Approach of a UML Profile for Berkeley Open Infrastructure for Network Computing (BOINC)

Christian Benjamin Ries
Computational Materials
Science and Engineering (CMSE)
University of Applied Sciences
Bielefeld, Germany
www.visualgrid.org

Christian Schröder
Computational Materials
Science and Engineering (CMSE)
University of Applied Sciences
Bielefeld, Germany
Christian.Schroeder@fh-bielefeld.de

Vic Grout
Centre for Applied
Internet Research (CAIR)
Glyndŵr University, United Kingdom
v.grout@glyndwr.ac.uk

***Abstract*—Despite the current enormous hype and popularity of Grid Computing environments like Amazons EC2 or Microsoft's Windows Azure , there exist open-source and free of cost software frameworks which allow to create high performance computing installation by means of Public Resource Computing (PRC). One PRC framework is BOINC (Berkeley Open Infrastructure for Network Computing) for solving large scale and complex computational problems. Each computer works on its own workunits independently from each other and sends back its result to a project server. Installing, configuring, and maintaining a BOINC based project however is a highly sophisticated task. Scientists and developers need a lot of experience regarding the underlying communication and operating system technologies, even if only a handful of BOINC related functions are actually needed for most applications. In this paper we present a Unified Modeling Language (UML) profile for BOINC called Visu@lGrid profile (VGP). A BOINC project installation for one or more hosts, a role-based access control, and modeling of scientific application is feasible by use of VGP. Based on our approach we provide a specification that allows the creation of BOINC projects with less development and implementation effort.**

***Keywords*—BOINC, UML, Profile, Stereotypes, Tag-Values**

I. Introduction

PUblic Resource Computing (PRC) technologies allow realising low-cost high-performance computing projects in certain application areas. Berkeley Open Infrastructure for Network Computing (BOINC) is a very prominent framework based in the principles of PRC and is based on a server-client communication infrastructure mechanism. Here, the client retrieves a project specific application from the server along with so-called workunit (WU), i.e. a number of parameter usually provided in data files of simple ASCII or binary format that are optionally needed by the application to perform specific tasks. Each BOINC project (BP) has its own infrastructure, i.e. few daemons, tasks, hosts, and scientific application (SAPP) for different target platforms.

This paper presents a Unified Modeling Language (UML) profile for Visu@lGrid (VG), called Visu@lGrid profile (VGP). VG is a research project with the goal to have specifications and definitions how to create a BP within a Model-driven engineering (MDE) process and graphical elements. This is the reason why we call it visual. VGP is used as a core specification which makes it possible to create UML models for a complete BP set-up with all components for a fully predictable infrastructure to have the possibility to use code-generation (CG) facilities. We show an approach which makes it possible to define how a BP will be installed on different ranges of unlimited count of hosts. Furthermore, we will show how to configure a complete role-based access control (RBAC) for different software components and system functionalities, e.g. database access or log-in on one host. There exists some promising RBAC approaches which will be adopted by VGP [4]. We have already presented a first modeling domain-specific language (DSL), i.e. textual and graphical model elements [11]. Additional, we have shown that only a handful BOINC functions are necessary to create a complete usable BP.

The remainder of the paper is organised as follows. Section 2 gives an overview of BOINC's problem domain. Next, Section 3 describes the UML modeling principles and which UML extensions are defined for VGP. In Section 4 some open tasks and research questions are refereed which we will focus on in additional work. Finally, Section 5 concludes this paper.

This project is funded by the German Federal Ministry of Education and Research.

II. BOINC: Usage and Challenges

BOINC is based on a simplified architecture, i.e. each task is separated in autonomous running applications like BOINC daemons or scripts and we call these BOINC services (BS). In fact, few steps allow to create a new BP: (1) download and install prerequisite software packages, (2) download the BOINC sources, (3) configure and build the BOINC software, and (4) call scripts for an automatic rudimentary and not really usable BP installation process [19]. Next, you can modify each part of your new BP, e.g.

- which and where BS should be executed, i.e. in case more than one hosts are used it is possible to define specific BS's should be executed with different or same start-up parameters on individual hosts,
- if a database uses a replication for read-only queries to speed-up BP's performance,
- which host has the main installation, i.e. if individual hosts are used on server-side, then one host has the BP

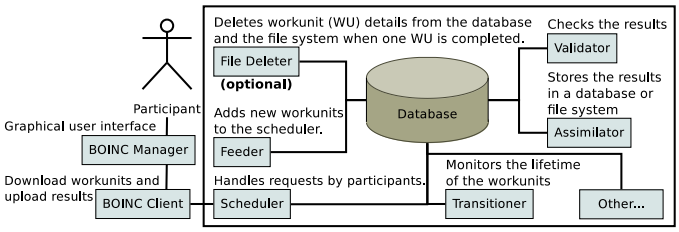

Fig. 1. Client and server components of BOINC's infrastructure

installation and has to distribute it to supplementary hosts. More detailed adjustments are possible. As in [11] mentioned, there exists typical errors that can occur due to manual editing of the BOINC server configuration files. As a consequence of these errors one can expect a significant effect on the system's integrity and application performance. In practice, wrong configured BS's do not work correctly, i.e. a BOINC Assimilator (BA) will not store results for later use or a BOINC Validator (BV) is not able to validate results received by participants.

On one hand if only one host is planned to use, it could be an uncomplicated proceeding from download of the BOINC sources to a running BP. In this case, all work is done on one host, i.e. BOINC sources could be accessed directly, BS's must be started on this host, database and web-server installations are also directly available, and only for this host relevant values are added to BOINC's configuration. Additionally, only one network interface card (NIC) is necessary to be configured and exclusive ports of Hypertext-Transfer Protocol [Secure] (HTTP[S]) must be allowed for network traffic. On the other hand it could be an enormous challenge to set-up a BP with more than one host and when daemons, tasks, database and web-server installations are distributed for execution to individual hosts. First established BP - Seti@Home (SAH), which is one of the most driven BP - has minimum count of 13 hosts on server side, and each host has a different functionality[1]. SAH has three databases[2] (DB) on respective host, and each DB has a different purpose but it is necessary that all BS's can interact with them.

A. Special Interest Groups and Architecture

Different special interested groups could administrate one BP, i.e. "administrators" configure and maintain the system installation, and "scientists" could create new WU's and add new SAPP versions. SAPP's could be implemented in different ways, e.g. scripts, hand-written code in different programming languages (C/C++ and Phyton are supported by default), or we can use ISV (Independent Software Vendor)-Applications or legacy applications [3], [7], e.g. in [10] we implemented a wrapper to handle COMSOL Multiphysics [5] computations on one computer. To a greater extent, an application can be written from scratch. This approach is the most difficult of all and fortunately BOINC's application programming interface (API) is not changing frequently with a new version. Fact is, one can show that only 23 different BOINC functions are necessary to implement a successfully running research relevant distributed SAPP [11]. We have to note, that we do not mention something about new target devices which could be implemented with additional application logic or extended functionalities, e.g. to support graphics processing units (GPU) with the help of BOINC's API. When a BP is installed and a SAPP is implemented, new problems and questions will arise:

- How can you manage errors or how is it possible to upgrade one SAPP during runtime?
- How could we replace software components on particular hosts or a whole host itself with same functionalities?
- How could we restrict access to groups of users and only for selected functions and system facilities?

Mentioned steps could be executed on different hosts, and it is highly error-prone to install and configure all required software components and interface settings and to keep an always valid configuration and stable system when for example software has to upgrade or a host has to be replaced. In addition, it is too cumbersome to create a fast and easy to use BP, just only for tests or quality assurance processes.

B. Proceeding

As seen on the left hand side of Fig. 1, participants need to download the BOINC Manager[3] (BM), register for one or more BP's, and let the BOINC Client (BC) organizes all system calls, in-/output handling, allocating compute resources, so-called scheduling [1] and handling of communication messages to and from individual BP's.

Fact is, with a minimum of five different BOINC daemons it is feasible to start-up a fully operable BP. Fig. 1 shows on the right hand side these five daemons with an optional sixth one. Scheduler is the interface for participant requests and supports two kinds of requests: (1) to send out WU's, and (2) to handle with computation results [1], [6]. Frequently, the shared-memory (SHM) based queue for schedule requests is refilled by the Feeder. File Deleter will keep the DB and file system clean of expired or not longer needed datasets and files. Lifetime of WU's is tracked by the Transitioner and this makes it necessary that all BS's have access to BOINC DB's, i.e.

- when an user creates WU's, information about these WU's are stored within BOINC's DB and they are marked as not *yet processed*,
- a request to the Scheduler will select few WU's and will change the state of these WU's to *processing*,
- after this request, different proceedings are possible: (1) each WU has a time-slot in which they should proceed and when time-slots upper limit - so-called deadline - is reached this WU will be marked as *out of date* and could be transmitted to participants again or will be deleted

[1] http://setiathome.berkeley.edu/sah_status.html - accessed 22.05.2011

[2] Components of SETI@home: (1) BOINC master database which is the BP back-end database, (2) BOINC replica database for read-only access, and (3) SETI@home science database to store results.

[3] A GUI to handle all BP's where volunteers are registered.

later[4], and (2) WU's are returned inside time-slots and the Scheduler will mark them in BOINC DB's as *ready for validation*,
- the BV will validate the result and mark each WU result as *valid* or *invalid*, and
- the BA will store valid and/or invalid results in an additional DB or within the file system hierarchy and mark assimilated WU's as *finished* and *ready for deletion*.

In Fig. 1 it could be seen, that each daemon needs a valid configuration to access BOINC's DB. Furthermore, a lot of knowledge is required to define runtime parameters of each daemon. Default values are fine, but in some cases it is more valuable to define various parameter sets, e.g. if load-balancing is necessary to increase server capacity. A file is used to define which SAPP's are offered by one BP. In addition, this file contains information about planned or supported target platforms, e.g. windows_intelx86[5] or x86_64-pc-linux-gnu[6]. Definitions in this file are not strict, it is necessary to have a SAPP within BP's file hierarchy, otherwise not used platforms are unrelated to SAPP's.

III. UML Profile: Visu@lGrid

UML profile is a kind of UML extension mechanism [4], [15]. It specializes some of the language elements, imposes new restrictions on them while respecting the UML metamodel and leaving the original semantics of the UML elements unchanged. Icons and symbols can be specified and applied for these specialized elements. The Object Management Group (OMG) maintains some common and widely accepted profiles, such as Systems Modeling Language (SysML) [16] and UML profile For Modeling and Analysis of Real-Time and Embedded Systems (MARTE) [17]. Stereotypes extend standard UML metaclasses. Furthermore, we can exchange UML models with the XML Metadata Interchange (XMI) language from one integrated development environment (IDE) to another.

UML profiles are defined in terms of three basic mechanisms: *stereotypes*, *tagged values*, and *constraints*. A stereotype defines how an existing metaclass may be extended. A tagged value is an additional meta-attribute, it could be seen as a variable. It has a name and a type, and is member of a specific stereotype. Constraints can be associated with stereotypes, they could be informal in plain-text or more specific defined with Object Constraint Language (OCL) [18]. OCL is part of the UML, and is used to express constraints and properties of model elements. Currently, only informal constraints are defined for VGP, because a high effort for implementation is needed to realize a fully support for all VGP requirements.

Most of the current OCL tools are academic tools and were developed by a team of a single university [2]. Although the quality of tools has improved considerably over the last years, it is not a surprise that these OCL tools cannot compete in terms of usability and the functionality they offer with IDE's for writing implementations.

Our UML profile approach is currently based on nine UML metaclasses and Table I lists these with our specified stereotypes which are used in this paper. With them we can set-up an automatic code generation (CG) of a BP. Here, VGP is currently based on 31 stereotypes which are usable to define BP's for a single host or a bundle of hosts, where each host runs different BS's. Our principles for VGP are directly based on the BOINC architecture. This means we are strongly focused on top-down analysis of BOINC's functionality, communication and runtime behavior, i.e. BP's are seen as UML packages with embedded UML components for hosts. BS's are seen as UML components with functionalities to outside world. Static configurations are UML properties which could be defined as UML classes, i.e. configuration of network interface cards (NICs) has fixed values. Main benefit of our VGP profile are based on the advantages of UML itself. With UML it is possible to specify Platform-Independent Models (PIMs) and Platform-Specific Models (PSMs) of processes and implementations. Here, we follow this methodology and define a model with a mix of PIM and PSM:

- On server side elements are fixed for one specific target platform[7] and PSM is used,
- creation and maintaining of workunits or scientific applications is not related to underlying operating system (OS) functionalities which means that PIM will be used, and
- participant's side could be highly heterogeneous where a PIM is reasonable.

A. Concept Idea behind Visu@lGrid Profile

As mentioned, our VGP concept is based on BOINC's implementation concept itself. Here, we make use of the opportunity that UML elements could be nested in other ones. With this in mind, we define a hierarchy with UML for a BP and each level in the hierarchy must have the potential to be replaceable. In our view, UML is the best choice to support this. Two engineering points of views exist: (1) outer view, and (2) inner view. Clearly, the outer view is the easiest to understand and it describes the world of participants / volunteers which are registered to one or more BP's and is illustrated on the left hand side of Fig. 1. In contrast, the inner view is more complex and needs more attention to specify the infrastructure and behavior on server-side.

B. UML Profile Itself

Fig. 2 shows an overview of all currently defined stereotypes, how they relate to each other stereotypes, and which UML metaclasses are extended. Constraints are not included, they are partly informal specified in Table I. «Projects» is our root of VGP and could own components extended by «SAN» and packages extended by «Project». The idea is,

[4]This behavior depends on sets of parameters during WU creation.
[5]Microsoft Windows running on an Intel x86-compatible CPU
[6]Linux running on an AMD x86_64 or Intel EM64T CPU
[7]BOINC supports only Unix platforms.

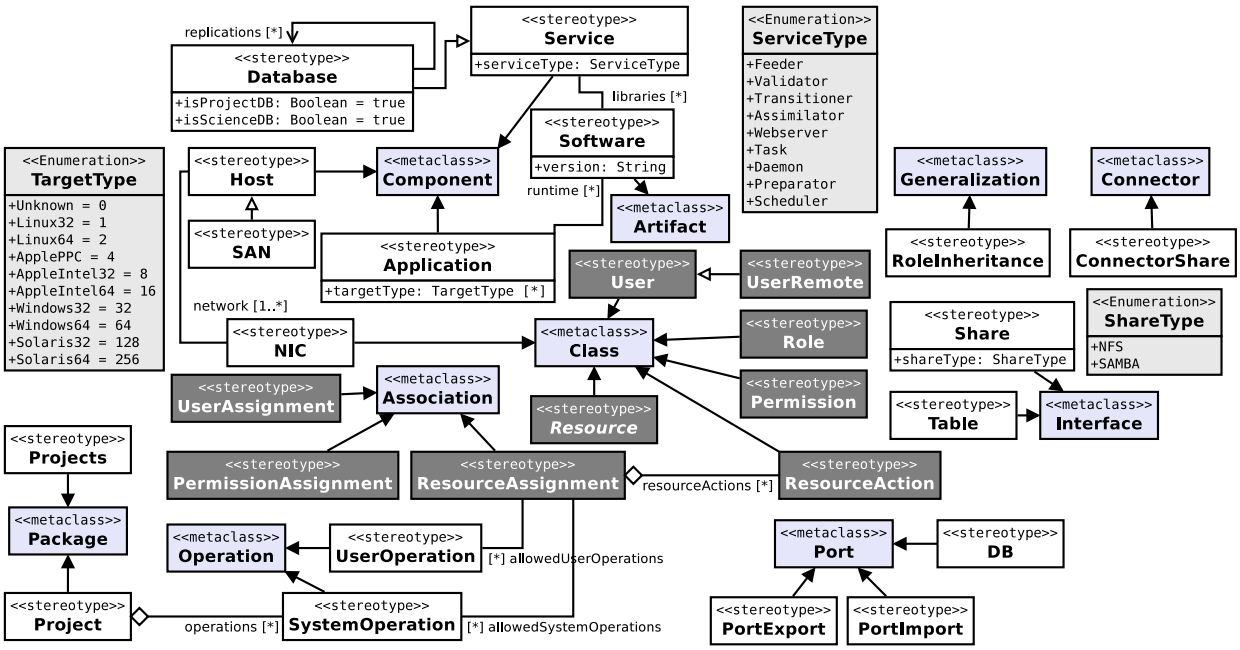

Fig. 2. VGP stereotypes to restrict the use of UML modeling aspects. This is an overview diagram and Table I names all metaclasses which are extended by these stereotypes.

that «Project» can be imported and exported any time within an IDE [8]. BS's can be added with components extended by «Service», which must be typed by «ServiceType». The enumeration does not provide any case of different BS types, for this purpose enumeration literals *Task* and *Daemon* are added and must be used if supplementary BS's are implemented, configured and wished to use for one BP. When a BS needs advanced software or libraries for runtime, they can be added in different versions by «Software». «Database» is a specialization of «Service» and can be used to add three different types of databases: (1) a BP database as shown in Fig. 1, (2) a science database (SDB) to store computational results or other datasets, and (3) database replications for the first and second database type. DB instances can be determined by two tag-values: *isProjectDB* and *isScienceDB*. First one is true when it is BP's database, and second one is true when this it is a SDB. DB replications are associated by *replications*, but the mentioned two tag-values are not necessary to specify while they can be retrieved automatically by check of owner's type. SAPP's can be modeled with components extended by «Application» and all planned supported target platforms must be added to *targetType*. Again, for runtime required software packages and libraries can be specified.

All hosts within one BP can share data among each other and for this reason, hosts get ports extended by «PortExport» and «PortImport» which must be associated to each other with interfaces extended by «Share». One «PortExport» with a provided «Share» can be used by more than one «PortImport», i.e. a «PortExport» is a global definition within one BP. If authorized access is wished, both ports must be extended by «Resource» and associated to a set of «Permissions» (described later). Same methodology is specified for databases and database-tables, i.e. «DB» is like mentioned two kind of ports and «Table» is like «Share». These principles are based on the UML specification to create components which are fully replaceable when all ports of another component are equal.

In Fig. 2, stereotypes with a dark-gray background color are used to define RBAC for one BP. It is not explicit defined where RBAC should be defined. Few ways are possible: (1) global definition in «Projects» or (2) fixed definitions for each «Project». We suggest to define them within «Project». In this case it would be possible to exchange BP's more effortless, otherwise it is necessary to create new RBAC or to duplicate existing ones. In [4] the authors give an idea of an approach to define an user control mechanism where each user could be added to roles and individual resources, our approach extends this idea. We define that permission rules could only associated to elements which are extended by «Resource». Individual users can be defined with classes extended by «User», roles are defined with classes extended by «Role», and a set of permissions are defined with classes extended by «Permissions». Based on this there is a strict coherence how associations must be used, i.e. «User» instances must be associated to «Roles» and these to «Permissions». Finally, «Permissions» could be associated with elements extended by «Resource». Fig. 3 shows a first example how our VGP could be applied to models.

C. Case-Study LMBoinc

Fig. 3 shows some applied stereotypes in a context of a real BP named *LMBoinc*. LMBoinc modifies a video stream, i.e. a video is fragmented in sequences and basic image processing algorithm are applied to these sequences, some results could

be seen on the project website [9]. WU's and computational results within LMBoinc include ZIP-archives (ZIP) with a specific number of sequences, wherefore the SAPP *lmboinc* is associated to one matching «Software» to make zip available during runtime. LMBoinc uses three hosts on server-side:

- **lmboinc-db:** This host has the BP database "DBProject" and SDB "DBScience" installed. SDB is used to store metadata of computational results, i.e. dimension, filesize, and filenames of sequences. Furthermore, BP's database has an additional replication to speed-up database queries.
- **lmboinc-services:** This host is composed by four BS's. The BA is associated to mentioned SDB and has an additional association to the third host and stores results within Webserver's file system, i.e. when a port is owned by one BS, additional constraints define which directories are usable for external use. Here, the exported directory is below the root directory for web-pages used by this web-server instance and network file system (NFS) is used between these hosts.
- **lmboinc-web:** Because of the provided Webserver component, participants just see this host of one BP. As a consequence, Feeder and Scheduler must be added to this host, due to the mentioned fact that Feeder is responsible to refill the shared-memory with information of next WU's which are selected for computation. When one participant requests new WU's, Feeder will check the shared-memory and send out informations where participants could download these WU's with HTTP requests. Therefore, necessary ZIP's must be below Webserver's document root directory, to make them accessible. *lmboinc-services* owns a BS to create WU's which are added to this download area.

In the top-right area three users are added with particular role associations. *Ries* is associated to role *Admin* and has permissions to access all hosts locally and remotely, additionally he is allowed to handle WU's for the SAPP *lmboinc*. The set of permissions for the association between *HandleWork* and *Preparator* include two operations: *createWU* and *cancelWU*. These two operations are implemented within *Preparator* and can be called by users when they get this permission. How this could be done must be provided by CG. Schroeder owns two roles, first is *Scientist* to interact with WU's as mentioned, and second role is *WebAdmin*. *WebAdmin* allows only to moderate posts within a BP related web forum. User *Grout* has only one role and the permission to moderate forum entries.

As defined in the top-left area, this BP targets Linux and Windows for 32 bit platforms where *lmboinc* will be distributed to. For more supported platforms it is required to extend the tag-value *targetType* by additional enumeration literals of «TargetType».

IV. Conclusion

We have proposed VGP, which provides an early approach to handle system and runtime relevant problems to implement a PRC project based on BOINC. Our approach is just the beginning of a MDE process to set-up a complete BP and to handle all necessary administration and maintaining tasks. In this paper, we have shown how our VGP can be applied to a real model within a cluster environment, how data can be shared between this cluster domain, where BS's are installed, which SAPP is used and which target platforms are supported. We have just shown an extract of VGP, due to the fact of the restriction of upper-page limit of this paper. A draft version of VGP is available and will be frequently changed [13]. An entire model would contain all design model elements and specifications to create a whole BP with a scientific application.

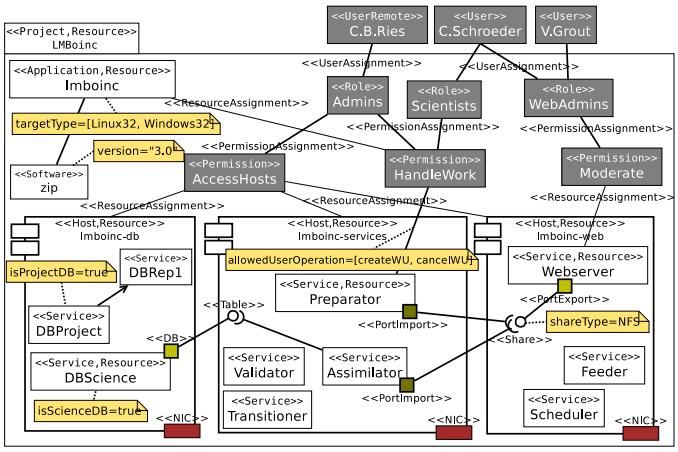

Fig. 3. A BP with three users, three hosts with different added BS and NIC's, a SAPP called *lmboinc* [9], and a RBAC set-up.

V. Future Work

Future work will focus on different approaches to make it possible to define processes for WU creation and maintaining of failed processed WU's. In general, currently it is not clear how we could monitor WU's lifetime, i.e. how many WU's are still in queue or if failed WU's should be processed again or not.

Additional work must be done for modeling concepts of adding and configuring of available ISV applications, e.g. Scilab. Some work is done and could be integrated in or used for VGP [8], [10], [11], [12]. As well some effort has to be spend on how we can define security restrictions on network traffic on «NIC» elements.

Currently, periodically executed tasks can be added by «Service» and *Task* assigned to service type. The period when one task should be executed is specified by a string value and is based on the format of the UNIX cron daemon. Future work will cover an approach how to describe this period with SysML timing diagrams [16] to make it more predictable.

Asynchronous messages by participants to one BP and vice versa are currently not supported, but suitable BS's can be added with «Service». How to react on incoming messages on client- and server-side is undefined. Additionally, it is unclear how we should react on unexpected exceptions, e.g. one BS is not responding anymore, or if there is not enough space to store computational results.

TABLE I
EXTRACT OF OUR UML STEREOTYPES WITHIN VGP

Stereotype name; extended Metaclass	**Description**	**Stereotype name; extended Metaclass**	**Description**
Projects; Package (from Kernel)	Top-level package which owns all sub-elements, i.e. hosts or users.	Host; Component (from BasicComponent)	Specifies one host within a BP, a host owns some or all BS.
Project; Package (from Kernel)	One specific BP.	NIC; Class (from Kernel)	Defines network settings for one host.
SAN; Component (from BasicComponent)	External host with capabilities to store data, e.g. WU's or computational results. Also known as storage area network (SAN).	OperatingSystem; Enumeration (from Kernel)	Unique literals for each planned or supported operating system, e.g. Linux32, Linux64, Windows32, or Windows64.
PortExport, PortImport; Port (from Ports)	Definition of network shared file directories between hosts.	ShareType; Enumeration (from Kernel)	Unique literal for each planned or supported share type, e.g. NFS or SAMBA [14].
Share; Interface (from Interfaces)	One specific directory which is ex- or imported. Exported directories are defined as provided interfaces, otherwise they are defined as required interfaces.	UserAssignment, PermissionAssignment, ResourceAssignment; Association (from Kernel)	Associates users, roles, permissions, and resources. UserAssignment assigns users to roles, PermissionAssignment assigns roles to permissions, and ResourceAssignment assigns permissions to resources.
User, UserRemote; Class (from Kernel)	A physical user which has different permissions within one BP described by roles and permissions.	Software; Artifact (from Artifacts, Nodes)	Software packages or libraries which are required during runtime, i.e. BA needs unzip to extract results out of an ZIP.
Resource; Class (from Kernel)	An user could only be assigned to resources when resources have this stereotype.	UserOperation; Operation (from Kernel, Interfaces)	Operations are executable by users.
ResourceAction; Class (from Kernel)	Actions which one user in a role or permission list could call.	Role, Permission; Class (from Kernel)	Defines roles for users and which permissions these roles include.
DB; Port (from Ports), Table; Interface (from Interfaces)	Allow BS's to access database tables.	SystemOperation; Operation (from Kernel, Interfaces)	Implementations of system or BS functionalities, e.g. routines for BV or BA.
Service; Component (from BasicComponents)	BS's on one host, e.g. Feeder and Transitioner.	ServiceType; Enumeration (from Kernel)	Type of a BS, e.g. Feeder or Transitioner.
Database; Component (from BasicComponents)	BOINC's master/science database and optional replications to increase DB performance.	ConnectorShare; Connector (from BasicComponents)	Connects exported shares with imports and vice versa. An export could be used by multiple imports.
Application; Component (from BasicComponents)	Describes one SAPP, which is used for computations.	Platform; Enumeration (from Kernel)	Literals for each target platform, e.g. Linux 32 bit or Windows 32 bit.

REFERENCES

[1] D. P. Anderson, C. Christensen, and B. Allen. "Designing a Runtime System for Volunteer Computing." in *Proc. ACM/IEEE SC*, 2006, Article No. 126

[2] T. Baar, D. Chiorean, A. Correa, M. Gogolla, H. Hußmann et.al. "Tool Support for OCL and Related Formalisms - Needs and Trends" in *Lecture Notes in Computer Science*, vol. 3844. J.-M. Bruel, Ed. Berlin Heidelberg: Springer-Verlag, 2006, pp.1-9

[3] O. Baskova, O. Gatsenko, G. Fedak, O. Lodygensky, and Y. Gordienko. "Porting Multiparametric MATLAB Application for Image and Video Processing to Desktop Grid for High-Performance Distributed Computing," presented at the 25th Int. Supercomputing Conf. (ISC), Hamburg, Germany, 2010

[4] Ç. Cirit and F. Buzluca. "A UML Profile for Role-Based Access Control," in *Proc. ACM SIN'09*, 2009, pp. 83-92

[5] "Multiphysics Modeling and Simulation Software – COMSOL." Internet: http://www.comsol.com [Oct. 29, 2011]

[6] D. Werthimer, J. Cobb, M. Lebofsky, D. Anderson, and E. Korpela. "SETI@HOME - massively distributed computing for SETI". *Computing in Science and Engineering*, vol. 3, pp. 78-83, Jan. 2001

[7] A. C. Marosi, Z. Balaton, and P. Kacsuk. "GenWrapper: A Generic Wrapper for Running Legacy Applications on Desktop Grids," in *Proc. IEEE-IPDPS*, 2009, pp. 1-6

[8] C. B. Ries, C. Schröder, and V. Grout. "Generation of an Integrated Development Environment (IDE) for Berkeley Open Infrastructure for Network Computing (BOINC)," in *Proc. SEIN*, 2011, pp. 67-76

[9] C. B. Ries and C. Schröder. "Public Resource Computing mit Boinc." Linux-Magazin, vol. 3, pp. 106-110, March 2011. Internet: lm-boinc.sourceforge.net

[10] C. B. Ries and C. Schröder. "ComsolGrid - A Framework For Performing Large-Scale Parameter Studies Using Comsol Multiphysics and Berkeley Open Infrastructure for Network Computing (BOINC)," in *Proc. COMSOL Conf.*, Paris, 2010

[11] C. B. Ries, T. Hilbig, and C. Schröder. "A Modeling Language Approach for the Abstraction of the Berkeley Open Infrastructure for Network Computing (BOINC) Framework," in *Proc. IEEE-IMCSIT*, 2010, pp. 663-670

[12] C. B. Ries. "ComsolGrid - Konzeption, Entwicklung und Implementierung eines Frameworks zur Kopplung von COMSOL Multiphysics und BOINC um hoch-skalierbare Parameterstudien zu erstellen." M.Sc. thesis, University of Applied Sciences Bielefeld, Germany, 2010.

[13] C. B. Ries. "UML Profile for Visu@lGrid (draft)." Internet: www.visualgrid.de/research/drafts/UMLProfileVG-draft.pdf

[14] "Samba – opening windows to a wider world." Internet: www.samba.org [Oct. 29, 2011]

[15] Object Management Group. "OMG Unified Modeling Language (OMG UML) Superstructure." formal/2010-05-05, May, 2010.

[16] Object Management Group. "OMG Systems Modeling Language (OMG SysML)." formal/2010-06-01, June, 2010

[17] Object Management Group. "OMG Profile For MARTE: Modeling And Analysis Of Real-time Embedded Systems." Version 1.1, June, 2011

[18] Object Management Group. "Object Constraint Language." Version 2.2, Feb., 2010

[19] "Setting up a BOINC server." Internet: boinc.berkeley.edu/trac/wiki/ServerIntro, July 25, 2011 [Oct. 29, 2011]