

# Towards On Demand IT Service Deployment

Jai Dayal<sup>2</sup>, Casey Rathbone<sup>2</sup>, Lizhe Wang<sup>1</sup>, and Gregor von Laszewski<sup>1</sup>

<sup>1</sup>Pervasive Technology Institute, Indiana University  
2729 E 10th St., Bloomington, IN 47408, U.S.A.

<sup>2</sup>Service Oriented Cyberinfrastructure Lab, Rochester Institute of Technology  
Bldg 74, Lomb Memorial Drive, Rochester, NY 14623-5608  
Email corresponding author: laszewski@gmail.com

**Abstract**—Advanced IT solutions allow users to create, organize and share user services and computing resources across a wide range of heterogeneous platforms. This makes managing (updating and deploying) the IT systems considerably more complex. Management tasks are so entangled within the existing components that it requires a IT professional with advanced training and intimate knowledge of the existing configuration to complete them.

In this chapter, we will present the deployment solution for a live complex IT system, the Emergency Services Directory (ESD). Many different technologies exist that attempt automate the application deployment process by allowing a user to describe the dependencies, provide the configuration parameters, and specify the application's required technologies, such as the operating system, database or Web server technology.

ESD's deployment solution takes advantage of the benefits provided by virtualization, and virtual appliances in particular. Each of ESD's components are wrapped and contained within a virtual appliance image. To automatically distribute and deploy the virtual appliance image on-demand, we utilize the Cyberaide Creative tool, which is a tool being developed in the on-going Cyberaide project.

## I. INTRODUCTION

Typically, an IT service or application consists of several components inter-operating to perform a set of functions or tasks. Service deployment is considered the process of acquisition and execution of the service, i.e., making the service ready for use [1]. Deployment typically consists of the release of the software, the configuration of the software, and the installation of the software [2].

Depending on the application, the deployment process can be quite complex requiring a user or developer to have a detailed understanding of the applications individual components. An IDC survey states that out of the \$95 billion dollars spent on application operating costs, 19% can be attributed to the cost of deployment alone [3].

With today's advanced IT infrastructures, such as compute Grids and Clouds, many users, who would benefit from their usage, avoid these technologies due to their high level of complexity and steep learning curve. The complexity and steep learning curve of these systems is largely due to the vast heterogeneity of the components and lack of more user-friendly software tools. Applications require specific operating system drivers and configurations as well as software libraries and packages for deployment [1]. In heterogeneous environments, access to resources matching the application's specific requirements can not always be guaranteed. Therefore a user

may be required to provide several version of the application to match the infrastructure's different resources which could be too cumbersome a task within a large environment [4].

To help assuage the problems within heterogeneous environments, users turn to virtual machine (VM) technology. VMs offer users [5]:

- Customized OS: The user has great flexibility to customize the operating system to meet the application's run-time requirements. These customizations can include, but are certainly not limited kernel level customizations to memory and storage customizations.
- Ease of Management: Virtual Machines can easily be shutdown or restarted, easing the system reconfiguration process. Additionally, VMs can easily be migrated to different physical machines, thus allowing the application to easily operate during a physical machine's downtime.
- System Security: VM users can define the operating system's privilege with out affecting the privileges of the underlying operating system. For example, root access is often required to install operating system modules. Installation of these modules only effect the VM. Additionally, if a configuration causes the system to crash, only the VM is affected, while the underlying machine and operating system remain operational.

While VMs provide us with a flexible and easy to deploy platform, VMs alone fall short of automatic and on-demand service deployment. For example, each time the VM is shutdown, the service will have to be re-deployed.

Virtual appliances take VMs a step further by containing both the platform and the service. Launching and executing virtual appliances, however, still requires the user to understand many technical details. For example, a user must understand which basic software packages or components are needed for the application. The user must also understand the functional dependencies between these packages. For example [1] stats that IBM DB2 has approximately 40 configuration parameters that must be resolved during the deployment process.

Our contribution uses the Cyberaide Creative tool to allow a user to select the appropriate virtual appliance image, and handles the resource selection, resolves the dependencies between components and packages, and automates the deployment, shielding the user from the process. Using ESD as a case study, we evaluate Cyberaide Creative's success.

The rest of this chapter is organized as follows. Section II presents some background information and compares our

solution to the existing solutions in the field. Section III discusses and formulates the deployment process. Cyberaide and Cyberaide Creative are presented in Section IV. Section V describes the ESD project, our case study, and Section VI presents our deployment solution. Section VII evaluates our solution and our conclusions are presented in Section VIII.

## II. BACKGROUND AND RELATED WORK

Automatic and on-demand service deployment is not a new idea there currently exist several tools to facilitate the deployment process. A common one is the package managers found in various Linux distributions such as Fedora and Ubuntu. These tools consist of a base repository containing a number of packages. These tools are fully capable of resolving dependencies amongst packages and fully automate the installation process. However, these tools are platform dependent and currently are not used in highly heterogeneous distributed environments.

Dearle [2] discusses the general concept of automatic on-demand service deployment, and states that the future of this paradigm will rely heavily on virtualization. Dearle also examines six current technologies that facilitate the automatic deployment of services. His discussion, however, does not present a architecture or framework, but suggests possible technological solutions.

Kecskemeti et al [4] presents a detailed framework to deploy a service in a heterogeneous environment, but the work seems to present the automatic deployment of computing platforms, such as the operating system and database technology. The work also does not provide a specific application and an evaluation to determine the effectiveness of their proposed framework.

In [1], Sun et al provide a good discussion and formulation of the deployment process. They also provide two typical service platforms, such as a LAMP stack on a single node, which closely relates to the ESD project, and a Web service architecture in a distributed environment. To facilitate the deployment process, they use virtual appliances. Their work, however, only provides the service platforms and does not discuss the steps needed to migrate a full application or service to the platform. Also, this work does not propose a design or framework for a tool to facilitate the deployment of the virtual appliance, which they appear to do manually.

Our work differs from the above studies by providing a framework that delivers two parts, the first is the automatic on-demand deployment of the service platform using virtual appliances, and the second is the on-demand automatic deployment of the service to the platform. We show how this can be achieved using Cyberaide Creative to deploy the virtual appliance, and a set of scripts to migrate the service codes and data.

## III. SERVICE DEPLOYMENT PROCESS

In this section, we formulate the general process of deployment and present a simple model to formulate and discuss the complexity of a deployment process.

### A. Deployment Process Overview

Deployment takes place at the end of the software life-cycle, hence it is a post-production activity [2]. There are many deployment guidelines, strategies, and tools, that typically share the same general deployment process [1]. These guidelines also tend to vary based on the type of deployed service. Services needing only a single machine typically have a more straight forward approach than do distributed services. Based on [1], [2], [6], we can generalize the deployment process:

- Determine the dependencies that exist in the deployment process. For example, if a component requires a database, the database installation and configuration should be included in the installation.
- Understand the communication and relationships between the different machines needed for the application.
- Resolve the dependencies at the platform layer, for example, the installation of a component might require a specific compiler.
- Install the application. This requires moving the source files, binaries, and etc. to the targeted environment. This step also requires the user to follow the dependencies required by the service.
- Activate the service and monitor the service's operation to identify any malfunctions.

### B. Deployment Model and Problem Definition

A deployment process consists of several identifiable elements, a service, a number of operations, and a set of dependencies which specify the order in which the operations must occur.

$$Deployment = \{Service, Operations, Dependencies\} \quad (1)$$

A software service is composed of any number of components, where a component is the minimum entity in the service. Formally, a service *service* is modeled as:

$$Service = \{component_i \mid 1 \leq i \leq I\} \quad (2)$$

where  $I$  is the number of individual components in the service. The set of operations *operations* can be formulated as:

$$Operations = \{op_j \mid 1 \leq j \leq J\} \quad (3)$$

where  $J$  is the total number of operations in the deployment process.

Typically, an operation  $op_j$  requires several configuration parameters *parameters*:

$$parameters = \{param_k \mid 0 \leq k \leq K\} \quad (4)$$

where  $K$  represents the number of configuration parameters for operation  $op_j$ .

The dependencies of a deployment process represent the order in which the operations must be performed. We model the dependencies as a directed graph, *Dependencies*:

$$Dependencies = \{Operations, D\} \quad (5)$$

where *Operations* is the set of vertices and  $D$  is a set of

edges, or dependencies, between operations:

$$D \subseteq \text{Operation } X \text{ Operation} \quad (6)$$

For example, a dependency  $(op_1, op_2)$  means that  $op_1$  depends on  $op_2$ , i.e.,  $op_2$  must happen before  $op_1$ . [7], [8] further discuss the modeling of software systems and dependencies. Figure 1 depicts a simple dependency graph. Deployment technologies such as RedHat’s RPM use dependency graphs to determine and resolve dependencies between software packages.

As we can see, as the number of components and operations grow, the more labor intensive the deployment process becomes for the user, especially in regards to the dependencies between operations. Our goal is not to reduce the number of dependencies or operations in a deployment process, but rather to *hide* the deployment complexity from the user.

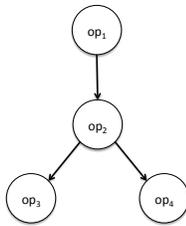


Fig. 1. Simple Dependency Graph

#### IV. THE CYBERAIDE PROJECT AND CYBERAIDE CREATIVE

This section provides an overview the Cyberaide project and Cyberaide Creative, a tool within the Cyberaide project. Cyberaide Creative is the tool we use to automatically and on-demand deploy our ESD service.

##### A. The Cyberaide Project

As mentioned in the introduction, users can benefit from complex IT infrastructures in a number of ways, but users often have a hesitant attitude towards using these technologies due to the amount of technical knowledge required. Cyberaide provides a possible solution to this problem. Several tools have been integrated into the Cyberaide project, such as the Cyberaide Toolkit and the Cyberaide Shell [9].

Cyberaide enjoys the following essential features [9]:

- *Ease of use*: make the JavaScript based API and interfaces useful for Grid and Web developers.
- *Low installation footprint*: support fast downloads as well as an easy maintenance through a small manageable code base.
- *Security*: gain access to Grid resources in order to avoid compromising the system. This is especially important due to known limitations of JavaScript.
- *Basic Grid functionality*: is provided for developers to create Grid-based client applications.
- *Advanced functionality*: is offered as many developers do not want to replicate functionality provided by other Grid middleware and upperware.

The framework is designed in layers and comprised of different components. (see also Fig. 2). A web client that provides access to Grid functionality and components that can be deployed in a web server are provided. A service called “mediator service” mediates tasks to the Grid and basically is a secure server that provides most of the functionalities in regard to the Grid.

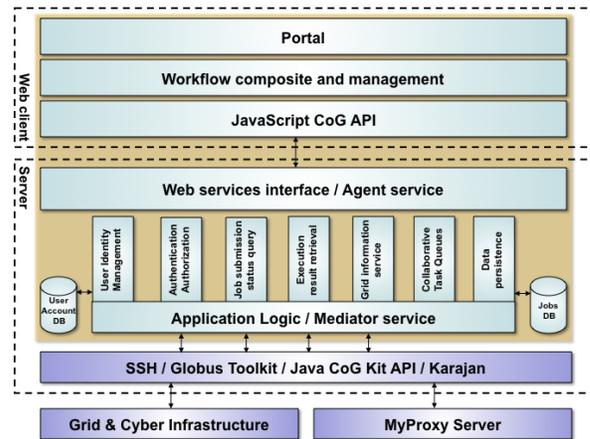


Fig. 2. System Architecture

- Web client: provides elementary functionality to access the Grid through a portal user interface.
- Server: contains two logical parts:
  - Agent service: is the intermediate service between Web client and mediator service; works as proxy for users to interact with the mediator service.
  - Mediator Service: is the bridge between the Grid and the client library. The mediator service offers different functionalities and contains the application logic.

Because of the separation between the service and the client the development of Cyberaide shell was possible. this is a system shell that facilitates the use of cyberinfrastructures. It contains four high level design components: object management, cyberinfrastructure backends, command line interpreters, and services (see Fig. 3) [9].

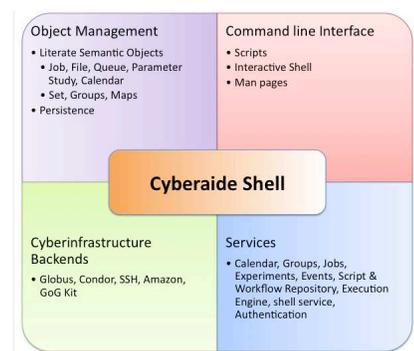


Fig. 3. High level design of Cyberaide Shell

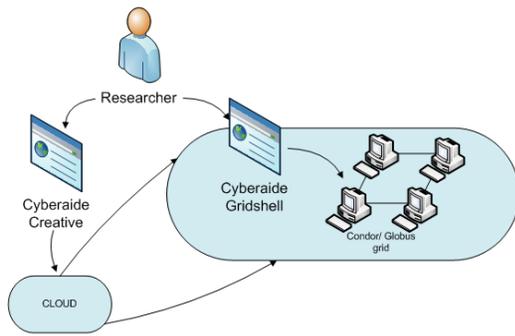


Fig. 4. Cyberaide Creative Paradigm [10]

### B. Cyberaide Creative Paradigm

The paradigm presented using Cyberaide Creative is an on-demand resource allocation system. This creates an environment where resources can be optimally utilized as demands request them. The Cyberaide Creative system is used as a tool for acquisition of the production grid running either Condor or Globus ToolKit and Cyberaide Gridshell is the operating interface to the grid. The benefits of this paradigm are the ability to outsource resources for less cost than to maintain a complete internal system for peak resource consumption.

Figure 4 shows the computing paradigm of Cyberaide Creative:

- 1) Users send requirement to Cyberaide creative to demand cyberinfrastructures from Clouds, for example, a condor cluster, or a computational Grid with Globus Toolkit as a middleware.
- 2) Cyberaide creative then construct a cyberinfrastructure for users, which is pre-installed some Grid middleware, like condor, Globus and Cyberaide shell.
- 3) Users then on demand access the cyberinfrastructure with aides of cyberaide shell.

[10] presents Cyberaide Creative's use cases. For our deployment, we are using the scenario where a user requests a single VM workstation. This work extends this scenario by deploying a fully configured and ready to use virtual appliance. Cyberaide Creative uses VM Ware ESXi technology to create and store the virtual appliance images. For a more in depth discussion on Cyberaide and Cyberaide Creative, see [9], [10].

## V. EMERGENCY SERVICES DIRECTORY

In this section, we will describe the ESD service and we will formulate the operations and parameters required to deploy this service.

### A. Overview

The Emergency Services Directory (ESD) is a nation wide directory containing information about emergency service resources, such as fire departments, ambulances, and law enforcement agencies. The directory aims to meet the needs of several different classes of users from the general public, to state and national level government officials. Each different class of user requires different levels of access to the data,

and also needs to perform role specific tasks. For example, a regional fire 5 chief may need to send an alert all near by fire stations, while ambulance providers may need to know what capabilities neighboring ambulance providers offer.

The directory also has an on-demand printing service that allows designated users to export information from the database into a customize format, such as a directory, booklet or pamphlet. The Society for Total Emergency Programs (STEP) council currently distributes a printed version of the directory to 911 dispatchers, ambulances, and other emergency service providers in Rochester, NY area. Additionally, the printing service must be scalable so multiple users can print on-demand customized documents simultaneously. The printing service consists of a set of Perl and shell scripts and can be operated independently from the Web site.

This real-life health-care application is categorized as a complex IT system as it relies on harmonious operation between independent technologies. Build on top of a LAMP (Linux, Apache, MySQL, and PHP), the on-line directory uses the Drupal content management system [6] to handle user requests, and to render the sites web pages. For added functionality, a developer can create custom PHP modules with in Drupal, or the developer can download and install pre-made modules supported by Drupal. ESD uses both methods.

Fine grained access to the site, and the underlying MySQL database, is controlled via role-based access control (RBAC). Drupal provides role-based access control out of the box, but this access control is limited only to content contained with in the Drupal system, such as the various Web pages and modules. Figure 5 shows ESD's design.

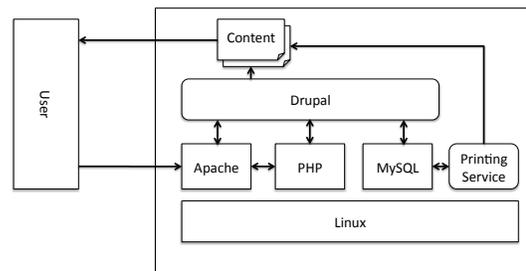


Fig. 5. ESD Design

ESD was chosen as a case study for this project because it is a real live example of a complex IT system, and has very detailed and well defined technical requirements. The deploy as a service paradigm must be able to accurately deploy a project with such detailed specifications.

### B. Deployment Strategy Overview

The first step in the deployment process is to determine the dependencies between the components and operations in the deployment process. First, we have our base operating system, which is Linux. For our distribution, we use Canonical's Ubuntu operating system. Next, the Drupal content management system requires a MySQL database, an Apache Web server with PHP capabilities. It does not matter which order we install MySQL and Apache, but PHP's installation

and configuration strongly depends on Apache. The printing service requires two tools, Perl and Latex. There are no dependencies between these tools, but naturally, they must be completed after the OS configuration. After we understand the dependencies, we can begin resolving the dependencies by performing the operations in order.

After the LAMP stack is in place, we can install and configure Drupal. Unfortunately, there are several steps in Drupal's configuration process that can not be automated, so the user will have to be present during the deployment to select some simple configuration options. Since Drupal is modular, however, it is possible to migrate any custom modules, codes, or themes by placing them in the appropriate location in Drupal's directory structure. Since the directory structure is well defined, the migration of the modules is included in our deployment scripts.

After the MySQL database has been installed, we can begin to load the database with the appropriate information. It may seem there are no dependencies between populating the MySQL database and installing Drupal, but Drupal maintains its user and site information in the database. The data containing information about ESD's emergency services is also stored in the MySQL database. While the ESD content is not related to Drupal, the process is simplified by having only one database migration step. If the ESD information and the Drupal information were stored in separate databases, then two steps may be required.

In summary, we can define the steps required for the ESD deployment process as follows:

- 1) Select the appropriate operating system or VM image
- 2) Install and configure Apache Server
- 3) Install and configure PHP support for Apache Server
- 4) Install and configure MySQL
- 5) Install Perl (usually not needed)
- 6) Install Latex for printing service
- 7) Install Printing Service source files
- 8) Install Drupal CMS
- 9) Configure Drupal CMS during installation
- 10) Migrate database contents for Drupal and ESD

Figure 6 displays the dependency chart for ESD's deployment process.

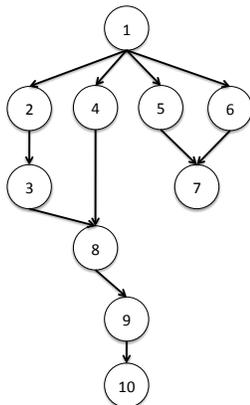


Fig. 6. ESD Dependency Graph

In the next section, we show how we can contain steps 1 - 6 within the virtual appliance. By adding a few installation scripts to the virtual appliance image, we can finish steps 7 - 9 automatically.

## VI. DEPLOYMENT SOLUTION

In this section, we describe how we create the virtual appliance on demand, and how we then deploy the application automatically. Both steps are implemented using the Cyberaide Creative tool.

According to VMWare, the creation of a virtual appliance can be generalized into three steps [11]: 1) Install the guest operating system; 2) Install the specific software, such as Apache and MySQL, needed by the service; 3) Provide an interface so the user can oversee and participate in the virtual appliance creation process.

As [11] points out, it is important that the user participates in the appliance creation process as there are several configuration parameters that require a user's input, for example the network configuration. Cyberaide Creative provides us with such an interface via a Web service.

Using Cyberaide Creative [10], here are the steps needed to deploy the virtual appliance and to move the service contents to the platform:

- 1) The user logs into the Cyberaide Creative Web service and selects the type of VM image required.
- 2) The Web service sends these parameters to the ESXi Server, which then provides the required VM image.
- 3) The user then specifies the packages needed and the dependencies between these packages. VMWare requires that the user specifies the dependencies.
- 4) The user provides the Web service with any custom installation or data migration scripts.
- 5) The Web service then forwards these requirements and scripts to the ESXi Server, which then installs the required packages.
- 6) The ESXi Server then launches the appliance to a host machine, runs the scripts and returns the address and login information to the Web service.
- 7) The Web service forwards the address and login information to the user. The user can now directly login to the instantiated appliance.

In our implementation, scripts were written to handle the download and installation of the Drupal CMS, the creation of the database tables needed for the ESD and Drupal applications, and then for the migration of the database contents.

It should be noted that the installation of Drupal is not completed until the user logs in to the virtual appliance and agrees to certain licenses and specifies certain parameters. It should also be noted, that the VMWare hypervisor only installs the appliance (OS and basic packages). The rest of the installation is done on the appliance via the provided scripts.

## VII. SOLUTION EVALUATION

Our goal is not reduce dependencies or operations in a deployment process, but it is to simply shield the user from the details of the deployment process. Furthermore, if the

deployment process is not automated, the user will have to repeat the deployment steps each time the service needs to be re-deployed. Using Cyberaide Creative, we can create a virtual appliance image, and store and retrieve the image as needed.

In the previous section, we discussed the operations required to configure and create the appliance on top of the VM image. This process only has to be performed once, as the fully configured and created image is then stored by VMWare ESXi. From this point on, the user only has to re-launch the appliance image, while all configurations, packages, and dependencies are retained.

An important metric is how long the on-demand automated deployment process takes. This metric is highly variable as it depends on the network connection, the virtual appliance size, and the amount of data to be transferred during the migration process. In our scenario, the basic virtual image and packages is around 600MB. The amount of data in the database was less than 20MB. Additionally, the appliance images and ESD data were stored at a local repository.

Another important metric deals with the overhead caused by using a virtual machine. In some cases, added overhead may not be acceptable. Many studies [5] have been conducted to evaluate overhead caused by virtual machines. Figure 7 displays the performance of virtual machines using Linpack to measure performance.

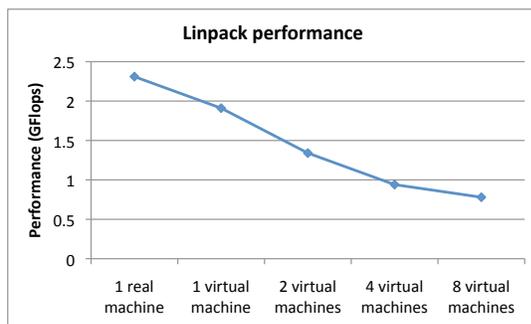


Fig. 7. VM Overhead [10]

## VIII. FUTURE WORK AND CONCLUSION

From the above discussion, we can see that there are still several steps in the deployment process that are not yet automated, such as the configuration of Drupal. Such scenarios are a result of the application itself, and possible cannot be handled by middleware.

There are several other steps that can be dealt with, such as the configuration of the virtual appliance to operate with the host machine's network. In the future, deployment projects should be extended to include a more general deployment description language. Several studies have investigated this process [12], [13]. These tools languages are typically XML based, but this is not required. [1] discusses a deployment description language, but only in the context of their framework. In the future, Cyberaide Creative plans to make use of a deployment description language.

In summary, we have shown that using the Cyberaide Creative tool, we can simplify a service's deployment process,

from the user's perspective. The user's service or application and a virtual machine image are combined into a virtual appliance. The virtual appliance is then deployed using Cyberaide Creative. Cyberaide Creative works to shield the user from the details of both the virtual appliance creation and deployment.

## REFERENCES

- [1] C. Sun, L. He, Q. Wang, and R. Willenborg, "Simplifying service deployment with virtual appliances," in *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 265–272.
- [2] A. Dearle, "Software deployment, past, present and future," in *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 269–284.
- [3] J. S. David, D. Schuff, and R. St. Louis, "Managing your total it cost of ownership," *Commun. ACM*, vol. 45, no. 1, pp. 101–106, 2002.
- [4] G. Kecskemeti, P. Kacsuk, G. Terstyanszky, T. Kiss, and T. Delaitre, "Automatic service deployment using virtualisation," *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, vol. 0, pp. 628–635, 2008.
- [5] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," in *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*. New York, NY, USA: ACM, 2006, pp. 125–134.
- [6] A. Aboulnaga, K. Salem, A. A. Soror, U. F. Minhas, P. Kokosiellis, and S. Kamath, "Deploying database appliances in the cloud." *IEEE Data Eng. Bull.*, vol. 32, no. 1, pp. 13–20, 2009. [Online]. Available: <http://dblp.uni-trier.de/db/journals/debu/debu32.html#AboulnagaSSMKK09>
- [7] C. Pich, L. Nachmanson, and G. G. Robertson, "Visual analysis of importance and grouping in software dependency graphs," in *SoftVis '08: Proceedings of the 4th ACM symposium on Software visualization*. New York, NY, USA: ACM, 2008, pp. 29–32.
- [8] P. Abate, J. Boender, R. Di Cosmo, and S. Zacchiroli, "Strong dependencies between software components," 2009. [Online]. Available: <http://www.citebase.org/abstract?id=oai:arXiv.org:0905.4226>
- [9] G. von Laszewski, F. Wang, A. Younge, X. He, Z. Guo, and M. Pierce, "Cyberaide JavaScript: A JavaScript Commodity Grid Kit," in *GCE08 at SC'08*. Austin, TX: IEEE, Nov. 16 2008. [Online]. Available: <http://cyberaide.googlecode.com/svn/trunk/papers/08-javascript/vonLaszewski-08-javascript.pdf>
- [10] C. Rathbone, L. Wang, and G. von Laszewski, "Cyberaide creative: Provision grid infrastructures in clouds."
- [11] VMWare, "Best practices for building virtual appliances," Tech. Rep. [Online]. Available: <http://www.vmware.com/resources/techresources/1011>
- [12] S. Lacour, C. Perez, and T. Priol, "Generic application description model: Toward automatic deployment of applications on computational grids," in *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 284–287.
- [13] W. Goscinski and D. Abramson, "Distributed ant: A system to support application deployment in the grid," in *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 436–443.