УДК 681.3.06

Л.П. БАБЕНКО

КАК ПОНЯТЬ ПРОГРАММУ. ХАРАКТЕРИСТИЧЕСКИЙ АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К СПЕЦИФИКАЦИИ ПРОГРАММ

Ключевые слова: *веб-сервис, интерфейс, информационный ресурс, онтология, спецификация программ, функциональность.*

ВВЕДЕНИЕ

Проблема понимания программ возникла после того, как программирование перестало быть предметом деятельности немногих «избранных» и перешло в разряд массовых профессий. Программы проникли почти во все отрасли современной цивилизации, но несмотря на это, существует постоянная потребность их обновления и создания новых программных систем. Наработано огромное количество готовых программных продуктов, использование которых в новых разработках существенно экономит трудозатраты и время разработки и повышает качество программных продуктов. Но такое использование

целиком определяется способностью новых разработчиков понять цели, функции и особенности, заложенные в готовые программные продукты при их разработке. Сказанное в значительной степени относится ко многим ресурсам (знаниям, онтологиям, публикациям, конституентам разработки программного обеспечения, стандартам и другим нормативным документам) современных информационных технологий, характерной чертой которых на данном этапе можно считать стремление к интеграции и открытому обмену знаниями.

Интерес к проблеме понимания информационных ресурсов стимулировал интенсивные разработки методов и средств их описания (спецификации). На первых этапах таких разработок усилия концентрировались на поисках формального математического аппарата спецификации, однако при этом уровень абстрагирования оказывался столь высоким, что спецификация становилась практически бесполезной. Затем произошла переориентация на концептуальные модели спецификации, т.е. ориентированные на их понимание профессионалами в выделенной области. Вместе с этим пришло осознание непригодности универсальных средств для всех категорий информационных ресурсов и целесообразности сосредоточить исследования на узких классах конечных продуктов, соответствующего накопления научных и инженерных знаний, необходимых для построения специализированных на эти продукты спецификаций.

Появление Интернета создало универсальную среду хранения спецификаций и доставки их пользователям, существенно увеличило число готовых ресурсов (ГОР), которыми можно воспользоваться при создании новых систем программирования, однако не решило проблемы поиска нужных спецификаций и их понимания. Именно поиск и понимание стали узким местом в использовании готовых ресурсов, поэтому на решение этих проблем нацелены многочисленные современные разработки. Далее в статье рассматриваются направления таких разработок, анализируются предлагаемые концептуальные и технические решения, получившие признание и распространение среди разработчиков и пользователей.

1. НАПРАВЛЕНИЯ ИССЛЕДОВАНИЯ СРЕДСТВ СПЕЦИФИКАЦИИ ИНФОРМАЦИОННЫХ РЕСУРСОВ

Мощные объединения и концерны, согласовывающие усилия ведущих разработчиков в области информационных технологий, включили в сферу своих интересов разработку стандартов спецификаций для отдельных классов информационных ресурсов, среди которых наиболее значимых результатов достигли следующие:

- Object Management Group (OMG) [1], курирующая такие успешные направления работ, как компонентное программирование (проект CORBA), развитие UML, ориентированные на сервисы архитектуры (SOA);
- World Wide Web concern (W3C) [2] — концерн, декларирующий своей целью интеграцию усилий по максимальному использованию потенциала веб-сообщества;
- The Organization for the Advancement of Structured Information Standards (OASIS) [3] — организация по продвижению структурированных информационных стандартов.

В рамках этих объединений и под их эгидой выполнено большинство проектов, повлиявших на развитие средств спецификации и включенных в настоящий обзор. Перечислим те из направлений исследования упомянутых проектов, результаты которых прямо или косвенно служат цели понимания программ.

Формулирование требований на разработку, или инженерия требований. Цель инженерии требований — добиться взаимопонимания между заказчиком и исполнителем, четко и однозначно зафиксировать для разработчика, кто его пользователи, что им нужно (по их мнению), что они хотят на самом деле, а для заказчика — сформулировать свои потребности так, чтобы они были понятны и ему, и разработчику, а впоследствии убедиться, что выполненная разработка удовлетворяет их требованиям.

Описание бизнес-процессов. Цель спецификации — сделать понятными и регламентировать процессы, позволяющие вести определенный бизнес и успешно взаимодействовать с другими деловыми кругами, понимая взаимные возмож-

ности и обязательства. Современные бизнес-услуги, как правило, поддерживаются программно, более того, интенсивно развиваются идеи электронного бизнеса (eBusiness), электронного правительства, электронной администрации, в которых минимизирована роль человека. В этих разработках речь идет не только о понимании программ человеком, но и о взаимопонимании между программами.

Описание повторно используемых ресурсов программной инженерии. Цель спецификации — понимание разработчиком тех готовых решений и ресурсов, которыми он может воспользоваться, сократив затраты и время. Спецификации должны сопровождать хранилища таких ресурсов, обеспечивать эффективный поиск ресурсов, полезных для новых разработок, понимание их функций и правил интеграции в новые разработки.

Описание ресурсов типа веб-сервисы. Цель спецификации — понимание тех услуг, которые обеспечивает веб-сервис, и правил доступа к ним. Заметим, что веб-сервис является также повторно используемым ресурсом, но, во-первых, в среде веб, во-вторых, он можеть исполняться как программно, так и с участием человека. Возникло специальное научное направление — Semantic Web — декларирующее цель всемирной системы распределенных веб-вычислений, что дало новый толчок разработкам средств понимания программ, хотя и реализованных как веб-сервисы.

Описание знаний в системах управления знаниями. Информационное общество как концепция новой ступени развития цивилизации, главным продуктом производства которой есть информация, получило широкое признание. Концепция предполагает создание глобального информационного пространства, которое обеспечивает эффективное информационное взаимодействие людей, их доступ к мировым информационным ресурсам для удовлетворения их информационных потребностей. Необходимая предпосылка ее реализации — увеличение роли информации и знаний в жизни общества, а именно, увеличение доли информационных коммуникаций, продуктов и услуг в валовом внутреннем продукте, систематическая управляемая деятельность в разных областях науки, нацеленная как на фиксацию мирового уровня достижений коллективных знаний, так и регулярное отслеживание соответствующего пополнения; появление в обществе единого виртуального пространства для взаимодействия групп исследователей разных направлений для решения актуальных проблем в той или иной области человеческого бытия. Спецификация знаний и их понимание играют здесь ключевую роль.

Описание ресурсов электронных библиотек. Перевод в цифровой формат разнообразных печатных документов (книг, газет, журналов, административных и правовых документов так называемого электронного правительства и пр.) и их хранение в электронных библиотеках в настоящее время приобретает массовый характер. Спецификация их смысла и назначения, понятность для пользователя целиком определит полезность таких библиотек.

Далее в статье анализируются подходы к спецификациям программ в ряде проектов, получивших наибольший резонанс среди специалистов перечисленных выше направлений. При выборе объектов анализа мы ввели в рассмотрение не только те проекты, что ставили своей прямой целью разработку средств описания (спецификации) программ, обеспечивающих понимание, но и те, реализация собственных целей которых в большей или меньшей мере способствовала появлению новых идей и продвижению в интересующем нас направлении. В качестве объектов анализа нами рассмотрены следующие проекты.

UML [4] — универсальный язык моделирования, широко используемый при разработке программного обеспечения и непосредственно, и как прародитель множества идей, получивших развитие в других проектах.

Reusing Assets Specifications [5] — стандартный каркас спецификации готовых ресурсов разработки программного обеспечения, пригодных для повторного использования (assets), разработанный под эгидой [1]. Предлагается иерархически упорядоченная совокупность аспектов описания информационного ресурса и соответствующих им характеристик. Выделяется базовое ядро спецификации,

общее для всех типов ресурсов на всех стадиях разработки. В него входят такие аспекты: идентификация ресурса (имя и другие характеристики), контекст (принадлежность готового ресурса к определенной стадии моделирования: модель бизнеса, модель требований или проектирования, компонента кода, модель теста, документация), использование (характеризует виды работ (пользователя или инструментального средства), необходимые для использования готового ресурса, в том числе для настраивания его вариантных свойств), решение (артефакты, соответствующие готовому ресурсу, среди которых различают главный тип, например компонента кода, и несколько вторичных типов, которые его объясняют, например соответствующие ему требования, отдельные диаграммы UML, разделы инструкций для пользователя и т.д.), так называемые точки вариантности (концептуальные позиции артефакта, в которые пользователь имеет возможность вносить изменения после передачи ему артефакта для использования), зависимости (определяет ссылки на артефакты, связанные с данным ресурсом определенными отношениями, например, ассоциация, агрегация, наследование, зависимость по изменениям и др.). Для ресурсов конкретного типа (продуктов конкретной деятельности на конкретном отрезке пути разработки) предлагается строить профили спецификации. Профиль — это расширение базового ядра спецификаций (базового профиля) для конкретного типа ресурсов, свобода детализации которого остается за разработчиком нужного профиля. Таким образом, понятие компонента повторного использования, принятое для программ, распространяется на метаданные: базовый профиль есть компонент повторного использования для определенного множества типов ресурсов. В [5] определены профили для программных компонентов и веб-сервисов. Данный проект, по нашему мнению, следует рассматривать как представительный портфель возможных аспектов спецификации и их характеристик. Отметим, что это единственный из попавших в наше поле зрения проектов, в котором предлагается рассматривать промежуточные продукты этапов жизненного цикла разработки программ как готовые ресурсы, заслуживающие самостоятельной аттестации для дальнейшего использования раздельно с продуктами других этапов разработки того же продукта. Такое решение представляется перспективным при смене среды исполнения.

Business Processes Modeling Notation (BPMN) [6] — проект языка моделирования процессов бизнеса, разработанный под эгидой OMG [1]. Одна из целей проекта — предоставить средства для формулирования требований к программам в привычной для бизнесменов нотации. Нотация нацелена на понимание внутренних процедур бизнеса (своего или партнерского), обеспечение их взаимодействия, представление требований на разработку программ, обеспечивающих функционирование бизнес-процессов. При этом она имеет визуальное представление, ориентированное на удобства человека, хотя степень ее формализации позволяет относительно простое отображение на языки, непосредственно транслируемые в исполнительский код (executable language), см., например, [7].

Web Services Description Language (WSDL) [8] — проект, имеющий статус рекомендаций для использования концерна W3C [2]. Хотя название свидетельствует о намерениях его создателей описывать веб-сервисы, им удалось лишь создать язык для описания их интерфейсов. Как следует из рассматриваемого документа, предлагаемые средства позволяют определить на синтаксическом уровне, как потенциальные клиенты вызывают сервис и какова структура входных и выходных сообщений его интерфейса. При этом предполагается, что он реализован и выполняет те функции, которые описаны неформально в виде комментария. В ряде рассматриваемых далее проектов WSDL используется для описания элементов интерфейсов (BPMN, UDDI, ebXML, WSMO). В то же время в ряде проектов, предлагаемых для рассмотрения W3C, содержатся механизмы для дополнительного аннотирования конструкций WSDL семантическими комментариями, позволяющими понять смысл и функциональное назначение сервиса (см. ниже).

Universal Description, Discovery and Integration (UDDI) [9] — проект стандарта OASIS [3] для спецификации сервисов ориентирован на устройство хранилища (регистра), в котором каждый может зарегистрировать свой бизнес и предлагаемые им услуги (сервисы), описывая их настолько понятно, чтобы будущие пользователи могли их обнаружить и интегрировать в свои проекты. Заметим, что под термином «бизнес» в контексте UDDI можно понимать любую предметную область (ПрО). В указанной публикации имеются сведения об успешных реализациях регистров для ряда ПрО.

ebXML [10] — совместный проект OASIS [3] и Центра ООН по торговле и электронному бизнесу, назначение которого — определить среду обмена коммерческой информацией для глобального электронного бизнеса, в котором любые два бизнес-партнера могут интегрировать свои бизнес-процессы, используя сведения об услугах, предоставляемых каждым из них. Эти сведения находятся в хранилище (репозитории), а описание того, как устроены эти сведения, — в так называемом регистре (хранилище спецификаций). Интегрированные в систему, названную ebXML Registry-Repository (репозиторий и регистр для електронного бизнеса), они предназначены для безопасного и объединенного управления большими объемами сложной информации путем использования стандартизованных и раширяемых метаданных. Реализован при построении ряда специализированных хранилищ информации, как, например, фирмой IBM при создании хранилища медицинских записей, фирмой SUN Microsystems при создании хранилища сервис-ориентированных архитектур и ряда других.

WEB Services Modeling Ontology (WSMO) [11] — предложения для использования в проекте Semantic Web Services [12], нацеленном на поддержку всемирной системы распределенных веб-вычислений как эффективной инфраструктуры для электронного бизнеса, где смысл и назначение сервисов и правила их использования понятны не только человеку, но и взаимодействующим программам, способным находить и использовать нужные веб-сервисы без участия человека. В рассматриваемых предложениях разделяются следующие компоненты: WSMO — концептуальная модель для описания различных аспектов веб-сервисов, представленная онтологией моделирования веб-сервисов; Web Service Modeling Language (WSML) [13] — язык для формального описания синтаксиса и семантики всех элементов собственно WSMO, основанный на ряде логических формализмов и предназначенный для формализованных аннотаций, ориентированных на понимание программными компонентами Semantic Web; Web Service Execution Environment (WSMX) [14], исполнительская среда для семантических веб-сервисов. Целям данной статьи отвечает первый из упомянутых компонентов как ориентированная на человека.

Semantic Annotations for WSDL and XML Schema (SAWSDL) [15] — рекомендации W3C по дополнительному аннотированию конструкций WSDL семантическими комментариями, позволяющими понять смысл и функциональное назначение сервиса (см. 2.1).

Web Service Semantics (WSDL-S) [16] — предложения для обсуждения W3C по расширению описаний операций веб-сервиса в WSDL семантическими комментариями, связывающими функциональное назначение операций сервиса с соответствующей ему семантической моделью (см. 2.1).

Для приведенных выше проектов авторы выделили и последовательно проанализировали совокупный спектр их общих и специфических концептуальных и проектных решений, нацеленных на выбор аспектов и характеристик для достижения понимания объектов спецификаций.

2. ОБЩИЕ КОНЦЕПТУАЛЬНЫЕ И ПРОЕКТНЫЕ РЕШЕНИЯ, СВОЙСТВЕННЫЕ БОЛЬШИНСТВУ ИЗ РАССМАТРИВАЕМЫХ ПРОЕКТОВ

• Большинство проектов предполагает наличие определенного хранилища информации, понимаемого как совокупность средств и инструментов, обеспечивающих возможности запоминать определенные «порции» информации как еди-

ницы хранения, имеющие описания, достаточные для последующего обнаружения единиц хранения.

• Как правило, предлагается раздельное хранение описания объекта спецификации (метаданных) и собственно его представления (контента) подобно тому, как в традиционных библиотеках раздельно хранятся каталоги и книги, причем в электронных хранилищах, как правило, обеспечивается прямой доступ из единицы каталога спецификаций к объекту спецификации.

• К описанию применяется аспектный подход. Целесообразность структурирования описаний программ отдельными категориями или аспектами осознана еще в ранних языках программирования, где отдельными средствами описывались аспект данных и аспект их преобразования, т.е. алгоритмический или поведенческий. В UML разделение описания на аспекты получило дальнейшее развитие — к аспектам данных (диаграмма классов) и алгоритмов их обработки (диаграммы состояний и действий) добавлены аспекты взаимодействия объектов программы между собой и с внешними по отношению к программе объектами, называемыми актерами (диаграммы последовательности и взаимодействия на основе посылки сообщений), аспекты возможных вариантов использования (Use Cases), т.е. услуг или задач, которые можно решить программой — данный аспект непосредственно ориентирован на понимание программы ее потенциальными пользователями, на выражение требований пользователем и на понимание разработчиком того, что нужно реализовать, чтобы разрабатываемая программа оказалась востребованной. Кроме того, диаграммы компонент, пакетирования и размещения отражали аспекты среды реализации программы. С развитием информационных технологий появляются новые аспекты, важные для понимания смысла, назначения и правил использования программ, и поэтому включенные в большинство рассматриваемых нами проектов.

• Аспект интерфейса выделен как самостоятельный в большинстве рассматриваемых проектов.

• Еще один аспект, общий практически для всех проектов, — аспект паспортных данных объекта спецификации, называемый также нефункциональными свойствами, административными правилами использования и др. Для представления этого аспекта обычно предлагаются дескрипторы стандарта, называемого Дублинским ядром (Dublin Core — Dc) [17]: название, ID (однозначный идентификатор ресурса, поддерживаемый определенным инструментом, как, например, Uniform Resource Identifier — URI [18]), необязательные атрибуты (дата, состояние, версия, права доступа, неформальное описание, контекст использования, автор, почтовый адрес или e-mail, телефон, дата создания или срок годности, правила приобретения и т.д.).

• В метаданных непосредственно используются модели ПрО, к сфере которых относятся решаемые программой задачи. При этом в мире наблюдается тенденция накопления стандартизованных моделей, значимых для практики ПрО (авиация, медицина, математика, химия, география и многие другие), фиксирующих состав и классификацию основных понятий, а также отношений между ними. Большинство стандартизованных моделей ПрО представлено онтологиями. Онтология разрешает удерживать пользователя в максимально возможном пространстве предопределенных возможностей, смысл которых зафиксирован и понятен и разработчику, и заказчику. Очевидно, что пребывание в таком пространстве гарантирует взаимопонимание обеих сторон договора (который материализован в виде требований).

• Для спецификации данных все рассмотренные проекты используют концепцию классов UML. При этом отличия заключаются в ограничениях на типы данных, атрибутов (слотов) классов, типы отношений между классами, допустимую кратность значений атрибутов и экземпляров классов.

• Аспектом, важным для жизнеспособности информационных ресурсов, в том числе и программ, является цена внесения изменений, необходимых

в быстро меняющемся мире. Явное определение тех качеств ресурса, которые можно менять, и точек внесения изменений описывается аспектом, который будем называть вариантностью (variability). Речь идет о предусмотренных на стадии проектирования возможностях, которые можно изменять локально в явно обозначенных позициях. Подробнее этот аспект рассмотрен в [19].

• Для представления спецификаций программ активно используются языки разметки XML, XML Schema [20], конкретизированные на требуемую проблемную область.

• Получают признание преимущественно те проекты, которые фокусируются на узких классах конечных продуктов (бизнес-процессы, веб-сервисы, специализированные электронные библиотеки знаний, диссертаций, услуг, онтологий и т.д.).

• Наряду с проектами, в которых ставится акцент — добиться взаимопонимания между создателем ресурса и его пользователем как двумя неформальными системами, появляются разработки, в которых формулируются намерения обеспечить понимание между машиной — создателем информационного ресурса и машиной — его пользователем. Акцент делается на правила преобразования концептуальных спецификаций, выполняемых человеком и адресуемых человеку, в исполнимые спецификации, понятные машине, и на генерацию спецификаций машиной.

• Как правило, общим для проектов является онтологический подход к спецификации аспекта функционального назначения ГОР в виде перечня дескрипторов (ключевых слов) с указанием классифицирующих их онтологий, в том числе стандартизованных, таких как онтологии продуктов и услуг, географических названий, математических, химических, медицинских терминов и др.

3. ХАРАКТЕРИСТИКИ АСПЕКТОВ СПЕЦИФИКАЦИИ

Важно отметить, что смысл термина «понять программу» зависит от ответа на вопрос «зачем». Ответ может иметь варианты.

• Найти среди множества предложенных программ ту, которая предоставляет нужные клиенту функции, т.е. сравнить спецификации предлагаемых программ с теми задачами, которые он хочет решать.

• Понять, как использовать выбранную программу: как обратиться к ней из другой программы, как развернуть ее в своей операционной среде, как настроить ее по нужным параметрам (если такая настройка предусмотрена в программе).

• Как изменить программу, если она не совсем удовлетворяет потребностям клиента.

Для понимания программы в каждом из перечисленных вариантов предлагаются свои аспекты спецификации и свои характеристики: первый вариант касается функционального назначения программы, которое выражается в обеспечиваемыми программой функциях, их интерфейсах и взаимодействии с внешней средой. Остальные варианты зависят от различий среды реализации программы и среды реализации, нужной клиенту, и требуют знания заложенных в программу проектных решений. При всей их важности определяющим является сопоставление функциональности программы и задач клиента. Исходя из этого, мы рассматривали только средства, нацеленные на понимание смысла и возможностей программ и услуг, не касаясь среды исполнения, операционных или организационных вопросов.

3.1. Аспект функциональности. Данный аспект ключевой для понимания объекта спецификации, его функций и предназначения. В то же время он труднее поддается формализации, поскольку является продуктом неформальной системы, каковой является человек, и выражает обычно его видение смысла программы, естественным стремлением при этом является использование терминов ПрО, к которой относится описываемая программа. В рассматриваемых нами проектах основную характеристику аспекта функциональности назовем единицей функциональности объекта спецификации. Этим термином будем обозначать одну услугу, которую пользователь может получить при одном обращении к программе.

Аналог такой характеристики, первоначально введеный в UML под названием вариант использования (Use Case), присутствует практически в каждом из проектов, отличия состоят в терминологии, отражающей область применения проекта, и в атрибутах описания варианта использования.

В проекте BPMN единицей функциональности является бизнес-процесс, акцент делается на спецификации динамики поведения бизнес-процесса. Процессы могут быть вложенными. В качестве характеристик функциональности выбраны следующие.

— Паспортные данные процесса (см. разд. 1).

— Выполняемые работы (activity), поток которых во времени для отдельного участника процесса, называемый пулом, выделяется графически. Управление потоком работ осуществляется так называемыми шлюзами (gateway). Графическое представление каждого из шлюзов маркируется типом соответствующего ему поведения (распараллеливание, слияние, циклы, условные переходы, прерывание в зависимости от происходящих событий). Атрибуты активности указывают на ее исполнителя (программа или человек), статус (готовность) и др.

— События, происходящие на протяжении бизнес-процесса и влияющие на его поведение. События могут происходить внутри пула или быть внешними по отношению к нему, пулы могут обмениваться сообщениями. Графическое представление события маркируется атрибутами, указывающими их причину и сферу влияния.

— Документация как произвольные комментарии к графическим диаграммам.

В проекте WSDL единицей функциональности веб-сервиса можно считать операцию интерфейса. Ее смысл задается неформально в виде комментария Documentation, а основное внимание уделяется синтаксической спецификации операций интерфейсов веб-сервисов. Более формализованное задание функциональности упомянутых выше операций предлагается в двух проектах, представляющих собой дополнительные аннотации синтаксических конструкций WSDL формализованными комментариями в терминах семантической модели, созданной вне спецификации WSDL. Такой моделью чаще всего является модель ПрО, в рамках которой рассматривается операция веб-сервиса, обычно некоторая онтология. При этом оба подхода не привязываются к конкретному языку представления семантической модели.

В предложениях WSDL-S [15] комментарии выражаются следующими средствами:

• дополнительный атрибут modelReference — ссылка на понятие семантической модели, ассоциированное с элементом WSDL, для которого указан данный атрибут;

• дополнительный атрибут schemaMapping, отражаюший структурное соответствие элементов схемы веб-сервиса и понятий семантической модели;

• дополнительный дочерний элемент precondition, задающий предусловие выполнения соответствующей операции WSDL в терминах понятий семантической модели;

• дополнительный дочерний элемент effect, задающий постусловие выполнения соответствующей операции WSDL в терминах понятий семантической модели;

• дополнительный атрибут category для элемента interface, задающий информацию о категоризации сервиса, которую можно использовать при размещении или поиске сервиса в регистрах, например в UDDI (см. далее).

В рекомендациях SAWSDL [16] предлагается набор атрибутов расширения WSDL и XML Schema, предназначенных для описания семантических свойств компонентов WSDL. Эти атрибуты позволяют ссылаться на понятия семантической модели, определенной, как и в предыдущем проекте (WSDL-S), вне спецификаций WSDL.

Проект UDDI своим полным названием (Universal Description, Discovery and Integration) нацеливает на возможность обнаружения клиентом нужного ему веб-сервиса среди хранимых в регистре. Предлагается четырехуровневая иерархия спецификаций. Высшим уровнем является спецификация так называемой бизнес-сущности (business entity). Она содержит описательную информацию

о бизнесе в целом, его провайдерах и их паспортных данных (контакты и др.). Бизнес-сущность можно считать аналогом ПрО, к которой приннадлежит веб-сервис. На втором уровне находятся спецификации так называемых сервисов бизнеса (businessService). Это услуги, известные в том бизнесе, логическим потомком которого является сервис. На третьем уровне располагаются спецификации веб-сервисов, осуществляющих каждую из услуг, а также техническую информацию для связывания с веб-сервисом (binding template). Наконец, на четвертом уровне располагаются технические модели (tmodel) спецификации отдельных характеристик веб-сервисов в WSDL или XML-схемах. Сервисы бизнеса и веб-сервисы играют в проекте роль единиц функциональности. Их характеристиками, помимо паспортных данных, являются принадлежность к определенной ПрО (бизнес-сущности) и так называемая таксономическая классификация. Она состоит из пары характеристик, первая задает систему классификации, вторая — таксономическое значение в этой системе (аналог пары предметная область — дескриптор применяемой в информационных системах). В частности, система классификации может быть представлена онтологией, а таксономическое значение — узлом этой онтологии. На каждом из уровней иерархии в спецификации аспекта функциональности можно одновременно применять несколько классификаций, т.е. характеризовать функции объекта спецификации с разных точек зрения. Механизм tmodel позволяет определять в XML характеристики аспекта функциональности, индивидуальные для клиента, и в дальнейшем их реиспользовать, например описание типов данных, используемых сервисом.

Проект WSMO нацелен на обогащение спецификаций веб-сервиса семантическими элементами, которые способна обрабатывать машина. Концептуальный каркас спецификации составляют:

- ontology — онтологии, определяющие понятия, отношения, функции, аксиомы и экземпляры, специфические для предметной области, к которой принадлежит веб-сервис, а также терминологию, используемую в остальных компонентах спецификации;
- goal — цели клиента, решаемые веб-сервисом;
- webService — функциональные и поведенческие аспекты веб-сервиса;
- mediator — так называемые посредники, служебные компоненты, обеспечивающие совместимость неднородных элементов модели (например, отображение возможностей веб-сервиса и целей клиента, онтологий, используемых клиентом и разработчиком и др.).

Единица функциональности в проекте называется capability(возможность). Это прямой аналог конструкции Use Case в UML. Каждая из предоставляемых capability описывается используемой онтологией, интерфейсом (см. 2.2) и моделью поведения, задаваемой следующими характеристиками:

- shared variables — переменные, значимые для поведения веб-сервиса, используемые в логических формулах характеристик, приведенных ниже;
- preconditions (предусловия) определяют пространство информации до выполнения веб-сервиса;
- postconditions (постусловия) определяют пространство информации после выполнения веб-сервиса;
- assumptions (предположения) определяют состояние мира до выполнения веб-сервиса;
- effect (эффект) определяет состояние мира после выполнения веб-сервиса.

Все условия, предположения и эффект выражаются аксиомами, представленными в языке формальной логики WSML [13], ориентированном на использование для автоматического обнаружения веб-сервисов при поиске в регистрах.

Спецификации capability как составные части входят в компоненты спецификации goal и webService, причем возможности, объявляемые в goal для клиента, могут отличаться от объявляевых в компоненте webService; в этом случае требуются соответствующие медиаторы.

Для облегчения понимания действий веб-сервиса человеком каждая из конструкций WSMO, таких как классы: онтология, понятие, отношение, экземпляр, веб-сервис, capability, interface, может быть снабжена нефункциональным свойством (см. разд. 2) Subject, которое (как и в рассмотреном выше проекте UDDI) рекомендуется представлять множеством пар характеристик, первая из которых задает систему классификации, вторая — таксономическое значение в этой системе. Таким образом, строго формальные характеристики для машинной обработки дополняются традиционными ключевыми словами.

Проект ebXML Registry-Repository — репозиторий и регистр для електронного бизнеса — составная часть более общего проекта ebXML — Electronic Business XML, языка разметки для электронного бизнеса. Как и в предыдущих проектах, для описания функций предлагаются онтология ПрО и множество пар система классификации — таксономическое значение.

3.2. Аспект интерфейса. Выделен как самостоятельный в большинстве рассматриваемых нами проектов. В нем определяются правила обращения к программе за получением объявленных в ней услуг — точки установления связи с услугой (например, сетевые адреса связывания), выполняемые в рамках услуги операции, их входные и/или выходные параметры, необходимость взаимодействия с другими программами, с внешними факторами и др.

BPMN. Поскольку объектами спецификации проекта являются бизнес-процессы, для которых характерен интенсивный обмен информацией с менеджерами, партнерами по бизнесу и другими внешними факторами, интерфейс бизнес-процесса описывается как цепь событий, определяющих выполнение бизнес-процесса. Событие имеет ряд характеристик:

— тип события определяет, на какой фазе процесса оно может происходить (стартовой, промежуточной или завершающей);

— характер влияния события на бизнес-процесс;

— причина события (внешний источник события или внутренняя ситуация процесса). В качестве стандартизованных причин событий в сфере бизнеса используются получение или отправка сообщения извне активности, таймер, заданное условие, сигнал от устройства, исключительная ситуация и т.д. Для каждой из перечисленных причин предусмотрены отдельные графические маркеры событий.

WSDL. Интерфейс описывается на двух уровнях: абстрактном и конкретном. На абстрактном уровне (синтаксическом) для компоненты описания, называемой интерфейс (interface), указываются абстрактные операции, которые может выполнить сервис (под операцией понимается простой обмен сообщениями с клиентом). Для операций указываются типы сообщений и так называемый паттерн обмена сообщениями (характер обмена — ввод, вывод или и то, и другое). Определяются также действия при исключительных ситуациях. На конкретном уровне (уровне реализации) элемент binding (связывания) определяет точки доступа к сервису (сетевые адреса и протоколы связывания), а также уникальное имя сервиса, позволяющее создавать однозначные ссылки на компоненты описания сервиса в соответствующих хранилищах.

UDDI. Для описания интерфейса используется WSDL.

ebXML. Для описания интерфейса используется WSDL.

WSMO. Кроме характетистик, предложенныхWSDL и используемых в проекте посредством механизма нефункциональных свойств, вводится две известные характеристики интерфейса [21].

Хореография (choreography) описывает взаимодействие веб-сервиса и его клиента. Клиентом может быть человек, другой веб-сервис или другое приложение. Концепция хореографии базируется на абстрактной машине состояний. Ее составляющими являются: состояние (state) (описываемое как множество явно указанных экземпляров понятий, отношений или функций и значений их атрибутов) и переходы (guarded transitions) (условие изменения состояния, заданное в форме аксиом WSML, а также требуемую модификацию состояния при его истинности).

Оркестровка (orchestration) определяет последовательность и условия вызовов других сервисов, необходимых данному сервису для реализации его функциональности. Эта характеристика также базируется на абстрактной машине состояний. Ее составляющие: состояние (state) (описываемое как множество явно указанных экземпляров понятий, отношений или функций и значений их атрибутов) и переходы (guarded transitions), но переход определяет условие вызова требуемого веб-сервиса и ссылку на используемый медиатор.

Заметим, что приведенные выше характеристики используют специально заданную онтологию интерфейса, отражающую внешние факторы, взаимодействующие с веб-сервисом, и характерные для них события.

ЗАКЛЮЧЕНИЕ

Сходство многих идей и решений в рассмотренных проектах свидетельствует о том, что в проблеме понимания и распознавания смыслового содержания программ для ряда аспектов найдены устоявшиеся подходы к их представлению. Построенные для таких аспектов соответствующие типовые онтологии могут эффективно использоваться при создании регистров повторно используемых программных ресурсов. Последние, в свою очередь, послужат информационной базой для процессов инженерии функциональных требований для разработки новых программных систем, а также для формулирования запросов на поиск готовых ресурсов, которые целесообразно использовать для новых разработок [22].

СПИСОК ЛИТЕРАТУРЫ

1. Object management group. — http://www.omg.org.
2. World Wide Web Concern. — http://www.w3.org.
3. The Organization for the advancement of structured information standards (OASIS). — //http.www oasis.org.
4. Unified Modellind Language. — http://www.uml.com.
5. Reusable asset specifications (RAS) OMG available specifications version 2.2., Date: November, 2005. — http://www.omg.org.
6. Business processes modeling notation. — http://www.omg.org/spec/BPMN/1.1.
7. Business Processes Language BPEL4 // IEEE Internet Comput. — 2004. — N 1. — P. 77.
8. Web services description language. — http://www.w3.org/TR/2007/REC-wsdl20-20070626.
9. Universal Description, Discovery and Integration. — http://uddi.org.
10. OASIS/ebXML Registry inform. model. — http://www.oasis-open/org/cojmmetees/specs/ebrim.
11. Roman D., Lausen H., Keller U. (eds.). Web service modeling ontology WSMO. — http://www.w3.org/Submission/wsmo.
12. Lara R. (Ed.): Semantics for Web service discovery and composition, knowledge Web deliverable D2.4.2, November 2004. — http://www.knowledgeweb.semanticweb.org.
13. de Bruijn J., Lausen H., Krummenacher R., Polleres A., Predoiu L., Kifer M., Fensel D. The Web service modeling language (WSML). — http://www.w3.org/Submission/wsml.
14. Web Service Execution Environment (WSMX). — http://www.w3.org/Submission/wsmx.
15. Semantic annotations for WSDL and XML schema. W3C recommendation. — http://www.w3.org/TR/sawsdl/.
16. Web Service Semantics — WSDL-S.W3C Member Submission. — http://www.w3.org/Submission/WSDL-S/.
17. Weibel S., Kunze J., Lagoze C., Wolf M. RFC 2413 — Dublin core metadata for resource discovery. — http://www.isi.edu/in-notes/rfc2413.txt.
18. Berners-Lee, Fielding R., Masinter L. RFC 3986. — Uniform resource identifiers (URI): Generic Syntax, IETF, January 2005. — http://www.isi.edu/in-notes/rfc3986.txt.
19. Бабенко Л.П. Спецификация прогнозируемой вариантности как инструмент управления изменениями программных продуктов UML // Кибернетика и системный анализ. — 2007. — № 3. — С. 156–163.
20. Дейтел Х.М. Как программировать на XML. — М.: Бином, 2001. — 874 с.
21. Web Service Glossary. — http://www.w3.org/TR/ws-gloss.
22. Бабенко Л.П. Онтологический подход к спецификации свойств программных систем и их компонентов // Кибернетика и системный анализ. — 2009. — № 1. — С. 180–187.

Поступила 12.08.2010