УДК 004.4

В.И. МЕЖУЕВ
Бердянский государственный педагогический университет

ПРИНЦИПЫ СОЗДАНИЯ И ПРИМЕНЕНИЯ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ МЕТОДОЛОГИЙ

В статье рассматриваются принципы моделирования и применения предметно-ориентированных методологий (Domain Specific Methodologies – DSMd). Предлагаемый подход позволяет строить методологии, наиболее полно соответствующие запросам рассматриваемой предметной области (предприятия, проекта, технологии и т.д.). Применимость подхода доказывается путем его применения для порождения таких известных методологий как ERD, SSA, OOD и других.

The principles of modelling and application of Domain Specific Methodologies (DSMd) are considered in the paper. The offered approach allows to develop methodologies, as full as possible corresponding to requirements of considered domain (enterprise, project, technology etc.). Applicability of the approach is proved by its application for producing such well-known methodologies as ERD, SSA, OOD and others.

Ключевые слова: предметно-ориентированное моделирование, предметно-ориентированные методологии.

Введение. В области информационных технологий (ИТ) в настоящее время разработано множество различных методологий моделирования и проектирования компьютерных систем. В конце прошлого столетия даже возникло понятие *войны методов*. Причиной является тот факт, что разработчики новых методологий моделирования игнорировали лежащие в их основе общие подходы. Именно поэтому появление новой методологии встречается с критическими взглядами как со стороны теоретических исследователей, так и практических разработчиков в области ИТ.

Идеи предлагаемого в статье подхода схожи с так называемой компьютерно-ориентированной инженерией методов (Computer Aided Method Engineering, CAME) [1; 2], которую можно рассмотреть по аналогии с компьютерно-ориентированной программной инженерией (Computer Aided Software Engineering, CASE). Как CASE облегчает процесс разработки программных систем, CAME осуществляет поддержку деятельности по разработке методов, технологических процессов, технологий и др. В то же самое время, CAME подход основан на ситуативной инженерии методов (Situational Method Engineering, SME) [3], чья главная цель – разработка специфичных для проекта методологий.

Объектом статьи являются принципы моделирования и применения предметно-ориентированных методологий (Domain Specific Methodologies, DSMd). Предложенный подход основан на методологии предметно-ориентированного моделирования (Domain Specific Modelling, DSM) [4; 5].

Идея DSMd состоит в определении специального механизма, который позволяет пользователю создавать не только *модели* ПрО (как в DSM), но также и *предметно-ориентированные методологии*. Таким образом, DSMd включает в себя DSM как частный случай.

Особенностью DSM является разработка моделей не в технических терминах данных и алгоритмов, а в содержательных понятиях предметной области (ПрО). Эти понятия порождаются из т.н. *метамодели* и служат предметно-ориентированным языком (Domain Specific Language, DSL) для разработки моделей ПрО. Заметим также, что методология DSM предназначена главным образом для разработки программных систем, в то время как мы не ограничиваем область применимости DSMd подхода.

Методология DSMd основана на определении двух метамоделей: первая служит для порождения предметно-ориентированных моделей, вторая – для построения предметно-ориентированных методологий (моделирования, проектирования и др.). Для разграничения этих метамоделей, мы будем называть первую *онтологической* (как модель, так и метамодель).

Заметим, что в философии онтология определяется как учение о бытии [6]. Иное (и более узкое) определение онтологии дается в Semantic Web, где оно означает формальную (обычно, основанную на графах) спецификацию концептуализации [7].

DSM и DSMd подходы весьма схожи с онтологическим моделированием, как оно имеет место в Semantic Web. Они основаны на концептуализации ПрО, то есть определении важнейших понятий ПрО и формализации отношений между ними. Различие состоит в том, что онтологическая модель в Semantic Web является полностью описательной. Для формализации отношений онтологии применяют логику первого порядка, а точнее ее DL (Description Logic) подмножество. Данный подход позволяет доказать некоторые свойства онтологической модели ПрО, однако не дает возможности определить методологию решения специфичных для предметной области задач.

Смысл DSMd состоит в определении *императивной методологии*, которая может быть применена к построенной ранее онтологической модели ПрО. Т.о. DSMd предоставляет пользователю поддержку в решении специфичных для ПрО задач.

В то же самое время, декларативный подход к моделированию ПрО остается важным частным случаем DSMd. Так, используя DSMd, мы можем породить методологию моделирования Semantic Web как частный случай. При этом могут быть проверены различные свойства онтологических моделей ПрО, например, валидность иерархии концептов онтологии, связанных отношением Is-A. С этой целью мы

используем существующие инструменты автоматизации умозаключений (reasoners), созданные для доказательства свойств онтологий Semantic Web (например, Fact++ [8]).

Следующий раздел статьи посвящен принципам создания и применения DSMd. Дальнейшие разделы доказывают применимость DSMd путем порождения существующих подходов и методологий моделирования. Завершают статью выводы и планы будущих исследований.

Принципы моделирования методологий. **Моделирование любых ПрО (в том числе и методологий) всегда имеет в основе некоторые общие принципы, среди которых выделим:**

абстрагирование, то есть выделение существенных свойств ПрО (в отвлечении от несущественных);

декомпозиция, то есть разделение сложных ПрО на отдельные элементы, свойства которых могут быть поняты и смоделированы;

структурирование, то есть связывание отдельных частей, полученных в результате декомпозиции, в определенные структуры;

формализация, то есть математически точное определение свойств и отношений объектов ПрО.

Заметим, что большинство методологий моделирования, проектирования и т.д. в области ИТ являются слабо или же вообще не формализованными. Например, объектно-ориентированная методология (ООМ), описанная в известной книге [9], вообще говоря, является множеством текстовых рекомендаций, которые весьма далеки от формальной техники проектирования систем.

В то же самое время заметим, чтобы породить методологию (например, ООМ), ее принципы должны быть ясно поняты и формализованы. Именно поэтому мы выделяем формализацию как один из базовых принципов нашего подхода. Имея целью доказательство применимости DSMd путем порождения существующих методологий, первым этапом всегда является понимание и формализация принципов этих методологий.

Следуя DSM, мы рассматриваем три уровня абстрагирования от ПрО:

- мета-метамодель (сокращено, ММММ или M^3) ПрО;
- метамодель (ММ или M^2) ПрО;
- модель (М) ПрО.

Основное направление приложения DSMd мы можем определить как *переход от абстрактного к конкретному*, то есть от мета-метамодели к метамодели, от метамодели к конкретной модели ПрО.

Абстрактные понятия уровня M^3 служат базовыми типами для порождения понятий M^2 (что близко к *наследованию*, как оно понимается в ООМ). В то же самое время, понятия M^2 являются типами для создания конкретных *экземпляров* объектов, составляющих модель ПрО. Основная идея DSMd (как и DSM) состоит в предоставлении возможности пользователю разработки метамодели, в понятиях которой будет осуществляться декомпозиция объектов модели ПрО.

Таким образом, *типизация, наследование* и *инстанциация* в DSMd являются методами реализации общенаучных принципов абстрагирования и декомпозиции.

Заметим, что традиционно *типизацию* рассматривают как классификацию объектов при переходе от единичного к общему (с последующей возможностью создавать копии этих типичных объектов). В DSM типы производятся из еще более абстрактных метатипов, таким образом, сохраняется общее направление от абстрактного к конкретному.

Инстанциация – механизм создания копий объекта одного и того же типа – является основным методом порождения модели из метамодели. После определения метамодели, мы рассматриваем ее понятия как типы для создания *экземпляров,* имеющих конкретные значения атрибутов.

На уровне метамодели типы можно рассмотреть как экземпляры мета-метамодели, к которым добавляются специфичные для предметной области атрибуты. Назовем этот метод *атрибутизацией* для разграничения с определенным в ООД термином *наследование*.

Описанные выше принципы не являются принципиально новыми и используются в уже существующих DSM инструментах (например, MetaEdit+ [10]). Мы можем также показать параллель с системами управления содержанием (Content Management Systems, CMS), где типы контента используются для порождения конкретных экземпляров содержания. Некоторые из CMS (например, Plone [11]), даже позволяют редактировать концептуальную схему ПрО, и, таким образом, создавать новые типы контента путем атрибутизации базовых понятий (составляющих концептуальную схему).

Отличие DSMd от CMS состоит в возможности *моделирования структуры ПрО*. Предложенная метамодель является не только набором абстрактных понятий (служащих типами для порождения экземпляров), она также определяет *способы отношений экземпляров*. Например, экземпляры понятий "система" и “подсистема”, определенных в метамодели, могут быть связаны на уровне модели только отношением “Часть-Целое”. Таким образом, принцип *структурирования* осуществляется путем наложения ограничений на возможные отношения экземпляров метамодели.

Определение мета-метамодели и метамодели онтологии. Как было упомянуто выше, идея DSMd состоит в формальном определении базовых понятий. Мы определяем мета-метамодель как формализм, который позволяет породить метамодели ПрО. В то же самое время метамодель – это формализм, позволяющий породить модели ПрО.

В этой статье мы будем рассматривать мета-метамодель и метамодели, основанные на

математической теории графов.

Заметим, что использование графа, как структурной части мета-метамодели не является новой идей. Мы можем сослаться, например, на мета-метамодель GOPHR, используемую в MetaEdit+ [10]. GOPRR есть аббревиатура от Graph, Object, Property, Role and Relationship (Граф, Объект, Свойство, Роль и Отношение).

Однако в DSMd мы используем графы не только для иерархического структурирования объектов ПрО, но и для построения методов решения возникающих в ПрО задач. Другими словами, мы используем методы теории графов как метамодель для порождения методологий ПрО.

Граф G определяется как упорядоченная пара вершин N и ребер E:

$$G = (N, E), \tag{1}$$

где N – есть множество вершин; E – множество ребер, являющихся подмножеством из двух элементов множества N (то есть ребро графа представляется двумя вершинами графа).

Таким образом, мета-метамодель M^3 онтологии мы определяем как граф G.

$$M^3 \stackrel{\Delta}{=} G. \tag{2}$$

Граф M^3 состоит из вершин и ребер, служащих базовыми типами для порождения типов метамодели M^2.

Переход от уровня мета-метамодели к метамодели осуществляется путем добавления к вершинам и ребрам графа специфичных для ПрО атрибутов.

Таким образом, метамодель M^2 может быть определена как атрибутизированный граф*:*

$$M^2 \stackrel{\Delta}{=} G_{attr}. \tag{3}$$

Каждая вершина n графа содержит множество атрибутов:

$$A^n = \{A_1, A_2, \ldots, A_K\}, \tag{4}$$

где K является общим количеством атрибутов вершины n графа.

Для обозначения принадлежности атрибута к вершине или ребру графа, мы будем использовать верхние индексы, например. A^n_k – k-й атрибут n-й вершины.

Граф M^2 определяет структуру модели, то есть возможный способ соединения экземпляров вершин посредством экземпляров ребер. Реализация данного принципа на уровне метамодели означает введение двух специальных атрибутов ребра: *субъекта* и *объекта* отношения, определяющих возможные типы вершин, которые может соединять данное ребро.

Модель M является графом, содержащим экземпляры вершин и ребер M^2*:*

$$M = (N_{inst}, E_{inst}). \tag{5}$$

На уровне модели атрибут определяется как множество из трех элементов: имя, тип, значение (Name, Type, Value).

$$A = \{N, T, V\}. \tag{6}$$

Например, атрибут "скорость" объекта автомобиль определяется как («скорость», «целое», «100»).

Обязательным атрибутом всех объектов является имя. Этот атрибут необходим для однозначной *идентификации* объекта. Идентификация применяется для получения значений атрибутов объекта, например, может быть использована импликация:

(Entity.Name=Car) → (Car.Speed.Value=100)

Значение имени объекта не должно меняться в течение жизненного цикла модели.

Заметим, что в рамках теории множеств можно определить некоторые другие важные понятия ПрО. Например, *состояние* можно рассмотреть как множество всех значений атрибутов объектов модели. Последовательность состояний определяет *поведение* модели и т.д.

Поскольку основанная на теории графов мета-метамодель может быть произведена из теории множеств, последняя может быть определена как мета-мета-метамодель ПрО. Однако данные рассуждения выходят за пределы DSMd подхода. Следуя DSMd, мы ограничиваем наше обсуждение тремя уровнями абстрагирования.

Заметим, что являются возможными и другие определения модели, метамодели и мета-метамодели, например, мы можем породить метамодель с такой базовой абстракцией, как вектор. Свойства таких метамоделей будут исследованы в наших следующих статьях.

Моделирование методологии. Методология определяется нами как упорядоченный набор методов, которые могут быть применены к онтологической модели с целью решения задач рассматриваемой ПрО. Вообще говоря, метамодель методологии – это любой формализм, который является частью ядра математической теории, используемой в качестве метамодели онтологической модели ПрО.

Основываясь на определении метамодели онтологии как графа (2), методология может быть определена как множество методов обработки и преобразования графов онтологий.

В пределах данного определения мы можем дать следующую классификацию методов:

- наложение ограничений на структуру графа (например, условий добавления или удаления вершин и ребер графа);
- определение различных маршрутов обхода графа, обладающих заданными свойствами (например, минимального пути);
- выделение деревьев в графах;
- отображение различных графов онтологий и т.д.

Построенная таким образом методология использовалась нами для решения задач над онтологическими моделями, отражающими структуру программных систем [12]. В данном подходе методология определяла структуру решения задачи, что, вообще говоря, может использоваться в случае достаточной формализованности ПрО. Однако, как было замечено нами выше, существующие методологии моделирования весьма далеки от математически точного определения.

Для таких ПрО целесообразность DSMd подхода состоит в порождении графических нотаций, а также формулировке правил, накладывающих ограничения на процесс разработки системы. Следующие разделы статьи иллюстрируют данные подходы.

Использование DSMd для порождения графических нотаций. В пределах системной инженерии существует несколько методологий и соответствующих им графических нотаций: диаграммы сущность-связь (Entity-Relationship Diagrams, ERD) [13; 14], диаграммы функционального моделирования (Structured Analysis and Design Technique, SADT) [15], диаграммы потоков данных (Data-Flows Diagrams, DFD) [16], ООМ [9] с UML [17] и т.д.

Заметим, что использование методологии сводится иногда только лишь к применению соответствующей графической нотации. В этом случае порождение методологии как DSMd означает разработку соответствующей графической системы обозначений.

Фактически, все существующие в системной инженерии графические нотации могут быть произведены на основе графов. Это является следствием того факта, что граф позволяет осуществить декомпозицию ПрО наиболее привычным для человека способом: в объекты (вершины) и их зависимости (ребра). Иным преимуществом является возможность выделения иерархических древоподобных структур и др.

Таким образом, задача порождения ERD как DSMd не является сложной. Фактически, ERD нотация один к одному соответствует основанному на теории графов подходу DSMd. В ERD сущность обладает множеством атрибутов, включающим также его уникальное имя (идентификатор). Зависимость определяется как соотношение или ассоциация между отдельными сущностями ERD.

Для порождения ERD как DSMd, мы определяем ERD-сущность как вершину графа, а ERD-отношение как ребро (что собственно и составляет метамодель ERD). DSMd предоставляет возможность добавления специфичных для ПрО атрибутов сущностей. Для графического обозначения вершин и ребер диаграмм ER используется библиотека графических символов.

Однако заметим, что применимость ERD подхода состоит главным образом в проектировании баз данных [13; 14]. DSMd может быть использован для порождения более общего ООМ подхода.

ООМ рассматривает ПрО (или будущую систему) как множество взаимосвязанных объектов. Объект определяется как часть действительности, имеющая определенные свойства и поведение.

Диаграммы ER также используются в контексте ООМ. Однако самой популярной является UML [17] нотация, имеющая 14 типов диаграмм (для UML v.2.3). Все эти типы UML диаграмм могут быть порождены как DSMd. Их отличия состоят только в графических обозначениях и атрибутах вершин и ребер.

Приведем как пример диаграмму классов UML (рис. 1):

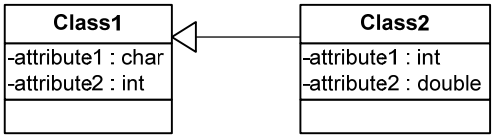

Рис. 1. Пример UML диаграммы классов

Однако DSMd подход позволяет не только породить графические нотации как то ERD или же UML, но и определить процесс моделирования ПрО. Следующие разделы статьи объясняют этот подход.

Моделирование методологии разработки систем. В наших предыдущих статьях (см., например, [21]) была определена методология разработки систем (Systems Development Methodology, SDM). Рассмотрим порождение SDM как DSMd. Базовыми понятиями SDM являются требование, спецификация, задача разработки, задача тестирования, задача валидации, задача верификации и др. Определения этих понятий были даны в [21].

Эти понятия имеют атрибуты и связаны отношениями, иными словами, теория графов также может использоваться в качестве метамодели SDM. Методы теории графов используются нами для выделения различных деревьев в графах онтологий построенных при помощи SDM моделей ПрО (например, зависимостей понятий).

Методологию SDM мы определяем как *формализм, управляющий процессом разработки системы*. Вообще говоря, методология SDM состоит в создании графа онтологической модели будущей системы и определении различных способов его обхода. Процесс построения онтологической модели есть создание экземпляров (*инстациация*) абстрактных понятий метамодели (т.е. требований, спецификаций, задач выполнения и т.д.) и связывание их отношениями, специфичными для ПрО.

Порядок процесса разработки системы определяется логическими условиями, положенными на величины атрибутов экземпляров понятий. Например, все понятия SDM имеют атрибут статус (*status*) с набором возможных значений «в работе» *(«in work»),* «одобрено» *(«approved»),* «не применимо» *(«not applicable»*). Связь между понятиями требования (*requirement*) и спецификации (*specification*) есть

логическая импликация:

$$\forall((\mathrm{Re}\,quirement.Status = Approved) \vee (\mathrm{Re}\,quirement.Status = Not\,applicable))$$
$$\rightarrow Specification.Status = Approved$$

То есть спецификация может быть одобрена, если соответствующие требования были одобрены (или же не применимы). Таким образом, переход к следующему уровню определения системы (уровень спецификаций) возможен только после одобрения предыдущего (уровень требований).

Такой подход есть не что иное, как определение возможных состояний онтологической модели системы. Иными словами, модель будущей системы мы рассматриваем как конечный автомат. Другая особенность DSMd – связывание декларативных условий с императивными действиями. Если формула истина, вызывается соответствующий условию метод (например, устанавливающий значение атрибута Status в "Approved").

Порождение методологии структурного моделирования. Рассмотрим применимость DSMd подхода на примере порождения методологии структурного моделирования.

Диаграммы структурного моделирования были созданы в середине 1960-х, когда Дуглас Т.Росс предложил специальную технику моделирования SADT (Structured Analysis and Design Technique). Позже SADT стала основой для методологии IDEF [18]. Далее в 1981 был создан стандарт IDEF0 (Icam DEFinition) в рамках программы интеграции компьютерных и индустриальных технологий ICAM (Integrated Computer Aided Manufacturing) [19].

Методология IDEF-SADT предоставляет набор методов, принципов и процедур для разработки функциональных моделей систем в любой ПрО. Функциональная модель SADT отражает структуру процессов как системы, так и ее отдельных подсистем. IDEF рассматривает систему как множество функций, которые преобразовывают входные в выходные данные.

Для создания методологии IDEF-SADT каждый элемент онтологической модели рассматривается нами как процесс. Процесс принимает участие (или используется) в выполнении некоторой функции или операции (преобразования, обработки, формирования и т.д.).

Функция является логической композицией операций (действий), которые преобразуют вход в выход. Последовательность взаимосвязанных входов и выходов функций составляет процесс. Функции могут также генерировать объекты любой природы. Вообще говоря, функции могут выполняться в любой последовательности (например, циклический). Порядок выполнения функций в нашем подходе имеет структуру графа. Такой порядок зависит от различных условий, как, например, события, происходящие в окружающей среде.

События вызывают выполнение функций, которые, в свою очередь, изменяют условия и формируют новые события, пока процесс не будет завершен. Таким образом, фиксация последовательности событий есть также один из способов описания выполнения процесса.

Структурный системный анализ (SSA) [20] есть метод моделирования систем, который начинается с наиболее общего описания с последующей детализацией отдельных аспектов свойств и поведения системы.

Ограничения SSA становятся видимыми, когда необходимо детализировать представление модели системы, например, кроме статических отношений показать поведение или же функционирование отдельных узлов системы. Для этих целей, в пределах SSA используются диаграммы потоков данных DFD.

Для порождения структурной методологии сначала представим в форме графов наиболее общие и абстрактные функции модели системы. Эти онтологии должны иметь ограниченное число узлов на каждом уровне иерархии, отражающие только наиболее существенные элементы системы.

Специфической особенностью порождения структурного подхода является декомпозиция системы в *функциональные* подсистемы. Эта декомпозиция итеративно применяется к подсистемам, пока они не становятся достаточно простыми, чтобы быть смоделированными. Такие подсистемы есть вершины графов предложенной онтологической модели.

В этом случае модель сохраняет целостность, в которой все подсистемы (вершины) связаны ребрами, отражающими отношения ПрО. Заметим, что при разработке системы снизу вверх, то есть от отдельных элементов к целой системе, ее целостность может быть потеряна.

Вместе с декомпозицией системы в вершины (подсистемы) и ребра (функциональные зависимости) используется метод иерархического упорядочивания. Это означает упорядочивание элементов системы в древоподобные структуры с последовательной детализацией на каждом уровне иерархии.

Вообще говоря, система декомпозируется в множество подсистем, представленных в форме "черных ящиков" с множеством входов и выходов. Такая декомпозиция со следующей организацией этих подсистем в иерархические древовидные структуры есть основной метод методологии структурного моделирования.

Кроме того, моделирование SSA требует разработки следующих диаграмм:

- *объектной структуры,* отражающей архитектуру системы;
- *функциональной структуры,* отражающей действия и взаимодействия объектов;
- *управляющей структуры,* отражающей поведение системы.

Обратим также внимание на основное различие между структурной и объектной методологией, которое состоит в способе объединения функций и данных. В ООМ данные и методы представляют единство (т.н. принцип *инкапсуляции*). Однако, структурная и объектная методология не являются

семантически противоположными и могут быть объединены в единую методологию при помощи DSMd. Эту задачу мы оставляем для наших будущих исследований.

Выводы. В статье рассмотрены принципы разработки и применения предметно-ориентированных методологий (DSMd). Целесообразность DSMd подхода состоит в теоретической и практической возможности разработки методологий, наиболее полно соответствующих запросам ПрО (предприятия, проекта, технологии и др.).

DSMd подход основан на методологии предметно-ориентированного моделирования (DSM). Отличие состоит в определении специального механизма (*метамодели*), который позволяет пользователю создавать не только *модели* ПрО (как в DSM), но также и предметно-ориентированные методологии.

Метамодель методологии – формализм, являющийся частью ядра математической теории, используемой в качестве метамодели для построения онтологической модели ПрО. Рассмотрена возможность применения теории графов в качестве метамодели для порождения методологий ПрО.

Наши будущие исследования будут посвящены дальнейшей разработке и формализации метамоделей методологий ПрО. Это позволит расширить классы задач, решаемых в пределах DSMd. Отдельно, предметом наших исследований будет разработка методологии стандартизации систем.

Литература

1. Ajantha Dahanayake. Computer-Aided Method Engineering: Designing CASE Repositories for the 21st Century. – IGI Global. – 2001. – 252 p.

2. Ali Niknafs and Raman Ramsin. Computer-Aided Method Engineering: An Analysis of Existing Environments // LNCS. – Springer. – Volume 5074. – 2008. – 525-540 pp.

3. Jolita Ralyte and Sjaak Brinkkemper. Situational Method Engineering: Fundamentals and Experiences: Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland. – Springer US. – 380 p.

4. Steven Kelly and Juha-Pekka Tolvanen. Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Pr. – 2008. – 427 p.

5. Richard C. Gronback. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional. – 2009. – 736 p.

6. http://en.wikipedia.org/wiki/Ontology

7. Gruber T.R. A translation approach to portable ontologies. Knowledge Acquisition, 5 (2). 1993. – pp. 199-220.

8. http://owl.man.ac.uk/factplusplus/

9. Grady Booch, Robert A. Maksimchuk, Michael W. Engel, and Bobbi J. Young. Object-Oriented Analysis and Design with Applications (3rd Edition). – Addison-Wesley Professional. – 2007. – 720 p.

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

11. http://plone.org/

12. Межуев В. И. Предметно-ориентированное моделирование распределенных параллельных приложений реального времени В. И. Межуев // Системи обробки інформації. – 2010. – Вип. 5 (86). – С. 98 – 103.

13. Sikha Bagui and Richard Earp. Database Design Using Entity-Relationship Diagrams (Foundations of Database Design). Auerbach Publications. – 2003. – 264 p.

14. Thalheim B. Entity-Relationship Modeling: Foundations of Database Technology. – Springer Berlin Heidelberg. – 2010. – 640 p.

15. David A. Marca and Clement L. McGowan. IDEF0 and SADT: A Modeler's Guide. – OpenProcess, Inc. – 2005. – 392 p.

16. DiCerto J. J. Planning and preparing data-flow diagrams. Hayden Book Co. – 1964. – 90 p.

17. Russ Miles and Kim Hamilton. Learning UML 2.0. – O'Reilly Media. – 2006. – 288 p.

18. http://www.idef.com/

19. http://www.icam.com

20. Introduction to Methodologies and SSADM. http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html

21. Vitaliy Mezhuyev, Eric Verhulst, and Bernhard H.C. Sputh. Interacting Entities Modelling Methodology for Robust Systems Design // Papers of the Second International Conference on Advances in System Testing and Validation Lifecycle. – CPS publishing. – 2010.

Надійшла 16.11.2010 р.