феномен, важнейшей искусственной составляющей которого является его знаковая модель в виде «формулировки вопроса» на языке, сопровождающем процесс проектирования АС. У «вопроса» как у любого природного образования имеются определённые свойства и отношения с его окружением, которые в системе WIQA представляются определённой атрибутикой и операциями над составляющими атрибутики, которые следует понимать как операции над «моделями вопроса». Всё сказанное про представления «вопросов» справедливо для «ответов», поскольку любой «вопрос» с соответствующим ему «ответом» представляют единое «вопросно-ответное целое».

Обобщая вышесказанное, подчеркнём, что структуры данных, представленные на рис.2, вместе с операциями над такими данными выполняют функции **интерфейса** между проектировщиками и их индивидуальным и коллективным опытом, а также опытом, который оперативно ими привлекается из любых полезных источников, находящимся за рамками опыта проектной организации.

Вопросно-ответный интерфейс, обеспечиваемый системой WIQA, активно обслуживает доступ, как к опыту проектировщиков, так и к моделям опыта, вложенным в базу опыта проектной организации. Средства вопросно-ответного интерфейса активно

используются в потоках работ «Взаимодействие с опытом». Именно такое понимание вложено в статье в спецификатор «вопросно-ответный» (Question-Answer, или короче QA).

3. Моделирование прецедентов

3.1. Типовые активы

Одним из основных понятий семейства АС являются «активы», под которыми понимают любые конструкты и средства предшествующих разработок АС, подготовленные для их повторных использований в будущих проектах. К типовым активам относятся следующие их виды:

- реализованные проекты, размещённые в архиве проектов;
- исполняемые проекты, каждый из которых при наличии разрешения на доступ открыт для заимствований из других исполняемых проектов;
- библиотека типовых нормативных документов;
- типовые архитектурные модели;
- стандарты и руководящие принципы;
- образцы;
- схемы моделирования;
- структуры программного обеспечения
- пакеты исходного текста;
- выполнимые компоненты;
- интерфейсные образцы;
- схемы протоколов;
- общая модель области, на которой построены другие модели;
- инструменты, технологии, платформы, инфраструктура.

Эффективность повторного использования представленных конструктов в существенной мере зависит от их представления, включающего идентифицирующую и сопроводительную информацию, например:

- имя, версии и варианты;
- отношения совместимости с предыдущими версиями;
- определенные, но не слишком многочисленные отношения с другими активами;
- идентификатор (или другая версия) адресного входа в каталог с разумными сроками поиска (так, чтобы проектировщики могли найти актив),
- документация для пользователей (сжатая, но достаточная, особенно относительно того, как специализироваться или параметризовать выбранный актив);
- образцы тестовых проверок в условиях комплексирования с другими активами.

Активы принято размещать в специальных репозиториях, которые часто называют «фабриками опыта» (Experience factory) [Tian, 2011], в которых, обычно, унифицируется представление активов в каталоге репозитория. А для каждого типа актива создаётся специализированная библиотека с собственными средствами доступа.

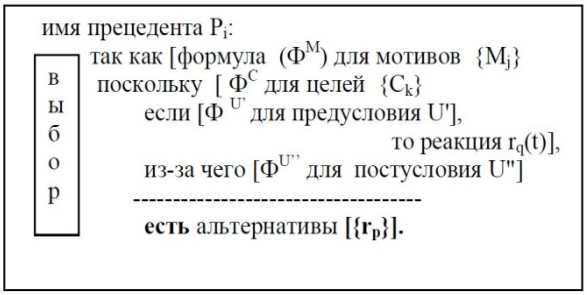
Представляемая БАЗА ОПЫТА также относится к классу «фабрик опыта», но отличается от известных хранилищ активов тем, что унификация в существенной мере распространена за рамки каталогов. Унификация распространена на задачи, которые помогают решать активы и на задачи, которые приходится решать пользователю, применяющему каждый выбранный им актив. Более того, все взаимодействия с базой опыта унифицированы в форме потоков работ, система которых названа «Взаимодействие с опытом». В названных средствах унификации центральное место занимают модели прецедентов.

3.2. Логическая схема прецедента

Специфику моделирования прецедентов для их представления в БАЗЕ ОПЫТА определяют:

- унифицированная логическая схема прецедента, ориентированная на представление и выбор прецедента с учётом его ценностных характеристик, конкретнее с учётом позитивов, которые достигаются от применения прецедента;
- связывание каждого прецедента с задачей, в результате решения которой была создана его модель;
- выделение в формировании (интегральной) модели прецедента подчинённых моделей, каждая из которых представляет состояние жизненного цикла прецедента;
- псевдокодовое программирование, по крайней мере, тех составляющих модели прецедента, в исполнении которых принимает участие человек (проектировщик).

Представление отмеченной специфики начнём с логической схемы прецедента, которая имеет следующий вид:



Ценностное содержание прецедента в этой схеме представляет учёт мотивов и целей, содержание которых фиксируется, например, в следующих формах «повышает точность ...., за счёт....», «снижает стоимость ...., за счёт .....», «предотвращает ошибки ....., обусловленные ....». В простейшем виде логическая формула Φ^M или Φ^C может быть подобна списку, с которым сопоставляются ценностные ожидания пользователя.

Одним из применений ценностных характестик является решение задачи выбора прецедента на

множестве альтернатив. Для обеспечения выбора в состав инструментальных средств включена библиотека методов принятия решений.

3.3. Интегральное представление прецедента

Включение очередной модели прецедента P_i в базу опыта осуществляется в определённый момент времени t_{ri} его жизненного цикла , который начинается с формулировки его исходной постановки задачи $P^T_i(t_{0i})$.

По ходу жизненного цикла постановка задачи изменяется $P^T_i(t_{0i})$, $P^T_i(t_{1i})$, ... , $P^T_i(t_{ki})$, исправляясь и детализируясь, отражая тем самым интеллектуальное освоение прецедента. Общий случай интеллектуального освоения прецедента свяжем с состояниями, а вернее со специализированными моделями прецедента, представленными на рис. 3.

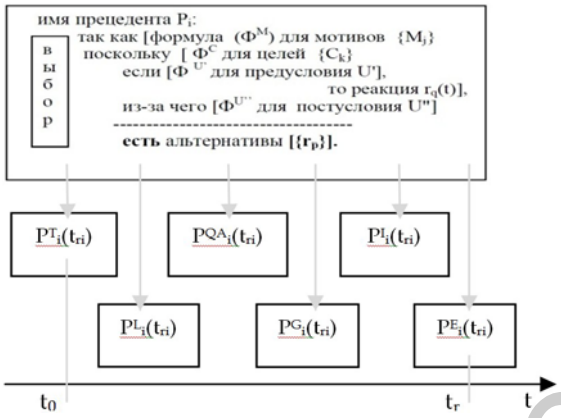


Рисунок 3 – Состояния жизненного цикла прецедента

4. Псевдокодовое программирование

4.1. Дополнительная атрибутика

В число полезных расширений потенциала QA-данных входит механизм дополнительной атрибутики, предназначенный для расширения атрибутики, вызванного любыми причинами (например, разработка новых плагинов или создание пользовательских QA-программ), причём, такого расширения, в котором не требуется вводить в QA-базу данных новые отношения или изменять уже существующие.

Такое расширение обеспечивает служебный плагин, получивший название «**Дополнительная атрибутика**» (**Additional Attributes, AA**). Разумеется, этот плагин, вводит в базу данных собственные дополнительные отношения, которые открывают возможность для **творческого приписывания** свйств любой единице дерева задач или любой единице QA-протокола, но они используются для эмуляции очередных полезных отношений.

В основу расширения набора (базовых) атрибутов любого из объектов типа «вопрос» или «ответ» положены механизмы объектно-реляционного преобразования, в результате которых для «объекта», хранимого в базе данных, создаётся его версия, адаптированная к выбранному языку программирования (в рассматриваемом случае к языку C#). В таком преобразовании можно расширить атрибутику и операционную часть (методы объекта).

По сути дела плагин **AA** предоставляет создателям объектов, использующих его средства, упакованные в объекты (на языке **C#**) атрибуты реляционных отношений, дополненные специализированными атрибутами. Плагин также обеспечивает связывание «**упакованных атрибутов**» в комплексы, а также предоставляет интерактивный доступ к таким конструктам.

Как отмечалось выше, функциональность плагина определяет набор отношений AA, дополнительно включённых в QA-базу, в частности, для эмуляции очередных отношений (виртуальных отношений), полезных для развития функционального потенциала инструментария WIQA или приложения, созданного с помощью инструментария WIQA. Объектно-реляционное развитие информационного потенциала QA-базы схематично представлено на рис. 4.

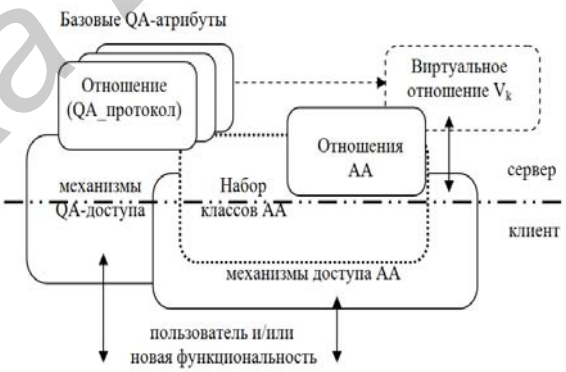


Рисунок 4 – Объектно-реляционное развитие базы данных

Набор отношений AA выбран и материализован из расчёта на объявление оказавшегося необходимы дополнительного атрибута AA_i, возможно включающего подчинённые атрибуты AA_{ij}, с учётом типа и необходимых свойств, причём, принадлежащих не только виртуальному отношению V_k. Созданный атрибут AA_i связывается с определённым узлом дерева задач или вопросно-ответного протокола или узами тех конструктов, которые QA-данные представляют.

Введение средств дополнительной атрибутики открывает возможность для создания из любого объекта QA-базы «перфокарты», на которой можно «записать» и специфицировать любой декларативный оператор (для объявления переменных любого типа) или любой императивный оператор (например, для представления операторов языка псевдокодов).

4.2. Средства псевдокодового программирования активностей

Ничто не мешает использовать структуры данных, представленные на рис. 2, для создания с их помощью информационных объектов, представляющих определённые единичные объекты (константы) или их классы (переменные) с помощью необходимой совокупности пар «ответов» и «вопросов».

В этом случае для именования объектов логично использовать «вопросы» (имя ставит вопрос о значении), а для представления их значений - «ответы». В дальнейшем тексте такие информационные объекты будем называть QA-данными, поскольку для их создания используются объекты QA-базы

Семантический потенциал QA-данных (как класса информационных объектов) существенно увеличивается за счёт присоединения к базовой атрибутике QA-данных («уникальный индекс – адрес», «текстовое описание», «тип», «автор», «время», «состояние завершённости» и другие атрибуты как вопросной, так и ответной частей) необходимых пользователю (проектировщику) дополнительных атрибутов.

Возможности создания и использования QA-данных были проверены в ряде приложений, созданных на базе инструментально-моделирующей среды WIQA, в частности в «Системе многоагентного моделирования окружающей обстановки судна».

Важнейшим результатом таких проверок оказалось то, что опыт, накопленный в использовании QA-данных, привёл к созданию, встроенного в инструментарий WIQA, комплекса средств псевдокодового программирования, ориентированного на прецеденты. Взаимодействие проектировщика с этим комплексом обобщённо представлено на рис 5.

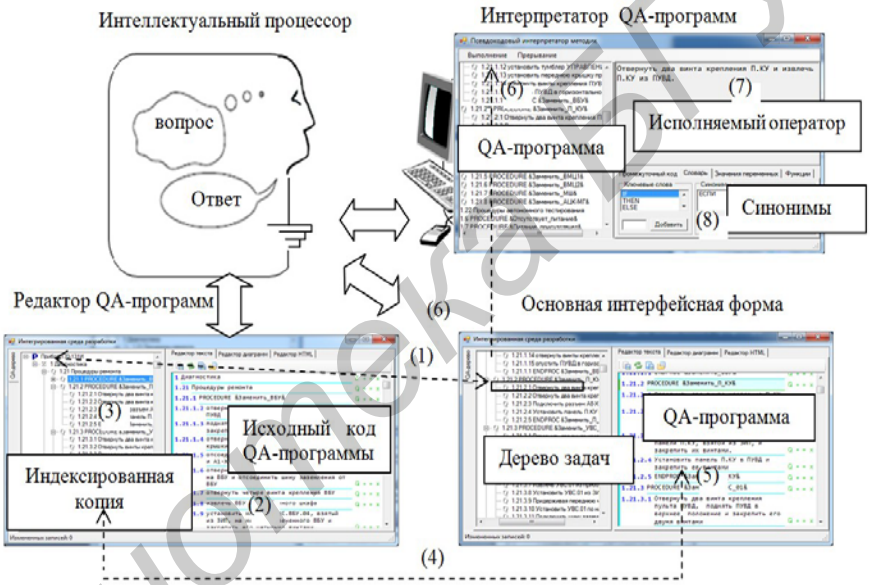


Рисунок 5 – Инструментальная среда псевдокодового

Создание в этой среде определённой псевдопрограммы (QA-программы для того, чтобы отличить от псевдопрограмм других типов) начинается с выбора «точки» в дереве задач и объявления новой задачи. Индексное имя новой задачи (1) будет использоваться как начальный адрес для приписывания индексных имён каждому оператору будущей программы, записанному в области (2) текстового редактора. Индексированная копия редактируемой QA-программы регистрируется в рабочей памяти редактора, а её копия (3), форма которой тождественна основной форме втизуализации QA-данных, визуализируется в левом окне редактора. После любого сохранения исходного кода в редакторе текущее состояние кода переносится (4) в QA-базу и визуализируется в области (5) основной интерфейсной формы. Любая QA-программа, хранимая в дереве задач в любом состоянии, в любой момент времени может быть загружена в редактор для продолжения её редактирования, исправления ошибок или для адаптации кода.

Любая QA-программа в любом её состоянии может быть загружена (6) в интерпретатор для исполнения. Каждый исполняемый оператор загруженного кода визуализируется в специальной области (7). В любой момент времени проектировщик может ввести (объявить) новый синоним для выбранного ключевого слова.

Реальность проектирования – параллельная работа проектировщика с совокупность задач. Для обеспечения такой работы в инструментарий WIQA всроена система прерываний, позволяющая превать любую исполняемую QA-программу и перейти к другой задаче (работе). Система прерываний поддерживает возврат в точку прерываний прерванной QA-программы.

4.3. Образцы QA-программ

В инструментальной среде, представленной на рис. 5, проектировщик специфицируется как «интеллектуальный процессор». Такая спецификация отражает тот факт, что основным исполнителем QA-программы является человек, считывающий оператор за оператором и выполняющий их действия, возможно, какие-то и без компьютера.

«Интеллектульный процессор» - это «субъект», создающий для решения задач псевдопрограммные модели прецедентов (и их совокупностей) и исполняющий такие псевдопрограммы вместе и согласованно с компьютерным процессором или процессорами. Имитируя роль процессора, человек инициирует переход к очередному оператору псевдопрограммы.

Примером такого вида псевдопрограмм может служить методика (прецедент) «Переустановить Outlook Express», в которой приведена только вопросная часть каждого оператора, помеченная символом «О», а не «Q»:

QA-Program: ***Установка Microsoft Outlook***
О1. Закройте все программы.
О2. На ***Запустить*** *меню, нажмите кнопку* ***Запустить****.*
03. В ***Открыть*** *поле, тип* ***regedit****, а затем нажмите кнопку* ***ОК****.*
О4. Переместить и выделите следующий раздел:
HKEY_CURRENT_USER/Software/Microsoft/Office/9.0/Outlook/
О5. В имени списка выбор ***FirstRunDialog****.*
О6. ***If*** *требуется включить только* ***Добро пожаловать в Microsoft Outlook*** ***Then*** ***Н****ажмите меню Правка* ***Изменить***
*О6.1. Введите****Значение true****в поле значение,*
*О6.2. Затем нажмите кнопку****ОК****.*
О7. ***If*** *вы хотите повторно создать все элементы вашему вниманию примеры* ***Then*** *переместить и выделите следующий раздел:*
HKEY_CURRENT_USER/Software/Microsoft/Office/9.0/Outlook/Setup
О8.В списке ***Имя*** *выберите и удалите следующие разделы:* ***Create Welcome*** *и* ***Первый запуск****.*
О9. В подтверждение удаления параметра диалоговом окне, нажмите кнопку ***Да*** *для каждой записи.*
*О10. Меню в реестр, нажмите кнопку****Выход****.*
О11. ***End***

Ответ на вопрос оператора фиксируется как факт его выполнения (интерпретатор регистрирует символ «*» в текстовом поле соответствующего «ответа»). Для QA-функций интерпретатор в поле «ответа» регистрирует вычисленное значение функции.

Напомним, что псевдопрограммирование, причём в вопросно-ответных формах, нацелено авторами на управление активностью проектировщика (интеллектуального процессора), создающего и использующего прецеденты. Разумеется, в своей работе, если это полезно, он будет использовать компьютер, но как подчинённый процессор.

Для того чтобы продемонстрировать некоторые детали языка псевдопрограммирования приведём фрагмент ещё одной QA-программы:

O 1.11 Procedure &DiscardPriority&
O 1.11.1 &P& := &Pmax&
O 1.11.2 Label &DP1&
O 1.11.3 &Priority& := &P&
O 1.11.4 CALL &GetTaskByPr&
O 1.11.5 &base& -> &TaskPriority& := &base& -> &TaskPriority& + 1
O 1.11.6 CALL &ChangeTask&
O 1.11.7 &P& := &P& - 1
O 1.11.8 IF &P& < &base& -> &Pmin& THEN &base& -> &NewPriority& := &Pmin& ELSE GOTO &DP1&
O 1.11.9 ENDPROC &DiscardPriority&

Процедура используется в Системе прерываний комплекса WIQA для вычисления приоритета прерванной QA-программы. Система прерываний специально создана с помощью средств псевдопрограммирования. Цель – определить потенциал созданных средств псевдо-программирования. Задачи такого рода уже не интерпретируются, а компилируются с помощью Компилятора псевдопрограмм, встроенного в инструментарий WIQA.

4.4. Псевдопрограммирование в прецедентах

В псевдопрограммировании прецедентов следует различать кодирование условий доступа и кодирование реактивной части. Общий случай программирования доступа представляет следующая схема:

OA-PROGRAM_1 (функциональный тип):
O1. Строка @Переменная V_1 @ Строка, Комментарий?
A1.Значение V_1.
O2. Строка @Переменная V_2 @ Строка, Комментарий?
A2. Значение V_2.
……
ON. Строка @Переменная V_M @ Строка, Комментарий?
AN. Значение V_M.
ON+1. Выражение_W1 (V_1, V_2, ..., V_M)?
AN+1. Значение выражения W1.
O N+2. Выражение_W2 (V_1, V_2, ..., V_M, W1)?
A N+2. Значение W2.
.....
O N+P. Выражение_WP(V_1, V_2, ..., V_M, W1, ..., WP-1),?
A N+P. Значение WP. (результат проверки условия)
End

Типовая **QA**-программ «условия» указывает на то, что в общем случае её экземпляр может включать связную совокупность алгебраических и логических выражений, используемых в оценках правомерности и полезности включения прецедента в текущую или предполагаемую активность человека. Так, что на самом деле за «условием» стоит совокупность взаимосвязанных алгебраических и условных функций, причём, одной из разновидностей содержания условных функций является «оценивание».

В то же время, так как формирование и проверки «условия» нацелены в основном на решение задач доступа к «реакции» и её выбора с позиций альтернатив и полезности, то главным для «условия» является устойчивое распознавание адекватного прецедента.

Для решения такой задачи обычно пригодны различные версии «условий» доступа, одна из которых обычно получает преференции и совершенствуется в многократных обращениях к прецеденту (обучается на примерах). А значит, для псевдокодовых программ «условий», исполняемых интеллектуальным процессором, принципиально важным является возможность их оперативного совершенствования (изменения) и исполнения.

Перейдём к псевдокодовому программированию «реакций», исходный псевдокод которой формируется как «процедура», поскольку за реакцией стоит поведение, состоящее из связной совокупности действий. Операторы «реакции» включают в общем случае целенаправленную совокупность действий двух типов, один из которых требует активности интеллектуального процессора, а второй – активности компьютерного процессора.

Типовая схема смешивания действий, исполняемых процессорами двух типов, имеет следующий вид:

QA-PROGRAM_2 (процедурный тип):
O1.K_i, K_j, …, PL_k ?
*A1. **
O2. K_m, QA-P_n, …, K_q?
*A2.**
……
ON. K_s, Pl_t, …, QA-P_v?
AN. #
End.

где ***K_i*** – команда, активизируемая пользователем в акте интерактивного взаимодействия с компьютером; ***PL_k*** – вызов пользователем плагина и переход к его функционалу; ***QA-P-*** вызов **QA**-программы.

4.5. Структура БАЗЫ ОПЫТА

В структурном плане БАЗА ОПЫТА представляет собой связную совокупность (Рис. 3) сетевых приложений WIQA.Net, каждое из которых обслуживает работу определённой группы проектировщиков [Соснин, 2010]. Связность обеспечивается как на уровне взаимодействий между группами, так и за счёт доступа к общему для всех групп репозиторию.

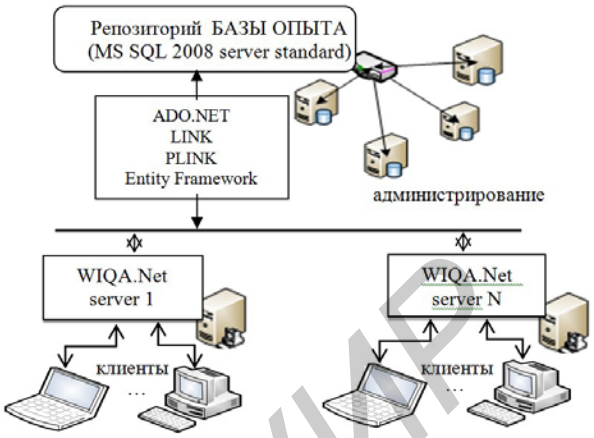


Рисунок 6 – Структура БАЗЫ ОПЫТА

Каталог активов (с учётом их типологии) в репозитории реализуется с помощью связной совокупности моделей, базовый тип которых приведён на рис. 2. В информационных потоках используются механизмы репликации и синхронизации. Типовой единицей обмена между репозиторием и серверами групп является представление актива, с каждым из которых связана «задача» и соответствующая задаче псевдопрограмма или связная совокупность псевдопрограмм.

5. Заключение

Создание предложенной в статье БАЗЫ ОПЫТА, аккумулирующей модели активов проектной организации, и её рациональное внедрение в проектную деятельность способно привести к увеличению производительности труда проектировщиков, существенному повышению качества проектов, уменьшению их стоимости. Внедрение БАЗЫ ОПЫТА в процессы проектирования согласовано с современными стандартами и практикой разработки семейств систем, которая подтверждает достижимость отмеченных позитивов.

К специфике БАЗЫ ОПЫТА относится комплексирование требований, извлекаемых из российских и международных стандартов для систем, интенсивно использующих программное обеспечение, а также применение для комплексирования инструментария вопросно-ответного моделирования задач.

В настоящий момент времени работы по созданию БАЗА ОПЫТА продолжаются. Основные проектные решения, которые вложены в БАЗУ ОПЫТА, прошли проверку на задачах концептуального проектирования АС, проектного документирования, информационной безопасности, контроля и диагностики изделий, технологической

подготовки производства, а также на предметных задачах многоагентного моделирования и экспертного мониторинга окружающей обстановки морского судна.

Потоки работ «Взаимодействий с опытом» прошли этап опытной отладки. Сборка и испытания БАЗЫ ОПЫТА как целого запланированы на 2012 год.

Библиографический список

[Соснин, 2011] Соснин, П.И. Вопросно-ответное программирование человеко-компьютерной деятельности / П.И. Соснин – Ульяновск:УлГТУ,2010. – 240 с.

[Charette, 2005] Charette, R.N. Why software falls // IEEE Spectrum, vol. 42, #9, pp. 36-43, 2005.

[Kroll, 2003] Kroll, P. and Ph. Kruchten, The Rational Unified Process Made Easy: A Practitioners Guide to the RUP / Addison-Wesley, 2003.

[Precedent, 2011] Precedent. – Режим доступа: http://dictionary.reference.com/browse/precedent. – дата доступа: 18.11. 2011)

[Reports, 2011] Reports of the Standish Group – Режим доступа: http:// www.standishgroup.com. – дата доступа: 18.11. 2011.

[Software, 2011] Software Intensive systems in the future // *Final Report//ITEA 2 Symposium,* – Режим доступа: http://symposium.itea2.org/symposium2006/main/publications/TNO_IDATE_study_ITEA_SIS_ in_the_ future_Final_Report.pdf – дата доступа: 18.11. 2011

[Sosnin, 2009] Sosnin, P.I. Means of question-answer interaction for collaborative development activity // Hindawi Publishing Corporation, Advances in Human-Computer Interaction. – 2009 – Vol.2009, Article ID 619405, 18 pages

[Sosnin, 2011] 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.

[Tian, 2011] Tian, J. An Emerging Experience Factory to Support High-Quality Applications Based on Software Components and Services// Journal of Software, Vol. 6, No. 2, 2011, pp. 14-32.

PRECEDENT-FOCUSED BASE OF EXPERIENCE OF THE DESIGN ORGANIZATION

V.A. Maklaev*, P.I. Sosnin**

Federal Science-Production Centre "MARS", Ulyanovsk, Russia

mars@mv.ru

Ulyanovsk State Technical University, Ulyanovsk, Russia

sosnin@ulstu.ru

In paper the model of collective experience of the design organization developing the family of automated systems is presented. Specificity of the model is defined by a repository of reuse actives for which the formal representation of precedents and question-answer logic are used. Requirements and the specifications enclosed in realization of the model are coordinated with basic standards on the development of the automated systems intensively using the software.

Introduction

Nowadays the most problematic area of computer applications is "Development of Software Intensive Systems (SIS)", within the frame of which the collaborative works of developers and other stakeholders are being fulfilled in corporate networks. "**A software intensive system is a system where software represents a significant segment in any of the following points: system functionality, system cost, system development risk, development time"**.

The significant number of SIS developments (about 65 percent) either is being stopped, or is exceeding planned time and/or finance, or reach the end in the poorer version. Such situation indicates that developers have not received very important means for successful developing the SIS till now.

In accordance of the author our opinion one of the problems of successful designing is caused by natural restrictions of designer intelligence which can be overcome with the help of adequate modeling of the human experience adjusted to the collective designing in the corporate network.

By the other words the developers of SISs need the effective means for adequate defining of the essential experience units, their modeling for achieving the necessary understanding and also for testing the units in appropriate conditions of designing and using. The named experience units are to be distinguished, defined, modeled, understood, coded and tested as precedents. "**Precedents are actions or decisions that have already happened in the past and which can be referred to and justified as an example that can be followed when the similar situation arises**"

In article the precedents models are used for the creation and usage the base of experience models of the organization developing the family of SISs. The specificity of the suggested solutions is being defined by the question-answer approach (QA-approach) to the work with the experience and its models. This work supports by the special instrumental system named WIQA (Working In Questions and Answers). The choice of the QA-approach to the experience base is explained that consciousness has a dialog nature the existence and process of which are being opened via the usage of the natural language.

Main Part

The positive changes in the development of the SIS can be connected with the creation and usage of means for programming the designer interactions with the experience and its models in the conceptual designing. The degree of the successfulness in the conceptual designing will be positively increased if any designer will play the role of the "processor" which executes the programs managing his(her) activity in the appropriate instrumental medium. Such role of the designer was named "intellectual processor". This role is being supported by means of WIQA.

The system WIQA has been developed as the QA-processor for the conceptual designing of the SIS by the method of conceptual solving the project tasks. This method is based on the stepwise refining and QA-reasoning which are being evolved in the frame of incremental designing.

The base component of WIQA is the QA-database supported the real time work of designers with solving tasks of designing for the current project. All tasks are combined in the tasks tree with the visual access of designers to its units. Each such unit is interpreted as the definite precedent used in design process or will be executed (in the future) by the user of the SIS.

The conceptual solution of any task Z_i is being built with the help of QA-reasoning (sequence of questions $\{Q_{ij}\}$ and answers $\{A_{ij}\}$) which are being registered in the QA-protocol interpreted as the QA-model of the corresponding task. Thus interactive tasks tree and a set of corresponding interactive QA-models of tasks present the "intermediate" between developers and the current state of the designed SIS existed in the computer environment.

This intermediate link opens opportunities for the realization of new forms human-computer interactions which are additional for forms of interactions put into the practice now. New forms are oriented to precedents and have the question-answer type. QA-means for the implementation of such forms consist of declarative and imperative parts.

The declarative part is based on data models named QA-data which present the tasks tree, QA-models, Z-, Q-, A-objects and all useful transformations of these units supported in WIQA. QA-data is their interpretation from the informational point of view.

The specificity of QA-data is defined by typical forms of Z-, Q- and A-objects each of which includes the numerous diversity of attributes (for example, index-name, type of object, creator, time of creation and many others) and the hierarchical combining of such objects. The central place among attributes occupies the attribute "textual description" presenting the content of the corresponding object ("task", "question" or "answer" of the definite type).

The imperative part consists of a set of commands (QA-commands) with QA-data, their useful sequences and more complicated behavioral units from QA-commands which are specified as a kind of QA-programs. For this reason WIQA is constructed as the specialized QA-processor.

Most essential feature of the QA-approach is the possibility of programming (preliminary or in the real time) of human-computer interactions (HCI). Moreover, this approach assumes that necessary means of HCI are being embedded to the definite model of the precedent. For supporting such possibility the means of QA-modeling have been evolved till their usage for pseudo-programming oriented on precedents and the execution of pseudo-programs by the intellectual and computer processors collaboratively.

One direction of broadening the interpretation of QA-data is defining the abstract type of data with named attributes and features including the accessible set of commands. Such interpretation allow to developers to use the abstract QA-type for the emulation of the types of data which are needed for pseudo-programming.

For solving the emulation tasks in WIQA there is a special mechanism for assigning the necessary characteristics to the definite unit of QA-data. It is the mechanism of additional attributes (AA) which gives the possibility to expand the set of basic attributes for any Z-, Q- or A-object keeping in the QA-database.

The mechanism of AA implements the function of the object-relational mapping of QA-data to programs objects with planned characteristics. One version of such objects is classes in C#. The other version is fitted for pseudo-code programming.

Broadening of the abstract type of QA-data by means of additional attributes helps to emulate any traditional data types such as scalars, arrays, records lists and the others. Moreover, means of additional attributes open the possibilities for assigning to simulated data their semantic features.

The set of basic operators includes traditional pseudo-code operators but each of which inherits the feature of the appropriate QA-units also. Hence, the basic attributes of the QA-unit and necessary additional attributes can be taken into account in processing the operator and not only in its translation. In order to underline the specificity of the operator emulation they will be indicated as QA-operators.

Means for QA-programming have been developed and embedded to WIQA as its evolution. They include the special editor, pseudo-code interpreter and compiler. The reality of the designer activity is a parallel work with many tasks at the same time. Therefore the special system of interruptions is included into WIQA.

CONCLUSION

Told above contains sufficient arguments to assert that the real time pseudo-programming of interactions of designers with experience and its models embedded to the base of experience leads to many positive effects in the usage of SISs and their development. QA-programming is the rational way for such work which can be implemented with the help of WIQA means. QA-programming of precedents can be implemented at the project level (as the creation of the tasks tree) and at the pseudo-code level (as writing QA-programs for intellectuals and computer processors).

QA-programs are useful means of HCI which are additional for traditional means of HCI. Such means of HCI are adjusted for the access to the human experience in the precedents forms which were used in creating the library of the usability metrics implemented as the set of tasks with embedded interfaces precedents.