УДК 004.89

В.И. Межуев

Одесский национальный политехнический университет, Одесса

ПРЕДМЕТНО-ОРИЕНТИРОВАННОЕ МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННЫХ ПАРАЛЛЕЛЬНЫХ ПРИЛОЖЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ

В статье предлагается метод предметно-ориентированного моделирования свойств распределенных параллельных приложений реального времени (на примере операционной системы OpenComRTOS). Сущность подхода состоит в определении основанной на теории графов метамодели и ее использовании для порождения моделей программных систем.

Ключевые слова: *предметно-ориентированное моделирование, моделирование программных систем, распределенные параллельные приложения реального времени.*

Введение

На сегодняшний день существует множество подходов к моделированию программных систем (ПС). Вообще говоря, моделирование становится необходимым этапом в разработке ПС, придя на смену классической (непосредственной) методике написания программного кода. Использование моделей для порождения программного кода является основной идеей MDE (Model-Driven Engineering) [1]. В общем случае для моделирования ПС используются как формальные текстовые языки (TLA, Promela, SDL и др.), так и различные диаграммные нотации (UML, IDEFX, MCS и др.). Моделирование программных систем предполагает отражение таких свойств ПС как структуры данных, последовательность выполнения операторов, время выполнения и др.

Среди преимуществ построения моделей ПС отметим:

– осуществление моделирования в терминах предметной области с абстракцией от деталей реализации;

– упрощение и ускорение разработки сложных программных систем ;

– автоматическая генерация программного кода;

– возможность валидации и верификации модели ПС;

– простота поддержки и сопровождения ПС и др.

На сегодняшний день разработано множество инструментальных сред, предназначенных для построения моделей ПС на одном из существующих формальных текстовых языков или же в одной из диаграммных нотаций. Заметим, что несмотря на мощность таких сред, разработка модели ПС осуществляется в фиксированной системе понятий, что ограничивает пользователя в описании семантики предметной области.

Иным направлением являются так называемые системы предметно-ориентированного моделирования – DSM (Domain Specific Modeling), позволяющие пользователю самому выбирать систему понятий и методов моделирования предметных областей в рамках определенной метамодели. Наиболее известным инструментом предметно-ориентированного моделирования на сегодняшний день является MetaEdit+ [2], осуществляющей построение моделей в рамках метамодели GOPPRR (Graph-Object-Property-Port-Role-Relationship) [3].

Соглашаясь с целесообразностью построения метамодели на основе графа, являющегося естественным способом структурирования иерархических систем, отметим ограниченность GOPPRR подхода, не использующей математические методы теории графов для решения задач предметной области.

Основная часть

Постановка задачи и метод ее решения

Предметом данной статьи является моделирование свойств распределенных параллельных приложений реального времени операционной системы OpenComRTOS в рамках метамодели G. Метамодель G основана на математической теории графов и имеет следующую структуру:

$$G \triangleq \{\{MT\},\{C\},\{R\}\}, \qquad (1)$$

где $\{MT\}$ – множество (мета)типов, соответствующих модельным объектам теории графов (узлы N и соединяющие их ребра E), и служащих для порождения типов модели ПС; $\{C\}$ – множество математических методов теории графов (правила обхода, поиск минимального пути и др.), а также правил манипуляции экземплярами $\{MT\}$ (перемещения, соединения узлов при помощи ребер и т.д.); $\{R\}$ – множество правил использования метамодели для построения моделей ПС.

Метамодель G используется для порождения множества моделей ПС

$$G \Rightarrow M_1, M_2 ... M_K. \qquad (2)$$

В рамках G структура модели ПС определяется как

$$M \triangleq \{\{T\},\{P\},\{L\}\}, \qquad (3)$$

где {T} – множество типов модели ПС, являющихся экземплярами {MT}; {P} – множество свойств, специфичных для предметной области (в нашем случае, программной системы); {L} – множество ограничений на модель ПС, вычислительных функций и правил генерации программного кода.

Использование методологии объектно-ориентированного дизайна

Структуру метамодели (1) и модели (3) ПС можно упростить, если использовать методологию объектно-ориентированного дизайна (ООД) [4].

В этом случае множество типов {MT} метамодели G необходимо определить как множество метаклассов, инкапсулирующих специфичные для них методы {C}.

Метаклассы {MT} являются базовыми для {T}, т.е. {T} наследуют свойства и методы {MT} (отношение *inheritance*).

Класс {T} расширяет {MT} специфичными свойствами и методами путем инкапсуляции (*encapsulation*) множеств {P} и {L} (в рамках структурного подхода данные множества рассматривается нами как внешние по отношению к {T}).

Входящие в {L} функции вычислений строятся на основе математических методов {C} метамодели (в нашем случае теории графов), но являются конкретными для предметной области (в нашем случае, программных систем).

Дальнейшие разделы статьи посвящены иллюстрации применимости метамодели G для построения моделей программных и аппаратных систем: M_{Top} - модели топологии вычислительных устройств; M_{App} - модели структуры приложения, распределенного в сети вычислительных устройств; M_{isc} – расширенной модели приложения M_{App}, включающей временные свойства и императивные конструкции.

Первые две модели были практически реализованы в рамках визуальной среды OpenVE [5], предназначенной для моделирования топологии вычислительных устройств и структуры распределенных на них параллельных приложений операционной системы реального времени OpenComRTOS [6; 7].

Построение модели топологии процессоров M_{Top} в рамках метамодели G

Как мы отмечали выше, типы {MT} метамодели G соответствуют модельным объектам теории графов – узлам N и соединяющим их ребрам E.

Построение модели топологии M_{Top} осуществляется в синтаксисе метамодели G при помощи совокупности правил {R}:

1. Выделение множества объектов *модели* топологии (а именно, вычислительных устройств – процессоров и аппаратных связей между ними).
2. Типизация объектов модели M_{Top} в рамках {MT}: процессора определяются как узлы, связи между процессорами как ребра графа.
3. Выделение множества свойств объектов модели (процессоров и аппаратных связей), их атрибутизация. Например, выделяются логические типы аппаратных связей (однонаправленные и двунаправленные).
4. Определение ограничений {L} на объекты модели (например, каждый процессор должен иметь как входящую, так и исходящую связь [8]).

Построение топологии вычислительных устройств в рамках модели M_{Top}

1. Порождение множества экземпляров вычислительных узлов и аппаратных связей.
2. Присвоение конкретных значений атрибутам созданных экземпляров.
3. Построение топологии путем визуального манипулирования созданными экземплярами (на рис. 1 приведен пример созданной в OpenVE кольцевой топологии вычислительных устройств, состоящей из Win32 и ARM типов узлов).

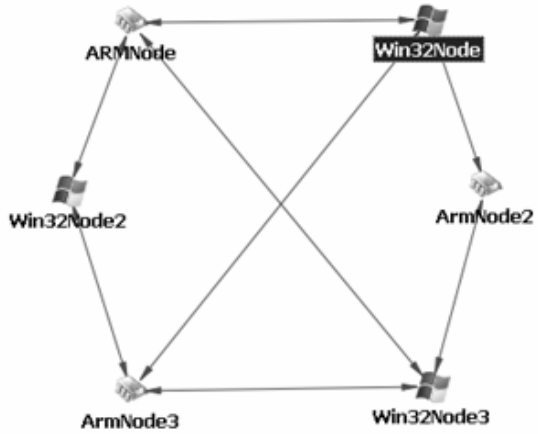

Рис. 1. Пример топологии вычислительных узлов

4. Применение основанных на теории графов методов вычислений (достижимости узла топологии, поиска минимального пути, распределения трафика в случае нескольких путей и др.).
5. Генерация программного кода (таблицы маршрутизации).

Семантику предметной области определяет именно модель. Практическим результатом построения модели топологии M_{Top} является построение (порождение программного кода) таблицы маршрутизации.

Таким образом, построение конкретной топологии вычислительных узлов осуществляется в синтаксисе метамодели G в рамках семантики модели

M_{Top}. Это означает, что множества типов {MT} и методов {C} (в рамках теории графов - обход, поиск минимального пути, распределение трафика и др.) метамодели G применимо ко всем порожденным из нее моделям.

Построение модели приложения M_{App} в рамках метамодели G

Основными понятиями модели M_{App} являются задача (*task*), сущность синхронизации (*hub*) и взаимодействие (*interaction*), соответствующие методологии OpenComRTOS [5 – 8]. Hub имеет следующие характерные для параллельного программирования типы: Port, Event, Semaphore, Resource, Memory Pool, FIFO.

Последовательность шагов для построения модели приложения M_{App} в рамках метамодели G является аналогичной приведенной выше для топологий. В M_{App} задачи и сущности синхронизации типизируются как узлы, взаимодействия между задачами через сущности синхронизации являются ребрами графа модели приложения. Специфичными для модели являются атрибуты и ограничения на модель, например, взаимодействие между задачами возможно только через сущности синхронизации. Ограничение на тип узлов, между которыми можно установить связь, фиксируется при помощи двух специальных атрибутов ребра взаимодействия:

- subject – субъект (инициатор) взаимодействия;
- object – объект взаимодействия.

Атрибуты *subject* и *object* являются ключевыми словами и имеют в M_{App} тип узла графа, таким образом, обеспечивая возможность установления связи между разрешенными узлами при визуальном построении модели приложения.

В практической реализации [5] мы используем XML для фиксации ограничения, например:

<interaction name="L1_PutPacketToPort_W" subject="task" object="port" />,

где L1_PutPacketToPort_W является именем взаимодействия (*interaction*) между задачей (*task*) и сущностью синхронизации (*port*). Имя взаимодействия используется генератором для построения кода приложения. Алгоритм генерации кода основан на обходе графа модели приложения (на рис. 2 приведен пример модели приложения OpenComRTOS).

M_{App} представляет модель ПС как упорядоченную последовательность взаимодействий между задачами через промежуточные объекты синхронизации. M_{App} выражает взаимодействие задач независимым от топологии процессоров способом. Иными словами, изменение структуры модели топологии не влияет на модель приложения и логику поведения задач.

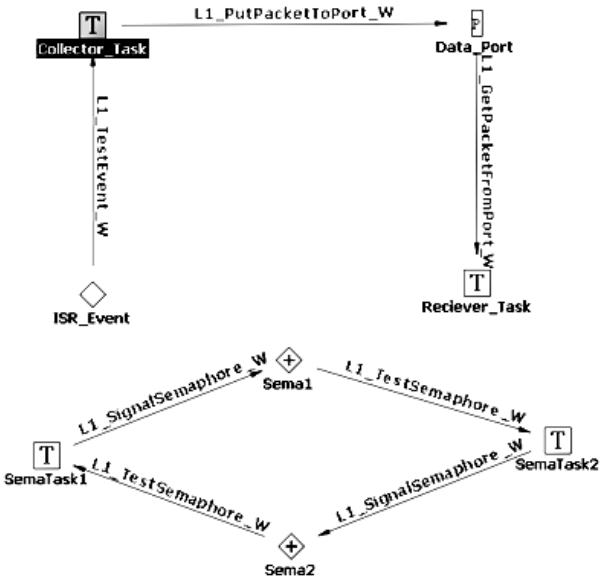

Рис. 2. Пример модели приложения OpenComRTOS

Моделирование способа распределения задач и сущностей синхронизации на вычислительных устройствах осуществляется путем определения специального атрибута, имеющего значение идентификатора процессора. Распределение задач и сущностей синхронизации в вычислительной сети осуществляется путем проецирования графа структуры приложения на граф топологии.

Поддержка модульности в M_{App}

Для поддержки структурного дизайна M_{App} предоставляет возможность группировки задач и сущностей синхронизации в едином компоненте или блоке. Такой блок имеет интерфейсы ко всем составляющим его структуру элементам. Например, при объединении в блок задач Task1 и Task2, взаимодействующих через Port1, блок предоставляет интерфейсы для управления задачами Task1 и Task2, а также синхронизации через Port1 (рис. 3).

Ребра взаимодействия могут быть связаны только с предопределенными *точками подключения* (интерфейсами) задач, сущностей синхронизации и компонент.

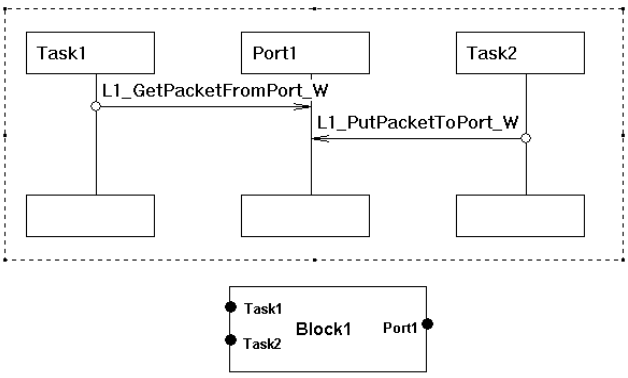

Рис. 3. Композиция задач и сущностей синхронизации в единый компонент (блок)

Таким образом, вводится понятие интерфейса, которое определяется нами как множество разрешенных для взаимодействия узлов.

Блок обозначается единым символом в M_{App} и позволяет создавать библиотеки компонентов. Определение отдельных компонент позволяет путем композиции определить структуру целого приложения. Генератор кода в этом случае преобразует в единое приложение код, полученный в результате обхода частичных графов (блоков).

Моделирование и проверка свойств приложений времени выполнения

Приведенная выше модель приложения M_{App} не позволяет определить структуру выполнения операторов между точками синхронизации задач, а также задать временные свойства приложения. Решение этой задачи требует включения в модель понятий императивного программирования (таких как линейная последовательность, условие, цикл), а также атрибутизации параметров времени выполнения.

Использование метамодели G позволяет строить различные диаграммные нотации для визуального моделирования программных систем в рамках представления структуры графа. Для моделирования структуры и временных свойств выполнения распределенных параллельных приложений операционной системы реального времени OpenComRTOS нами была предложена диаграмма последовательности взаимодействия задач - Interaction Sequence Chart (ISC).

ISC является расширением MSC (Message Sequence Chart) [9] - диаграммы последовательности сообщений и позволяет специфицировать как параллельное, так и последовательное поведение программной системы. Основное отличие ISC от диаграммы MSC состоит в использовании для разделения взаимодействующих задач сущности синхронизации (*hub)*. Иным отличием от MSC является то, что ISC поддерживает императивные конструкции программирования (линейные блоки, условия, циклы), что позволяет сгенерировать из ISC полный код приложения на языке программирования высокого уровня (обычно это C).

Одна из основных целей разработки ISC состоит в том, чтобы предоставить возможность проверки валидности модели программной системы (включая свойства времени выполнения) еще на этапе ее дизайна. Для этого из ISC генерируется формальная модель программной системы в синтаксисе MAST (Modeling and Analysis Suite for Real Time Applications) [10]. Благодаря тому, что из ISC также генерируется программный код, могут также использоваться статические валидаторы программного кода.

Иным способом валидации модели ПС является проверка соответствия между диаграммой ISC (созданной во время дизайна) и файлом событий задач OpenComRTOS, формирующегося во время их выполнения.

Применение различных методов валидации модели ПС гарантирует, что сгенерированный из ISC программный код формально правилен.

ISC нотация также может использоваться, когда системные спецификации не являются полными, например, на ранних этапах анализа в целях визуализации и документирования приложения. Таким образом, использование ISC осуществляет поддержку пользователя в течение полного цикла разработки программной системы.

M_{ISC} модель программной системы

В основе ISC модели ПС также лежат базовые понятия OpenComRTOS: задача (*task*), сущность синхронизации (*hub*) и взаимодействие (*interaction*). Модель ПС также находится в рамках основанной на теории графов метамодели *G*, предоставляющей способы построения, графическое представление и правила манипуляции базовыми понятиями модели.

Целью разработки ICS является определение временного сценария выполнения приложения, т.е. порядка взаимодействия между задачами в распределенной параллельной системе. Этот порядок определяется как операторами управления, так и временными ограничениями на процесс выполнения приложения.

Таким образом, ICS модель приложения OpenComRTOS строится из экземпляров задач и сущностей синхронизации, взаимодействий, операторов программирования и аннотаций, выражающих временные ограничения. Для реализации этих понятий модели вводятся дополнительные типы узлов, отражающие операторы программирования – блок (*block*), условие (*condition*), цикл (*loop*), а также дополнительный тип ребра графа – линия времени (*timeline)*, необходимая для отражения измерения времени.

Пример структуры модели приложения в ISC показан на рис. 4. Ребра графа, идущие вертикально вниз от задач, отражают измерение времени (*timeline*) выполнения приложения. Ребра графа, идущие вертикально вниз от сущностей синхронизации, называются линиями взаимодействия (*hub interaction line*). Они необходимы для установления точек синхронизации и не отражают процесс времени (иными словами, сущность синхронизации *hub* существует вне понятия времени).

Визуально задачи и сущности синхронизации представлены на ISC как прямоугольники (узлы графа), содержащие имя и другие атрибуты. Взаимодействия отражаются на ISC как горизонтальные стрелки (ребра графа), соединяющие *линию времени* (*timeline)* задачи и линию взаимодействия сущности синхронизации (*hub interaction line*).

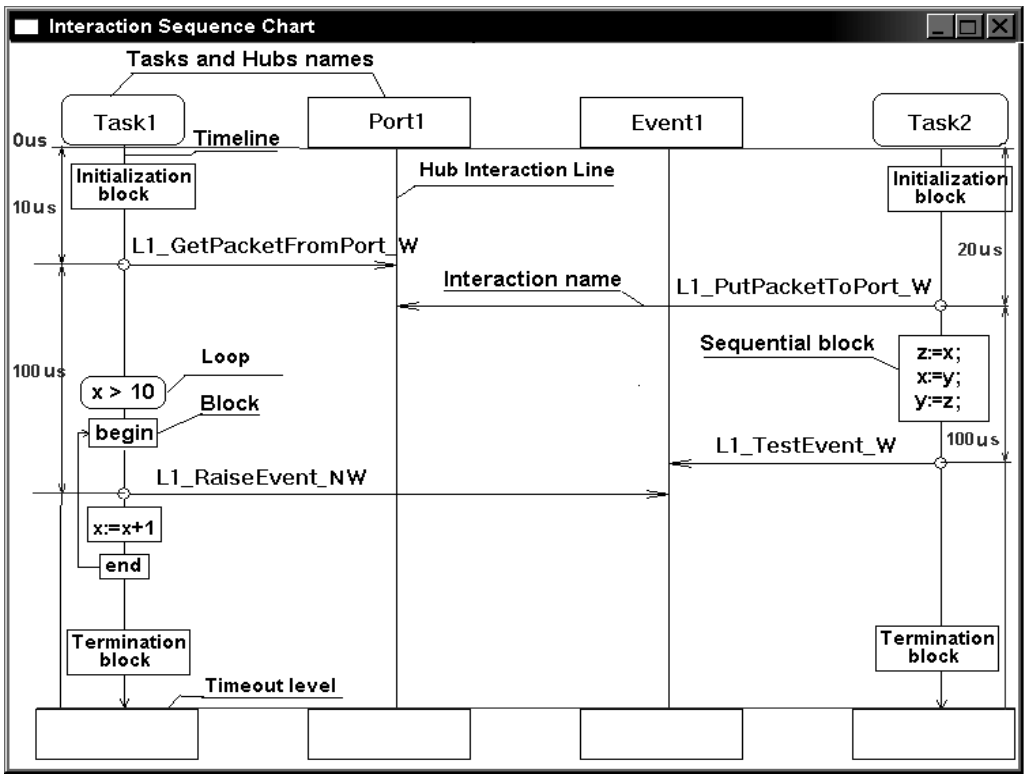

Рис. 4. Структура модели приложения в ISC

OpenComRTOS модель времени в ISC

Моделирование поведения приложений OpenComRTOS и генерация валидного программного кода требует определения в ISC модели времени.

ISC отражает логическое (определяющее относительный порядок следования взаимодействий) и абсолютное время (определяющее возникновение взаимодействия в момент времени, заданный значением глобальных часов).

Логическое время отражает причинно-следственный механизм взаимодействий задач.

Абсолютное (реальное, физическое) время является единым для всей системы и имеет монотонно возрастающее значение. Абсолютное время в OpenComRTOS синхронизируется с глобальным регулярным импульсом.

Логическое время используется для дизайна приложения и отражает частичный порядок взаимодействий между задачами в приложении OpenComRTOS. Последовательность значений времени между взаимодействиями задач формирует структуру синхронизации приложения OpenComRTOS и отражает его поведение во времени.

Модель логического времени тесно связанна с характеристиками взаимодействий, например, можно рассмотреть синхронное и асинхронное время.

На диаграмме ISC время отражается при помощи идущего вертикально вниз ребра *timeline*. Начало *timeline* задает нулевое значение в определении времени выполнения приложения.

ISC предоставляет возможность для визуального определения временных атрибутов поведения приложения. Например, расстояние между ребрами взаимодействия пропорционально времени между точками синхронизации задач. Такая информация используется при генерации кода, как параметра времени в сервисах OpenComRTOS. Генерация модели в синтаксисе MAST позволяет осуществить планирование приложения (так называемый *schedulability* анализ).

ISC конструкции программирования и генерация кода

ISC расширяет MSC, обеспечивая поддержку генерации кода посредством отражения следующих конструкций программирования:

1. Линейная последовательность операторов.
2. Блок операторов (например, блок завершения и инициализации).
3. Условный оператор (цикл рассматривается нами как частный случай условия).
4. Визуальный блок для определения глобальных и локальных переменных.
5. Комментарии.

Множество $\{L\}$ модели M_{isc} связывает понятия ISC с шаблонами кода языка программирования или моделирования. В качестве примера начального шаблона кода для задачи можно привести:

```
while (1) {
<тело цикла >
}
```

то есть бесконечный цикл выполнения задачи, что является традиционной практикой для проектирования встроенных (*embedded*) систем. (Естественно, пользователь в любое время может изменить определение шаблона кода). Дальнейший текст кода задачи генерируется из диаграммы ISC путем обхода графа выполнения в соответствии с *timeline*.

Дальнейшим этапом в разработке ISC является ее рассмотрение как дерева взаимодействий - Interaction Tree Chart (ITC). В этом случае вводятся дополнительные типы ребер графа, с которыми пользователь может определить взаимодействие задач. Таким ребром может быть ребро истинности условия, в зависимости от типа которого, рассматриваются как двоичные деревья, так и n-арные разветвления, соответствующие операторам-переключателям. Пользователь может определить взаимодействие с отдельной веткой дерева ITC.

Выводы

1. В статье предложена метамодель для построения моделей распределенных параллельных приложений реального времени (на примере операционной системы OpenComRTOS). Метамодель основана на теории графов и предоставляет возможность использования математических методов для моделирования программных систем.

2. Предложенный подход позволяет изменять модель путем добавления новых типов узлов и связей, находясь в рамках метамодели – теории графов, и тем самым создавать модели ПС для других платформ, ОС и АПИ.

3. Метамодель позволяет строить различные диаграммные нотации для визуального моделирования программных систем. Предложена диаграмма последовательности взаимодействия задач (ISC), позволяющая специфицировать поведение программных систем реального времени.

Список литературы

1. John D. Poole. Model-Driven Architecture: Vision, Standards and Emerging Technologies / Poole John D. // ECOOP Workshop on Metamodeling and Adaptive Object Models. – 2001.

2. Metacase.com [Электронный ресурс]. – Режим доступа к ресурсу: http://www.metacase.com.

3. Smolander Kari. MetaEdit: a flexible graphical environment for methodology modeling / Kari Smolander, Kalle Lyytinen, Veli-Pekka Tahvanainen, Pentti Marttiin // Proceedings of the third international conference on Advanced information systems engineering, May 1991, Trondheim, Norway. – P. 168-193.

4. Booch G. Object-Oriented Analysis and Design with Applications. Addison-Wesley / G. Booch. – 2007. - 608 p.

5. Mezhuyev Vitaliy. OpenComRTOS Visual Modelling Environment: the Tool for Distributed Parallel Applications Development / Vitaliy Mezhuyev, Eric Verhulst // Науковий вісник Чернівецького університету: Збірник наук. праць. Вип. 423: Фізика. Електроніка.: Тематичний випуск "Комп'ютерні системи та компоненти". Частина І. – Чернівці: ЧНУ, 2008. – С. 88-94.

6. Verhulst Eric. An Industrial Case: Pitfalls and Benefits of Applying Formal Methods to the Development of a Network-Centric RTOS / E. Verhulst, G. Jong, V. Mezhuyev // Lecture Notes in Computer Science. FM 2008: Formal Methods. – Heidelberg: Springer Berlin, 2008. – P. 411-418.

7. Verhulst Eric. OpenComRTOS: A Runtime Environment for Interacting Entities / E.Verhulst, V. Mezhuyev, Bernhard H.C. Sputh [et al.]; P. Welch, H. Roebbers, T. Announced (eds.) // Communicating Process Architectures 2009. – IOS Press. – 2009. – P. 173-184.

8. Межуев В.И. Моделирование свойств операционной системы реального времени OpenComRTOS при помощи OWL-DL онтологий / В.И. Межуев // Збірник наукових праць ДонНТУ серії "Інформатика, кібернетика та обчислювальна техніка". – 2009. – Вип. 10(153). - С. 39-46.

9. Message Sequence Chart (MSC). ITU-T Recommendation Z.120. [Электронный ресурс]. – Режим доступаа к ресурсу: http://www.itu.int/ITU-T/studygroups/com17/ languages/ Z120.pdf.

10. Michael González Harbour. MAST: A Timing Behaviour Model for Embedded Systems Design Processes / Michael González Harbour // MoCC - Models of Computation and Communication International Workshop, November 16-17, 2006, Zurich, Switzerland.

11. Serrano J.A. VCt – A Formal Language for the Specification of Diagrammatic Modelling Techniques / J.A. Serrano, R. Welland // Information and Software Technology, 1998. – 40(9). – P. 463-474.

Поступила в редколлегию 14.04.2010

Рецензент: д-р физ.-мат. наук, проф. В.В. Кидалов, Бердянский государственный педагогический университет, Бердянск.

ПРЕДМЕТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ РОЗПОДІЛЕНИХ ПАРАЛЕЛЬНИХ ДОДАТКІВ РЕАЛЬНОГО ЧАСУ

В.І. Межуєв

У статті пропонується метод предметно-орієнтованого моделювання властивостей розподілених паралельних додатків реального часу (на прикладі операційної системи OpenComRTOS). Сутність підходу полягає у визначенні основаної на теорії графів метамоделі та її використанні для породження моделей програмних систем.

Ключові слова: *наочно-орієнтоване моделювання, моделювання програмних систем, розподілені паралельні додатки реального часу.*

DOMAIN-SPECIFIC MODELLING THE DISTRIBUTED PARALLEL REAL TIME APPLICATIONS

V.I. Mezhuyev

The method of domain-specific modelling properties of the real time distributed parallel applications (on example of OpenComRTOS operation system) is considered in the paper. The essence of the approach is in the definition of the based on the graph theory metamodel and its using for generation of models of program systems.

Keywords: *in-oriented design, design of the programmatic systems, up-diffused parallel appendixes of the real time.*