УДК 681.3

Моделирование свойств операционной системы реального времени OpenComRTOS при помощи OWL-DL онтологий

Межуев В. И.
Бердянский государственный педагогический университет
mejuev@ukr.net

Abstract

Mezhuyev V. Modelling properties of OpenComRTOS Real-Time Operation System by OWL-DL ontologies. *In the paper the necessity and possible advantages of an ontology development of the Open Communication Real Time Operating System, its Real-Time applications and related distributed networks of processing nodes are explored. The properties of OpenComRTOS which can be expressed with OWL Ontology and checked by OWL-DL reasoners are shown.*

1. Введение

Онтологии сейчас широко используются для описания понятий и их отношений в различных предметных областях (ПрО). Эта статья посвящена разработке OWL онтологии операционной системы реального времени OpenComRTOS [1]. Язык OWL (Web Ontology Language) был создан WWW консорциумом [2]. Чтобы показать возможный потенциал разработки онтологий для OpenComRTOS и ее приложений, мы используем OWL-DL (то есть основанный на Description Logic) вид OWL. Description Logic является подмножеством логики первого порядка и используется в системах автоматизации умозаключений (в работе мы используем FACT ++ [3]).

Рынок современного программного обеспечения (SW) и аппаратных средств (HW) требует увеличения производительности труда разработчиков, более низкой стоимости и большего количества функциональных возможностей компьютерных систем. Более того, современные SW и HW системы достигли того уровня сложности, для которого основополагающими становятся требования отказоустойчивости и безопасности. OpenComRTOS позиционируется на рынке встроенных систем (embedded systems) как первая RTOS, разработанная с использованием формальных методов. Метод моделирования использовался нами, прежде всего, на этапе проектирования OpenComRTOS, что имело результатом четкую архитектуру системы с исключительными свойствами и сервисами. После реализации системы, для подтверждения ее безопасности и отказоустойчивости, сервисы OpenComRTOS были смоделированы снова при помощи языка TLA+ (Temporal Logic of Actions) и проверены программным валидатором TLC (Temporal Logic Checker) [4].

Заметим, что онтологический подход возник из концепций Семантической Паутины (Semantic Web) [5] и предназначен для использования программными агентами, понимающих семантику данных Интернет. Следуя этому подходу, онтология OpenComRTOS помещается в Интернет, чтобы агенты сети смогли понять и использовать распределенные в Интернет сервисы данной системы. ‘Смогли понять’ означает ‘должным образом сконфигурировать’ на основе онтологии OpenComRTOS некоторое SW-приложение, запрашивающее ресурсы OpenComRTOS. Таким образом, онтологии предназначены для выражения семантики ПрО с целью ее последующего использования программными агентами сети.

В то же время концептуальное моделирование остается первым этапом разработки любой системы во всякой ПрО. ***Задачей*** данной статьи является анализ преимуществ концептуального моделирования OpenComRTOS при помощи OWL-онтологий, а ***предметом*** - свойства OpenComRTOS, которые могут быть смоделированы с OWL-DL и проверены соответствующими системами автоматизации умозаключений. Таким образом, в работе предлагается объединить две важнейшие стадии разработки системы - концептуального моделирования и проверки (верификации) ее свойств.

Для построения OWL-онтологии OpenComRTOS использовался Protégé [6]. Для визуализации классов и иерархических отношений использовался Jambalaya [7] модуль Protégé.

2. Методология Взаимодействующих Сущностей и ее соответствие концепциям RDF и OWL

Теоретической основой OpenComRTOS является парадигма Взаимодействующих Сущностей (IE – Interacting Entities). Исходя из этого, основными понятиями IE являются **сущность** и **взаимодействие**. Сущность характеризуется **атрибутами** (сущность также может включать другие сущности) и внутренними **функциями**.

Внешнее поведение сущностей определяется их **взаимодействием**.
Взаимодействие сущностей в IE также определяется атрибутами - **именем**, **объектом** и **субъектом**, что является близким к концепциям Resource Description Framework (RDF) [8]. В RDF взаимодействие определяется тройкой: предметом, объектом и отношением, каждый элемент которой имеет URI (Uniform Resource Identifier).
Подобие этих подходов свидетельствует как о возможностях использования RDF для архитектурного моделирования программных систем, так и методологии взаимодействующих сущностей в контексте идей Semantic Web.

Вообще говоря, любая онтология решает задачу уникального описания и представления знаний о гетерогенных системах, как например:
- различных HW-платформ (например, Intel, MLX, MBlaze, Leon3 и т.д.);
- различных SW-платформ (например, Win32, Linux, OSE, CMX, Solaris, OpenComRTOS и т.д.).

Чтобы использоваться программными агентами, занимающимися в самом известном случае поиском и категоризацией содержания сети, такие гетерогенные системы должны иметь уникальную форму своего представления (например, на основе RDF). Заметим также, что использование уникальной формы представления знаний и формулировка правил дедукции позволяет автоматизировать другие связанные классы задач, как например:
- портирование OpenComRTOS и других RTOS в различные HW платформы;
- эмуляция OpenComRTOS и других RTOS на различных SW платформах;
- моделирование сервисов и приложений OpenComRTOS при помощи других подходов (например, SDL-RT - Specification and Description Language Real Time [9]);
- конфигурирование работающих на различных HW платформах SW приложений [10] (например, конфигурирование web-приложений для получения сервисов OpenComRTOS).

Поиск методов решения данных проблем состоит в анализе концептуальных моделей, например, выраженных в форме онтологии методологии разработки приложений OpenComRTOS и онтологии SDL-RT. Оба данных подхода отражают семантически эквивалентные концепции активных (процессов или задач) сущностей и сущностей синхронизации (например, событие, семафор, ресурс и т.д.). Моделирование сервисов и приложений OpenComRTOS при помощи SDL-RT сводится к задаче генерации кода для приложений OpenComRTOS, выраженных в понятиях SDL-RT. Практическая ценность применения онтологий состоит в возможности формулировки и использования соответствующих правил дедукции. Например, правилом дедукции необходимо установить эквивалентность семантики поведения таких сущностей синхронизации как *событие* OpenComRTOS и *бинарный семафор* SDL-RT.

Здесь мы приходим к проблеме разработки онтологий решения задач как перспективном направлении наших исследований. Имеются классы стандартных задач, возникающих в приложениях RTOS, как например осуществление различных форм синхронизации. Классификация таких возникающих в предметных областях проблем и формулировка правил дедукции (в самом простом случае, выбора метода на основе некоторых критериев) позволяет автоматизировать решение проблемы (например, использование сущности *событие*, если проблема была классифицирована как необходимость синхронизации двух выполняющихся задач). Разработке онтологии решения задач OpenComRTOS будут посвящены наши дальнейшие работы.

Также обратим внимание на соответствие между концепциями OWL и IE, как методологии разработки OpenComRTOS. Компонентами OWL онтологий являются классы, свойства и экземпляры. Концепции IE - это сущности, атрибуты, взаимодействия, т.е. абстрактные понятия (близкие по смыслу к понятию класса OWL), которые инстанциируются в процессе выполнения приложения. Например, экземплярами сущности Порт (Port) OpenComRTOS являются $Port_1$, $Port_2$, $Port_3$ и т.д., что близко к концепции экземпляра класса OWL.
Экземпляры класса OWL также представляют собой конкретные объекты предметной области. Но понятия ‘класс’ и 'экземпляр' в OWL практически идентичны: например, экземпляр может стать классом, если он обобщает некоторые понятия ПрО.

В IE методологии разработки OpenComRTOS сущности и экземпляры сущностей являются различными концепциями. Экземпляры сущностей OpenComRTOS рассматриваются в программном смысле и имеют собственный набор атрибутов и функций.
Т.о. экземпляры сущностей OpenComRTOS не имеют семантической значимости для концептуального моделирования OpenComRTOS при помощи OWL онтологий. Другими словами, более существенным для нашего исследования является hasSubclass (иметь подкласс), а не hasInstance (иметь экземпляр) отношение.

В OWL классы интерпретируются как содержащие элементы (экземпляры)

математические множества. Чтобы быть членом OWL класса, экземпляр должен соответствовать выраженному при помощи DL предикату. Такой подход используется нами для построения *валидных множеств экземпляров сущностей OpenComRTOS*, то есть множеств экземпляров, соответствующих определению (системной спецификации) сущностей OpenComRTOS в виде OWL-классов.

Т.о. определение сущности OpenComRTOS как OWL класса означает построение *множества* валидных (то есть соответствующих выраженным при помощи DL ограничений) экземпляров сущности. Применение логики первого порядка и операций теории множеств к таким содержащим экземпляры множествам и определяет выразительность предложенного подхода.

3. OWL Онтология OpenComRTOS

Выраженные как OWL классы сущности OpenComRTOS организованы в иерархию (таксономию) подкласс-суперкласс. Рисунок 1 отражает упрощенную онтологию сущностей OpenComRTOS с отношениями hasSubclass между классами и hasInstanse между классом и его экземплярами (например, ApplicationTask hasInstanse AppTask_1, Port hasInstanse Port_1). В данной таксономии как корень используется сущность OpenComRTOS (в OWL же традиционно все классы являются подклассами класса 'Вещь').

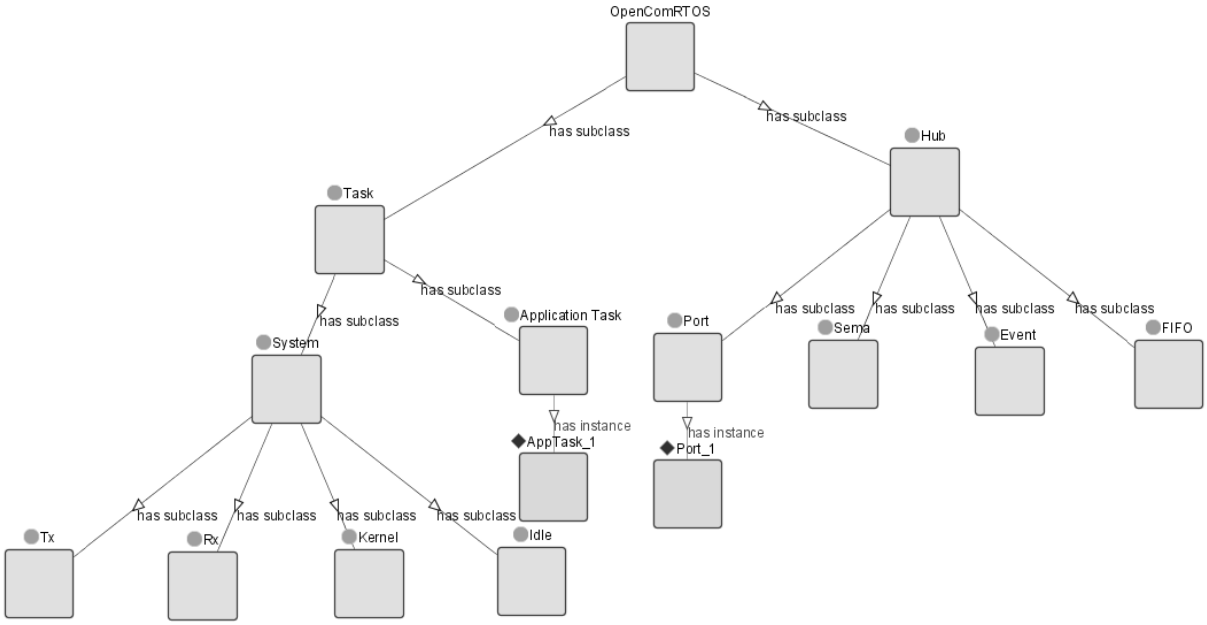

Рисунок 1. Онтология OpenComRTOS

Отношение hasSubclass в OWL подразумевает, что когда Port является подклассом сущности Hub, все члены множества Port также являются членами множества Hub. Аналогичное правило применимо и к другим OWL классам.

На основании данного отношения мы формулируем такое правило дедукции как *наследование свойств*: например, если все Hub имеют предикат синхронизации, а Port является Hub, то сущность Port также должна иметь свойство ‘предикат синхронизации’.

Классы OWL могут пересекаться. Например, рассмотрение задачи (Task) возможно как понятия, которое принадлежит одновременно к суперклассам системы (System) и приложения (Application) в онтологии OpenComRTOS.

Задача Task и сущность Hub в онтологии OpenComRTOS определены как являющиеся непересекающимся. Это означает, что любой экземпляр онтологии OpenComRTOS не может быть экземпляром более чем одного из этих классов. Следующий предикат выражает это отношение.

$$(\text{Entity} \in \text{Task}) \Rightarrow \neg\, (\text{Entity} \in \text{Hub}) \qquad (1)$$

Заметим, что данное предположение основывается на существовании только двух базовых классов OpenComRTOS: задачи **Task** и сущности синхронизации **Hub.**

Не все классы в онтологии OpenComRTOS могут иметь прямые экземпляры. Например, Hub является OWL суперклассом для Port, Event, FIFO и других сущностей синхронизации, которые могут иметь экземпляры в приложениях RTOS. Сам же Hub прямых экземпляров не имеет.

Иными словами, в онтологии OpenComRTOS класс может иметь экземпляры, если он не имеет подклассов.

$$\exists \text{ hasSubclass Entity} \Rightarrow \neg(\exists \text{ hasInstance Entity}) \quad (2)$$

Одной из ключевых особенностей OWL-DL является то, что подкласс-суперкласс отношения могут быть вычислены посредством систем автоматизации дедукции (САД), использующим DL логику. САД (к которым относится и FACT++ [3]) позволяет нам проверять соответствие понятий, вычислять выведенные иерархии и эквивалентные классы.

Заметим, что OWL онтология охватывает только структурные и статические свойства модели ПрО (в данном случае OpenComRTOS). Чтобы отразить динамическое поведение системы реального масштаба времени необходимо также определить классы для функциональных, временных и других свойств ПрО. Проверка таких свойств остается за пределами возможностей OWL-DL онтологий и связанных с ними САД.

4. Атрибуты сущностей OpenComRTOS и взаимодействия как OWL-свойства

Существуют два основных типа OWL свойств в которых мы выражаем атрибуты сущностей OpenComRTOS и их взаимодействия: свойство объекта (Object) и свойство ‘типы данных’ (Datatype).
Так как свойства объекта OWL связывают экземпляр класса с экземпляром класса, для атрибутов сущностей OpenComRTOS используется свойства Datatype, которые связывают экземпляры сущностей со XML схемой (например, атрибут PacketSize, имеющий некоторое значение целого типа).

Взаимодействие сущностей OpenComRTOS выражаются *свойствами объектов OWL*. Примерами взаимодействий в OpenComRTOS являются L1_PutPacketToPort_W (отослать пакет в порт) и L1_GetPacketFromPort_W (забрать пакет из порта), объединяющие сущности задачи (Task) и порта (Port).
Приставка L1_ отражает логический слой дизайна OpenComRTOS и не имеет семантического значения для построения OWL онтологии OpenComRTOS, поэтому далее для краткости мы будем использовать PutPacketToPort и GetPacketFromPort. Суффиксы _W, _NW, _WT отражают *временные* свойства взаимодействий (Wait - ждать, т.е. блокирующий сервис, Non Wait - не ждать, Wait with Timeout - ждать интервал времени), и не могут быть смоделированы в OWL-DL.

OWL свойства объектов связывают экземпляры из области определения с экземплярами из области значений. Областью определения PutPacketToPort является множество экземпляров класса ApplicationTask. Область значений - множество экземпляров класса Port. Области определения и значений свойств OWL используются нами как предикаты в рассуждениях, для проверки валидности приложений RTOS. Например, если мы определяем, что некоторый $class_1$ применяет свойство PutPacketToPort к некоторому $class_2$, использование САД доказывает, что $class_1$ является подклассом ApplicationTask, а $class_2$ является подклассом Port.

Свойства OWL могут иметь подсвойства, что дает возможность строить иерархии свойств. Подсвойства *специализируют* свои суперсвойства (например, свойство hasName может специализировать более общее свойство hasAttribute). Мы делим взаимодействия OpenComRTOS на сервисы управления задачами (например, StartTask и StopTask) и сервисы Hub (например, PutPacketToPort и GetPacketFromPort). Таким образом, в данный момент иерархия взаимодействий в онтологии OpenComRTOS построена соответственно семантике иерархии сущностей. Однако дальнейшая разработка онтологии OpenComRTOS, прежде всего онтологии решения возникающих задач, приведет к иному структурированию иерархии взаимодействий.

Каждое свойство класса OWL онтологии OpenComRTOS может иметь соответствующее обратное свойство. Например, свойство isPartOf имеет обратное свойство consistsOf (например, если “Topology consistsOf Nodes” простым правилом дедукции мы можем вывести что “Nodes isPartOf Topology”).

Из-за симметрии взаимодействий задач OpenComRTOS мы имеем семантически противоположные пары свойств, например, StartTask и StopTask или же PutPacketToPort и GetPacketFromPort. Однако заметим, что эта семантика взаимодействий не может быть смоделирована с использованием свойства инверсии OWL свойств, например из того положения, что $Task_1$ StartTask $Task_2$ не следует, что $Task_2$ StopTask $Task_1$. Принцип симметрии взаимодействий используется нами для верификации приложений OpenComRTOS, что будут показано в следующей главе.

5. Разработка онтологии приложений OpenComRTOS

OWL-ограничения используются нами для построения множества всех семантически

корректных (то есть соответствующих спецификациям) экземпляров класса приложений (Application) OpenComRTOS.
В приложениях OpenComRTOS порт (Port) является промежуточной сущностью, предназначенной для синхронизации взаимодействий задач (Tasks). Семантически правильное приложение OpenComRTOS подразумевает, что если $Task_1$ вызывает PutPacketToPort, то должна существовать $Task_2$, которая вызывает GetPacketFromPort (в предположении, что обе задачи использует один и тот же порт).
Выразим эту схему взаимодействия сущностей OpenComRTOS в форме предиката:

$$\exists \text{ PutPacketToPort } (Task_i, Port_k) \Rightarrow \exists \text{ GetPacketFromPort } (Task_j, Port_k) \quad (3)$$

Заметим, что экзистенциальные ограничения OWL могут определить *существование* (по крайней мере одного) отношения (например, PutPacketToPort) между двумя экземплярами (например, $Task_i$ и $Port_k$), но не *импликацию существования* соответствующего отношения (например, GetPacketFromPort между $Task_j$ и $Port_k$).

Формула (3) отражает фундаментальное свойство OpenComRTOS - симметрию взаимодействия задач. Свойство симметрии является одним из основных принципов дизайна OpenComRTOS и используется нами для проверки валидности приложений OpenComRTOS.

Какие же механизмы OWL могут быть использованы для моделирования данного принципа?

Заметим, что каждое OWL-DL ограничение определяет анонимный класс, который содержит экземпляры, удовлетворяющие данной логической формуле.
Например, следующее OWL ограничение для класса задачи:

$$\exists \text{ PutPacketToPort Port} \quad (4)$$

определяет анонимный класс, экземпляры которого являются членами класса задачи и участвуют во взаимодействии PutPacketToPort.
Другое OWL ограничение для класса задачи:

$$\exists \text{ GetPacketFromPort Port} \quad (5)$$

определяет анонимный класс, экземпляры которого являются членами класса задачи и участвуют во взаимодействии GetPacketFromPort.

Пересечение этих анонимных классов дает нам *суперкласс* задачи, элементы которого принимают участие в PutPacketToPort и GetPacketFromPort взаимодействиях.

$$(\exists \text{ PutPacketToPort Port}) \cap (\exists \text{ GetPacketFromPort Port}) \quad (6)$$

Такое OWL ограничение можно рассмотреть как спецификацию валидного (т.е. отражающего принцип симметрии) приложения OpenComRTOS, а его результат - как множество валидных экземпляров приложений OpenComRTOS.

Данный подход является конкретным примером применимости OWL онтологии для проектирования приложений OpenComRTOS и проверки их свойств.

Можно привести также другие примеры, иллюстрирующие применимость предложенного подхода. Например, для класса приложений ограничение

$$\exists \text{ hasFunction main} \quad (7)$$

описывает все экземпляры класса приложений, которые имеют (по крайней мере одну) главную функцию. Чтобы определить свойство, что экземпляр класса приложения должен иметь *одну и только одну* главную функцию, должно использоваться OWL ограничение на количество элементов. OWL ограничение количества элементов определяет точное количество отношений, в которых экземпляр класса может участвовать.

Валидное приложение OpenComRTOS всегда имеет главную функцию, таким образом, свойство (7) есть необходимое, т.е. является частью спецификации валидных приложений OpenComRTOS.

Заметим, что конкретизация свойства hasFunction - hasMainFunction является функциональным, то есть для данного экземпляра класса приложений может существовать только один экземпляр класса функций (главная функция), который связан с экземпляром приложения через свойство isMainFunctionOf.

6. Моделирование топологии вычислительных узлов

OpenComRTOS предоставляет сервисы в сети гетерогенных и распределенных вычислительных узлов. Поэтому мы используем единое (унифицированное) представление свойств каждого узла (Node) в определенной конфигурации (Parameters, Paths, Compiler Options и т.д.). Таким образом, разработка онтологии является задачей, предшествующей

задаче построения реальной топологии вычислительной сети (рисунок 2 является упрощенным примером онтологии вычислительных узлов OpenComRTOS).

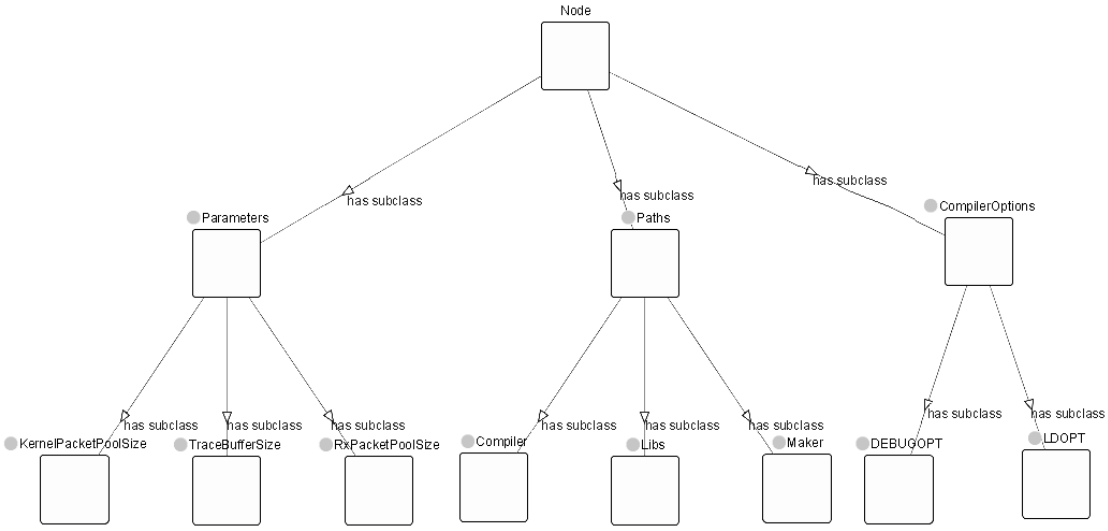

Рисунок 2. Онтология вычислительных узлов OpenComRTOS

В онтологии OpenComRTOS вычислительная сеть составлена из узлов (Node) и связей (Link), соединяющих узлы. Связи могут быть однонаправленными (Unidirectional) или двунаправленными (Bidirectional). Чтобы быть достижимым, каждый узел в данной топологии должен иметь, по крайней мере, один вход и один выход. Т.е. между любыми двумя узлами должен существовать, по крайней мере, один путь.

Поскольку однонаправленные, так же как и двунаправленные связи являются отношениями между двумя узлами, их можно рассмотреть как OWL бинарные свойства, например hasUniLinkWith (имеет однонаправленную связь с), и hasBiLinkWith (имеет двунаправленную связь с).
Двунаправленная связь является симметричной:

$$(\mathbf{Node_i\ hasBiLinkWith\ Node_j}) \Rightarrow (\mathbf{Node_j\ hasBiLinkWith\ Node_i}) \quad (8)$$

То есть, если $Node_i$ имеет связь с $Node_j$, из этого следует, что $Node_j$ имеет связь с $Node_i$ (см. рисунок 3). Другими словами, свойство hasBiLinkWith является собственным обратным свойством.

Рисунок 3. Симметрия двунаправленной связи

Двунаправленная связь также транзитивна:

$$(\mathbf{Node_i\ hasBiLinkWith\ Node_j}) \wedge (\mathbf{Node_j\ hasBiLinkWith\ Node_k}) \Rightarrow (\mathbf{Node_i\ hasBiLinkWith\ Node_k}) \quad (9)$$

Если $Node_i$ имеет связь с $Node_j$, и $Node_j$ имеет связь с $Node_k$, то $Node_i$ достижимо из $Node_k$, то есть между ними существует путь (см. рисунок 4). Заметим, мы не различаем здесь понятия (логической) связи и (физического) пути.

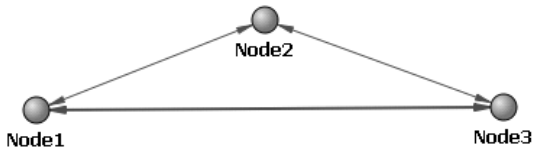

Рисунок 4. Транзитивность двунаправленной связи

С принятыми положениями мы можем строить модели валидных топологий, накладывая ограничения на OWL классы, например:

$$\mathbf{\exists\ hasBiLinkWith\ Node} \quad (10)$$

Выражение (10) определяет анонимный класс узлов, которые имеют, по крайней мере, одну двунаправленную связь с другим узлом. Такой класс OWL можно рассмотреть как модель некоторой реальной топологии.
Чтобы смоделировать то свойство, чтобы быть достижимым, каждый узел данной топологии должен иметь как входящую, так и исходящую связь, мы можем определить два новых свойства как подсвойства **hasUniLinkWith**: **hasInUniLinkWith** (имеет входящую однонаправленную связь) и **hasOutUniLinkWith** (имеет исходящую однонаправленную связь).

Пересечение таких OWL классов, которые имеют входящие и исходящие однонаправленные связи, дает нам анонимный класс, который соответствует спецификации:

$$(\exists \text{ hasInUniLinkWith Node}) \cap (\exists \text{ hasOutUniLinkWith Node}) \quad (11)$$

Принимая во внимание, что узлы могут иметь двунаправленные связи, мы можем расширить определение валидных (то есть соответствующих требованию наличия хотя бы одного пути между любыми двумя узлами) топологий следующим образом:

$$((\exists \text{ hasInUniLinkWith Node}) \cap (\exists \text{ hasOutUniLinkWith Node})) \cup (\exists \text{ hasBiLinkWith Node}) \quad (12)$$

Имея такие спецификации как, например (12), мы можем проверить валидность разрабатываемой топологии еще на стадии концептуального моделирования. Наши будущие исследования будут посвящены расширению множества свойств ПрО, которые могут быть смоделированы и проверены на этапе концептуального моделирования и построения онтологии. Одним из подходов является включение в онтологию ПрО временной логики для исследования и проверки динамических свойств систем.

Выводы

1. В работе проведен анализ соответствия между концепциями RDF, OWL и IE. Подобие этих подходов свидетельствует как о возможностях использования онтологий для архитектурного моделирования сложных систем, так и использования методологии IE в контексте идей Semantic Web.
2. Определение сущностей OpenComRTOS как OWL классов и использование OWL-ограничений предоставляет возможность построения валидных (т.е. соответствующих системным спецификациям) множеств экземпляров сущностей OpenComRTOS. Введенные положения также позволяют строить модели валидных топологий вычислительных узлов.
3. Проведен анализ взаимодействий сущностей OpenComRTOS (рассмотрены свойства симметричности, транзитивности, инверсии, функциональности). Формализация взаимодействий сущностей OpenComRTOS в виде OWL свойств дает возможность их использования в качестве предикатов в рассуждениях.
4. Показано, какие механизмы OWL могут быть использованы для моделирования базовых принципов OpenComRTOS. Например, свойство симметрии взаимодействия задач выражается как OWL-DL ограничение, а возникающий в результате его применения анонимный OWL суперкласс содержит множество валидных экземпляров приложений OpenComRTOS.
5. Практическая ценность разработки онтологии OpenComRTOS также состоит в формулировке соответствующих правил дедукции, что позволяет автоматизировать решение возникающих классов задач (например, моделирование приложений реального времени при помощи SDL-RT или же IE подходов).
6. Таким образом, построение онтологий мы рассматриваем не только как методику концептуального моделирования в различных ПрО, но также и как способ проверки свойств системы на ранних этапах ее разработки (начиная со спецификации требований). Расширение выразительности онтологий временными свойствами, модальной логикой и др. позволит объединить этап концептуального моделирования и стадию проверки динамических свойств системы.

Литература

1. Eric Verhulst, Gjalt de Jong, Vitaliy Mezhuyev. An Industrial Case: Pitfalls and Benefits of Applying Formal Methods to the Development of a Network-Centric RTOS. Lecture Notes in Computer Science. FM 2008: Formal Methods. - Springer Berlin / Heidelberg. - 2008. - P. 411-418.
2. OWL Web Ontology Language Guide. http://www.w3.org/TR/owl-guide/
3. http://owl.man.ac.uk/factplusplus/
4. Leslie Lamport. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers / Addison-Wesley. - 2002. http://research.microsoft.com/en-us/um/people/lamport/tla/book-02-08-08.pdf
5. Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. // Scientific American Magazine. - May, 2001. http://www.scientificamerican.com/
6. http://protege.stanford.edu
7. http://www.thechiselgroup.org/jambalaya
8. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/
9. SDL-RT standard V2.2. Specification & Description Language - Real Time. http://www.sdl-rt.org/standard/V2.2/pdf/SDL-RT.pdf
10. Hong Zhou; Jian Kang; Feng Chen; Hongji Yang OPTIMA: An Ontology-Based Platform-specific software Migration Approach // Quality Software, 2007. - 143–152 p.

Поступила в редколлегию 12.03.2009