УДК 004.04

ОНТОЛОГИЧЕСКИЕ МОДЕЛИ ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

А.И. Водяхо[1], В.В. Никифоров[2]

[1]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), Санкт-Петербург, Россия
aivodyaho@mail.ru

[2]Санкт-Петербургский институт информатики и автоматизации РАН (СПИИРАН), Санкт-Петербург, Россия
nik@iias.spb.su

Аннотация

Современный этап развития информационных технологий характеризуется ужесточением требований, предъявляемых к эффективности функционирования создаваемых информационных систем, к стоимости и срокам их разработки, а также к уровню их «интеллекта». Действенным средством решения этих задач является использование онтологических моделей на всех этапах жизненного цикла информационных систем. В значительной части создаваемые информационные системы являются системами реального времени. Для систем реального времени специфическим является архитектурный этап проектирования. На этом этапе предлагается использование архитектурных онтологических моделей, которые позволяют реализовывать систему поддержки принятия архитектурных решений и накопления архитектурного знания. Рассматривается структура архитектурной онтологической модели, выделяются три уровня этой модели. Рассматривается состав знаний, который должен быть включен в ядро модели. Встроенные онтологические модели в системах реального времени могут быть использованы при реализации функций, связанных с реализацией когнитивного поведения. Делается вывод о целесообразности интегрирования рассматриваемой архитектурной онтологической модели в широко используемый архитектурный фреймворк Захмана.

Ключевые слова: *онтология, фреймворк Захмана, проектирование, системы реального времени.*

Цитирование: *Водяхо, А.И.* Онтологические модели для систем реального времени / А.И. Водяхо, В.В. Никифоров // Онтология проектирования. – 2018. – Т. 8, №2 (28). - С.240-252. – DOI: 10.18287/2223-9537-2018-8-2-240-252.

Введение

Современный этап развития информационных технологий (ИТ) характеризуется постоянным увеличением сложности и расширением сферы применения информационных систем (ИС), повышением требований к быстродействию. Значительная часть создаваемых ИС относится к системам реального времени (СРВ). СРВ – это один из важных подклассов ИС, который можно определить как ИС, относящиеся к определённому домену требований, т.е. ИС, объединенные общностью требований в плане быстродействия. СРВ находят применение в самых различных предметных областях, таких как системы управления подвижными объектами (автомобили, самолеты, ракеты, роботы), системы управления оборудованием и др. Для достижения требуемого качества функционирования необходимо реализовывать достаточно сложную логику работы ИС и перестраивать её в зависимости от наблюдаемых ситуаций в условиях ограниченных ресурсов. СРВ могут быть построены с использованием различных архитектурных стилей, таких как конвейеры и фильтры (pipes&filters), системы, управляемые событиями, репозитарии [1]. Несмотря на то, что разработана общая теория построения СРВ, предлагаются многочисленные методики

проектирования СРВ, имеются рекомендации по проектированию, доступны библиотеки паттернов, создание и сопровождение СРВ продолжают оставаться нетривиальной задачей, решение которой требует специальных знаний и высокой квалификации, особенно когда речь идет об экстремальных характеристиках по быстродействию.

Для решения задач проектирования и моделирования в ИТ-сфере широко применяется онтологический подход, который активно развивается. В частности, наблюдается всё более усиливающееся взаимовлияние между онтологическим проектированием и программной инженерией. Этот процесс уже идёт, по крайней мере, полтора десятка лет, но онтологический подход до сих пор не превратился в архитектурный стиль и в среде прикладных программистов продолжает оставаться редким явлением.

В настоящей статье рассматриваются вопросы применения онтологических моделей (ОМ) при проектировании СРВ с учётом особенностей этого класса систем.

1 Применение ОМ при проектировании СРВ

Основные типовые этапы проектирования СРВ включают:

- формирование требований;
- архитектурное проектирование:
  - разработка требований к архитектуре;
  - принятие архитектурных решений;
  - разработка архитектурных моделей;
  - разработка архитектурного описания;
  - оценка архитектуры с помощью моделей;
- проектирование приложения по моделям;
- отладка, тестирование;
- модернизация.

Общий подход. Большинство этапов идентично как для ИС, не относящихся к классу СРВ, так и для систем мягкого и жёсткого реального времени (ЖРВ). С точки зрения реализации процесса проектирования на верхнем уровне онтологический подход к проектированию подробно описан в [2], хотя этот подход относительно слабо связан с реальным процессом проектирования программного обеспечения [3]. Это относится, в частности, к архитектурному проектированию ИС. Рассмотрим возможные подходы к использованию ОМ на различных этапах проектирования ИС.

Формирование требований. На этапе формирования требований доменные онтологии, описывающие область применения целевой системы, могут использоваться в качестве словаря, который могут использовать заинтересованные стороны для исключения неправильного понимания требований. Кроме того, ОМ могут быть использованы как средство формализованного описания требований [4, 5] и управления требованиями [4, 6].

Архитектурное проектирование СРВ с использованием ОМ. Архитектурное проектирование – это тот этап, на котором специфика СРВ проявляется в полной мере. В первую очередь это относится к программной архитектуре. Онтологический подход к архитектурному проектированию СРВ различного назначения может быть реализован, в частности, в рамках архитектурного фреймворка Захмана (ФЗ), который, по существу, представляет собой онтологию [7]. В ФЗ заложены и отчасти реализованы идеи, связанные с накоплением и повторным использованием архитектурного знания [8] в форме архитектурных моделей. ФЗ можно рассматривать как контейнер для хранения моделей и метамоделей, в частности ОМ, разного уровня и поддерживающих разные архитектурные перспективы и точки зрения [9].

В настоящее время доступны реализации ФЗ для разных платформ, которые активно используются на практике, например в популярном продукте Enterprise Architect [10].

В основе ФЗ лежит идея, состоящая в том, что функционирование ИС можно описать в терминах ответа на шесть простых вопросов: ***что, как***, ***где, кто, когда, почему***: используемые данные (***что?),*** процессы и функции (***как?),*** места выполнения процессов (***где?),*** организации и персонал (***кто?),*** управляющие события ***(когда?),*** цели и ограничения, определяющие работу системы (***почему?).*** Ответы на эти вопросы можно давать с использованием различных понятий, т.е. с разной степенью детализации. При этом выделяют шесть уровней детализации: уровень контекста, уровень бизнес-описаний, системный уровень, технологический уровень, технический уровень, уровень реальной системы.

Обозначенные вопросы определяют шесть аспектов рассмотрения с точки зрения разных заинтересованных сторон, в число которых входят: аналитики, менеджеры, архитекторы, разработчики, системные администраторы, пользователи.

Таким способом формируется матрица размером 6х6, каждой клетке которой ставятся в соответствие модели и артефакты. Фрагмент матрицы ФЗ, относящийся к архитектурному уровню, приведён в таблице 1. Прочие клетки заполняются стандартным для ФЗ способом.

Первая строка (контекст) соответствует уровню позиционирования системы с учётом внешних факторов. Данная строка определяет контекст всех последующих строк, отражает самый общий взгляд на организацию проектируемой ИС и отражает точку зрения бизнес-аналитика. Вторая строка описывает функционирование ИС в бизнес-терминах. Третья строка соответствует видению системы архитектором. Четвертая строка – это привязка данных и операций над ними к определенным программным и аппаратным платформам и инструментальным средствам. Пятая строка отражает видения системы администратором (системный администратор, администратор баз данных и т.п.). Шестая строка описывает функционирующую систему. Особо следует отметить, что ФЗ не накладывает ограничений на типы используемых моделей, и ОМ могут быть помещены в этот контейнер без проблем.

Таблица 1 – Матричное представление ФЗ (фрагмент)

№	Степень детализации	Что	Как	Где	Кто	Когда	Почему	Заинтересованная сторона
1	Контекст							Аналитики
2	Бизнес-модель ИС							Топ менеджеры
3	Системная модель ИС	Концептуальные модели данных	Архитектура приложения	Архитектура распределенной ИС	Интерфейсы пользователя	Модель работы с событиями	Бизнес-правила	Архитекторы
4	Технологическая модель ИС							Разработчики
5	Детальное описание							Администраторы
6	Функционируюшая ИС							Пользователи

Специфика СРВ проявляется наиболее отчетливо на этапе архитектурного проектирования (третья строка). Это этап, на котором архитектором принимаются архитектурные решения и формируется архитектурное описание, включающее, в частности, архитектурное обоснование [11]. При этом первоочередной интерес представляют архитектурные перспективы, относящиеся к обеспечению быстродействия.

Архитектурное знание можно определить как знание, необходимое архитектору для принятия архитектурных решений и составления архитектурного описания. Архитектурное знание накапливается преимущественно в виде правил, а составление архитектурного

описания требует обоснования выбора тех или иных архитектурных решений. Для решения перечисленных выше задач могут быть успешно использованы ОМ. При этом ОМ, относящиеся к архитектурному (третьему) уровню, можно определить как архитектурные ОМ (АОМ).

2 Предлагаемый подход

В основу предлагаемого подхода к использованию АОМ для решения задач архитектурного проектирования СРВ положены следующие принципы:

- АОМ – это часть онтологического описания ИС в рамках ФЗ;
- АОМ может использоваться либо как база знаний, к которой архитектор может обращаться со SPARQL-запросами, либо может использоваться совместно с другими не онтологическими моделями для формирования моделей более высокого уровня;
- АОМ имеет трехуровневую структуру, к каждому из уровней можно обращаться с запросами соответствующего уровня.

Уровни могут рассматриваться как уровни зрелости АОМ: уровень 1 (ядро онтологии, которое содержит общие знания о СРВ), уровень 2 (инкапсулирует знания, позволяющие работать на уровне архитектурных тактик [9] и архитектурных обоснований), уровень 3 (позволяет проверять корректность созданных архитектурных моделей).

При работе на первом уровне к АОМ СРВ можно обращаться с запросами типа:

- *Какие механизмы управления могут использоваться в СРВ?*
- *Какие механизмы могут использоваться при данных ограничениях?*
- *Какие инструментальные средства существуют для поддержки данного механизма?*
- *Какие паттерны доступны для реализации данного механизма?*
- *Какие механизмы ранее использовались в похожих проектах?*

При работе на втором уровне АОМ может отвечать на вопросы типа:

- *Возможно ли использование данного механизма при заданных ограничениях?*
- *Является ли механизм лучшим для использования в системе при заданных ограничениях?*
- *В чём состоят достоинства и недостатки разных механизмов при заданных ограничениях?*

При работе на третьем уровне АОМ можно получить ответы на вопросы типа:

- *Является ли созданная архитектурная модель корректной?*
- *Является ли созданная АОМ оптимальной при заданных ограничениях?*

Очевидно, что при работе на втором и третьем уровнях требуется использование внешних сервисов, таких, например, как моделирование в терминах сетей Петри и др.

В настоящее время разработан каркас первого уровня АОМ, осуществляется его тестирование. В него заложены базовые знания из области современных СРВ. В основу положены такие сущности, как *требования, протокол, ресурс, инструмент, процесс, управление процессами*. АОМ СРВ строится с использованием традиционных подходов к проектированию онтологий [2]. Уровни 2 и 3 находятся на стадии проработки.

3 Знания, заложенные в АОМ СРВ уровня 1

Требования к составу знаний. Основная проблема при построении ядра АОМ - отбор знаний, которые должны быть представлены в разрабатываемой ОМ. Модель должна быть, с одной стороны, практически полезной, т.е. достаточно подробной, а с другой стороны, иметь ограниченные размеры, поскольку в противном случае её поддержка в актуальном состоянии становится неразрешимой задачей в условиях ограниченности ресурсов, выделяемых на

сопровождение. В основу были положены результаты теоретических исследований и практических разработок, проводившихся СПИИРАН в области СРВ в последние годы. Ниже в сжатом виде приведены знания, которые «зашиты» в ядро АОМ.

Специфика СРВ. Ключевое свойство СРВ состоит в том, что они работают в «структуре времени», определяемой ходом внешних процессов [12]. Такое соответствие между ходом исполнения компонентов программных приложений СРВ и ходом внешних процессов часто обусловлено наличием жёстких временных рамок для информационных обменов с внешними процессами. Одним из главных следствий этого обстоятельства является требование организации программного приложения СРВ в виде комплекса кооперативных задач - комплекса задач, совместно исполняемых для достижения общих целей функционирования СРВ, при этом возникает нетривиальная задача управления таким комплексом.

В ходе работы многозадачного программного комплекса составляющие его кооперативные задачи разделяют общие системные ресурсы: исполнительные ресурсы (в первую очередь – процессоры, ядра многоядерных процессоров) и информационные ресурсы (глобальные массивы данных, интерфейсные регистры периферийных устройств, элементы человеко-машинного интерфейса и т.п.). В случае СРВ на построение многозадачных программных приложений накладываются, во-первых, требования корректной реализации доступа задач к разделяемым информационным ресурсам и, во-вторых, требования эффективного использования исполнительных ресурсов.

Знания о способах организации доступа к разделяемым ресурсам. Требования корректной организации доступа к разделяемым ресурсам состоят в выполнении следующих условий: целостности разделяемых информационных ресурсов и отсутствия опасности возникновения взаимного блокирования задач. Выполнение этих условий необходимо для обеспечения логической корректности программных приложений не только в случае СРВ, но и в случае любых программных приложений, в которых для обеспечения согласованности исполнения кооперативные задачи обмениваются данными и синхронизирующими (сигнальными) информационными сообщениями.

В случае СРВ имеет место дополнительное условие динамической корректности — обеспечение своевременности исполнения задач. В классической постановке продолжительность каждого очередного исполнения задачи реального времени ограничена предельным сроком. Для задач, отвечающих требованиям ЖРВ, нарушение предельного срока считается недопустимым, поскольку может повлечь, например, материальные потери (в частности, вывод из строя оборудования) или даже человеческие жертвы.

В частном случае программные приложения СРВ могут состоять из независимых задач, в которых отсутствуют межзадачные синхронизирующие связи. Если такие связи имеют место, задачи являются взаимозависимыми. Для приложений с независимыми задачами в ряду перечисленных требований корректной работы системы остаются лишь требования динамической корректности, для удовлетворения которых выбирается подходящий порядок предоставления задачам имеющихся исполнительных ресурсов — выбор подходящих алгоритмов планирования вычислительных процессов. Исследования, посвященные разработке эффективных дисциплин планирования вычислительных процессов для СРВ на базе одиночных одноядерных процессоров, начались более сорока лет назад [13]. Были определены эффективные дисциплины планирования со статическими, фиксированными приоритетами задач (например, RM - Rate monotonic), и с динамическими, изменяемыми в ходе работы системы приоритетами задач (например, EDF - Early Deadline First).

Особенно интенсивно развиваются методы повышения эффективности использования вычислительных ресурсов многоядерных процессоров [14]. Дисциплины планирования со статическими приоритетами задач относительно просты в реализации, но в условиях

жёстких предельных сроков выполнения задач уступают дисциплинам с динамическими приоритетами задач в отношении эффективности использования процессорного времени.

Стандартный подход к обеспечению целостности разделяемых информационных ресурсов сводится к использованию «мьютексов» (англ. mutex, от mutual exclusion — «взаимное исключение») [15]. Для каждого из разделяемых ресурсов формируется программный объект мьютекс – синхронизирующий элемент, фиксирующий занятость ресурса. Участок кода, в рамках которого задача имеет доступ к определённому ресурсу (критический интервал по доступу к этому ресурсу), ограничивается синхронизирующими операторами захвата и освобождения ресурса. Состав и порядок действий, реализуемых этими операторами, определяется используемым протоколом доступа к разделяемым ресурсам. Для простейшего протокола PP (Primitive Protocol) единственное условие входа в критический интервал состоит в том, что требуемый ресурс свободен. В условиях применения протокола PP возможна инверсия приоритетов, ведущая к нарушению своевременности исполнения высокоприоритетных задач. Протокол наследования приоритетов PIP (Priority Inheritance Protocol) обеспечивает устранение возможности возникновения инверсии приоритетов, но не гарантирует решения проблемы взаимного блокирования. Протокол пороговых приоритетов PCP (Priority Ceiling Protocol) предотвращает и инверсию приоритетов, и возможность возникновения взаимного блокирования задач. Но его применение допустимо только в условиях использования дисциплин планирования со статическими приоритетами задач [15]. Применение PCP требует специальной статической (на этапе проектирования программного приложения) обработки схемы программного приложения, отражающей структуру межзадачных связей в многозадачном программном комплексе СРВ.

Новый протокол междольных контуров PIPC (Protocol Inter-Partite Contours) не только предотвращает инверсию приоритетов и возникновение взаимного блокирования задач, но при этом оставляет возможность использования эффективных дисциплин планирования с динамическими приоритетами [16]. В случае применения PIPC, как и в случае применения PIP, требуется специальная статическая обработка схемы приложения. Такая обработка опирается на построение графа связок критических интервалов - специального многодольного ориентированного графа, отражающего особенности межзадачных синхронизирующих связей [17].

Знания о способах работы с независимыми задачами. Программным приложениям СРВ, состоящим из независимых задач, не требуются услуги типа протоколов доступа к разделяемым ресурсам. Для таких приложений выбор дисциплины планирования выполняется в порядке поиска разрешения противоречий между требованиями снижения объёма выделяемых вычислительных ресурсов и требованиями динамической корректности (требованиями сохранения своевременности выполнения задач).

Для программных приложений СРВ, содержащих взаимозависимые задачи, на выбор дисциплины планирования накладывается дополнительное ограничение: выбираемая дисциплина планирования должна быть допустима в условиях применения требуемого протокола доступа к разделяемым ресурсам. Отмеченное дополнительное ограничение обусловлено тем, что применение конкретного протокола доступа может сужать состав допустимых дисциплин планирования.

Применением PCP защищает исполнение программного приложения от возникновения взаимного блокирования задач. При работе СРВ взаимное блокирование задач может привести к недопустимым последствиям. В этой связи разработчики СРВ обычно ориентируются на применение PCP при создании любых программных комплексов с взаимозависимыми задачами, автоматически сужая возможности выбора эффективных дисциплин планирования вычислительных процессов.

Наличие взаимозависимых задач не обязательно означает возможность возникновения их взаимного блокирования. Из самого этого факта не следует необходимость применения PCP, влекущего отказ от использования эффективных дисциплин планирования. Взаимное блокирование невозможно, если в кодах взаимозависимых задач отсутствуют пересекающиеся (вложенные или сцепленные) критические интервалы. Каждая пара пересекающихся критических интервалов называется связкой критических интервалов [18]. Таким образом, первым шагом анализа структуры программного приложения на возможность взаимного блокирования является проверка наличия в кодах задач связок критических интервалов. Отсутствие таких связок означает, что возникновение взаимного блокирования невозможно, что вместо PCP можно применять протоколы, не сужающие возможности выбора эффективных дисциплин планирования (например, PP или PIP).

Наличие связок критических интервалов является необходимым, но не достаточным условием возможности возникновения взаимного блокирования. Необходимое и достаточное условие такой возможности приведено в [18]. Для проверки программного приложения на возможность взаимного блокирования задач вводится отношение зависимости связок и предлагается строить многодольный ориентированный граф, каждая вершина которого соответствует отдельной связке, а дуги отражают отношение зависимости связок. В этих терминах необходимое и достаточное условие возможности взаимного блокирования формулируется так: взаимное блокирование задач возможно в том и только в том случае, если построенный многодольный ориентированный граф (граф зависимостей связок) содержит междольные контуры.

Таким образом, вторым шагом проверки структуры программного приложения на возможность взаимного блокирования является построение графа связок и проверка наличия в этом графе междольных контуров. Отсутствие междольных контуров означает, что в применении PCP нет необходимости, можно применять протоколы, допускающие возможность использования эффективных дисциплин планирования.

Граф связок не только обеспечивает проверку необходимого и достаточного условия возможности взаимного блокирования, он позволяет локализовать причины такой возможности. В работе [19] представлен подход к устранению этих причин путём локальной модификации кодов задач. В результате модификации можно обеспечить разрыв междольного контура графа связок, устраняя тем самым возможность взаимного блокирования. Если анализ показывает, что по каким-то причинам требуемая модификация кода недопустима, то ещё остаётся проверка возможности применения представленного в [17] протокола PIPC, который, с одной стороны, предотвращает взаимное блокирование и, с другой стороны, оставляет возможность использования эффективных дисциплин планирования с динамическими приоритетами.

Знания, которые закладываются в уровни 2 и 3. Знания, заложенные в АОМ первого уровня, не предназначены напрямую для генерации архитектурных решений. Они позволяют только получать информацию или ссылки на информацию о СРВ, которая необходима для принятия архитектурных решений. В первую очередь, эти знания полезны для архитекторов, не имеющих большого опыта при построении СРВ. Кроме того, эту модель можно использовать в учебных целях, в частности для тестирования обучаемых.

Для создания практически полезного сервиса поддержки принятия архитектурных решений при проектировании СРВ требуется создавать приложения на базе ядра, способные отвечать на вопросы, относящиеся к уровню 2. Реализация описанной выше процедуры требует постоянного учёта временных отношений. Следует заметить, что современные механизмы работы с онтологиями позволяют работать со временем [20], но описания получаются достаточно громоздким и число доступных ОМ этого типа ограничено. Поэтому

авторы ориентировались на использования внешних систем моделирования, в частности, разработанных в СПИИРАН. Приложения этого уровня строятся с использованием микросервисных решений. Эти работы находятся в завершающей стадии.

Приложения, относящиеся к уровню 3, строятся на базе сервисов, предоставляемых уровнями 1 и 2, и должны обеспечивать решение задач верификации и оценки эффективности алгоритмов, используемых в СРВ. Приложения этого уровня также строятся с использованием микросервисных решений. Реализация этого уровня должна позволить выйти на реализацию систем поддержки принятия решений (СППР) архитектурного этапа проектирования, т.е. обеспечивать получение информации о возможных решениях СРВ, реализацию архитектурных тактик [9], относящихся к достижению требуемых показателей быстродействия, автоматическое формирование архитектурного обоснования, увязывание показателей быстродействия с другими функциональными и нефункциональными характеристикам, верификацию механизмов управления вычислительными процессами, способность оценивать эффективность алгоритмов управления вычислительным процессом, а также накапливать знания об эффективности применения тех или иных механизмов управления вычислительным процессом. В настоящее время заканчивается проработка основных технических решений приложений уровня 3.

4 Проектирование приложения СРВ по моделям

Современный подход к проектированию ИС предусматривает широкое использование методов проектировании на основе моделей, предполагающее использование моделей, метамоделей и механизмов трансформации моделей, однако по сравнению с онтологиями выразительные способности используемых моделей ограничены, в частности они не позволяют применять процедуры логического вывода. Современные методологии проектирования предполагают использование MDA-ориентированных языков, таких как UML и SysML. Следует отметить, что вопросы интеграции ОМ и MDA-ориентированных языков проработаны достаточно хорошо [21-24].

ОМ могут также использоваться на этапе тестирования и сопровождения программной системы. В частности, возможно использование ОМ для генерации тестов как в статике так и в динамике. Этот подход представляет интерес, прежде всего, когда используются динамические архитектуры. В более простом случае ОМ может использоваться в режиме экспертной системы для поиска неисправностей [25].

Можно утверждать, что на всех этапах проектирования, кроме архитектурного проектирования, использование ОМ определяется не столько спецификой СРВ, сколько архитектурным стилем, комбинацией функциональных и нефункциональных требований, предъявляемых к разрабатываемой ИС, а также используемой методологией проектирования.

В таблице 2 приведены типовые варианты использования ОМ в СРВ и потенциальные преимущества от их использования на различных этапах проектирования.

Таблица 2 – Типовые варианты использования ОМ в СРВ и потенциальные преимущества от их использования

№	Вариант использования	Преимущества от использования ОМ
1	Формирование требований	Возможность формально описывать, анализировать и отслеживать требования
2	Архитектурное проектирование	Уменьшение числа фатальных ошибок проектирования. Накопление и повторное использование архитектурного знания
3	MDA- проектирование	Уменьшение стоимости и сроков проектирования, уменьшение числа ошибок проектирования
4	Тестирование и сопровождение	Возможность реализации накопления и обработки знаний о сбоях, ошибках и неисправностях. Возможность автоматической генерации тестов

5 Перспективы интеллектуализации систем СРВ

Проблема повышения уровня интеллекта создаваемых ИС, в частности СРВ и систем ЖРВ, является одной из ключевых задач современного этапа развития ИС. Применительно к СРВ основная проблема заключается в том, что реализация интеллектуальных элементов поведения требует перехода на работу со знанием. Манипуляции со знаниями, представленными в форме ОМ, предполагают реализации логического вывода, что, в свою очередь, требует выполнения большого числа машинных операций, которые не всегда можно выполнять параллельно, что затрудняет использование ОМ не только в системах ЖРВ, но и в СРВ с умеренными требованиями к строгости соблюдения сроков выполнения задач. Кроме того, время, требуемое для реализации логического вывода, не всегда возможно точно предсказать, поскольку потоковые онтологии ориентированы преимущественно на использование в рамках архитектурного стиля pipes&filters, а не на построение ИС, управляемых событиями. Для ядра СРВ, отвечающего за планирование вычислительных процессов, это, как правило, неприемлемо. В этом плане, по крайней мере при использовании современных платформ, встраивать ОМ напрямую в ядро СРВ возможно только при очень умеренных требованиях к строгости соблюдения сроков выполнения задач.

Более перспективным представляется направить усилия на разработку конверторов, которые могут строить на базе ОМ более простые, например автоматные, модели. В этом случае речь идёт о синтезе автоматов из ОМ. Ещё более эффективным представляется подход, при котором ядро системы СРВ строится по модульному принципу. Модули в этом случае реализуют политики управления вычислительными процессами, а ОМ используются для управления на уровне политик. В этом случае можно использовать многоуровневую автоматную модель с изменяемой структурой. Алгоритм построения таких автоматов описан, в частности, в [26].

В простейшем случае, при очень жёстких требованиях по времени, автоматы можно зашивать в код, а в более сложном случае при изменении контекста их можно повторно генерировать и загружать в форме модулей. Отдельные модули реализуют критичные к времени функции, которые могут генерироваться из архитектурного описания как в статике, так и в динамике. Для загрузки модулей можно предусмотреть отдельные временные слоты. В последнем случае можно говорить об адаптивной (agile) архитектуре в смысле [27]. Перспективы практического применения данного подхода определяются доступностью сервисов трансформации, которые позволяют переходить от онтологического описания к другим описаниям, таким как автоматы, таблицы, деревья решений и другим моделям, которые позволяют получить высокое быстродействие. Возможные подходы к использованию онтологии при построении когнитивных СРВ приведены в таблице 3.

В СРВ ОМ могут использоваться для реализации только части функционала. Чаще всего, они используются в многоуровневых системах для реализации функционала, относящегося к одному из уровней. Как правило, это верхние уровни, где требования по быстродействию менее жёсткие. Часто вспомогательный функционал представлен такими системами как системы мониторинга состояния системы и реконфигурации в случае необходимости. Онтологии в этом случае используются для представления модели самой системы в виде автоматной модели с изменяемой структурой. В качестве элементов интеллектуального поведения можно рассматривать способность СРВ собирать информацию о собственном функционировании и изменять поведение по результатам анализа. Это может быть сделано посредством генерации и загрузки отдельных модулей [25].

Таблица 3 – Возможные подходы к использованию онтологии при построении когнитивных СРВ

№	Функции	ЖРВ	РВ	Способ использования
1	Реализация основного функционала (управление ресурсами)	Использование жёсткого кода, синтезированного из ОМ	Использование небольших ОМ, оптимизированных для СРВ	1. Выделение фрагментов онтологического архитектурного описания 2. Построение специальной онтологии, оптимизированной для СРВ 3. Получение автоматного или другого описания из ОМ и помещение его в загружаемый модуль
2	Реализация отдельных функций основного функционала	Возможно использование загружаемых модулей	Использование небольших фрагментов ОМ	1. Использование загружаемых модулей 2. Встраивание фрагментов ОМ в код
3	Реализация вспомогательного функционала	Накопление результатов оценки эффективности функционирования	Использование значительных по размеру фрагментов ОМ	Встраивание фрагментов ОМ в код

Заключение

Несмотря на значительные усилия, затраченные на устранение разрыва между традиционными модельно-ориентированными подходами и онтологическим подходом к проектированию, и достижение определённых успехов в данном направлении, этот разрыв продолжает оставаться достаточно большим. На практике необходимо, чтобы СРВ реализовывали всё более сложное интеллектуальное поведение, что в свою очередь требует, чтобы СППР, входящие в состав СРВ, работали на уровне знаний.

Благодарности

Работа выполнена в рамках государственного задания № 0073-2018-0004, утверждённого Федеральным агентством научных организаций России 12 января 2018 г.

Список источников

[1] ***Shaw, M.*** Software Architecture: Perspectives on an Emerging Discipline / M. Shaw, D. Garlan. - Prentice-Hall. NJ, 1996. — 242 p.

[2] ***Pan, J.*** Ontology-Driven Software Development / J. Pan, S. Staab, U. Aßmann, J. Ebert,. ZhaoY. - Springer NY, 2013. – 337 p. – DOI: 10.1007/978-3-642-31226-7.

[3] ***Bass, L.*** Software Architecture in Practice. 3rd ed. / L. Bass, P. Clements, R. Kazman. — Upper Saddle River, NJ.: Addison-Wesley. 2013. — 661 p.

[4] ***Mayank, V.*** Requirements Engineering and the Semantic Web, Part II. Representation, Management, and Validation of Requirements and System-Level Architectures / V. Mayank, N. Kositsyna, M. Austin. - Technical Report. TR 2004-14, University of Maryland. 2004.

[5] ***Decker, B.*** Selforganized Reuse of Software Engineering Knowledge supported by Semantic Wikis / B. Decker, J. Rech, E. Ras, B. Klein, C. Hoecht. - In: Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE). November 2005.

[6] ***Lin, J.*** A Requirement Ontology for Engineering Design / J. Lin, M.S. Fox, T. Bilgic. - Enterprise Integration Laboratory, University of Toronto, Manuscript, September 1996. – DOI: 10.1177/1063293x9600400307.

[7] The Zachman International e-Commerce - http://www.zachmaninternational.com.

[8] ***Babar, M.A.*** Software Architecture Knowledge Management / M.A. Babar, T. Dingsøyr, P. Lago, van H. Vliet. - Dordrecht Heidelberg London New York: Springer. 2009. - 279 p. – DOI: 10.1007/978-3-642-02374-3.

[9] ***Rozanski N.*** Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives / N. Rozanski, E. Woods // Viewpoints. - 2005. V.8, No 2. - p. 576.

[10] Enterprise Architect User Guide - https://www.sparxsystems.com

[11] International Standard ISO/IEC/IEEE 42010 Systems and software engineering — Architecture description - http://www.iso.org
[12] ***Давиденко, К.Я.*** Технология программирования АСУТП. Проектирование систем реального времени, параллельных и распределенных приложений. - М.: Энергоатомиздат. 1985. - 183 с.
[13] ***Liu, C.*** Scheduling Algorithms for Multiprocessing in a Hard Real-Time Environment / C. Liu, J. Layland // Journal of the ACM. - 1973. V.20. No.1. - pp.46–61.
[14] ***Andersson, B.*** Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38% / B. Andersson // Proceedings of the 12th International Conference on Principles of Distributed Systems. 2008. pp.73–88. – DOI: 10.1007/978-3-540-92221-6_7.
[15] ***Liu, J.W.S.*** Real-Time Systems. NJ: Prentice Hall. 2000. - 590 p.
[16] ***Никифоров, В.В.*** Протокол предотвращения взаимного блокирования задач в системах реального времени / В.В. Никифоров // Известия ВУЗов. Приборостроение. 2014. №12. - С.21-27.
[17] ***Никифоров, В.В.*** Структурные модели для анализа многозадачных программных систем / В.В. Никифоров, В.А. Павлов // Информационно-измерительные и управляющие системы. 2011. №9. - С.19-29.
[18] ***Никифоров, В.В.*** Статическая проверка корректности разделения ресурсов в системах реального времени Статическая проверка корректности разделения ресурсов в системах реального времени / В.В. Никифоров, С.Н. Баранов // Труды СПИИРАН, №3 (52), 2017. - С.132-141.
[19] ***Никифоров, В.В.*** Доступ к разделяемым ресурсам в системах реального времени с переменными приоритетами задач / В.В. Никифоров, А.А. Тюгашев // Известия ВУЗов, Приборостроение, т.59, №11, 2016. - С. 964-970.
[20] ***Petnga, L.*** Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems / L. Petnga, M. Austin. - http://www.isr.umd.edu/~austin/reports.d/CSER2013-LP-MA-Preprint.pdf.
[21] ***Gasevic, D.*** Model Driven Architecture and Ontology Development / D. Gasevic, D. Djuric, V. Devedzic. - Springer-Verlag, 2006. – DOI: 10.1007/3-540-32182-9.
[22] ***Kiko, K.*** Integrating Enterprise Information Representation Languages / K. Kiko, C. Atkinson. - In: Proc. of Int. Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005), Enschede, The Netherlands (2005).
[23] ***Cranefield, S.*** UML and the Semantic Web. In: Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford. 2001.
[24] Ontology Definition Metamodel. OMG: Ontology Definition Metamodel RFP. 6th Revised Submission. 2006. - http://www.omg.org/dontology/.
[25] ***Жукова, Н.А.*** Архитектурный подход к построению систем обработки многомерных измерений параметров пространственно распределенных объектов» / Н.А. Жукова, А.И. Водяхо // Известия СПбГЭТУ “ЛЭТИ”, №2, 2013. - С.21-26.
[26] ***Osipov, V.Yu.*** Automatic Synthesis of Action Programs for Intelligent Robots, Program. Comput. Software. 42 (3) 2016. - pp.155–160. – DOI: 10.1134/s0361768816030063.
[27] ***Babar, M.A.*** Agile Software Architecture / M.A. Babar, A.W. Brown, I. Mistrik. - Waltham, MA: Elsevier Inc. 2014. - 392 p. – DOI: 10.1016/c2012-0-01208-2.

ONTOLOGY MODELS FOR REAL TIME SYSTEMS

A.I. Vodyaho[1], V.V. Nikiforov[2]

[1] *Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, Russia*
aivodyaho@mail.ru

[2] *Saint Petersburg Institute for Information and Automation RAS. Saint Petersburg, Russia*
nik@iias.spb.su

Abstract

The modern state of information technologies is characterized by permanently increasing level of requirements to operation effectiveness, cost, duration of development and the level of intelligence of information systems (IS). The effective approach to solution of these problems is the usage of ontology models in all stages of IS life cycle. An essential part of modern IS are real time systems (RTS). In the article the mechanisms of processes management which are used in modern RTS and possible aspects of ontology models used for solving problems appearing in different stages of RTS development are analyzed. From the development point of view, RTS requires a specific architectural design stage. This

stage uses architectural ontological models which allow implementation of architectural decision support system and mechanisms of architectural knowledge saving and processing. The structure of architectural ontological model is considered, three levels of model maturity are defined. The composition of knowledge to be included into a model kernel is considered. The built-in ontological models in RTS can be used for implementation of the functions connected with realization of cognitive behavior. It is suggested to use presented architectural ontological model as an element of well known and widely used Zachman's framework.

Key words: *ontology, Zachman's framework, design, real time systems.*

Citation: *Vodyaho AI, Nikiforov VV.* Ontology models for real time systems [In Russian]. *Ontology of designing*. 2018; 8(2): 240-252. DOI: 10.18287/2223-9537-2018-8-2-240-252.

Acknowledgment

The work was financially supported within the government task No. 0073-2018-0004, approved by Federal Agency of Scientific Organizations on January 12, 2018.

References

[1] ***Shaw M, Garlan D.*** Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall. NJ, 1996.
[2] ***Pan J, Staab S, Aßmann U, Ebert J, ZhaoY***. Ontology-Driven Software Development. Springer NY, 2013. – DOI: 10.1007/978-3-642-31226-7.
[3] ***Bass L, Clements P, Kazman R.*** Software Architecture in Practice. 3rd ed. – Upper Saddle River, NJ: Addison-Wesley. 2013.
[4] ***Mayank V, Kositsyna N, Austin M.*** Requirements Engineering and the Semantic Web, Part II. Representation, Management, and Validation of Requirements and System-Level Architectures. Technical Report. TR 2004-14, University of Maryland. 2004.
[5] ***Decke B, Rech J, Ra E, Klein B, Hoecht C.*** Selforganized Reuse of Software Engineering Knowledge supported by Semantic Wikis. In: Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE). November 2005.
[6] ***Lin J, Fox MS, Bilgic T.*** A Requirement Ontology for Engineering Design. Enterprise Integration Laboratory,, University of Toronto, Manuscript, September 1996. – DOI: 10.1177/1063293x9600400307.
[7] The Zachman International e-Commerce Site: http://www.zachmaninternational.com.
[8] ***Babar MA, Dingsøyr T, Lago P, van Vliet H.*** Software Architecture Knowledge Management. Dordrecht Heidelberg London New York: Springer. 2009. – DOI: 10.1007/978-3-642-02374-3.
[9] ***Rozanski N, Woods E.*** Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison-Wesley. 2005. - 546 p.
[10] Enterprise Architect User Guide - https://www.sparxsystems.com.
[11] International Standard ISO/IEC/IEEE 42010 Systems and software engineering — Architecture description http://www.iso.org
[12] ***Davidenko KYa.*** Automated systems programming technologies. Real time systems development, parallel and distributed systems [In Russian]. – Moscow: Energoatomizdat. 1985.
[13] ***Liu C, Layland J.*** Scheduling Algorithms for Multiprocessing in a Hard Real-Time Environment // Journal of the ACM. 1973; 20(1): 46–61.
[14] ***Andersson B.*** Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38% // Proceedings of the 12th International Conference on Principles of Distributed Systems. 2008. - P.73–88. – DOI: 10.1007/978-3-540-92221-6_7.
[15] ***Liu JWS.*** Real-Time Systems. NJ: Prentice Hall. 2000.
[16] ***Nikiforov VV.*** Deadlock avoidance protocol for real time systems [In Russian]. Izvestiya VUZov. Priborostroenie. 2014; 12: 21-27.
[17] ***Nikiforov VV, Pavlov VA.*** Structural models for multitask systems analysis [In Russian]. Informatsionno-ismeritelniye i upravlyayustchie systemy. 2011; 9: 19-29.
[18] ***Nikiforov VV, Baranov SN.*** Statistical testing of correctness of resources distribution in real time systems [In Russian]. Trudy SPIIRAS, 2017; 52(3): 132-141.
[19] ***Nikiforov VV, Tyugashev AA.*** The access to shared resources in real time systems [In Russian]. Izvestiya VUZov. Priborostroenie, 2016; 59(11): 964-970.
[20] ***Petnga L, Austin M.*** Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems. - http://www.isr.umd.edu/~austin/reports.d/CSER2013-LP-MA-Preprint.pdf.

[21] ***Gasevic D, Djuric D, Devedzic V.*** Model Driven Architecture and Ontology Development. Springer-Verlag, 2006. – DOI: 10.1007/3-540-32182-9.
[22] ***Kiko K, Atkinson C.*** Integrating Enterprise Information Representation Languages. In: Proc. of Int. Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005), Enschede, The Netherlands. 2005.
[23] ***Cranefield S.*** UML and the Semantic Web. In: Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford. 2001.
[24] Ontology Definition Metamodel. OMG: Ontology Definition Metamodel RFP. - http://www.omg.org/dontology/, 6th Revised Submission. 2006.
[25] ***Zhukova NA, Vodyaho AI.*** Architectural approach to development of information systems for processing results of multidimentional parameters of distributed object measurements. [In Russian]. Izvestiya LETI, 2013; 2: 21-26.
[26] ***Osipov VYu***. Automatic Synthesis of Action Programs for Intelligent Robots, Program. Comput. Software. 2016; 42(3): 155–160. – DOI: 10.1134/s0361768816030063.
[27] ***Babar MA, Brown AW, Mistrik I.*** Agile Software Architecture. Waltham, MA: Elsevier Inc. 2014. – DOI: 10.1016/c2012-0-01208-2.

Сведения об авторах

Водяхо Александр Иванович, 1947 г. рождения. Окончил Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) (СПбГЭТУ ЛЭТИ) в 1973 году, к.т.н. (1977), д.т.н. (1992), профессор кафедры вычислительной техники СПбГЭТУ ЛЭТИ. В списке научных трудов более 50 работ в области архитектуры вычислительных и информационных систем.
Alexandre Ivanovich Vodyaho (b. 1947) graduated from the Saint Petersburg Electrotechnical University "LETI" (ETU) in 1973. Ph.D (1977), DrS (1992) Professor, ETU University. He is co-author of more than 50 publications in the field of computer and information systems architecture.

Никифоров Виктор Викентьевич, 1941 г. рождения. Окончил Ленинградский политехнический институт им. М.И. Калинина (ныне Санкт-Петербургский политехнический университет Петра Великого) в 1965 году, к.т.н. (1984), д.т.н. (1991), профессор (2001), главный научный сотрудник лаборатории вычислительной техники и технологий программирования СПИИРАН. В списке научных трудов более ста работ в области разработки программных средств для робототехнических комплексов, программных систем моделирования дискретных процессов, операционных систем для программных приложений реального времени.
Viktor Vikentievich Nikiforov, (b. 1941) graduated from the Leningrad Polytechnic Institute in 1965. Ph.D (1984), DrS (1991), professor (2001), chief research fellow Saint-Petersburg Institute for Information and Automation RAS. He is author and co-author of over a hundred publications in the field of software application design.