УДК 519.711.3

ПРОЕКТИРОВАНИЕ СИСТЕМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОД УПРАВЛЕНИЕМ ОНТОЛОГИЙ: МОДЕЛИ, МЕТОДЫ, РЕАЛИЗАЦИИ

В.Ф. Хорошевский

Вычислительный центр им. А.А. Дородницына РАН, ФИЦ «Информатика и управление» РАН, Москва, Россия
V.Khor@mail.ru

Аннотация

В представленном обзоре обсуждаются вопросы проектирования систем программного обеспечения под управлением онтологий. На основе краткого анализа «классических» технологий, подходов и методов показано, что в настоящее время акценты исследований и разработок всё больше смещаются из области технологий программирования в сторону моделирования процессов проектирования программного обеспечения, а также спецификации разрабатываемых систем и тестирования результатов разработок. Общим трендом является использование методов и средств онтологического моделирования процессов проектирования и спецификаций разрабатываемых систем. Обсуждаются парадигмы моделирования процессов разработки и инжиниринг программного обеспечения на основе повторного использования компонент. При этом общим подходом, поддерживающим рассмотренные парадигмы, является программирование через генерацию. Показано, что исследования и разработки в области создания семантических моделей проектирования программного обеспечения развиваются в настоящее время в рамках нового направления – *Ontology-Based (-Driven) Software Engineering*. При этом наряду с разработками по онтологическому моделированию процессов проектирования программного обеспечения активно ведутся работы по созданию инструментов онтологического инжиниринга в данной предметной области. Теоретические основы онтологического моделирования процессов проектирования программного обеспечения позволяют сместить центр тяжести от моделей сборки программного обеспечения прикладных систем из крупных функциональных блоков к моделям генерации блоков из согласованной системы онтологических паттернов их внутренних спецификаций. Перспективы в рассматриваемой области связаны с использованием методов и средств машинного обучения для генерации онтологических моделей проектирования программного обеспечения.

Ключевые слова: *технологии проектирования, программное обеспечение, онтологическое моделирование, проектирование под управлением онтологий, система онтологий проектирования, прикладная система.*

Цитирование: *Хорошевский, В.Ф.* Проектирование систем программного обеспечения под управлением онтологий: модели, методы, реализации / В.Ф. Хорошевский // Онтология проектирования. – 2019. – Т. 9, №4(34). - С.429-448. – DOI: 10.18287/2223-9537-2019-9-4-429-448.

Введение

Автоматизация проектирования систем программного обеспечения (ПО) имеет более чем 40-летнюю историю и, вместе с тем, остаётся в фокусе современных исследований и разработок. Постановка задачи автоматизации проектирования систем ПО предполагает наличие адекватных моделей всего жизненного цикла (ЖЦ), включая технологии разработки и реализации, модели архитектуры систем и их программных компонент, а также модели генерации ПО и модели предметных областей (ПрО), в которых предполагается функционирование соответствующих систем. Основным трендом в настоящее время является создание и использование семантических моделей автоматизации проектирования ПО.

В работе обсуждаются вопросы проектирования ПО прикладных систем под управлением онтологий. Выполнен ретроспективный обзор используемых в этой ПрО подходов, моде-

лей и методов. Показано, что в настоящее время общим трендом в исследованиях и разработках по автоматизации проектирования систем ПО является использование методов и средств онтологического моделирования как самих процессов проектирования, так и спецификаций разрабатываемых систем. В заключительной части работы указываются перспективы развития рассматриваемого научно-технического направления.

1 Автоматизация проектирования систем ПО

1.1 Модели ЖЦ процессов проектирования ПО

В рамках создания технологий программирования разработаны и активно используются на практике различные модели процессов проектирования ПО [1-3], среди которых наиболее известными являются каскадная модель (*Waterfall model*) [4], спиральная модель Боэма (*Boehm Spiral model*) [5], модель экстрим-программирования (*Extreme programming*) [6], модель «проворного» программирования (*The Agile Software Process*) [7], модель RUP (*Rational Unified Process*) [8], а также стандарты ISO [9, 10] и др. Каждая из перечисленных моделей обладает достоинствами и недостатками, но общим их несовершенством является то, что все они скорее методологические рекомендации и требования, чем формальные модели ЖЦ, которые могут быть основой для полной автоматизации процессов проектирования ПО. Как показывает анализ публикаций, на современном этапе всё большее внимание исследователей и разработчиков привлекают парадигмы и модели проектирования ПО.

1.2 Парадигмы и модели проектирования ПО

Одна из известных парадигм моделирования процессов разработки ПО опирается на понятие ПО-семейства – *Software Product Lines* (SPLs), которое представляет ПО с общей архитектурой и повторно используемыми компонентами [11], что повышает качество, сокращает время, стоимость и сложность разработки [12]. Основная цель SPLs - отказ от разработок «с нуля» путём реконфигурирования и повторного использования компонент существующих в других проектах SPLs. Разработка на базе SPLs включает два процесса: *Domain Engineering* (DE) и *Application Engineering* (AE). Задача DE - разработка общей архитектуры систем семейства и формирование планов для «членов» семейства. Процесс AE ориентирован на создание конкретных продуктов вовремя и с минимальной ценой при высоком качестве.

Сходная с предыдущей парадигма разработки ПО на основе компонент – *Component-Based Development* (CBD или CBE) – базируется на повторном использовании компонент в стиле «вставил и заработало» (*plug-and-play*) и направлена на осуществление давней мечты о создании сложных систем ПО из компонент «с полки» (*off-the-shelf*) [13].

Парадигма инжиниринга ПО под управлением моделей – *Model-Driven Engineering* (MDE) обращается к модельным абстракциям и автоматической генерации кода. В рамках парадигмы MDE основные свойства проектируемой системы выражаются в соответствующих моделях, а генераторы используются для получения исходного кода ПО для различных объектов модели [14, 15]. Модель представляет собой описание будущей системы, которое рассматривается не как документ спецификации, а как повторно используемый артефакт на всём ЖЦ создания ПО [16]. В MDE используются техники «послойного» проектирования с автоматической трансформацией моделей в исходные коды ПО. В рамках MDE используются три слоя: вычислительно-независимая модель – *Computation Independent Model* (CIM), специфицирующая структурные аспекты будущей системы; платформенно-независимая модель – *Platform Independent Model* (PIM), фиксирующая систему как технологически нейтральную виртуальную машину или вычислительную абстракцию; платформенно-зависимая модель –

Platform Specific Model (PSM), которая включает модель платформы и модель реализации. В MDE разработчики создают первую версию разрабатываемого ПО на языке спецификаций UML (*Unified Modeling Language*), а затем путём итераций уточняют её до тех пор, пока она не будет отвечать заданным критериям и не станет достаточно конкретной, чтобы из неё можно было сгенерировать исходные коды ПО разрабатываемой системы.

Общим подходом, поддерживающим рассмотренные выше парадигмы, является программирование через генерацию – *Generative Programming* (GP) [14]. При этом генератор рассматривается как программа, на вход которой поступают высокоуровневые спецификации фрагментов ПО, а на выходе формируется соответствующий исходный код ПО разрабатываемой системы. GP использует техники DE и может применяться в SPLs AE. Дополнительно к UML-моделированию в подходе GP применяется метод *Feature-Oriented Domain Analysis* (FODA), который описывает общие и частные свойства моделируемых объектов и связей между ними. Предметная GP-модель фокусируется на отображении «пространства» проблемы в «пространство» решения, а сам генератор использует техники метапрограммирования для чтения, генерации, анализа или трансформации других программ и даже для модификации своего кода в процессе исполнения [12].

Приведённый обзор парадигм, моделей и методов показывает, что они ориентированы на формальную спецификацию процессов проектирования ПО, но остаются в «традиционном» русле автоматизации программирования. В последнее время всё большее внимание исследователей и разработчиков привлекают семантические модели проектирования ПО.

2 Семантические модели проектирования ПО

Исследования и разработки в области создания семантических моделей проектирования ПО развиваются в настоящее время в рамках нового направления – *Ontology-Based Software Engineering* (OBSE) или *Ontology-Driven Software Engineering* (ODSE). Иногда это направление называют программной инженерией - *Software Engineering* 2.0 [17-22].

Так, в работе немецкого специалиста В. Гессе (W. Hesse) [17] даётся сравнительный анализ атрибутов ЖЦ программной инженерии и онтологического инжиниринга (таблица 1).

В рамках OBSE-ODSE можно выделить несколько тематик исследований и разработок. Наиболее развитым является онтологическое моделирование процессов управления разработками ПО [21], которое исторически имеет своим базисом работы в рамках канадского проекта TOVE (*TOronto Virtual Enterprise*) и проекта *Enterprise Project* [23, 24].

С точки зрения целей и задач настоящей работы интересными являются исследования и разработки по созданию онтологий проектирования ПО [25-31], а также по разработке и использованию соответствующих инструментальных средств в конкретных прикладных системах на основе онтологий [32-40].

2.1 Семантические модели процессов создания ПО

Как показывает анализ публикаций, все семантические модели процессов создания ПО опираются на ту или иную классификацию используемых моделей.

В работе [26] на основе анализа фаз ЖЦ построения систем ПО предложена следующая классификации системы онтологических моделей для этой ПрО:

- разработка под управлением онтологий – *Ontology-driven development* (ODD) – предполагает использование онтологий в процессе создания ПО для спецификации ПрО разработки;

Таблица 1 – Сравнительный анализ атрибутов ЖЦ программной инженерии и онтологического инжиниринга (адаптировано из [17])

	Software Engineering	***Ontology Engineering***
Длительность	• детерминирована, ограничена рамками одного проекта	• недетерминирована, неограничена
Структура	• фазы, группировка в итерации и активности	• эволюционные, группировка в итерации и активности
Подпроцессы	• для компонент, инкрементные или специальные задачи	• для разработки или ревизии ПрО
Парадигмы	• водопадная • инкрементная • на базе компонент • прототипирование, спиральная	• инкрементная • на базе компонент • эволюционная
Активности и артефакты	• поиск и описание случаев • определение требований • построение концептуальной модели / класса структур • построение архитектуры системы, спецификация компонент и модулей • кодирование: использование / генерация на конкретных языках программирования • тестирование и отладка • интеграция, тестирование подсистем • развёртывание • валидация, адаптация к окружению • получение обратной связи от пользователей • ревизия (при необходимости)	• идентификация потенциальных приложений • анализ терминологии • построение таксономий • построение и заполнение глоссариев • определение фактов и правил • сравнение глоссариев, разрешение терминологических конфликтов • формализация: трансляция в формальный язык спецификации онтологий • интеграция: связывание и валидация под-онтологий, адаптация с онтологиями верхнего / соседнего уровня • публикация онтологий • валидация, получение обратной связи • определение требований для ревизии
Языки / средства спецификации	• диаграммы случаев • естественный язык • ER-диаграммы, UML • псевдокод • языки программирования	• естественный язык • ER-диаграммы, UML • глоссарии, таблицы • семантические сети • тематические карты • логики • концептуальные графы • фреймы, DAML+OIL, OWL
Целевые группы	• пользователи • клиенты • (другие) разработчики	• предметные эксперты • разработчики • агенты
Результаты и продукты	• ориентация на проект • краткосрочная ориентация • (чаще) не используемые повторно • изолированы, ограничены конкретными приложениями	• охватывающие много проектов • ориентированы на долгосрочную перспективу • повторно используемые • «разделяемые» многими организациями и между многими проектами

- поддержка разработки на основе онтологий – *Ontology-enabled development* (OED) – тоже использует онтологии в процессе разработки, но здесь они ориентированы на поддержку разработчиков при решении их задач;
- архитектура ПО на базе онтологий – *Ontology-based architectures* (OBA), здесь онтологии используются как первичные артефакты на этапе исполнения и играют центральную роль в логике приложения;
- поддержка архитектуры на основе онтологий – *Ontology-enabled architectures* (OEA) – предполагает использование онтологий для обеспечения поддержки инфраструктуры во время работы программной системы (хорошим примером здесь могут быть базы знаний интеллектуальных прикладных систем, где онтологии добавляют семантический слой поверх существующих описаний, расширяя функциональные возможности разработанного ПО).

Отдельные кластеры данной классификации могут перекрываться, а онтологии из этих кластеров – использоваться на разных этапах ЖЦ разработки ПО.

Общепризнанные в области онтологического инжиниринга специалисты Н. Гуарино (N. Guarino[1]) и Д. Фензел (D. Fensel[2]) придерживаются несколько другой классификации онтологий, которая базируется на уровнях общности онтологических моделей:

- онтологии верхнего уровня (*Generic* или *Upper*), специфицирующие общие знания о мире и применимые в различных ПрО;
- онтологии представления (*Representational*) сущностей без привязки к определённой ПрО;
- предметные (*Domain*) онтологии, где специфицируются знания из заданной ПрО;
- онтологии методов и задач (*Method & Task*).

В работе [41] предложена расширенная классификация онтологий на основе предмета концептуализации:

- онтологии представления знаний (*Knowledge representation ontologies*), которые используются для спецификации знаний в рамках конкретной парадигмы их представления;
- общие онтологии (*Common or generic ontologies*) представляют повторно используемые знания здравого смысла (*common-sense knowledge*);
- онтологии высшего уровня (*High-level ontologies*) описывают общие понятия;
- предметные онтологии (*Domain ontologies*), предлагающие словари для понятий в определенной ПрО;
- онтологии задач (*Task ontologies*), где специфицируются словари базовых (общих) активностей;
- онтологии задач ПрО (*Domain task ontologies*), повторно используемые в некоторой ПрО;
- онтологии методов (*Method ontologies*), которые применяются для спецификации процессов вывода при выполнении частных задач;
- прикладные онтологии (*Application ontologies*) зависят от приложений и часто являются специализацией предметных или задачных онтологий.

Работа специалистов из Юго-Восточной Азии [29] посвящена созданию и использованию онтологии процессов ПО (*Software Process Ontology*), фрагмент которой приведён на рисунке 1. Как утверждают авторы, разработанная ими онтологическая модель отвечает требованиям ISO/IEC 15504.

---

[1] См. например: ***N. Guarino***. Formal Ontology and Information Systems. Proc. of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15. - *Прим.ред.*

[2] См. например: ***D. Fensel.*** Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag, Berlin, 2001. 2nd Edition, Springer-Verlag, Berlin, 2003. - *Прим.ред.*

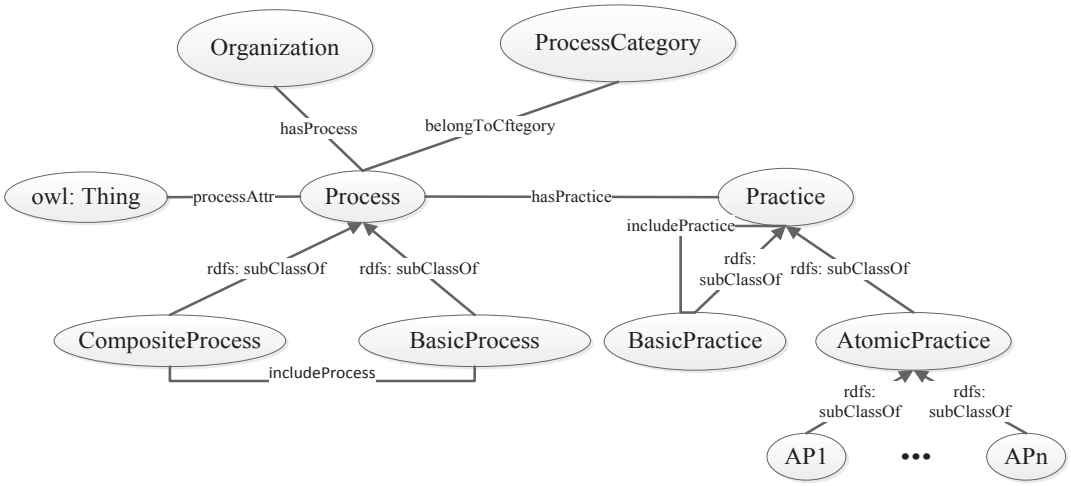

Рисунок 1 – Фрагмент онтологии процессов ПО [29]

В работе [42] для поддержки процессов сопровождения ПО описан комплекс онтологий сопровождения ПО, включающий онтологию ПО, где представлены такие концепты, как ПО, пользователи и документация; онтология знаний и умений в области *computer science* с концептами компьютерные технологии и языки моделирования; онтология процессов модификации с концептами запрос на модификации и обслуживание; онтология организационных структур с такими понятиями, как организационная единица и директива; прикладная онтология, которая связывает понятия ПрО с задачами, которые должны в ней быть выполнены. С использованием метода извлечения знаний в области создания систем ПО (*Post-Mortem Analysis*) [43] разработана методология, которая обеспечивает точное представление знаний о разных стадиях процесса сопровождения ПО согласно ISO/IEC 14764 [44] на базе использования собственного комплекса онтологий сопровождения ПО.

Многолетние исследования и разработки в области онтологий ведутся и в департаменте Компьютерной науки и Информационных систем (Department of Computer Science and Information Systems) университета Яваскила (University of Jyvaskyla) в Финляндии. Одной из базовых публикаций этого исследовательского коллектива является работа его руководителя М. Леппанена (M. Lepparen) [45], в которой обсуждается методология проектирования многоуровневых онтологий на базе контекстов (рисунок 2).

В подходе финской школы выделяются семь контекстных областей, из которых «получают» концепты для уточнения и интерпретации контекстных явлений: цель, лицо, действие, объект, возможность, место и время. Структурирование понятий внутри и между областями осуществляется по схеме 7S: *for **S**ome purpose, **S**omebody does **S**omething for **S**omeone, with **S**ome means, **S**ometimes and **S**omewhere* (с какой целью кто-то делает что-то для кого-то с помощью некоторых средств время от времени и в некотором месте).

Что представляется важным в подходе финской школы? Во-первых, это спецификация онтологий на основе теории контекстов. Во-вторых, спецификация отдельных контекстных областей с их «внутренними» отношениями, а также введение в рассмотрение «внешних» связей между объектами разных контекстных областей. В-третьих, подход может быть использован для интеграции разноуровневых по семантике и целям проектирования онтологий.

Из приведённого обзора можно видеть, что в настоящее время общепризнанной классификации онтологических моделей не существует. Имеющиеся классификации являются либо слишком общими, чтобы использоваться непосредственно в качестве основы для построения системы онтологических моделей процессов проектирования ПО, либо настолько частными, что их использование в данной ПрО представляется затруднительным.

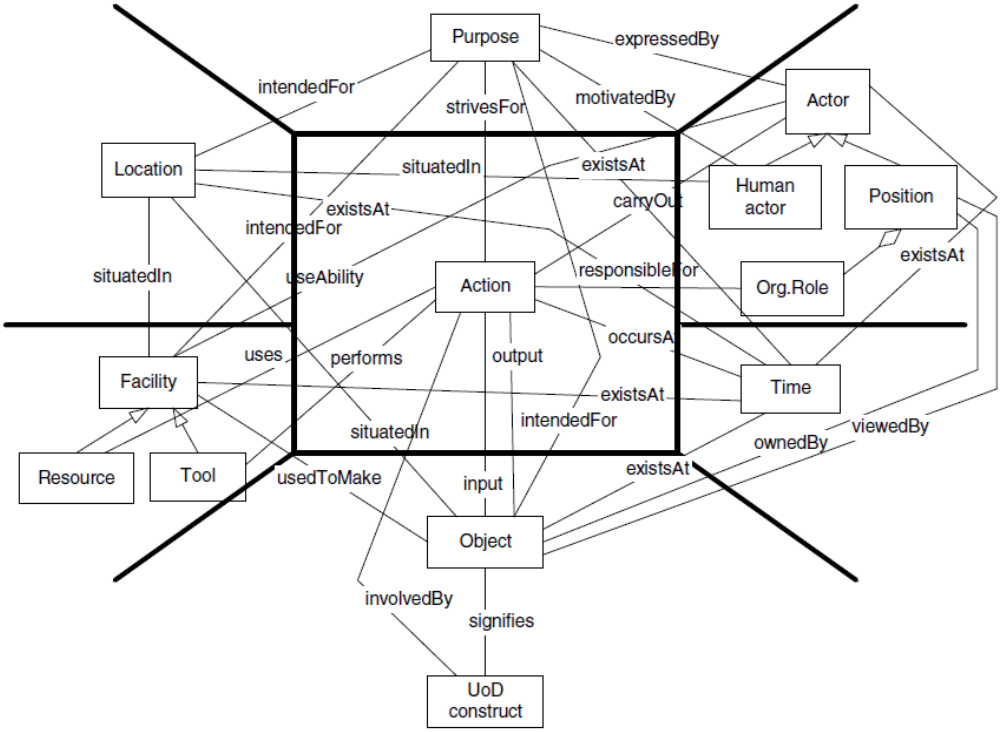

Рисунок 2 – Структура системы онтологий, основанной на контекстах [45]

Практически значимая онтологическая модель процессов проектирования систем ПО должна отвечать следующим требованиям:

- адекватное онтологическое представление проектируемых систем ПО, включая их архитектуру, спецификацию отдельных компонент и связей между ними;
- эффективная генерация исходных кодов разрабатываемого ПО под управлением системы сбалансированных онтологий;
- правильное отражение процессов проектирования ПО.

Структурно такая система онтологий должна быть многоуровневой. И здесь, по нашему мнению, может быть использован подход 7S М. Леппанена [45].

2.2 Семантические модели прикладных систем ПО

Наряду с исследованиями и разработками по онтологическому моделированию процессов проектирования ПО в рамках направления OBSE-ODSE активно ведутся работы по созданию инструментов онтологического инжиниринга в данной ПрО, см. например, работы [32, 34, 37-39]. При этом характерна ориентация на комплексное решение существующих в данной области проблем.

Хорошим примером является европейский проект MOST (*Marrying Ontology and Software Technology*) [46], ориентированный на повышение качества разработок ПО на основе использования семантических технологий и методов логического вывода. В рамках проекта реализована бесшовная интеграция семантических технологий в разработку ПО на основе моделей (MDSD - *Model Driven Software Development*), что привело к созданию методов и средств разработки ПО на основе онтологий (ODSD - *Ontology-Driven Software Development*). Интеграция касается, прежде всего, языков моделирования, что обеспечивает возможности преобразования интегрированных моделей с использованием унифицированных метамоделей. Онтологии и модели интегрируются таким образом, чтобы онтологии можно было использовать в моделях и наоборот. Запросы на основе онтологий интегрированы на уровне преобра-

зований MDA (*Model Driven Architecture*) и помогают разработчику выполнять требования и принимать решения по моделированию. Программный комплекс, основанный на онтологиях и управляемый моделями MOST, обеспечивает управление процессами разработки, которое формализует правила, условия и действия инженеров-программистов. Таким образом, проект MOST покрывает и фундаментальные исследования, и прикладные разработки.

В ряде публикаций предлагаются определённые методы онтологического моделирования процессов проектирования ПО и инструменты, разработанные на его основе.

Так, в работе [32] предложено повторное использование ПО на основе онтологического подхода. Представлена база знаний KOntoR, в которой онтологии повышают ценность библиотек для повторного использования за счёт семантической интеграции явных и неявных метаданных, что обеспечивает средства для получения новых фактов. Фрагмент онтологии KOntoR приведён на рисунке 3. Авторы приводят примеры, которые показывают, как пользователи библиотеки ПО могут извлечь выгоду из формализованных знаний, например, о лицензиях на ПО или технологиях программирования.

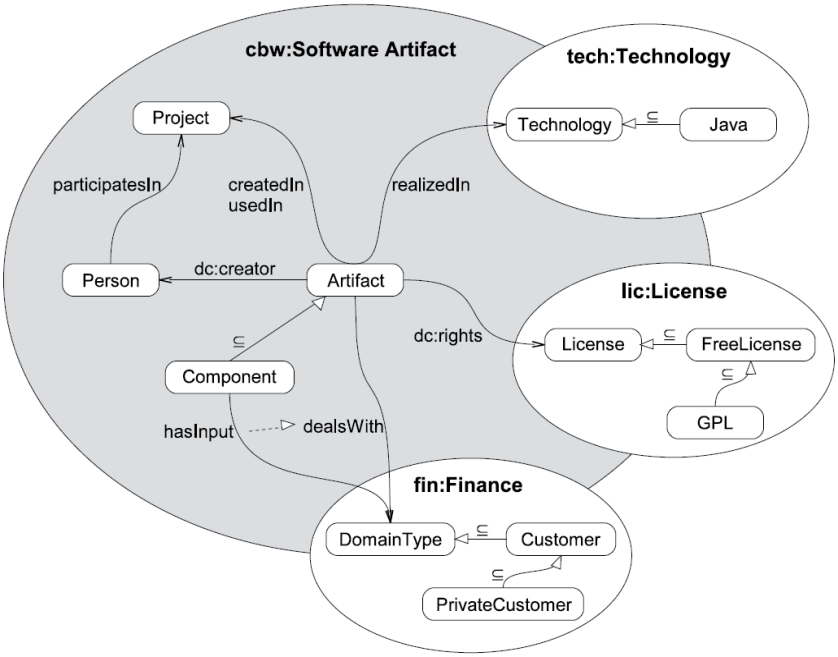

Рисунок 3 – Фрагмент онтологии KOntoR [32]

В работе [34] акцент сделан на использовании *Semantic Web* в качестве среды для представления знаний об эволюции ПО с помощью онтологий, технологий связанных данных (*Linked Data*) и рассуждений. Авторы разработали пирамиду онтологий для формального описания эволюции ПО (SEON), где специфицируются участники процесса разработки, их деятельность, артефакты, которые они создают, и отношения между ними (рисунок 4).

Реализация инструментария поддержки SEON выполнена в виде веб-сервисов с интерфейсом запросов на естественном языке и визуализацией результатов. Представлены три приложения, которые работают с SEON в качестве семантической основы. Первое – платформа веб-сервисов анализа эволюции ПО (рисунок 5); второе – ЕЯ-интерфейс для ответов на вопросы по пониманию программ (*Hawkshaw*); третье приложение представляет собой рекомендующую систему *Semantic Visualization Broker* (SVB) для анализа семантики заданного набора данных и составления списка визуализаций, которые могут быть полезны для более глубокого понимания анализируемой системы ПО.

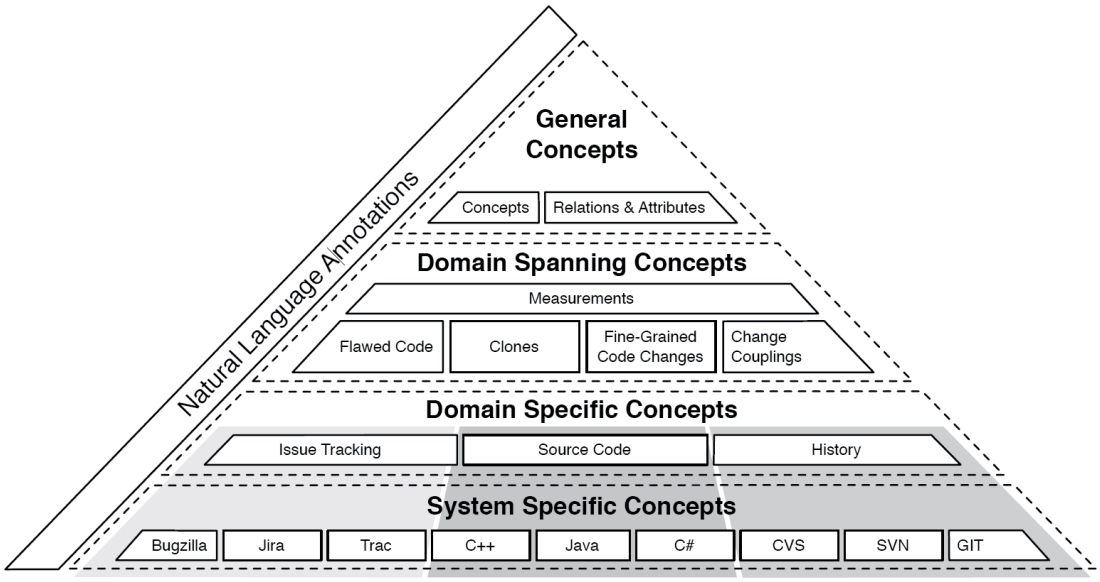

Рисунок 4 – Пирамида онтологий эволюции ПО [34]

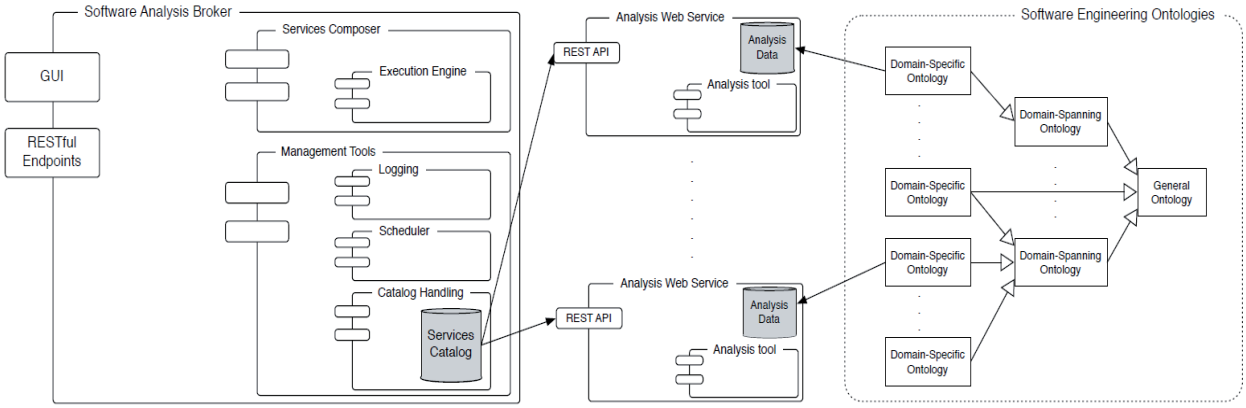

Рисунок 5 – Платформа Веб-сервисов анализа эволюции ПО [34]

В работе [37] представлен инструментарий помощи интеллектуальным агентам в решении задач разработки и настройки ПО. Предложенный метод – это онтологический подход, помогающий в принятии решений и обмене информацией посредством использования онтологии ODYSSEY для спецификации ЖЦ разработки ПО. Онтология ODYSSEY разработана в соответствии с методологией *Enterprise Ontology* и закодирована в формализме *Descriptive Logic* (DL). Здесь интересна попытка формального описания классических моделей ЖЦ разработки ПО. Например, DL-спецификация верхнего уровня водопадной модели ЖЦ разработки ПО определяется как

$$\begin{aligned}\text{Waterfall_Model} \subseteq \text{Model} \ & \wedge \exists\ \text{hasPhase}=\{\text{P1=System_Requirement}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P2=Software_Requirement}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P3=Analysis}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P4=Program Design}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P5=Coding}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P6=Testing}\}\\ & \wedge \exists\ \text{hasPhase}=\{\text{P7=Operations}\},\end{aligned}$$

а представление этой спецификации в системе Protégé показано на рисунке 6.

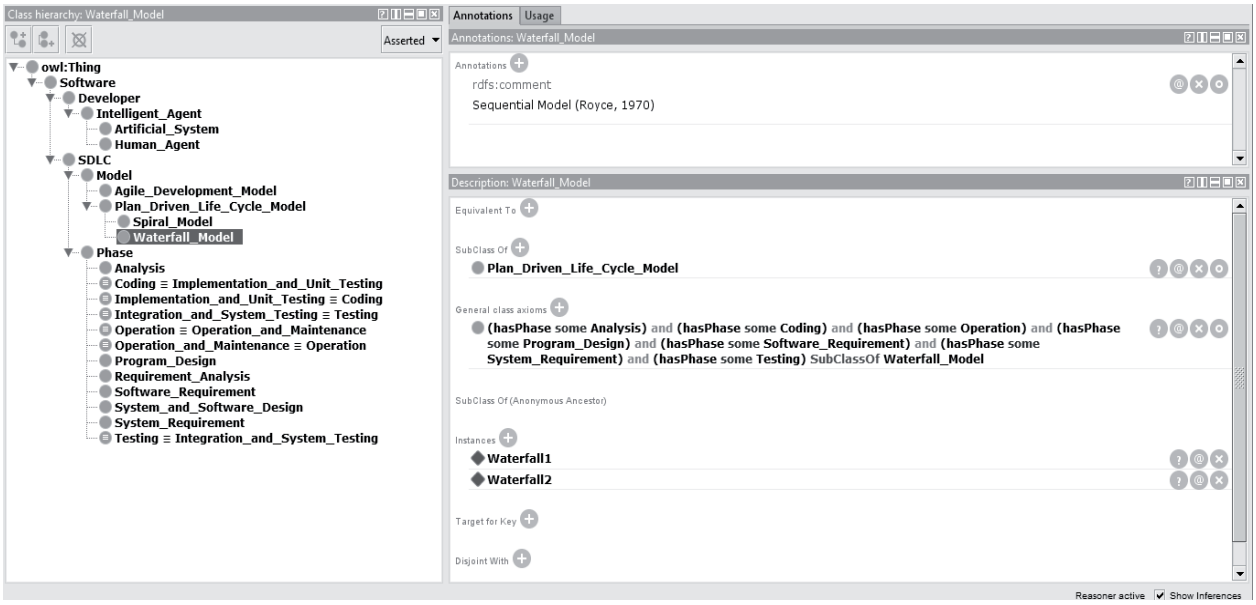

Рисунок 6 – Фрагмент онтологии водопадной модели ЖЦ [37]

В исследовании [38] авторами предлагается DKDOnto – онтология ПрО для распределённых программных проектов. Цель создания этой онтологии - поддержка проектов путём определения общего словаря для рассредоточенных команд разработчиков. Пользователи DKDOnto могут консультироваться с базой знаний, которая возвращает по их запросам лучшие практики, найденные в той же конфигурации проекта. Полученные ответы могут использоваться другими командами, вовлечёнными в тот же проект, или другими командами из других проектов. Фрагмент онтологии DKDOnto представлен на рисунке 7.

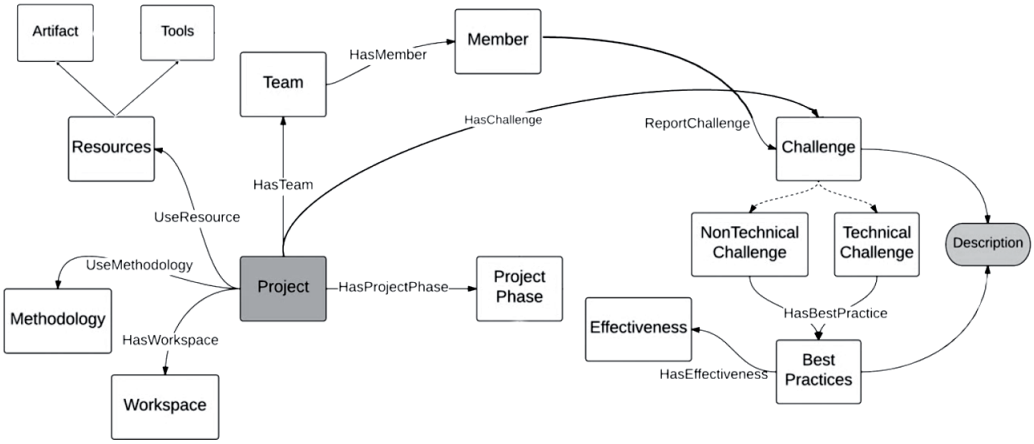

Рисунок 7 – Фрагмент онтологии DKDOnto [38]

В работе [39] отмечается, что онтологии в настоящее время являются частью большого числа информационных систем. В результате команды разработчиков ПО, которым приходится объединять деятельность по разработке онтологий с практиками разработки ПО, сталкиваются с рядом проблем, поскольку эти две области развивались, как правило, раздельно. Подход авторов *OnToology* состоит в том, чтобы обеспечить:

- автоматизацию крупномасштабных вспомогательных действий, связанных с разработкой онтологий, включая документирование, управление версиями, оценку и публикацию онтологий, которые поддерживаются и управляются версиями в известной среди разработчиков ПО среде распределённой системе управления версиями *Git*;

- рабочее место для поддержки непрерывной интеграции, когда регистрируются новые изменения в онтологии;
- интеграцию всех работ с *Git*.

Архитектура программного комплекса *OnToology* представлена на рисунке 8.

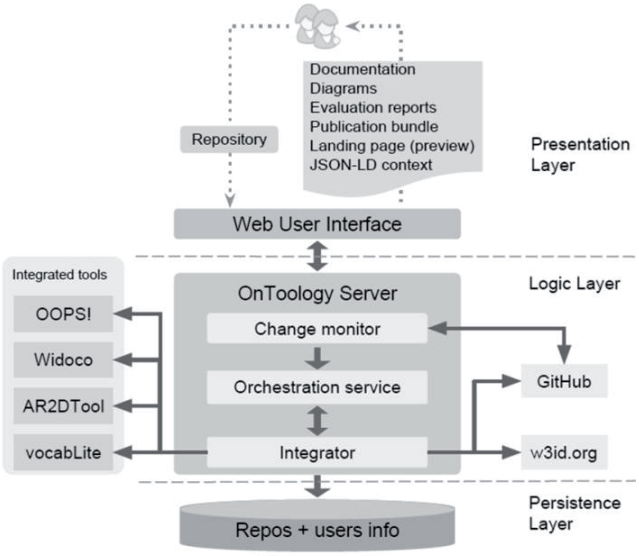

Рисунок 8 – Архитектура программного комплекса OnToology [39]

При этом *OnToology* может поддерживать как отдельных проектировщиков, так и команды разработчиков в рамках ЖЦ *Git*. *OnToology* – это веб-система, которая объединяет набор существующих инструментов для документации, оценки и публикации. Таким образом, инструментарий *OnToology* претендует на внедрение средств онтологического инжиниринга в «классические» инструменты совместного проектирования ПО и конкурирует с *WebProtege* [47, 48]. Интересным в данном решении является то, что в инструментарии *OnToology* поддерживается преобразование онтологий, спроектированных с использованием OWL, в JSON-представление. Важно и то, что данный инструментарий интегрирован с привычной для разработчиков ПО средой совместного проектирования *Git*.

Вопросы создания ПО прикладных систем на базе онтологических моделей рассматриваются в работах [49-53], выполненных в нашей стране. Так, в работе [49] обсуждаются общие принципы использования онтологий при разработке программных систем и их сопровождении, даётся классификация онтологий, используемых при проектировании, проведён анализ способов их применения.

В работе [51] предлагается подход к построению ПО для проектирования теплоснабжающих систем (ТСС) на основе концепции модельно-управляемой разработки. Используются технологии метапрограммирования и знания о ПрО в виде онтологий. Приведено описание программного комплекса СОСНА, разработанного на базе изложенного подхода. Программная система автоматически строится на основе следующих компонент: компьютерной модели конкретной ТСС; предварительно разработанных моделей элементов ТСС; программных компонент, реализующих методы и алгоритмы решения прикладных задач; онтологической системы предметных знаний (рисунок 9).

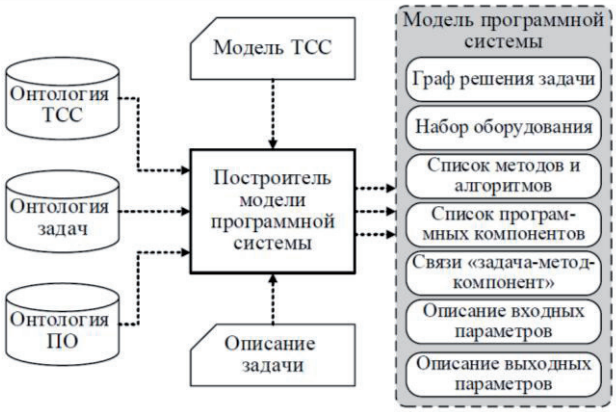

Рисунок 9 – Схема построения модели программной системы [51]

Наиболее интересным здесь является то, что алгоритм верхнего уровня, определяющий схему взаимодействия программных компонент и ход вычислительного процесса, строится автоматически во время построения программной системы на основе описания

прикладной задачи и методики её решения, а преобразование математических и графических моделей элементов ТСС в программный код осуществляется на основе XSLT-правил (*eXtensible Stylesheet Language Transformations* — язык преобразования XML-документов).

В работе [52] обсуждаются вопросы использования логической модели представления знаний для обеспечения системы требований к ПО. В рамках модели определяются различные типы требований и устанавливаются иерархические и ассоциативные отношения между ними. На основе критического анализа существующих моделей требований строится онтология типов требований, определяющая отношения между типами требований (классами онтологии), с использованием теории полевой структуры частей речи. Здесь представлена одна из немногих попыток построения онтологии проектирования ПО на основе лингвостатистического анализа текстов требований к разрабатываемому ПО. Для решения проблемы обеспечения выполнения свойств спецификации предлагается гибридная модель знаний, включающая фреймовую онтологию, содержащую иерархическое описание типов (классов) требований и ассоциативных связей между ними (рисунок 10) и систему правил, основанную на продукционной (логической) модели представления знаний.

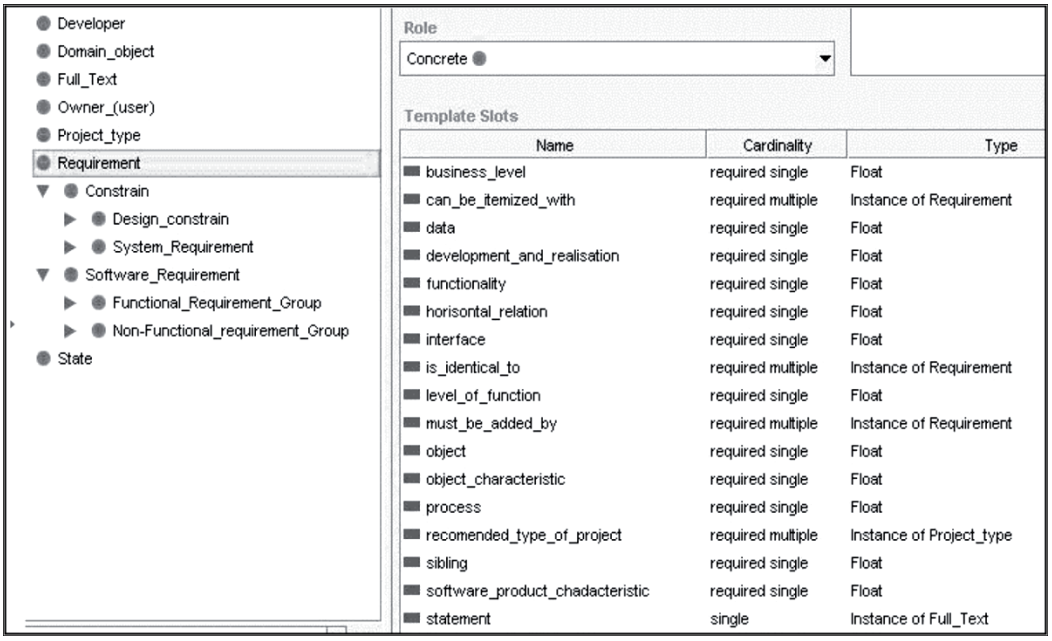

Рисунок 10 – Фрагмент онтологии требований в инструментарии Protégé [52]

Работа [53] посвящена обсуждению вопросов повторного использования знаний и организации системы поддержки принятия решений при управлении взаимодействующими бизнес-процессами в проектно-ориентированных организациях. Интересным здесь представляется разработка онтологии прецедентов проблемных ситуаций (рисунок 11) и онтологической модели управления взаимодействующими процессами в ИТ-проектах.

Нетрадиционный подход к разработке методов и средств семантического моделирования и проектирования представлен в работе [54]. Здесь каждая единица gB-системы (*good Base*) представляет собой стандартную Excel-книгу, а каждая родовидовая разновидность онтологических моделей имеет свой Excel-шаблон. При этом ядро gB-системы образует объектно-ориентированная СУБД на таблично-реляционном остове Excel, а программный интерфейс СУБД является составной частью онтологии решателей задач.

Вопросам онтологического моделирования процессов создания ПО и прикладных исследований в данной области посвящены работы [55-57], выполненные на Украине.

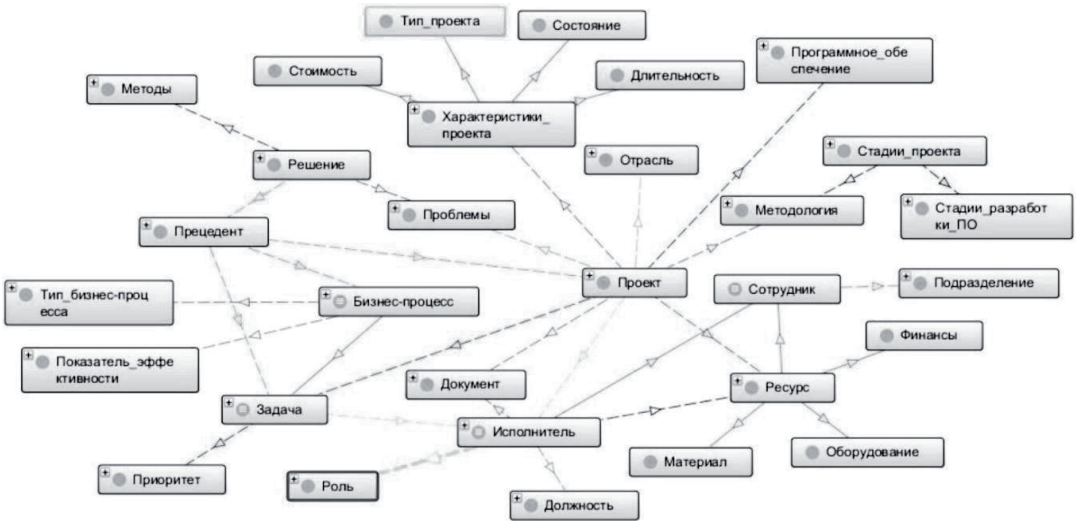

Рисунок 11 – Фрагмент онтологии прецедентов проблемных ситуаций [53]

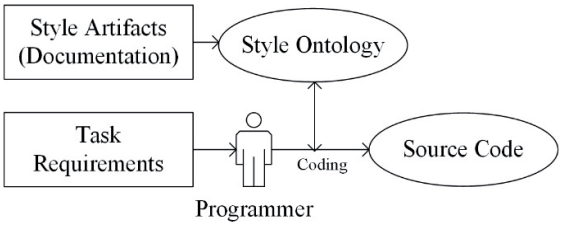

Рисунок 12 – Система онтологических моделей стилей программирования [56]

Так, например, в работе [56] представлен онтологический подход к моделированию процессов использования стилей программирования (рисунок 12). В частности, рассматривается представление стандартов на стили программирования в форме онтологии и применение решателя на основе дескриптивной логики в качестве ассистента программиста. По сути, здесь предлагается собственно онтология стилей программирования (рисунок 13) и инструмент для поддержки управляемого этой онтологией применения стилей, реализованный на языке Java (рисунок 14).

В работе [57] ставится и решается практически важная задача семантического анализа спецификаций (на примере требований к медицинскому ПО). Обсуждается реализация интеллектуального агента (ИА), оценивающего входную информацию для формирования таких спецификаций. В разработанной системе использована онтология требований для медицинской информационно-аналитической системы учёта терапевтических и диагностических мероприятий, предоставляемых раненым во время перевозки. Экспериментально показано, что именно ИА обеспечивает повышение уровня достаточности входной информации для определения качества ПО на 24%.

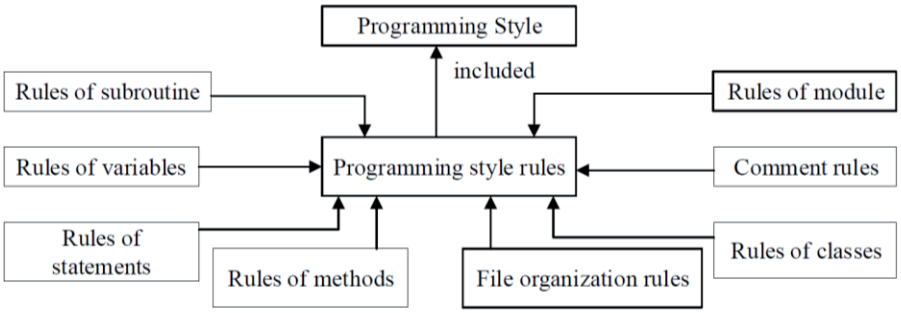

Рисунок 13 – Концептуализация онтологии стилей программирования [56]

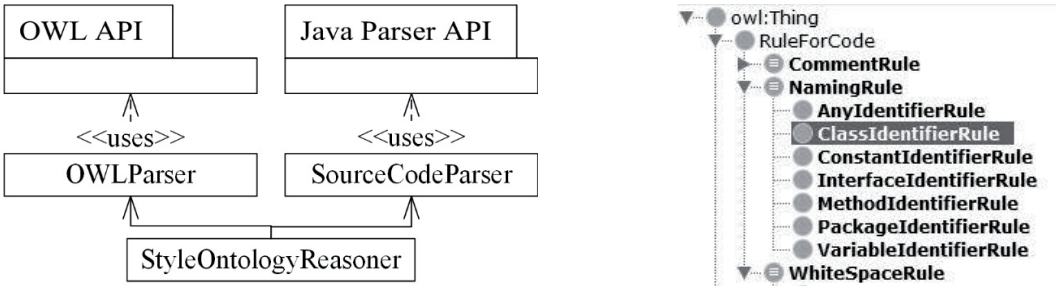

Рисунок 14 – Схема машины вывода и фрагмент онтологии стилей программирования [56]

В заключение автор хотел бы отметить, что основным направлением его собственных дальнейших исследований и разработок является применение методов и средств OBSE-ODSE для автоматизации проектирования систем извлечения информации из ЕЯ-текстов. Теоретические аспекты работ в этом направлении представлены в [58], а первые результаты – в [59].

Заключение

В работе представлен аналитический обзор подходов, моделей, методов проектирования ПО прикладных систем под управлением онтологий.

На основе проведённого анализа можно сформулировать следующие утверждения:

- современная постановка задачи автоматизации проектирования систем ПО предполагает наличие адекватных моделей всего ЖЦ, включая технологии разработки и реализации, модели архитектуры систем и их программных компонент, а также модели генерации ПО и модели ПрО, в которых предполагается функционирование соответствующих систем;
- общим трендом настоящего времени является создание и использование методов и средств онтологического моделирования как самих процессов проектирования, так и спецификаций разрабатываемых систем;
- основное внимание в современных исследованиях уделяется созданию онтологических моделей процессов проектирования ПО.

Можно отметить следующие особенности и перспективы развития данного научно-технического направления.

- Теоретические основы онтологического моделирования процессов проектирования ПО находятся на том уровне, когда центр тяжести исследований и разработок будет смещаться от моделей сборки ПО прикладных систем из крупных функциональных блоков в модели генерации самих блоков из согласованной системы онтологических паттернов их внутренних спецификаций.
- Тренд в области OBSE-ODSE связан с использованием методов и средств машинного обучения для генерации онтологических моделей проектирования ПО. Для выхода этого тренда на плато продуктивности потребуется ещё много фундаментальных исследований.
- Перспективными кандидатами на использование методов и средств проектирования ПО под управлением онтологий являются прикладные интеллектуальные системы, где основными компонентами можно считать базы знаний и машины вывода, которые, в свою очередь, могут строиться на основе онтологических моделей.

Список источников

[1] ***Липаев, В.В.*** Проектирование и производство сложных заказных программных продуктов / В.В. Липаев. – М.: СИНТЕГ, 2011. – 408 с.

[2] ***Bhuvaneswari, T.*** A Survey on Software Development Life Cycle Models / T. Bhuvaneswari, S. Prabaharan // International Journal of Computer Science and Mobile Computing. - 2013. - Vol. 2, Issue 5. - P.262-267.

[3] ***Half, R.*** Basic SDLC Methodologies: Which One is Best? / R. Half. – https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best.

[4] ***Winston, W.R.*** Managing the Development of Large Software Systems / W.R Winston // Proceedings, IEEE WESCON. - 1970. - P.1-9.

[5] ***Boehm, B.*** A Spiral Model of Software Development and Enhancement / B. Boehm // ACM SIGSOFT Software Engineering Notes. - 1986. - 11(4). - P.14-24.

[6] ***Beck, K.*** Extreme Programming Explained: Embrace Change / K. Beck - Addison-Wesley. 1999. – 237 p.

[7] ***Beck, K.*** Manifesto for Agile Software Development / K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas // Agile Alliance. - 2001. – https://agilemanifesto.org/.

[8] ***Jacobson, S.*** The Rational Objectory Process - A UML-based Software Engineering Process / S. Jacobson // Rational Software Scandinavia AB, 2014. – 17 p.

[9] ***Henderson-Sellers, B.*** An ontology for ISO software engineering standards: 1) Creating the infrastructure / B. Henderson-Sellers, C. Gonzalez-Perez, T. McBride, G. Low. // Computer Standards & Interfaces. – 2014. - 31, 36(3). - P. 563-576.

[10] ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes. – https://www.iso.org/standard/63712.html.

[11] ***Asikainen, T.*** A domain ontology for modelling variability in software product families / T. Asikainen, T. Männistö, T. Soininen // Adv. Eng. Inform. - 2007. – 21. - P.23-40.

[12] ***Strmečki, D.*** An Overview on the use of Ontologies in Software Engineering / D. Strmečki, I. Magdalenić, D. Kermek // Journal of Computer Science. – 2016. - 12 (12). - P.597-610.

[13] ***Ringert, J.O.*** Multiplatform generative development of component and connector systems using model and code libraries / J.O. Ringert, B. Rumpe, A. Wortmann // Proc. of the 1st International Workshop on Model-Driven Engineering for Component-Based Systems, CBS' 14 (Valencia, Spain, 2014). - P.26-35.

[14] ***Magdalenić, I.*** Autogenerator: Generation and execution of programming code on demand / I. Magdalenić, D. Radošević, T. Orehovački // Expert Syst. Applic. - 2013. – 40. - P.2845-2857.

[15] ***Lilis, Y.*** Staged model-driven generators: Shifting responsibility for code emission to embedded metaprograms / Y. Lilis, A. Savidis, Y. Valsamakis // Proc. of the 2nd International Conference on Model-Driven Engineering and Software Development (Jan. 7-9, 2014). - IEEE Xplore Press, 2014. - P. 509-521.

[16] ***Rodrigues da Silva, A.*** Model-driven engineering: A survey supported by the unified conceptual model / A. Rodrigues da Silva // Comput. Lang. Syst. Struct. - 2015. – 43. - P. 139-155.

[17] ***Hesse, W.*** Ontologies in the Software Engineering process / W. Hesse // Proc. of the 2nd GI-Workshop on Enterprise Application Integration, EAI-05 (Marburg, Germany, June 30 / July 1st, 2005). – https://pdfs.semanticscholar.org/7812/c8ac3e35a78d35c1df93f09108c69884e7f1.pdf.

[18] ***Dillon, T.S.*** Ontology-based Software Engineering - Software Engineering 2.0 / T.S. Dillon, E. Chang, P. Wongthongtham // Proc. 19th Australian Conference on Software Engineering. - 2008. - P.13-23.

[19] ***Henderson-Sellers, B.*** Bridging metamodels and ontologies in software engineering / B. Henderson-Sellers // Journal of Systems and Software. – 2011. - 84(2). - P.301-313.

[20] ***Isotani, S.*** Ontology Driven Software Engineering: A Review of Challenges and Opportunities / S. Isotani, I.I. Bittencourt, E.F. Barbosa, D. Dermeval, R.O.A. Paiva // IEEE Latin America Transactions. – 2015. - Vol. 13, No. 3. - P.863-869.

[21] ***Bhatia, M.P.S.*** Ontologies for Software Engineering: Past, Present and Future / M.P.S. Bhatia, A. Kumar, R. Beniwal // Indian Journal of Science and Technology. – 2016. - 9(9). - P.8-17.

[22] ***Боргест, Н.М.*** Границы онтологии проектирования / Н.М. Боргест // Онтологии проектирования. – 2017. – Т. 7, №1(23). – С.7-33. – DOI: 10.18287/2223-9537-2017-7-1-7-33.

[23] ***Fadel, F.G.*** A generic enterprise resource ontology / F.G. Fadel, M.S. Fox, M. Gruninger // Proc. of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'94 (April 1994, Morgantown, West Virginia), 1994. - P.117–128.

[24] ***Uschold, M.*** The Enterprise Ontology / M. Uschold, M. King, S. Moralee, Y. Zorgios // The Knowledge Engineer Review. - 1998. - 13(1). - P.31-89.

[25] ***Mendes, O.*** Software engineering ontology: A development methodology / O. Mendes, A. Abran // Metrics News. - 2005. – 9. - P.68–76.
[26] ***Happel, H.*** Applications of ontologies in software engineering / H. Happel, S. Seedorf // Proc. of the 2nd International Workshop on Semantic Web Enabled Software Engineering, ESE' 06. - 2006. - P.1-14.
[27] ***Bossche, M.V.*** Ontology Driven Software Engineering for Real Life Applications / M.V. Bossche, P. Ross, I. MacLarty, B. van Nuffelen, N. Pelov // Proc. 3rd International Workshop on Semantic Web Enabled Software Engineering, SWESE 2007 (Innsbruck, Austria, 2007). – https://www.odaseontologies.com/wp-content/uploads/2016/03/Ontology-Driven-Software-Engineering-for-Real-Life-Applications-OK.pdf.
[28] ***Nianfang, X.*** Software components description based on ontology / X. Nianfang, Y. Xiaohui, L. Xinke // Proc. of the 2nd International Conference on Computer Modeling and Simulation (Jan. 22-24, 2010). - IEEE XPlore Press; 2010. - P.423-426.
[29] ***Liao, L***. Software Process Ontology and Its Application / L. Liao, Y. Qu, H.K.N. Leung // Studies on the Semantic Web. Vol. 17: Semantic Web Enabled Software Engineering. - IOS Press, 2014. - P.207-217.
[30] ***Pileggi, S.F.*** Ontology in Software Engineering / S.F. Pileggi, A. Lopez-Lorca, G. Beydoun // Proc. Australasian Conference on Information Systems 2018 (Sydney, Australia, 2018). – http://www.acis2018.org/wp-content/uploads/2018/11/ACIS2018_paper_200.pdf.
[31] ***Jabar, M.A.*** General Domain Ontology in Enterprise Software Development Process / M.A. Jabar, M.S. Khalefa // International Journal of Engineering and Advanced Technology. – 2019. - Vol. 8, Is. 3S. - P.398-402.
[32] ***Happel, H.J.*** KOntoR: An Ontology-enabled Approach to Software Reuse / H.J. Happel, A. Korthaus, S. Seedorf, P. Tomczyk // Proc. SEKE 2006: the 18th International Conference on Software Engineering & Knowledge Engineering (July 5-7, 2006, California, USA, 2006). - P.349-354.
[33] ***Chaves, A.P.*** OntoDiSENv1: an Ontology to Support Global Software Development / A.P. Chaves, I. Steinmacher // CLEIej 2011. - Vol.14, No.2. – http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002011000200002.
[34] ***Würsch, M.*** SEON: A pyramid of ontologies for software evolution and its applications / M. Würsch, G. Ghezzi, M. Hert, G. Reif, H. Gall // Computing. - 2012. - 94(11). - P.857-885.
[35] ***Aßmann, U.*** Ontology-Guided Software Engineering in the MOST Workbench / U. Aßmann, S. Zivkovic, K. Miksa, K. Siegemund, A. Bartho, T. Rahmani, E. Thomas, J.Z. Pan // In: Ontology-Driven Software Development. - Springer Berlin Heidelberg, 2013. - P.293-318.
[36] ***Bhatia, M.P.S.*** An ontology based framework for automatic detection and updation of requirement specifications / M.P.S. Bhatia, R. Beniwal, A. Kumar // International Conference on Contemporary Computing and Informatics, IC3I (2014 Nov 27). – IEEE, 2014 - P.238-242.
[37] ***Olszewska, J.I.*** ODYSSEY: Software Development Life Cycle Ontology / J.I. Olszewska, I.K. Allison // Proc. International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (Seville, Spain 18 Sep 2018-20 Sep 2018). - P.303-311.
[38] ***Rochaa, R.*** DKDOnto: An Ontology to Support Software Development with Distributed Teams / R. Rochaa, A. Ara´ujoa, D. Cordeiroa, A. Ximenesa, J. Teixeiraa, G. Silvaa, D. da Silvaa, D. Espinharaa, R. Fernandesa, J. Ambrosiob, M. Duartec, R. Azevedo // Proc. 22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. - Procedia Computer Science 126, 2018. - P.373-382.
[39] ***Alobaid, A.*** Automating ontology engineering support activities with OnToology / A. Alobaid, D. Garijo, M. Poveda-Villalón, I. Santana-Perez, A. Fernandez-Izquierdo, O. Corcho // Journal of Web Semantics 57, 100472, 2019. – https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3260516.
[40] ***Alsanad, A.A.*** Domain Ontology for Software Requirements Change Management in Global Software Development Environment / A.A. Alsanad, A. Chikh, A Mirzaa // IEEE Access. - 2019. - Vol. 7. - P.349-353.
[41] ***Calero, C.*** Ontologies for Software Engineering and Software Technology. 1st Ed. / C. Calero, F. Ruiz, M. Piattini. - Springer, 2006. - 340 p.
[42] ***Anquetil, N.*** Software Maintenance seen as a Knowledge Management Issue / N. Anquetil, K.M. de Oliveira, D. Kleiber, M.G. de Sousa // Inf. Softw. Technol. - 2007. - 49(5). - P.515-529.
[43] ***Fairley, R.E.*** A Post-Mortem Analysis of the Software Engineering Programs at Wang Institute of Graduate Studies / R.E. Fairley // Issues in Software Engineering Education. Eds.: Fairley R., Freeman P. -Springer, New York, 1989. – 97 p.
[44] ISO/IEC 14764:2006 Software Engineering - Software Life Cycle Processes – Maintenance. – https://www.iso.org/standard/39064.html.
[45] ***Leppanen, M.*** A context-based enterprise ontology / M. Leppanen // Proc. of the 10th international conference on Business information systems, BIS'07. - Springer-Verlag Berlin, Heidelberg, 2007. - P. 273–286.
[46] Project MOST Marrying Ontology and Software Technology. 2011. – https://cordis.europa.eu/project/rcn/85351/factsheet/en.

[47] ***Tudorache, T.*** WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web / T. Tudorache, C. Nyulas, N.F.Noy, M.A. Musen // Semantic Web Journal. – 2013. - Vol. 4, No 1. - P.89-99.
[48] ***Musen, M.A.*** The protégé project: a look back and a look forward / M.A. Musen // Newsletter AI Matters. – 2015. - Vol. 1, Issue 4. - P.4-12.
[49] ***Шалфеева, Е.А.*** Возможности использования онтологий при разработке и сопровождении программных систем / Е.А. Шалфеева // Владивосток: ИАПУ ДВО РАН, 2011. - 16 с.
[50] ***Лаврищева, Е.М.*** Подход к формальному представлению онтологии жизненного цикла программных систем / Е.М. Лаврищева // Весник КГУ. Серия физ.-мат. наук. - 2013. - № 4. - С.140-149.
[51] ***Стенников, В.А.*** Применение онтологий при реализации концепции модельно-управляемой разработки программного обеспечения для проектирования теплоснабжающих систем / В.А. Стенников, Е.А. Барахтенко, Д.В. Соколов // Онтология проектирования. – 2014. - 4(14). - С.54-68.
[52] ***Пустовалова, Н.В.*** Построение согласованной модели требований для процесса программной инженерии / Н.В. Пустовалова, Т.В. Авдеенко // Труды СПИИРАН. – 2016. - Выпуск 44. - С.31-49.
[53] ***Бармина, О.В.*** Интеллектуальная система управления взаимодействием бизнес-процессов в проектно-ориентированных организациях / О.В. Бармина, Н.О. Никулина // Онтология проектирования. – 2017. – Т. 7, №1(23). - С.48-65. – DOI: 10.18287/2223-9537-2017-7-1-48-65.
[54] ***Смирнов, С.В.*** Опыт создания средств семантического моделирования и проектирования на массовой программной платформе // Open Semantic Technologies for Intelligent Systems, OSTIS-2015: Материалы V международ. науч.-тех. конф. (19-21 февраля 2015 г. Минск, Беларусь) / Отв. ред.: В.В. Голенков. – Минск: БГУИР, 2015. – С.413-416.
[55] ***Тарасов, А.Ф***. Онтологический подход к построению логических моделей программных систем / А.Ф. Тарасов, О.А. Лябик // Вісник Донбаської державної машинобудівної академії. - 2006. - №1Е(6). - С.50-54.
[56] ***Sidorov, N.*** Ontology driven tool for utilizing programming styles / N. Sidorov, N. Sidorova, A. Pirog // Proc/ of the National Aviation University. - 2017. - 2(71). - P.84-92.
[57] ***Hovorushchenko, T.*** Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle / T. Hovorushchenko, O. Pavlova // Advances in Intelligent Systems and Computing. - 2019. - Vol. 836. - P.169-178.
[58] ***Efimenko, I.V.*** Identification of Promising High-Tech Solutions in Big Text Data with Semantic Technologies: Energy, Pharma, and Many Others (Chapter 16. Advanced Methods) / I.V. Efimenko, V.F. Khoroshevsky // In: Innovation Discovery. Network Analysis of Research and Invention Activity for Technology Management. / Eds: T. Daim, A. Pilkington. 443 p. - 2018. – P.429–467.
[59] ***Хорошевский, В.Ф.*** Генерация лингвистических процессоров для платформы GATE под управлением онтологий / В.Ф. Хорошевский, А.С. Булгаков, А.В. Демин // В сб.: Шестнадцатая Национальная конференция по искусственному интеллекту с международным участием КИИ-2018. / Труды конференции: в 2-х томах. 2018. - С.288-296.

ONTOLOGY DRIVEN SOFTWARE ENGINEERING: MODELS, METHODS, IMPLEMENTATIONS

V.F. Khoroshevsky

Dorodnicyn Computing Centre of Russian Academy of Sciences,
Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russia
V.Khor@mail.ru

Abstract

Ontology-driven software engineering is the scope of the paper. A retrospective analysis of the approaches, models and methods is carried out. In the first section, a brief overview of "classical" technologies, approaches and methods for software engineering is presented. It is shown that the emphasis of research and development is currently shifting from the field of technologies of programming towards the modeling of the software design processes, as well as towards the specifications and testing of the developed systems. It is outlined that the general trend in the domain is the use of methods and tools of ontology modeling of both the design processes per se and the specifications of the developed systems. The second section is devoted to a review of publications in the field of ontology-based software engineering.

Such paradigms for modeling software development processes as Software Product Lines, plug-and-play software development - Component-Based Development, model-driven software engineering - Model-Driven Engineering, etc. are discussed. It is shown that "programming through generation", i.e. Generative Programming, is a general approach supporting the considered paradigms. It is shown that R&D in the domain of the semantic models of software engineering is currently focused on a new approach - Ontology-Based (-Driven) Software Engineering. Within the framework of this area, ontology engineering tools are developed along with modeling the software development processes. The scientific and technical perspectives of ontology driven software engineering are specified and discussed in the final part of the paper. In particular, it is outlined that the theoretical foundations of ontology-based modeling of software design processes form the solid basis for shifting the focus from the models of compiling the modules to their pattern-based generation from internal specifications. It is noted that the next trend in the field of OBSE-ODSE will be the usage of machine learning methods and tools to generate ontological models of software design.

Key words: *software engineering, software product lines paradigm, component-based development, model-driven engineering, generative programming, ontology modeling, ontology-driven software engineering, system of ontologies of designing, ontology-driven applied system.*

Citation: *Khoroshevsky VF.* Ontology Driven Software Engineering: Models, Methods, Implementations [In Russian]. *Ontology of designing*. 2019; 9(4): 429-448. – DOI: 10.18287/2223-9537-2019-9-4-429-448.

References

[1] ***Lipaev VV.*** Complex Custom Software Products Design and Manufacture [In Russian]. – Moscow: SINTEG publ.; 2011.

[2] ***Bhuvaneswari T., Prabaharan S.*** A Survey on Software Development Life Cycle Models. International Journal of Computer Science and Mobile Computing 2013; 2(5): 262-267.

[3] ***Half R.*** Basic SDLC Methodologies: Which One is Best? – https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best.

[4] ***Winston WR.*** Managing the Development of Large Software Systems. Proceedings, IEEE WESCON 1970: 1-9.

[5] ***Boehm B.*** A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes 1986; 11(4): 14-24.

[6] ***Beck K.*** Extreme Programming Explained: Embrace Change. - Addison-Wesley; 1999.

[7] ***Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D.*** Manifesto for Agile Software Development. Agile Alliance 2001. – https://agilemanifesto.org/.

[8] ***Jacobson S.*** The Rational Objectory Process - A UML-based Software Engineering Process. Rational Software Scandinavia AB 2014.

[9] ***Henderson-Sellers B, Gonzalez-Perez C, McBride T, Low G.*** An ontology for ISO software engineering standards: 1) Creating the infrastructure. Computer Standards & Interfaces 2014; 31, 36(3): 563-576.

[10] ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes. – URL: https://www.iso.org/standard/63712.html (access date 04.10.2019).

[11] ***Asikainen T, Männistö T, Soininen T.*** A domain ontology for modelling variability in software product families. Adv. Eng. Inform. 2007; 21: 23-40.

[12] ***Strmečki D, Magdalenić I, Kermek D.*** An Overview on the use of Ontologies in Software Engineering. Journal of Computer Science 2016; 12(12): 597-610.

[13] ***Ringert JO, Rumpe B, Wortmann A.*** Multiplatform generative development of component and connector systems using model and code libraries. Proc. of the 1st International Workshop on Model-Driven Engineering for Component-Based Systems, CBS' 14 (Valencia, Spain, 2014): 26-35.

[14] ***Magdalenić I, Radošević D, Orehovački T.*** Autogenerator: Generation and execution of programming code on demand. Expert Syst. Applic. 2013; 40: 2845-2857.

[15] ***Lilis Y, Savidis A, Valsamakis Y.*** Staged model-driven generators: Shifting responsibility for code emission to embedded metaprograms. Proc. of the 2nd International Conference on Model-Driven Engineering and Software Development (Jan. 7-9, 2014). - IEEE Xplore Press, 2014: 509-521.

[16] ***Rodrigues da Silva A.*** Model-driven engineering: A survey supported by the unified conceptual model. Comput. Lang. Syst. Struct. 2015; 43: 139-155.

[17] ***Hesse W.*** Ontologies in the Software Engineering process. Proc. of the 2nd GI-Workshop on Enterprise Application Integration, EAI-0 (Marburg, Germany, June 30 / July 1st, 2005). – https://pdfs.semanticscholar.org/7812/c8ac3e35a78d35c1df93f09108c69884e7f1.pdf.

[18] ***Dillon TS, Chang E, Wongthongtham P.*** Ontology-based Software Engineering - Software Engineering 2.0. Proc. 19th Australian Conference on Software Engineering, 2008: 13-23.
[19] ***Henderson-Sellers B.*** Bridging metamodels and ontologies in software engineering. Journal of Systems and Software 2011; 84(2): 301-313.
[20] ***Isotani S, Bittencourt II, Barbosa EF, Dermeval D., Paiva ROA.*** Ontology Driven Software Engineering: A Review of Challenges and Opportunities. IEEE Latin America Transactions 2015; 13(3): 863-869.
[21] ***Bhatia MPS, Kumar A, Beniwal R.*** Ontologies for Software Engineering: Past, Present and Future. Indian Journal of Science and Technology 2016; 9(9): 8-17.
[22] ***Borgest NM.*** The Boundaries of Ontology of Designing [In Russian]. Ontology of Designing 2017; Vol. 7, 1(23): 7-33. – DOI: 10.18287/2223-9537-2017-7-1-7-33.
[23] ***Fadel FG, Fox MS, Gruninger M.*** A generic enterprise resource ontology. Proc. of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'94 (April 1994, Morgantown, West Virginia), 1994: 117–128.
[24] ***Uschold M, King M, Moralee S, Zorgios Y.*** The Enterprise Ontology. The Knowledge Engineer Review 1998; 13(1): 31-89.
[25] ***Mendes O, Abran A.*** Software engineering ontology: A development methodology. Metrics News 2005; 9: 68-76.
[26] ***Happel H, Seedorf S.*** Applications of ontologies in software engineering. Proc. of the 2nd International Workshop on Semantic Web Enabled Software Engineering, ESE' 06. 2006: 1-14.
[27] ***Bossche MV, Ross P, MacLarty I, van Nuffelen B, Pelov N.*** Ontology Driven Software Engineering for Real Life Applications. Proc. 3rd International Workshop on Semantic Web Enabled Software Engineering, SWESE 2007 (Innsbruck, Austria, 2007). – https://www.odaseontologies.com/wp-content/uploads/2016/03/Ontology-Driven-Software-Engineering-for-Real-Life-Applications-OK.pdf.
[28] ***Nianfang X, Xiaohui Y, Xinke L.*** Software components description based on ontology. Proc. of the 2nd International Conference on Computer Modeling and Simulation (Jan. 22-24, 2010). - IEEE XPlore Press; 2010: 423-426.
[29] ***Liao L, Qu Y, Leung HKN.*** Software Process Ontology and Its Application. Studies on the Semantic Web. Vol. 17: Semantic Web Enabled Software Engineering. - IOS Press; 2014: 207-217.
[30] ***Pileggi SF, Lopez-Lorca A, Beydoun G.*** Ontology in Software Engineering. Proc. Australasian Conference on Information Systems 2018 (Sydney, Australia, 2018). – http://www.acis2018.org/wp-content/uploads/2018/11/ACIS2018_paper_200.pdf.
[31] ***Jabar MA, Khalefa MS.*** General Domain Ontology in Enterprise Software Development Process. International Journal of Engineering and Advanced Technology 2019; Vol. 8, Is. 3S: 398-402.
[32] ***Happel HJ, Korthaus A, Seedorf S, Tomczyk P.*** KOntoR: An Ontology-enabled Approach to Software Reuse. Proc. SEKE 2006: the 18th International Conference on Software Engineering & Knowledge Engineering (July 5-7, 2006, California, USA, 2006): 349-354.
[33] ***Chaves AP, Steinmacher I.*** OntoDiSENv1: an Ontology to Support Global Software Development. CLEIej 2011; Vol. 14, No.2. – http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002011000200002.
[34] ***Würsch M, Ghezzi G, Hert M, Reif G, Gall H.*** SEON: A pyramid of ontologies for software evolution and its applications // Computing 2012; 94(11): 857-885.
[35] ***Aßmann U, Zivkovic S, Miksa K, Siegemund K, Bartho A, Rahmani T, Thomas E, Pan JZ.*** Ontology-Guided Software Engineering in the MOST Workbench // In: Ontology-Driven Software Development. - Springer Berlin Heidelberg; 2013: 293-318.
[36] ***Bhatia MPS, Beniwal R, Kumar A.*** An ontology based framework for automatic detection and updation of requirement specifications. International Conference on Contemporary Computing and Informatics, IC3I (2014 Nov 27). – IEEE; 2014: 238-242.
[37] ***Olszewska JI, Allison IK.*** ODYSSEY: Software Development Life Cycle Ontology. Proc. International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (Seville, Spain 18 Sep 2018-20 Sep 2018). - P.303-311.
[38] ***Rochaa R., Ara´ujoa A, Cordeiroa D, Ximenesa A, Teixeiraa J, Silvaa G, da Silvaa D, Espinharaa D, Fernandesa R, Ambrosiob J, Duartec M, Azevedo R.*** DKDOnto: An Ontology to Support Software Development with Distributed Teams. Proc. 22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. - Procedia Computer Science 126; 2018: 373-382.
[39] ***Alobaid A, Garijo D, Poveda-Villalón M, Santana-Perez I, Fernandez-Izquierdo A, Corcho O.*** Automating ontology engineering support activities with OnToology. Journal of Web Semantics 57, 100472; 2019. – URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3260516 (access date 01.10.2019).
[40] ***Alsanad AA, Chikh A, Mirzaa A.*** Domain Ontology for Software Requirements Change Management in Global Software Development Environment. IEEE Access 2019; Vol.7: 349-353.
[41] ***Calero C, Ruiz F, Piattini M.*** Ontologies for Software Engineering and Software Technology. 1st Ed. – Springer; 2006.

[42] ***Anquetil N, de Oliveira KM, Kleiber D, de Sousa MG.*** Software Maintenance seen as a Knowledge Management Issue. Inf. Softw. Technol. 2007; 49(5): 515-529.
[43] ***Fairley RE.*** A Post-Mortem Analysis of the Software Engineering Programs at Wang Institute of Graduate Studies. Issues in Software Engineering Education. Eds.: Fairley R., Freeman P. - Springer, New York, 1989.
[44] ISO/IEC 14764:2006 Software Engineering - Software Life Cycle Processes – Maintenance. – https://www.iso.org/standard/39064.html.
[45] ***Leppanen M.*** A context-based enterprise ontology // Proc. of the 10th international conference on Business information systems, BIS'07. Springer-Verlag Berlin, Heidelberg; 2007: 273–286.
[46] Project MOST. Marrying Ontology and Software Technology., 2011. - https://cordis.europa.eu/project/rcn/85351/factsheet/en.
[47] ***Tudorache T, Nyulas C, Noy NF, Musen MA.*** WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. Semantic Web Journal 2013; 4(1): 89-99.
[48] ***Musen MA.*** The protégé project: a look back and a look forward. Newsletter AI Matters 2015; 1(4): 4-12.
[49] ***Shalfeeva EA.*** Possibilities of Using Ontologies in the Development and Maintenance of Software Systems [In Russian]. Vladivostok, IAPU DVO RAS; 2011.
[50] ***Lavrishcheva EM.*** The Approach to the Formal Representation of the Ontology of the Software Systems Life Cycle [In Russian]. Vestnik KGU 2013; 4: 140-149.
[51] ***Stennikov VA.*** The use of ontologies in the implementation of the concept of model-driven development of software for the design of heat supply systems [In Russian]. *Ontology of Designing*. 2014; 4(14): 54-68.
[52] ***Pustovalova NV, Avdeenko TV.*** Building a Consistent Requirement Model for a Software Engineering Process [In Russian]. Proc. of SPII RAS 2016, 44: 31-49.
[53] ***Barmina OV, Nikulina NO.*** Intelligent system for interactive business processes management in project-oriented organizations [In Russian]. *Ontology of Designing*. 2017; 7(1): 48-65. – DOI: 10.18287/2223-9537-2017-7-1-48-65.
[54] ***Smirnov SV.*** The experience of semantic modeling and designing tools on widely used platform [In Russian]. Proc. of the V International Scientific Conf. on Open Semantic Technologies for Intelligent Systems OSTIS-2015 (February 19-21, 2015 Minsk, Belarus). Eds.: VV Golenkov and al. - Belarusian State University of Informatics and Radioelectronics, 2015: 413-416.
[55] ***Tarasov AF, Lyabik OA.*** Ontological approach to the construction of logical models of software systems [In Russian]. Vestnik Donbass Derjavnoi Mashinobudivnoi Academii. 2006; 1E(6): 50-54.
[56] ***Sidorov N, Sidorova N, Pirog A.*** Ontology driven tool for utilizing programming styles. Proc. of the National Aviation University 2017; 2(71): 84-92.
[57] ***Hovorushchenko T, Pavlova O.*** Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. Advances in Intelligent Systems and Computing 2019; 836: 169-178.
[58] ***Efimenko IV, Khoroshevsky VF.*** Identification of Promising High-Tech Solutions in Big Text Data with Semantic Technologies: Energy, Pharma, and Many Others (Chapter 16. Advanced Methods). In: Innovation Discovery. Network Analysis of Research and Invention Activity for Technology Management. Eds: T. Daim, A. Pilkington. 443 p. 2018: 429–467.
[59] ***Khoroshevsky VF, Bulgakov AS, Demin AV.*** Ontology-Driven Generation of Linguistic Processors for the GATE Platform [In Russian]. Proc. of 16th National Conference on Artificial Intelligence, CAI-2018. 2018: 288-296.

Сведения об авторе

Хорошевский Владимир Федорович, 1946 г. рождения. Окончил Московский инженерно-физический институт по специальности «Прикладная математика» в 1971 г., к.т.н. (1976), д.т.н. (1995). Профессор кафедры «Интеллектуальные системы» Московского физико-технического института. Один из основателей советской, затем Российской ассоциации искусственного интеллекта (РАИИ). Член Научного Совета РАИИ. Член редколлегий научных журналов «Искусственный интеллект и принятие решений», «Программные продукты и системы». В списке научных трудов более 130 публикаций в области ИИ (5 монографий, учебных пособий и учебников для ВУЗов).

Vladimir Fedorovich Khoroshevsky (b. 1946) graduated from Moscow Physical-Engineering Institute in 1971, PhD (1976), DrSc (1995). Hi is Professor at Moscow Institute of Physics and Technology (Department of Intelligent Systems). One of the founders of the Soviet, then Russian Association of Artificial Intelligence (RAAI), RAAI Scientific Council member. Member of the editorial boards of scientific journals “Artificial Intelligence and Decision Making”, “Software Products and Systems”. The list of scientific papers contains more than 130 publications in the AI domain (5 monographs and textbooks).