УДК 004.942 , 004.042 , 338.46
DOI: 10.20535/SRIT.2308-8893.2016.4.05

ПРОЦЕССНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ КОМПЛЕКСОВ КАК СИСТЕМ СЕРВИСОВ

А.И. ПЕТРЕНКО

Аннотация. Идея перехода на новую парадигму программирования, связанную не с объектами, а с бизнес-процессами и их составной частью — бизнес-функциями, — компоновка приложений путем обнаружения и вызова сервисов, доступных в сети, для выполнения необходимой задачи. Этот подход не зависит от конкретных языков программирования и операционных систем и предполагает использование сервис-ориентированной архитектуры (SOA) приложений, построенной на основе формализованных бизнес-процессов, функции которых представлены в виде многократно используемых сервисов с прозрачно описанными интерфейсами. Рассмотрено построение систем сервисов для реализации бизнес-процессов с помощью процессно-ориентированных моделей функций предприятия и онтологий соответствующих предметных областей.

Ключевые слова: сервис-ориентированная архитектура, семантические сервисы, модельно-управляемое моделирование, онтология, онтолого-управляемое моделирование, бизнес-модель, универсальный язык моделирования UML.

ВВЕДЕНИЕ

В последнее время возникла ***необходимость интеграции и взаимодействия приложений*** в рамках совокупности большого количества информационных систем предприятия или нескольких предприятий. Наблюдаются тенденции перехода к бизнесу реального времени и созданию систем расширенного предприятия, объединяющего само предприятие, его поставщиков, партнеров и клиентов в единую систему. Поэтому требуется *взаимодействие приложений* как в одной информационной системе, так и между системами отдельных участников бизнес-процесса.

Выход из создавшегося положения, активно развиваемый ведущими поставщиками информационных технологий, такими как IBM, Microsoft, Oracle и другими, состоит в переходе от централизованной инфраструктуры информационных технологий и замкнутого на себе функционала прикладных систем к архитектуре, обеспечивающей возможности быстрого создания новых систем из набора доступных сервисов, т.е. переходе на **сервисные модели взаимодействия** между приложениями общей системы в рамках сервис-ориентированной архитектуры (**Service-oriented Architecture SOA**) и ее реализации с помощью модельно-ориентированных архитектурных решений (**Model-driven Architecture MDA**).

Сервис-ориентированная архитектура (SOA) является стилем архитектуры программного обеспечения, которая обеспечивает использование информационных технологий в виде модульных бизнес-сервисов для достижения конкретных бизнес-целей. Архитектура SOA — концептуальная архитектура бизнеса, где бизнес-функциональность, или логика приложе-

ний, становится доступной для пользователей SOA, как и множество многократно совместно используемых сервисов в ИТ-сети. Сервисы в SOA соответствуют модулям бизнес-процесса, или модулям функциональности приложения с открытыми интерфейсами, вызываемыми с помощью сообщений. Основная мотивация появления SOA — желание индустрии программирования заменить «ручное» кодирование программ «от и до» на «промышленную» сборку приложений из «стандартных комплектующих», как это происходит в автомобильной или других «традиционных» отраслях промышленности. Компоненты программы могут размещаться на различных узлах сети и представлять собой независимые, слабосвязанные, заменяемые сервисы-приложения.

Для бизнеса SOA означает ускоренное удовлетворение потребностей клиентов, реальную гибкость бизнеса, быстрое время выхода на рынок, простоту сотрудничества и низкую стоимость бизнеса. Для ИТ-организаций SOA — это повышение производительности, снижение затрат на ИТ-решения за счет ускорения разработки приложений, повторное использование сервисов, улучшение качества приложений, и в целом быстрое реагирование на запросы бизнес-клиентов для модификации системы. Дополнительно к этому существует возможность использования сервисов независимых поставщиков, что обеспечивает еще большую ценность SOA. Более того, даже интеграцию *унаследованных приложений* (***legacy***) целесообразно проводить с применением данной технологии, когда определенная, наиболее важная часть существующей функциональности как бы «инкапсулируется» и обеспечивается стандартизированным интерфейсом. В случаях, когда сервисы представляют собой реализации вычислительных процедур, говорят о системах ***сервис-ориентированных вычислений SOC (Service-orientedComputing).***

Цель работы — теоретическое обоснование, разработка и развитие сервис-ориентированной архитектуры и моделей систем сервисов для решения задачи поиска web-сервисов в репозитариях, композиции и оркестровки сервисов и построения их рабочих потоков (workflows) для поддержки определенных бизнес-процессов.

НАУКА О СЕРВИСАХ, МЕНЕДЖМЕНТЕ И ИНЖИНИРИНГЕ

Усилиями ведущих ИТ-компаний мира созданы методология и инструментарий проектирования сервис-ориентированных систем, наиболее полно отраженные в многосерийной публикации [1]. Они ориентированы на разработку приложений в основном с использованием ***web-сервисов*** (служб). Под web-сервисами понимаются программные системы, которые используют ***XML (Extensible Markup Language)*** в качестве формата данных, стандарты ***Web Services Description Language (WSDL)*** для описания своих интерфейсов, S***imple Object Access Protocol (SOAP)*** для описания формата принимаемых и посылаемых сообщений и стандарт ***Universal Description, Discovery and Integration (UDDI)*** для создания каталогов доступных сервисов. Для комбинирования сервисов в необходимые приложения предложены процедуры ***хореографии*** и ***оркестровки***, при этом хореография web-сервисов определяет взаимодействие между сервисами с помощью обменных уведомле-

ний, а оркестровка описывает взаимодействие сервисов в рамках одного бизнес-процесса, в частности, с использованием языка типа ***WS-BPEL (Web Services Business Process Execution Language).*** Наиболее просто бизнес-процесс можно представить в качестве простой оркестровки web-сервисов, которая составляет **поток задач (*workflows***), согласно которому последовательно подключаются сервисы, при этом каждый сервис следует за предыдущим без промедления или пробела и его действие заканчивается как раз перед началом действия следующего сервиса [2–5].

Компанией IBM для удовлетворения потребностей сервис-ориентированного ИТ-предприятия предложена операционная среда реализации компонентов SOA (ODOE — On Demand Operating Environment). Приведем ее основные составляющие [6]:

Сервисы приложений (Application Services) — все компоненты, необходимые для построения скомпонованной информационной системы.

Сервисная шина предприятия (Enterprise Service Bus — ESB) — интеллектуальный слой для подключения информационных систем, различных данных и других сервисов, которые обычно распределены по всей ИТ среде предприятия.

Бизнес-сервисы (Business Services) — открытая часть бизнес-процессов и бизнес-функциональности, к которой обращаются и которая обеспечивает реализацию заданной ценности на запрос.

Общие сервисы (Common Services) — обеспечение доставки и индивидуальной обработки данных, а также других утилит, таких как формирование отчетности. Эти сервисы позволяют определять и выполнять рабочие потоки, в основном для логических сервисов бизнес-функций.

Сервисы информационного менеджмента (Business Performance Management) — унифицированный способ представления, доступа, поддержания, управления, анализа и интеграции данных и контента из гетерогенных информационных источников.

Сервисы инфраструктуры (Infrastructure Services) — набор независимых от платформы сервисов, который позволяет всем другим сервисам быть установленными, выполняться и контролироваться в конкретной инфраструктуре, составленной из операционных систем и сетевых и аппаратных систем.

Информация о наиболее известных средствах разработки сервисов приведены в табл. 1. На сегодня существует ряд известных репозитариев сервисов общего назначения, например:

Amazon Web Service Developer Connection (http://Developer. amazonweb services.com);

XML Web Services Repository (www.xmlwebservices.cc);

UDDI service registry (www.uddi.org);

Xmethod service registry (xmethods.net/);

IBM UDDI (http://www-306.ibm.com/software/solutions/webservices/uddi/);

Internet of Things (www.fi-ppp.eu/projects/).

Таблица 1. Наиболее известные средства разработки SOA и SOC

Средство	Реализуемые функции SOA	Компания	Комментарий
Web-sphere	Композиция сервисов, их поиск и моделирование	IBM	Флагман SOA инструментария компании IBM. Требует значительных знаний SOA
.NET	Создание сервисов, их обнаружение, композиция, развертывание	Microsoft	Основной продукт компании Microsoft для SOA. Основан на системе Windows и использует C # как основной язык программирования
HP SOA Center	Композиция сервисов, моделирование, интеграция, менеджмент	HP	Базовый SOA инструментарий компании HP. Требует значительных знаний SOA
Weblogic	Развертывание сервисов, их композиция, менеджмент, политика использования	BEA	Основной продукт компании BEA для SOA. Предназначен для профессионального использования
Oracle SOA Suite	Создание сервисов, развертывание, композиция, оркестровка	Oracle	Флагманский продукт Oracle для SOA. Требует использования высококачественных серверов, а не настольных компьютеров
Enterprise SOA	Создание сервисов, развертывание, композиция	SAP	Продукт компании SAP интегрирован со многими из существующих продуктов SAP. Достаточно сложный и часто требует, чтобы пользователь знал о SAP продуктах

ТИПЫ СЕРВИС- ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

С увеличением внимания предприятий к web-основанным решениям и сервисам наблюдается повышение спроса на бизнес-сервисы и бизнес-функциональность. Крайне необходимы исследования по выявлению возможного выбора ***инвариантных сервисов для систем***, которые фокусируются на человеческой деятельности (электрические сети, системы водоснабжения, транспортные системы, система здравоохранения, система образования, банковско-финансовые системы, сети розничной торговли, системы туризма, медиа и развлечений и др.). Это позволит создать ***репозитарий междисциплинарных инвариантных сервисов как строительных блоков*** соответствующих систем сервисов. Объем исследований настолько велик, что требует коллективных усилий многих партнеров и соучастников.

Такой репозитарий мог бы стать основой для *создания среды знаний о сервисах*, или *Service Science Knowledge Environment*, которая объединит научные круги, промышленность и правительство, а также другие европейские институты. При этом первоначальные ресурсы знаний могут относиться к различным доменным приложениям, таким как *электронная администрация, электронное правительство, электронное здравоохранение, производство, энергопотребление с акцентом на возможности SmartGrid сети, пе-*

редовые программные сервисы, цепочки поставок и логистики, сервисы туризма и отдыха, передовые сервисы в области телекоммуникаций. Список сервисных секторов, обеспечивающих реальный вклад в современную экономику, не ограничивается перечисленными приложениями. У*мный транспорт, умные здания, интеллектуальное водоснабжения* — это только некоторые дополнительные сервисные сектора возможного применения, для которых новые сервисы могут быть задуманы и разработаны.

Web-сервисы, описанные на языке WSDL, содержат преимущественно синтаксическую информацию, и это затрудняет организацию автоматического поиска web-сервиса и выполнение композиции сервисов. Поэтому (кроме введения в описания семантических аннотаций) предложено непосредственно использовать семантическую информацию в web-сервисах, в результате чего появилось понятие ***семантических web-сервисов (SWS***) со своими языками описания, представителями которых являются ***OWL-S (Web Ontology Language for Web Services)*** и ***WSMO (Web Service Modeling Ontology).*** Сервисы SWS создаются на базе существующей технологии web-сервисов для обеспечения динамического поиска сервисов, их композиции, вызова web-сервисов. Результатом этого является возможность автоматизации процедур, которые требуют вмешательства разработчиков программного обеспечения. Это может быть достигнуто использованием более *точных описаний семантических web-сервисов, которые понятны для компьютера.*

SOA с помощью web-сервисов становится основой для объединения разнородных бизнес-систем через Интернет. С помощью SOA функциональные потребности предприятия в корпоративных системах удовлетворяются набором сервисов, а не отдельным приложением, что фундаментально меняет подходы разработчиков к реализации. Конфигурация и координация сервисов в архитектуре, основанной на сервисах, и композиция сервисов и процессов одинаково важны в современных системах сервисов [7]. Сервисы взаимодействуют друг с другом с помощью сообщений. Сообщение можно осуществить с помощью ***шаблона «запрос–ответ»***, когда в определенный момент времени только один из конкретных сервисов вызывается одним потребителем (связь «один-к-одному», или синхронная модель); с использованием шаблона ***«публикация / подписка»***, когда на одно конкретное событие может откликнуться много сервисов (связь «один-ко-многим», или асинхронная модель); с ***привлечением интеллектуальных агентов*** (***MAS***), которые определяют координацию сервисов, поскольку каждый агент имеет в своем распоряжении определенные знания о бизнес-процессе и может обмениваться этими знаниями с другими агентами.Такая система способна сочетать качества SOC, такие как ***функциональная совместимость*** и ***открытость***, со свойствами MAS, такими как ***гибкость*** и ***автономия***.Но необходимы дополнительные исследования по созданию практических механизмов для управления агентными сервисами и их коммуникации в интернет-среде.

В ИПСА НТУУ «КПИ» исследуются гибридные ***сервис-ориентированные архитетуры, управляемые событиями (EDSOA)***, в которых сервисы генерируют события и тем самым переводят бизнес-процесс из одного состояния в другое. В свою очередь, благодаря событиям сервисы подключаются для передачи состояния процесса и данных от одного сервиса, который об-

наружил и опубликовал событие, к другим сервисам, запускаемым при появлении конкретных событий. Такое формирование бизнес-сервисов и бизнес-событий, а затем их целевая увязка для решений задач бизнеса позволяют добиться стратегических преимуществ для предприятия. При этом необходимо «накинуть» на всю программную инфраструктуру организации «сеть» программных датчиков, программных агентов и аппаратных датчиков, которые отслеживают события во всех аппаратных и программных компонентах, а также значимые для бизнеса события и передают в центр принятия решений сигналы, ассоциированные с этими событиями. Это позволяет технологически управлять бизнесом не вслепую, а иметь четкое представление о всем происходящем в данный момент времени на предприятии.

При проектировании систем сервисов важное значение имеет *зернистость* сервисов, т.е. уровень детализации обслуживания. Обычно используют модули бизнес-логики достаточно высокого уровня, благодаря чему взаимодействие между ними сводится к ограниченному количеству сообщений, при этом снижается нагрузка на сеть. Но в последнее время широкое распространение получают *микросервисы*, реализующие только одиночные функции [8].Микросервисы представляют собой бизнес-функции, а не общие функции программного обеспечения, например, поиск в базе данных. Они являются логическими компонентами, развернутыми в виде RESTful сервисов и доступными через HTTP и URL.

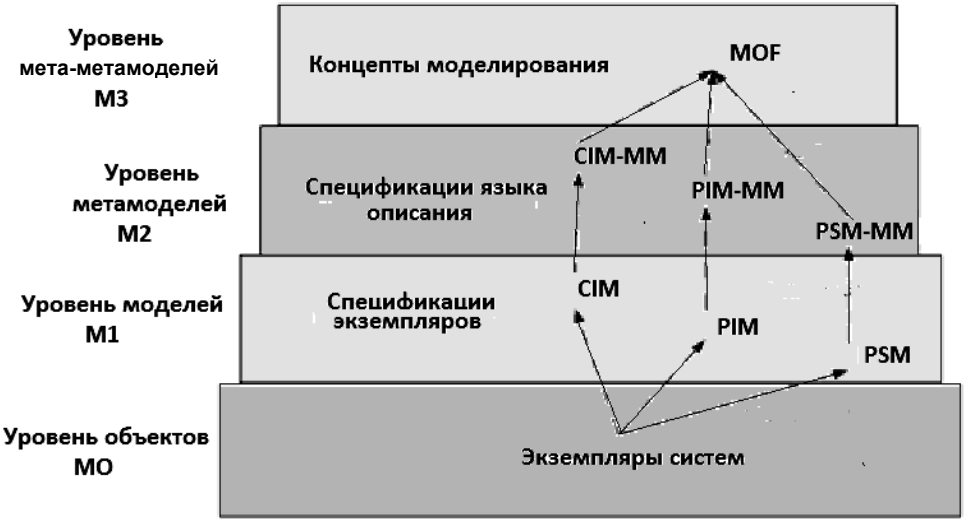
РАЗРАБОТКА СИСТЕМ СЕРВИСОВ НА ОСНОВЕ ОНТОЛОГИЙ ПРЕДМЕТНОЙ ОБЛАСТИ

В последние годы исследуется концепция разработки систем сервисов на базе ***онтологий*** разного уровня (*Domain, Application, Task Ontologies*). Онтологии, описанные на языке OWL, — это формальное явное описание понятий в данной предметной области. В центре большинства онтологий находятся ***классы.*** Например, класс *Action* включает все процедуры (запуск задач, передача данных, контроль потока данных и т.д.). Конкретные процедуры — это ***экземпляры*** этого класса. Класс имеет подклассы, которые содержат более конкретные понятия, чем сверхкласс. ***Слоты*** описывают свойства классов и экземпляров: процедура Task может содержать файл (*containsFile*), создавать ресурс (*createsResource*) или зависеть от определенных условий (*hasDependency*).

Слоты могут иметь различные ***фацеты***, которые описывают тип значения, числовые значения, разрешенные пределы их изменения и др. Онтология вместе с набором индивидуальных экземпляров классов образует ***базу знаний***. Но определить, где заканчивается онтология и где начинается база знаний, трудно. Группой управления объектами OMG (Object Management Group) [9] предложена иерархическая система уровней описания семантики системы сервисов при ее моделировании, которая состоит из *информационного уровня* (M0), *уровня моделей* (M1), *уровня метамодели* (M2) и *уровня мета-метамодели* (M3), который часто называют ***Meta Object Facility*** (MOF) (см. рисунок).

Процесс проектирования системы сервисов может быть сведен к *процессу преобразований моделей*, начиная с метамодели в качестве входных

данных и используя набор правил преобразования [9–14]. При этом преобразование само по себе является также моделью. Модели преобразования имеют различные области применения:



Связь между моделями MDA и метамоделями

– создание моделей нижнего уровня и в итоге ***программного кода***, двигаясь от моделей более высокого уровня;

– отображение и синхронизация между моделями на том же уровне или на различных уровнях абстракции;

– создание представления о системе на базе запросов;

– эволюция задач модели, таких как рефакторинг модели, т.е. изменение внутренней структуры системы с целью сделать её проще для понимания и внесения дальнейших изменений, не изменяя существующей функциональности;

– обратное проектирование от моделей нижнего уровня или кода до модели более высокого уровня.

Преобразование моделей является ключевым элементом в проектировании систем сервисов для обеспечения плавного средства обработки входных онтологических моделей с целью генерирования, фильтрации и обновления целевых моделей. Языки преобразования моделей реализуют различные типы преобразования, такие как ***модель-модель*** **(**ММ) или ***модель-код*** **(**МК**)**. Особенностью данного подхода является то, что разработчик большую часть времени работает не с кодом, а с моделями.

Для трансформации моделей предложен **модельно-управляемый подход (**MDA**)**. MDA использует MOF-основанные модели для создания и манипулирования точными, подробными, компьютерно-считываемыми описаниями структуры приложений, не зависящими от языков программирования, операционных систем или баз данных, которые могут быть использованы для их реализации. Ключевым стандартом MDA **является MOF, а не UML,** как считают до сих пор некоторые пользователи.

Подход MDA можно использовать для получения контроля и систематического улучшения всего жизненного цикла ИТ-решений: от моделирова-

ния общего бизнеса (содействия эффективной связи между бизнес-аналитиками и ИТ-разработчиками и выполнения конкретных требований к решению) до разработки, внедрения, интеграции и управления многими видами программных артефактов. MDA на основе трехслойного подхода обеспечивает такие типы моделей:

Вычислительная независимая модель CIM (Computing Independent Model) — описывает систему с вычислительной независимой точки зрения, освещая структурные аспекты системы, изменяя акцент от моделирования домена на моделирование архитектуры.

Независимая от платформы модель PIM (Platform Independent Model) — может рассматриваться как определение системы с точки зрения технологии нейтральной виртуальной машины или вычислительной абстракции.

Модель для конкретной платформы PSM (Platform Specific Model) — охватывает технические концепции и сервисы, составляющие платформу реализации. Эта модель направлена на конкретную технологию реализации системы сервисов.

Метамодель описывает свойства и конструкции каждой из CIM, PIM и PSM моделей точно (см. рисунок). В контексте работы выбирается платформа на основе web-сервисов. Поставщик сервисов создает абстрактное описание интерфейса сервиса, который может быть использован потенциальными пользователями сервисов для поиска и вызова этого сервиса с помощью сообщений на основе XML. Параметры web-сервисов при обращении к репозиторию, в котором потенциальные пользователи могут искать соответствующие сервисы и протоколы для связи сервисов и их клиентов, образуют платформу реализации как сервис-ориентированную архитектуру.

Различные типы платформы могут отличаться друг от друга. Общим свойством платформы является ее сервис-ориентированная архитектура, а конкретными технологиями реализации — сами web-сервисы и сервисные механизмы (например, Apache Axis или Oracle BPEL). Метамоделирование является одним из самых важных понятий MDA. ***Онтологии рассматриваются как CIM модели***, поэтому концептуальные модели PIM и PSM также отражают семантику для данного домена приложений, соответствующую семантике доменов реального мира (Real World) независимо от конкретных потребностей приложений. Таким образом, онтология выступает в качестве основы метамодели для генерирования концептуальных моделей для реализации конкретных информационных систем.

Возможное слияние онтологического и MDA подходов приводит к следующему возможному процессу разработки программного обеспечения:

– построить независимые общую (GO) и доменную (DO) онтологии. Часть DO использовать в качестве вычислительной независимой модели (CIM);

– преобразовать CIM в независимую от платформы модель (PIM);

– преобразовать PIM в модель конкретной платформы (PSM) и артефакт зависимых моделей (например, модель схемы базы данных); определить методику сравнения и поиска сервисов;

– превратить части онтологий DO и GO в онтологию контекста (CO);

– применить преобразования PSM в коды приложений и программных артефактов [15].

При создании модели конкретной платформы (PSM) осуществляется поиск необходимых сервисов [16, 17] в сети, при этом используются онтологии для повышения точности поиска сервиса и открытие такого сервиса, в котором упоминается в точности искомое понятие, а не произвольных сервисов, в описаниях которых встретилось заданное ключевое слово. Известно, что в различных предметных областях одни и те же понятия могут быть представлены различными терминами. Механизм онтологии в этих случаях позволяет формировать осмысленные иерархические взаимосвязи между сервисами, т.е. реализовать композицию сервисов, способную удовлетворить запрос пользователя, хотя в ее описании может не быть некоторых ключевых слов из входного запроса.

Рассмотренный подход к проектированию систем сервисов хорошо адаптируется к особенностям *современных распределенных вычислительных сред* (типа мультиоблака или грид), когда web-сервисы находятся в различных географически разделенных репозитариях (возможно, что в нескольких эквивалентных вариантах реализации), а их композиции в виде созданных приложений выполняются на различных вычислительных узлах среды, ресурсы которых освободились к моменту начала выполнения очередного сервиса композиции. В результате реализованное ***приложение имеет динамически изменяемую архитектуру и переменный компонентный состав***.

ОСОБЕННОСТИ ПРОЦЕССНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ

Применительно к разработке программных приложений особенности процессно-ориентированного проектирования обобщены в табл. 2

Таблица 2. Сравнение различных подходов к разработке программного обеспечения

№ п/п	Объектно-ориентированный	Процессно-ориентированный
1	**ОТВЕСТВЕННЫЕ ИСПОЛНИТЕЛИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**	
	Разработчики программного обеспечения, большую часть времени работающие с кодом	*Архитекторы программного обеспечения*, большую часть времени работающие совместно с экспертами предметной области *с онкологиями и моделями*
2	**ЭТАП АНАЛИЗА ТЕХНИЧЕСКОГО ЗАДАНИЯ**	
	Требования приведены на естественном языке для системных аналитиков, которые преобразовывают их технические характеристики для разработчиков	Анализ домена выполнен поставщиками сервисов, что позволяет разработчикам приложений сосредоточиться только на поиске и комбинировании сервисов, которые отвечают бизнес-техническим характеристикам

Продолжение табл. 2

3	**ЭТАП ПРОЕКТИРОВАНИЯ**	
	Уточнение структуры классов, проектирование архитектуры системы (структуры компонентов и модулей), выбор строительных блоков. Обновление их с учетом технических изменений является сложной задачей	Выбор или построение онтологии прикладного домена и проектируемого приложения, выбор репозитария сервисов
4	**ЭТАП РЕАЛИЗАЦИИ**	
	Кодирование на конкретном языке программирования (Java, C ++, C #). Разработка выполняется одной (виртуальной или физической) командой, которая создает функции, классы, модули	Модели сервисов представлены в машиночитаемой форме, что позволяют разработчикам сервисов автоматически перевести их в исполняемый код (CIM-> PIM -> PSM->UML).Новые сервисы динамически создаются с использованием существующих
5	**ЭТАП ТЕСТИРОВАНИЯ И ПОДДЕРЖКИ**	
	Тестирование обычно проводится тестеровщиками в той же организации на основе исходного кода и функциональных спецификаций. Тестовые сценарии определяются разработчиками / тестеровщиками	Тестирование разделяется между поставщиком сервисов, брокером и клиентом без взаимодействия между ними. Сервисы могут быть проверены до их регистрации и использования. Тестовые сценарии могут быть автоматически сгенерированы на месте совместно брокером и клиентом на основе сервисных метаданных и спецификаций
6	**ОБЩИЕ ХАРАКТЕРИСТИКИ РЕШЕНИЙ**	
	Проектно-зависимые решения, которые трудно использовать повторно, с относительно краткосрочным изолированным применением в выбранном домене	Кросс-проектные решения, обслуживающие несколько доменов и предназначенные для повторного долгосрочного совместного использования многими проектантами и организациями

ВЫВОДЫ

Ориентация мировой экономики на индустрию сервисов, появление междисциплинарной науки о сервисах, распространение сервисных подходов на технические системы (в частности, структуры программного обеспечения) мотивируют экономистов, социологов, математиков, программистов и законодателей сотрудничать для достижения очень важной цели: анализа, построения, управления и развития сложных систем сервисов. Главная задача состоит в выявлении логики сложных систем сервисов и внедрении общей методологии их моделирования. Решение этой задачи требует специалистов нового типа — сервисных специалистов, о которых в Украине почти ничего неизвестно [17]. Для ее решения в 2015 г. в ИПСА НТУУ «КПИ» введена специализация «Системное проектирование сервисов» в рамках новой специальности 122 «Компьютерные науки и информационные технологии». Наука о сервисах (которая развивается довольно активно) имеет несомненные междисциплинарные признаки с долгосрочной целью стать действительно междисциплинарной отраслью.

ЛИТЕРАТУРА

1. *Jürgen Kress.* Industrial SOA / Jürgen Kress, Berthold Maier, Hajo Normann, Danilo Schmeidel, Guido Schmutz, Bernd Trops, Clemens Utschig-Utschig, Torsten Winterberg. — Available at: http://www.oracle.com/technetwork/ articles/soa/ind-soa-toc- 1934143.html.
2. *Handbook of* Service Science. Service Science: Research and Innovations in the Service Economy / P. Maglio, C.A. Kieliszewski, J. Spohrer // Springer, New York, 2010.
3. *Service* Systems Implementation. Service Science: Research and Innovations in the Service Economy / H. Demirkan, J. Spohrer, V. Krishna // Springer, New York, 2011.
4. *Services* Science: Fundamentals, Challenges and Future Developments / B. Stauss, K. Engelmann, A. Kremer, A. Luhn // Springer, Berlin, 2007.
5. *Петренко А.А.* Объекты и методы науки о сервисах / А.А. Петренко // Системні дослідження та інформаційні технології. — 2015. — № 2. — С. 75–82.
6. *Succeeding* through service innovation: A service perspective for education, research, business and government, ISBN: 978-1-902546-65-0. // University of Cambridge Institute for Manufacturing (IfM) and International Business Machines Corporation (IBM), April 2008. — 30 p.
7. *Петренко О.О.* Порівняння типів архітектури систем сервісів / О.О. Петренко // Системні дослідження та інформаційні технології. — 2015. — № 4. — С. 48–62.
8. *Newman Sam.* Building Microservices: Designing Fine-Grained Systems / Sam Newman // O'REILLY, 2014. — 182 p.
9. *World* Wide Web Consortium. Web Services Architecture. — Available at: http://www.w3.org/TR/ws-arch, 2006.
10. *Valiente María-Cruz*. An Ontology-Based and Model-Driven Approach for Designing IT Service Management Systems / María-Cruz Valiente, Cristina Vicente-Chicote, Daniel Rodríguez // International Journal of Service Science, Management, Engineering, and Technology, 2(2), 65–81, April-June 2011.
11. *Alahmari S.* A Model-Driven Architecture Approach to the Efficient Identification of Services on Service-oriented Enterprise Architecture / S. Alahmari, D. De Roure, Ed. Zaluska // At The Second Workshop on Service oriented Enterprise Architecture for Enterprise Engineering in conjunction with the 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brazil, 2010.
12. *Pahl Claus*. Semantic Model-Driven Architecting of Service-based Software Systems / Claus Pahl. — Available at: http://ceur-ws.org/Vol-244/paper3.pdf
13. *Pahl C.* An Ontology for Software Component Matching / C. Pahl // International Journal on Software Tools for Technology Transfer, Special Edition on Component-based Systems Engineering. — 2006. — № 7. — P. 1–10.
14. *Петренко А.А.* Семантическое модельно-управляемое моделирование архитектуры системы сервисов на основе доменных онтологий / А.А. Петренко, А.И. Петренко // Electronic Modeling. 2016. — **38**, —№ 5. — ISSN 0204–3572.
15. *Kalyanpur A.* Automatic mapping of OWL ontologies into Java / A. Kalyanpur, D. Pastor, S. Battle, J. Padget // Proceedings of the International Conference on Software Engineering &Knowledge Engineering (SEKE), 2004.
16. *Петренко І.А.* Автоматизованіметодипошукуівідкриттянеобхіднихсервісів / І.А. Петренко, О.О. Петренко // Вісник Університету «Україна». Серія «Інформатика, обчислювальна техніка та кібернетика». — 2015. — №1 (17). — С. 55–64.
17. *Андон П.* Роль семантики в интеграции приложений на основе веб-сервисов / П. Андон, В. Дерецький // Проблеми програмування. — 2010. — № 2–3. Спец. вип. — С. 469–478.
18. *Петренко О.О.* Підготовка кадрів для індустрії сервісів / О.О. Петренко // Information Technologies in Education. — 2015. — № 22. — С. 158–164. — DOI: 10.14308/ite000528

Поступила 28.10.2016