УДК 004.75

А. С. Поспешный, аспирант, **С. Г. Стиренко**, канд. техн. наук
Национальный технический университет Украины
«Киевский политехнический ин-т»
(Украина, 03056, Киев, пр-т Победы, 37,
тел. (044) 4068013, E-mail: pospishniy@kpi.in.ua)

Онтология ресурсов для семантического информационного сервиса грид

Представлена онтология ресурсов для семантического информационного сервиса грид и рассмотрены особенности применения семантических технологий в грид-системах.

Подано онтологію ресурсів для семантичного інформаційного сервісу грід та розглянуто особливості застосування семантичних технологій в грід-системах.

К л ю ч е в ы е с л о в а: грид, онтологии, информационный сервис, база знаний.

Динамично развивающаяся сфера высокопродуктивных вычислений существенно преобразила современную научную и инженерно-техническую деятельность. Появились новые инструменты, открылись новые возможности, но при этом возникли и новые проблемы, требующие решения.

В области научных исследований, где особенно остро ощущается нехватка вычислительных ресурсов, зародилась концепция вычислительной системы, способной интегрировать географически распределенные ресурсы и предоставлять к ним коллективный доступ. Грид является платформой для коллективной деятельности ученых и инженеров в рамках виртуальных организаций, которые обеспечены свободно предоставленными ресурсами. Такой подход позволяет наиболее эффективно использовать имеющиеся в наличии ресурсы для решения поставленных задач.

Эффективность грид зависит от доступности, точности и актуальности информации обо всех подключенных ресурсах, их характеристиках и состоянии. В грид-системах такую информацию обеспечивают информационные сервисы. Механизм доступа к этой информации является одним из ключевых компонентов любой грид-системы, так как он обеспечивает работу всех остальных служб. Доступ к этой информации должен быть максимально понятным широкому кругу пользователей и в то же время обеспечивать решение сложных задач.

С дальнейшим развитием грид неизбежно возникнут трудности в управлении такой большой и сложной гетерогенной системой, однако широко используемые в настоящее время для распределенных вычислительных систем традиционные технологии имеют существенные недостатки.

Возможность успешного применения семантических технологий в грид-системах обсуждалась неоднократно. В работе [1] сформированы основные принципы функционирования семантического информационного сервиса, рассмотрены выдвигаемые к нему требования и его ключевые компоненты, в частности онтология ресурсов. Был рассмотрен прямопоточный процессор правил TRIPLE/XSB и малоэкспрессивный язык RDFS. Разработанные онтологии не отображали специфики грид-ресурсов, а представляли собой описание аппаратных средств.

В модифицированной архитектуре грид с интегрированными семантическими сервисами [2] особое внимание уделено семантической аннотации ресурсов на базе GLUE схемы и применению агентов. Предложено использовать OWL-DL для создания онтологий, которые подразделены на две группы — онтологии компонентов и событий.

В [3] использована идея связывания онтологий разнородных грид-систем с целью их взаимной интеграции.

В работе [4] представлен семантический сервис мониторинга и поиска как надстройка над Globus Toolkit 4 и описан способ генерирования доменных онтологий по описанию свойств ресурса в формате XML. Агрегированные данные сохранялись в хранилище RDF триплетов Sesame и запрашивались с помощью языка запросов SPARQL.

В работе [5] предложен способ интеграции разнородной информации из различных сервисов грид с помощью онтологий. Такой подход позволяет аккумулировать и связывать информацию из разных источников и получать к ней доступ, используя язык запросов SPARQL.

Однако существенные недостатки описанных подходов [1—5] не позволили создать качественно новую информационную систему грид. В частности использование SPARQL запросов, генерируемых онтологий и малоэкспрессивных языков онтологий не обеспечило достаточно весомого преимущества перед традиционными технологиями.

Постановка задачи. Используя гипотезу о том, что с помощью семантических технологий, развивающихся в рамках концепции семантической сети, может быть создан семантический информационный сервис, основанный на технологиях интеллектуального управления знаниями [6], рассмотрим онтологию грид-ресурсов.

Объединим семантические технологии с одним из базовых компонентов любой грид-системы — информационным сервисом. Такой подход

дает возможность повысить эффективность использования грид-системы внедрением в механизмы ее работы элементов интеллектуальных систем, акцентируя внимание на задаче организации и поиска ресурсов.

Для достижения поставленной цели в первую очередь необходимо реализовать один из базовых элементов семантического информационного сервиса — онтологию грид-ресурсов. Под термином «онтология» будем понимать детальную формализацию некоторой области знаний с помощью концептуальной схемы. Онтология описывается множествами классов, атрибутов классов, доменов атрибутов, отношений и правил вывода на основе этих отношений. Разработанная онтология должна максимально отображать специфику грид-системы, опираться на стандартизированные технологии и языки и иметь высокую экспрессивность. Кроме того, необходимо наличие эффективных инструментов по разработке и использованию онтологии, а также возможность дальнейшего ее расширения, в частности на системы логического вывода и процессоры правил.

Семантические технологии. Консорциумом W3C был разработан и стандартизирован набор технологий представления знаний в виде, пригодном для машинной обработки с учетом децентрализованных систем. Среди таких технологий особый интерес представляют следующие:

XML — синтаксис для структурированных документов (без семантики).

RDF — модель представления данных в виде ориентированного графа.

RDFS — средства описания свойств, классов, иерархий RDF-ресурсов.

OWL — обширный язык онтологий для выражения сложных понятий и отношений.

SPARQL — язык запросов к данным, представленным в модели RDF.

RDF [7] — это универсальный расширяемый язык представления информации о ресурсах. Базовый строительный блок в RDF — триплет «объект — атрибут — значение», который обычно записывают в виде $A(O,V)$. Эту связь можно также представить как ребро с меткой A, соединяющее два узла, O и V : $[O]$ — A —> $[V]$. Таким образом, любой объект может играть роль значения, что в графическом представлении соответствует цепочке из двух ребер с метками. Кроме того, RDF допускает форму представления, в которой любое выражение RDF в тройке может быть объектом или значением, т.е. графы могут быть как вложенными, так и линейными.

Однако RDF не обеспечивает механизмов ни для описания атрибутов ресурсов, ни для определения отношений между ними. Эту роль выполняет RDFS. С его помощью определяются иерархии классов и свойств, а также такие понятия как «домен» и «диапазон». Но для решения конкретных задач возможностей RDF(S) часто оказывается недостаточно. Для

выражения более сложных отношений между понятиями в RDF разработан OWL — язык онтологий, расширяющий набор терминов RDFS. Язык OWL имеет три диалекта: OWL-Lite (соответствует уровню $SHIF^{(D)}$ дескриптивной логики), OWL-DL (обладает выразительной мощью, эквивалентной дескриптивной логике $SHOIN^{(D)}$) и OWL-Full (наиболее выразительный диалект с синтаксической свободой RDF, но не обладающий качеством разрешимости) [8].

Таким образом, онтология формирует понятийный аппарат, определяя сущности с их атрибутами и отношения между ними. Онтологию с экземплярами можно рассматривать как базу знаний, которая может быть использована для логического вывода (на основе принципа резолюций) новых фактов из имеющейся, предварительно структурированной информации, а также для совершения интеллектуального поиска экземпляров.

Существует множество реализаций процессоров умозаключений (reasoning engine) для OWL онтологий, различных по возможностям, области применения и качеству исполнения [9]. Обобщенный анализ позволяет разделить их на три группы в зависимости от метода реализации:

1. *Табличные DL-процессоры*. Традиционно использовались первыми для решения подобных задач. Имеют низкую производительность, но способны производить умозаключения на очень сложных онтологиях с множеством нетривиальных конструкций. К этому классу относятся Pellet [10], RacerPro [11], FaCT++ [12], а также HermiT [13] и SHER [14].

2. *Дизъюнктивные Datalog-процессоры*. Трансформируют онтологию в дизъюнктивную Datalog программу и используют методику дедуктивных баз данных и правило резолюций. Такие процессоры обладают удовлетворительным быстродействием при использовании некоторых оптимизаций, однако не поддерживают определенные OWL конструкции, в частности кардинальные ограничения и номиналы. К этой группе относится KAON2 [15].

3. *Процессоры правил.* Используют системы обработки правил для умозаключений в онтологиях. Имеют высокую производительность, но могут обрабатывать лишь простые онтологии, лишенные многих важных конструкций. Представители: Sesame/OWLIM [16], Jena [17], OWLJessKB [18].

В работах [13, 19, 20] дана оценка производительности упомянутых процессоров умозаключений и сделаны выводы об их слабых и сильных сторонах.

Онтология ресурсов грид. Рассматриваемая онтология основана на специально разработанной схеме для именования ресурсов в грид — Grid Laboratory Uniform Environment, GLUE [21]. Эта схема фиксирует все доступные для существования в грид компоненты и их характеристики и

используется в таких современных информационных сервисах как MDS [22] и BDII [23].

Разработанная онтология [24] содержит 65 классов, 33 объектных и 106 типизированных свойств и соответствует уровню SHIF $^{(D)}$ дескриптивной логики. Таким образом, данная онтология соответствует нормам Lite диалекта языка OWL.

На верхнем уровне представлены три класса: GridEntity, DomainConcept и Enumeration. Первый класс выполняет роль суперкласса всех базовых элементов грид, второй определяет доменные вспомогательные концепции, а третий используется для перечисляемых сущностей.

Онтология определяет следующие базовые элементы грид-системы (рис. 1):

CoreEntity — базовая сущность: Service и Site (сервис и площадка).

ComputingResource — вычислительный ресурс: Cluster, SubCluster, ComputingElement (кластер, подкластер и вычислительный элемент).

StorageResource — накопительный ресурс: StorageElement и Storage Area (накопительный элемент и область хранения).

На рис. 2 и 3 изображена иерархия соответственно DomainConcept (вспомогательных понятий) и Enumeration (перечисляемых понятий). Обе группы понятий используются для описания базовых элементов грид.

Понятие «площадка» определяет множество ресурсов, установленных и управляемых определенной организацией или персоной, а понятие «сервис» — абстракцию определенного программного модуля, реализующего взаимодействие между клиентом и сервером посредством пересылки сообщений. В табл. 1 и далее для определения элементов онтологии используется Манчестерский синтаксис [25]. Так, экзистенциальная конструкция $\exists$ определена ключевым словом **some**, а квантор всеобщности $\forall$ — словом **only** и т. д.

В табл. 2 приведены определения онтологических понятий «кластер», «подкластер» и «вычислительный элемент». Кластер определяет множество вычислительных машин под управлением единой системы управления. Подкластер описывает множество гомогенных хостов в составе кластера. Вычислительный элемент определяет службу, управляющую задачами, и предоставляет вычислительные ресурсы для их выполнения.

Накопительный элемент абстрагирует ресурс хранения данных, определяемый группой сервисов, протоколом, типом хранилища информации и др. (табл. 3). Область хранения — это логический раздел общего хранилища данных, выделенный для конкретной виртуальной организации.

Определенные в онтологии вспомогательные и перечисляемые сущности помогают описать базовые элементы грид, представляя отдельные их аспекты, характеристики и режимы работы [26].

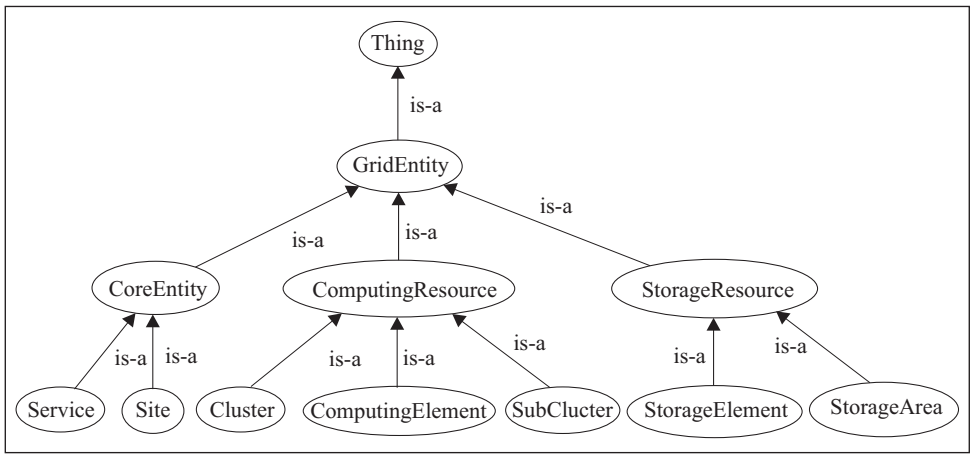

Рис. 1. Иерархия базовых элементов грид

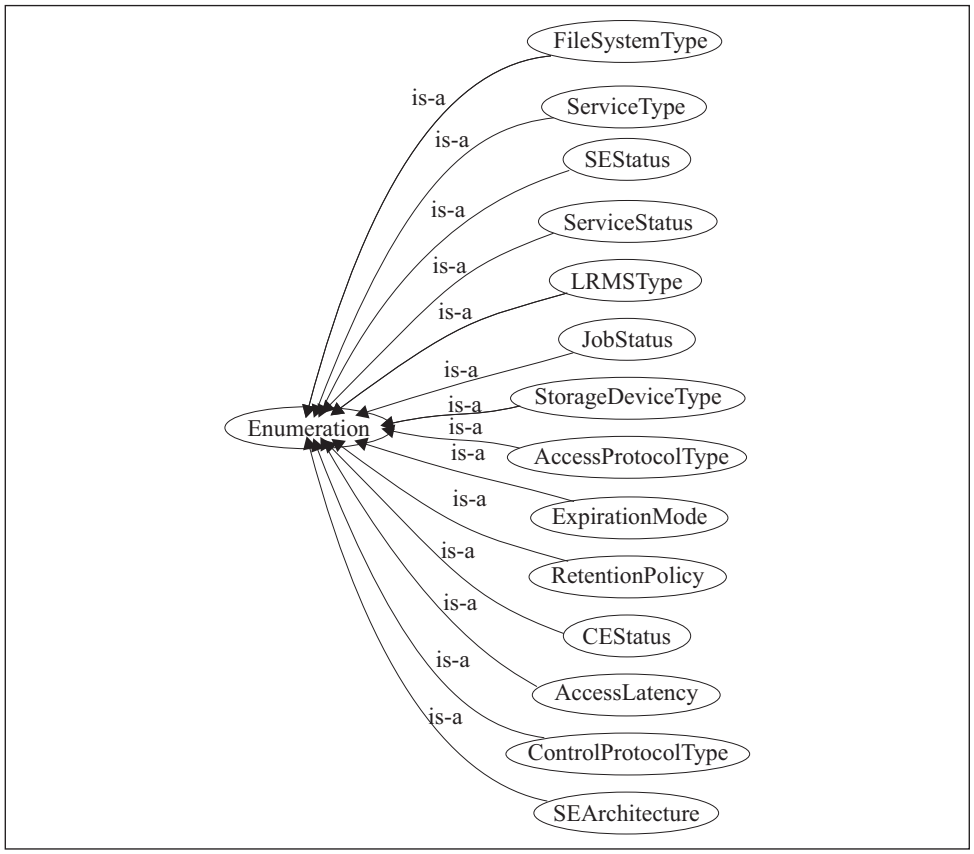

Рис. 2. Иерархия вспомогательных понятий

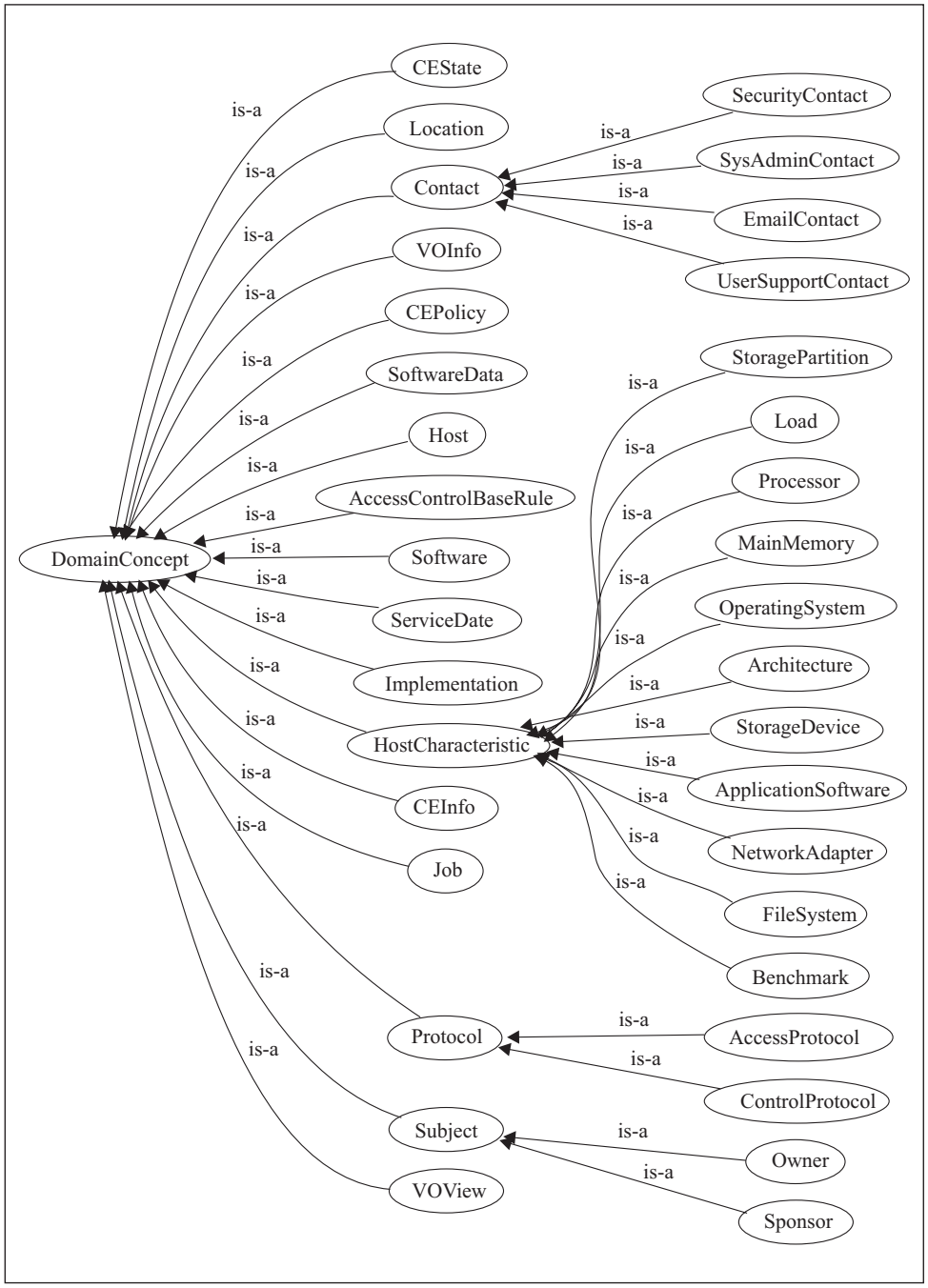

Рис. 3. Иерархия перечисляемых понятий

Таблица 1

Площадка	Сервис
Class: **Site** **SubClassOf:** CoreEntity, **contains some** (Cluster **or** Service **or** StorageElement), hasContact **some** Contact, hasLocation **some** Location, hasSponsor **some** Sponsor, hasDescription **some** string, hasName **some** string, hasOtherInfo **some** string, hasUniqueID **some** string, hasWeb **some** string	Class: **Service** **SubClassOf:** CoreEntity, hasAccessControlBaseRule **some** AccessControlBaseRule, hasOwner **some** Owner, hasStatus **some** ServiceStatus, hasType **some** ServiceType, hasServiceData **only** ServiceData, inRelationshipWith **only** Service, hasEndpoint **some** string, hasName **some** string, hasSemantics **some** string, hasStartTime **some** string, hasStatusInfo **some** string, hasUniqueID **some** string, hasVersion **some** string, hasWSDL **some** string

Таблица 2

Кластер	Подкластер	Вычислительный элемент
Class: **Cluster** **SubClassOf:** ComputingResource, **contains some** (ComputingElement **or** SubCluster), partOf **some** Site, hasName **some** string, hasTmpDir **some** string, hasUniqueID **some** string, hasWNTmpDir **some** string	Class: **SubCluster** **SubClassOf:** ComputingResource, describedBy **some** Host, describedBy **some** Software, partOf **some** Cluster, hasLogicalCPUs **some** integer, hasName **some** string, hasPhysicalCPUs **some** integer, hasTmpDir **some** string, hasUniqueID **some** string, hasWNTmpDir **some** string	Class: **ComputingElement** **SubClassOf:** ComputingResource, hasAccessControlBaseRule **some** AccessControlBaseRule, hasImplementation **some** Implementation, hasInfo **some** CEInfo, hasPolicy **some** CEPolicy, hasState **some** CEState, hasVOView **only** VOView, hasCapability **some** string, hasInformationServiceURL **some** anyURI, hasName **some** string, hasUniqueID **some** string

Использование онтологии. Онтология разработана с помощью редактора онтологий и инструмента построения баз знаний Protégé 4.1 [26]. Редактор Protégé не выполняет обработку знаний в онтологии, т.е. не содержит процессора умозаключений. Для этих целей внешние процессоры должны быть подключены посредством интерфейса OWLAPI [27].

Однако онтология — это всего лишь набор терминологических аксиом или так называемый TBox(T). Для того чтобы онтология стала работоспособной, необходимо наполнить ее набором утверждений об индивидах, т.е. создать ABox(A). Вместе ABox и TBox составляют базу знаний (K), которая может быть использована для представления знаний и их логического анализа. Для этих целей разработана програма [28] импортирования данных из LHC Computing Grid (WLCG), самой масштабной в настоящее время грид-системы, обслуживающей проведение экспериментов на Большом адронном коллайдере.

Разработанное приложение устанавливает соединение с корневым сервером информационного сервиса грид (top-BDII) по протоколу LDAP и формирует множество утверждений об экземплярах грид-ресурсов, опираясь на данную онтологию. При этом результат выдается в виде OWL файла со ссылкой на TBox онтологию, определенную в отдельном файле. Программа позволяет задать пользователю типы обрабатываемых ресурсов, необходимых к импортированию. Приложение не ограниченно LHC гридом и может быть использовано для импорта данных из других грид-систем, использующих BDII или MDS в качестве информационного сервиса.

Таблица 3

Накопительный элемент	Область хранения
Class: **StorageElement** **SubClassOf:** StorageResource, hasArchitecture **some** SEArchitecture, hasImplementation **some** Implementation, hasProtocol **some** (AccessProtocol **or** ControlProtocol), hasStatus **some** SEStatus, manages **some** StorageArea, partOf **some** Site, hasInformationServiceURL **some** string, hasName **some** string, hasTotalNearlineSize **some** integer, hasTotalOnlineSize **some** integer, hasUniqueID **some** string, hasUsedNearlineSize **some** integer, hasUsedOnlineSize **some** integer	Class: **StorageArea** **SubClassOf:** StorageResource, hasAccessControlBaseRule **some** AccessControlBaseRule, hasAccessLatency **some** AccessLatency, hasExpirationMode **some** ExpirationMode, hasRetentionPolicy **some** RetentionPolicy, managedBy **some** StorageElement, hasVOInfo **only** VOInfo, hasCapability **some** string, hasFreeNearlineSize **some** integer, hasFreeOnlineSize **some** integer, hasLocalID **some** string, hasName **some** string, hasPath **some** string, hasReservedNearlineSize **some** integer, hasReservedOnlineSize **some** integer, hasTotalNearlineSize **some** integer, hasTotalOnlineSize **some** integer, hasUsedNearlineSize **some** integer, hasUsedOnlineSize **some** integer

Таким образом, с помощью сформированной базы знаний пользователь может совершать запросы необходимых ресурсов, опираясь на определения в онтологии. Важным является то, что пользователь может расширять набор терминологических аксиом фактами, дополняя базовую онтологию и, следовательно, базу знаний все большим числом фактов, которые будут использованы для извлечения неявной информации на основе уже имеющихся фактов.

Рассмотрим практический пример. Определим класс FreeCE как вычислительный узел, который в данный момент не выполняет никаких заданий, имеет пустую очередь задач и способен принимать новые задания:

```
FreeCE ≡ ComputingElement and hasState some
(CEState and hasRunningJobs value 0
        and hasWaitingJobs value 0
        and hasFreeJobSlots some integer[>0])
```

Определим класс MPI_Enabled_Cluster как кластер, поддерживающий технологию MPI (состоит из хостов, в которых заявлена поддержка пакета OpenMPI):

```
MPI_Enabled_Cluster ≡ SubCluster and describedBy some
(Host and describedBy some
    (ApplicationSoftware and hasRunTimeEnvironment value “OPENMPI”))
```

Опираясь на эти концепции, определим класс SiteForMyAp для нахождения площадки грид, которая содержит кластер с поддержкой MPI, минимум 100 процессоров и свободен для выполнения задач:

```
SiteForMyApp ≡ Site and (contains some FreeCE ) and (contains some
      (MPI_Enabled_Cluster and hasPhysicalCPUs some integer[>100]))
```

После процесса классификации индивиды, принадлежащие типу SiteForMyApp, будут соответствовать искомым ресурсам, а класс доступен для использования при создании более сложных запросов.

Таким образом, создавая свой словарь терминов и понятий, пользователь облегчает задачу организации и поиска ресурсов в грид. Например, можно определить множество программ и их требования по ресурсам и совершать поиск запросом «найти все ресурсы, удовлетворяющие требования программы X» или определить задачи, которые решают эти программы, и искать ресурсы, необходимые для выполнения этих задач (например, «какие ресурсы грид подходят для решения задачи моделирования жидкости?»).

Для дальнейшего расширения возможностей онтологии целесообразно использовать язык правил SWRL [29]. С его помощью можно создавать свои правила логических заключений, не предусмотренные языком OWL 2

[30], и таким образом еще больше расширять возможности использования онтологий. Для выполнения поставленной задачи ключевую роль играет процессор умозаключений. Среди указанных выше процессоров лишь несколько удовлетворяют предъявляемым требованиям. Так, например, среди продуктов первой группы практический интерес представляет Pellet. Процессоры HermiT и FaCT++ не оптимизированы для работы с экземплярами (большим ABox), RacerPro является закрытым коммерческим продуктом, а SHER пока недостаточно стабилен для промышленного использования.

Представитель второй группы — проект KAON2 — не обеспечивает поддержку языка OWL2 и интерфейса OWLAPI и в данный момент прекратил свое развитие.

В третью группу входят наиболее перспективные процессоры. Скорость обработки данных в Jena и OWLIM более чем достаточна для практического применения. Однако процессоры данной группы способны поддерживать относительно небольшую экспрессивность онтологий, ограниченную в основном простыми конструкциями. Например, отсутствует поддержка логического анализа типизированных свойств, что делает невозможным выражение концепций, используемых выше в качестве примера.

Выводы

Представленная онтология ресурсов грид становится фундаментом для семантического информационного сервиса, который помогает эффективно и удобно использовать грид-системы для решения задач. Применение семантических технологий открывает множество возможностей и перспектив для дальнейшего совершенствования базовых элементов грид-систем, способствует появлению новых моделей взаимодействия пользователя с ними и дает толчок к «интеллектуализации» и созданию более комфортных и доступных сервисов.

The ontology of resources for semantic information service of grids have been presented, peculiarities of using semantic technologies in grid-systems have been considered.

1. *Tangmunarunkit H., Decker S., Kesselman C.* Ontology-based resource matching in the Grid — the Grid meets the semantic web // The SemanticWeb-ISWC 2003. — 2003. — P. 706—721.
2. *Ambrosi E., Bianchi M., Gaibisso C., Gambosi G.* A Description Logic based Grid Inferential Monitoring and Discovery Framework // Proc. of the 2005 International Conference on Grid Computing and Applications. — 2005. — P. 18—23.
3. *Parkin M., Van den Burghe S., Corcho O. et al*. The knowledge of the grid: A grid ontology // Proc. of the 6-th Cracow Grid Workshop. — 2006. — P. 111—118.

4. *Said M., Kojima I.* S-MDS: Semantic Monitoring and Discovery System for the Grid // Journal of Grid Computing. — 2008. — Vol. 7, N 2. — P. 205—224.
5. *Xing W., Corcho O. et al.* An ActOn-based semantic information service for Grids // Future Generation Computer Systems. — 2010. — Vol. 26, N 3. — P. 324—336.
6. *Стиренко С. Г., Поспешный А. С., Барабаш М. А.* Информационные сервисы GRID-систем. Их недостатки и семантический подход к их решению // Розподілені комп'ютерні системи: Зб. праць ювілейної міжнародної науково-практичної конференції РКС-2010. — Київ: НТУУ «КПІ», 2010. — С. 26—28.
7. *Спецификация* Resource description framework (RDF). — http://www.w3.org/ RDF/.
8. *Спецификация* Web Ontology Language (OWL). — http://www.w3.org/ TR/owl-ref/.
9. *http:*//www.w3.org/2007/OWL/wiki/Implementations
10. *Sirin E., Parsia B., Grau B. et al.* Pellet: A practical OWL-DL reasoner. // Web Semantics: science, services and agents on the World Wide Web. — 2007. — Vol. 5, N 2. — P. 51—53.
11. *Веб-ресурс* проекта RacerPro. — http://www.racer-systems.com/
12. *Веб-ресурс* проекта FaCT++. — http://owl.man.ac.uk/factplusplus/
13. *Shearer R., Motik B., Horrocks I.* Hermit: A highly-efficient owl reasoner. // Proc. of the 5-th International Workshop on OWL: Experiences and Directions (OWLED 2008). — 2008. — Vol. 432.
14. *Dolby J., Fokoue A., Kalyanpur A. et al.* Scalable highly expressive reasoner (SHER). // Web Semantics. —2009. — P. 357—361.
15. *Веб-ресурс* проекта KAON2. — http://kaon2.semanticweb.org/
16. *Веб-ресурс* проекта OWLIM. — http://www.ontotext.com/owlim/
17. *Веб-ресурс* проекта Jena. — http://jena.sourceforge.net/
18. *Веб-ресурс* проекта OWLjessKB. — http://edge.cs.drexel.edu/assemblies/software/owljesskb/
19. *Rock J., Haase P., Ji Q., Volz R.* Benchmarking OWL reasoners // ARea2008 — Workshop on Advancing Reasoning on the Web: Scalability and Commonsense. — 2008. — Vol. 5021. — P. 1—15.
20. *Weithoner T., Liebig T., Luther M et al.* Real-world reasoning with OWL // The Semantic Web: Research and Applications. — 2007. — P. 296—310.
21. *Andreozzi S., Burke S., Donno F et. al*. GLUE Schema Specification (version 1.3). — http://gluschema.forge.cnaf.infn.it/Spec/V13 .
22. *Czajkowski K., Fitzgerald S., Foster I., Kesselman C.* Grid information services for distributed resource sharing // Proc. of the 10-th IEEE International Symposium on High Performance Distributed Computing. — IEEE Press. — 2001. — P. 181—195.
23. *Документация* к проекту Berkeley Database Information Index V5. —https://twiki.cern.ch/twiki/bin/view/EGEE/BDII/.
24. http://grid-ontology.googlecode.come/files/GLUE.owl
25. *Horridge M., Drummond N., Goodwin J.* The manchester owl syntax // Second International Workshop «OWL: Experiences and Directions (OWLED 2006). — 2006. — Vol. 216.
26. *Веб-ресурс* проекта Protege. — http://protege.stanford.edu/
27. *Веб-ресурс* проекта OWLAPI. — http://owlapi.sourceforge.net/
28. http://code.google.com/p/grid-ontology/source/checkout/
29. *Предложение* стандарта SWRL. — http://www.w3.org/Submission/SWRL/
30. *Andreozzi S., Burke S., Ehm F. et al.* GLUE Schema Specification (version 2.0). — http://www.gridforum/org/documents/GFD.147.pdf

Поступила 11.01.11

ПОСПЕШНЫЙ Александр Сергеевич, аспирант кафедры вычислительной техники Национального технического университета «КПИ», который окончил в 2009 г. Область научных исследований — построение схем метаданных и онтологий для повышения качества интеллектуального поиска и семантической совместимости разнородных, географически распределенных ресурсов национальной грид-инфраструктуры.

СТИРЕНКО Сергей Григорьевич, канд. техн. наук, директор центра суперкомпьютерных вычислений Национального технического университета «КПИ». В 1995 г. окончил Национальный технический университет «КПИ». Область научных исследований – организация эффективных вычислительных процессов в гетерогенных высокопроизводительных средах; экстремальные параллельные вычисления для проблемно-ориентированных предметных областей; построение схем метаданных и онтологий для повышения качества интеллектуального поиска и семантической совместимости разнородных, географически распределенных ресурсов национальной грид-инфраструктуры.