Performance Evaluation of a Generic Deployment Infrastructure for Component-based S/W Engineering

Abdelkrim BENAMAR
Department of computer sciences
University of Abou Bekr Belkaid
Tlemcen, Algeria
a_benamar@mail.univ-tlemcen.dz

Noureddine BELKHATIR
Adele S/W Eng. Team,
LIG Laboratory
University of Grenoble, France
Noureddine.Belkhatir@imag.fr

***Abstract*—We present a generic deployment infrastructure for distributed component-based applications. This infrastructure is based on OMG's deployment and configuration specification and model driven architecture paradigm. Even though our approach is experimented for enterprise Java beans model, it can be extended to other specific models. We suggest the use of a classical measurement method in decision making for the proposed generic deployment platform of component-based applications. This method is based on graph theory and k-median algorithm. It allows optimization of the cost of any transaction in component deployment planning.**

Keywords-deployment and configuration specification; model driven architecture; computer network graph.

I. INTRODUCTION

Software deployment [6] is a very complex and important process covering many activities. This complexity becomes more significant with the evolution of networks and component based systems. Many component based systems [13] are used both by industries and academics. We illustrate our approach on currently used industrial component systems, such as Enterprise Java Beans (EJB), Microsoft corporation .Net and OMG's CORBA Component Model (CCM).

In the following, we present a generic deployment infrastructure for distributed component-based applications. Furthermore, we layout a general method to design made-to-measure distances for any given deployment transactions. The optimal distances are computed with classical graph algorithms such as, k-median and contribute to the improvement of the decision making process for deployment of component-based applications.

The remainder of this paper is structured as follows: section II presents the state of the art on deployment of component-based systems. Section III focuses on the state of the practices. In section IV we synthesize our previous work [3] [17] on defining a generic deployment framework for component-based applications. In Section V, the main approaches to assessing the performance of distributed applications are reviewed. They are followed by a measurement method we apply to the deployment specifically for deployment planning. Finally the main achievements and perspectives are summarized in the concluding section.

II. STATE OF THE ART

Recently, due to the availability of high-speed networks and advances in packaging and interface technologies, there has been considerable interest in building deployment platforms for component-based applications [8] and evaluating the performance of distributed applications [9].

A. Building Deployment Platforms

Hnětynka [13] introduces the Deployment Factory (DF) and model-oriented environment, based on Deployment and Configuration (D&C) specification for deploying software components. Since the DF is based on a plug-in thought, the deployment of the existing component technologies becomes more easer.

Merle and Belkhatir [17] propose a distributed environment called ORYA, for deploying ordinary applications. In fact, ORYA supports the basic stages of deployment process, such as install, configure, reconfigure and uninstall. Nevertheless, the planning stage of ORYA is very simple, because it supports only the deployment of ordinary applications.

Deng et al. [7] introduce a deployment engine called DAnCE based on D&C specification. This environment is now under construction and supports just the deployment of CCM components. However, it does not provide functionalities of D&C specification.

III. STATE OF THE PRACTICE

In this section we survey the main deployment platforms for component-based applications (e.g., EJB, .Net and CCM) developed by industrials and used in practice. The complete comparative study presented in our previous work [3] proves the robustness of these models and therefore the rationality of selecting only them.

A. Corba Component Model (CCM)

CCM [20] is a component specification proposed by the international consortium called OMG. The objective of CCM is to facilitate the development of heterogeneous distributed components. In fact, the first specification of Corba was entirely oriented towards interoperability, so that all features related to the deployment were omitted. Nevertheless, the CCM 4.0 standard supports all the functionalities of software deployment and distribution. Precisely, this specification includes four models that are summarized hereafter:

- Abstract model: it designs the component interfaces and their interactions.
- Programming model: it designs the component code sources and their non-functional properties (e.g., transaction, persistence and security).
- Deployment model: it defines the component system assemblies.
- Execution model: it is represented by containers.

B. Enterprise Java Bean (EJB)

The EJB [18] is a framework developed by Sun Microsystems. The purpose of EJB is to allow the development of distributed and object-oriented applications in the Java language. Components in EJB are called beans. The bean interface is directly implemented in Java language. Each bean has two interfaces (e.g., remote, home). The remote interface allows performing the component business. The home interface allows the production of a novel component, or getting an existing component. Unlike CCM, the EJB specification includes two models that are summarized above:

- Abstract model: It represents the specification of component interfaces.
- Deployment model: It allows to assemble a component-based application, pack it into a package, and install it on selected sites.

C. Microsoft's .Net

The .Net is a framework developed by Microsoft Corporation. The objective of this framework is to provide the development of distributed applications. The .NET framework is based on the concept of class that is also called component. The class code is developed in classical programming languages (e.g., C#, visual Basic, Java…). The manifest file is created thanks to the classes' compilation process. All these files are packaged into another file called assembly that is manually deployed through network. In fact, the concept of assembly was introduced by Microsoft. They try to determine the versioning and deployment problems that were cause by the DLLs. Those one were known as DLL hell. The versioning problem appears like when a new application installs a new version of a shared component that is not backward compatible with the version already installed on the machine.

IV. TOWARD A GENERIC DEPLOYMENT PLATFORM

Although there are many environments for making unified the deployment of software component. None of them is generic sufficiently, and they do not perform automatically the deployment of heterogeneous applications. Furthermore, we suggest to use a generic methodology that makes unified the deployment component systems. More precisely, this methodology is based on a model transformation approach that employs suitable Platform Independent Model (PIM) and Platform Specific Model (PSM).

A. Model Transformation Overview

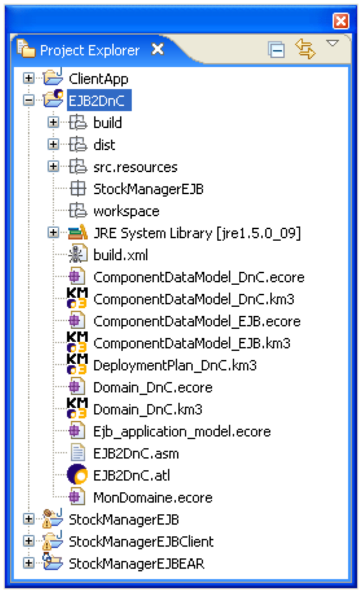
There are several projects aiming to make generic the deployment of software component. None of them fulfills completely the required features (e.g., release, install, activate, update, adapt) [6]. OMG contributes to the resolution of this problem with its D&C specification [19]. This specification matches the Model Driven Architecture (MDA) paradigm. This paradigm proposes a methodology to software development through modeling and transformation of models to code implementations. Among other approaches to model transformation, providing tools, we can mention VIATRA [23], Tefkat [10], AMW [5], ATL [14][4], Kent [1], and C-SAW [12].

B. Implementation

We outline in this section some implementation details. The prototype we developed relies on the D&C application meta models as PIM and EJB meta model as PSM. We use the Eclipse SDK,

In the following, we summarize the main tools (see Figure 1) used in the prototype. More details are given in [3]

- The Eclipse Modeling Framework (EMF) is used to develop the main project named 'EJB2DnC'.
- The Atlas Transformation Language (ATL) is EMF plug-in that is used for mapping meta models of EJB to D&C application.
- The Eclipse Web Tools Platform (WTP) is used to develop a specific EJB application named stock management.
- The Ant Build Tool (ABT) is tool used in EMF to run java applications.



```
ClientApp
EJB2DnC
    build
    dist
    src.resources
    StockManagerEJB
    workspace
    JRE System Library [jre1.5.0_09]
    build.xml
    ComponentDataModel_DnC.ecore
    ComponentDataModel_DnC.km3
    ComponentDataModel_EJB.ecore
    ComponentDataModel_EJB.km3
    DeploymentPlan_DnC.km3
    Domain_DnC.ecore
    Domain_DnC.km3
    Ejb_application_model.ecore
    EJB2DnC.asm
    EJB2DnC.atl
    MonDomaine.ecore
StockManagerEJB
StockManagerEJBClient
StockManagerEJBEAR
```

Figure 1. Project explorer view.

V. Performance Evaluation

This section highlights the currently used measurements to determine the performance of generic deployment platform. It proposes to use a classical measurement methodology for component-based applications, and proves the utility in decision making for deployment planning.

A. *Motivation*

As stated in [3] [6] [13], the deployment is a complex process constituted of many steps and activities, starting with the installation stage. Generally, the component is inserted into the target site (i.e, repository). The configuration stage succeeds the installation stage, and provides several configurations for further utilizations. During this stage, no deployment decisions (i.e, optimal placement and instance number) for components are performed. Naturally, these decisions are achieved in the planning stage. Therefore, we will contribute in planning stage by using measurement methodology in decision making for component deployment. Within the scope of this methodology, several decisions are carried out:

- Which component instance will be used?
- How many component instances will be deployed ?

B. *Related works*

There are several projects aiming to use end-to-end distances to achieve decision-making in computer networks. Nevertheless almost these works represent the end-to-end distance thanks to raw network metrics measurement. In below, we survey briefly some relevant examples:

Wolski et al. [25] present the Network Weather Service (NWS), which capture the condition of both network and hosts. It can provide the raw measurements of the classical metrics (e.g., bandwidth, latency, connection time, CPU availability) as well as forecasts based on aggregations of the set of raw measurements.

In AppLes project, Berman et al. [2] assume that each application is integrated with its own AppLes agent, which uses the performance model and dynamic information regarding resources to predict the run-time of its application on a given set of resources. Among a set of available possible candidate schedules, AppLes agent selects the one that is predicted to provide the best performance.

In Network Measurements Working Group (NMWG), Lowekamp et al. [16] highlight the used measurements to determine network performance for grid applications. They focus on a set of indicators as bandwidth, latency, throughput and CPU availability. They present also the characteristics of several measurement methodologies.

Seymour et al. [22] build a NetSolve infrastructure for providing domain-specific high-end network services. NetSolve provides a complete run-time infrastructure, as well as server management tools and client interfaces to languages as C, Fortran, Java, and MATLAB.

Karlsson and Mahalingam [15] present an illustration of using raw metric (e.g., latency and number of hops) for decision-making. More precisely, they propose an evaluation framework for replication algorithms. Moreover, they present a survey on replica placement algorithms with comparison study. Nevertheless, the used raw metrics seem to be quite irrelevant for monitoring the performances of high-level applications.

Qiang Xu [21] presents a use case of other raw metrics (e.g., latency and Round Trip Time) in grid environment. Furthermore, he proposes an approach for automatic hosts clustering, by mapping them to a geometric space. Even though the used raw metrics have advantages which are their stability and easiness of its measurement, they appear to be insufficient to supervise the performance of computer networks.

Gossa and Pierson [11] propose a novel technique to represent derived distances (e.g., computation task cost and data transfer cost) for any transaction in pervasive grid environment. The computation of these distances is based on the measurement of different raw metrics (e.g., latency and bandwidth) that can be provided by any monitoring systems. This work is set apart because it uses the derived metrics which are hard and expensive to measure. They appear to be pertinent on the topic of to data transfer concerns. In addition, the metric computation has been implanted in a grid service, called Network Distance Service (NDS) and developed with Globus Toolkit 4.

Therefore, we were very motivated by the last work [11] because authors use a derived distances which are well-suited with decision making for deployment planning.

C. *Overview of Measurement Representation*

Since networks are constituted of hosts and links, they can be represented in graph form. We define a network as a graph $G = (\upsilon, \varepsilon)$ where:

- υ is a set of vertices representing the hosts.
- ε is a set of edges representing paths between vertices that are labeled with measurements from source to destination hosts.

According to Lowekamp et al. [16], a metric is a quantity corresponding to the performance of computer networks. There are kinds of measurements (e.g., raw or derived). Raw measurements are something that can be measured easily such as measuring latency using pings. Derived measurements might be an aggregation on a set of low-level measurements. The main useful metrics are:

- the bandwidth (BW) in Megabits/second,
- the latency (L) in Milliseconds,
- the CPU availability (CPU_a) in percents,
- the free memory space (RAM) in Megabytes.

These observations can be represented by matrices called BW, L, CPU_c, CPU_a and RAM. We note $m_{i,j}$ the measurement of the metric m from the host i to the host j. Here, we assume that: $BW_{i,i} = \infty$ and $L_{i,i} = 0$ (i.e, the cost of local data transfer is null).

D. *Experimentation*

The objective of this section is to take the best decision for components placement related in the generic deployment platform (presented in the previous section). More precisely, our experimentations are made on a test network which is composed of four hosts (e.g., H_1, H_2, H_3 and H_4) and that is shown in Figure 2. We have evaluated our proposition on a classic planning scenario of component deployment. We forecast the effect of component data size with respect to their locations. Therefore, we consider different component sizes increasing from 1 (or 10^0) to 10^{10} with multiplier factor equal to 10.

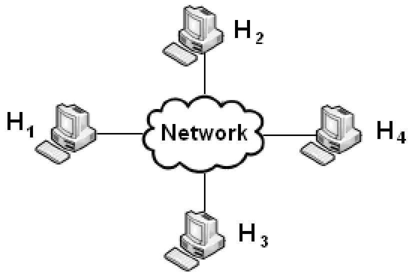


Figure 2. Deployment infrastructure.

Besides, we will assume that:

- all network hosts (e.g., H_1, H_2, H_3 and H_4) undertake the deployment of components,
- the component software is deployed on three hosts (e.g., H_1, H_2, and H_3)

These hypotheses are illustrated in Figure 3.

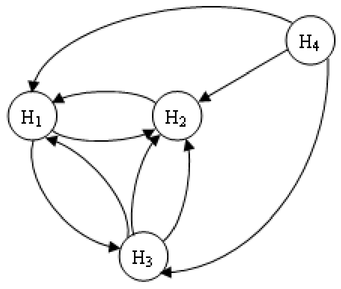


Figure 3. Graph of deployment infrastructure.

In order to optimize the time of component deployment, we only opt for transfer time. Thus, we use a compound metric called the Data Transfer Cost (DTC) [9], and represented by the formula:

$$DTC_{i,j}(\text{dataSize}) = \frac{\text{dataSize}}{BW_{i,j}} + 3(L_{i,j} + L_{j,i})$$

We use two measurement matrices corresponding to bandwidth (BW) and latency (L) (as shown in the Table 1) for computing the matrices DTC with respect to component sizes (as shown in the Table 2).

TABLE I. MEASUREMENT MATRICES

$$BW = \begin{pmatrix} \infty & 1.56 & 2.48 & 2.17 \\ 5.37 & \infty & 3.17 & 3.14 \\ 3.36 & 3.27 & \infty & 87.68 \\ 3.44 & 3.24 & 87.19 & \infty \end{pmatrix}$$

$$L = \begin{pmatrix} 0 & 16.5 & 10.0 & 9.5 \\ 16.5 & 0 & 15.6 & 15.2 \\ 9.8 & 15.7 & 0 & 15.2 \\ 10.0 & 15.7 & 0.6 & 0 \end{pmatrix}$$

TABLE II. REPRESENTATIVE RESULTS OF DTC CONPUTATION.

$$DTC(1) = \begin{pmatrix} 0 & 0.099 & 0.059 \\ 0.099 & 0 & 0.093 \\ 0.059 & 0.093 & 0 \\ 0.058 & 0.092 & 0.047 \end{pmatrix}$$

$$DTC(10^3) = \begin{pmatrix} 0 & 0.104 & 0.062 \\ 0.100 & 0 & 0.096 \\ 0.061 & 0.096 & 0 \\ 0.060 & 0.095 & 0.047 \end{pmatrix}$$

$$DTC(10^5) = \begin{pmatrix} 0 & 0.611 & 0.381 \\ 0.247 & 0 & 0.346 \\ 0.297 & 0.338 & 0 \\ 0.291 & 0.339 & 0.056 \end{pmatrix}$$

$$DTC(10^7) = \begin{pmatrix} 0 & 51.38 & 32.31 \\ 14.99 & 0 & 25.33 \\ 23.86 & 24.55 & 0 \\ 23.31 & 24.78 & 0.964 \end{pmatrix}$$

$$DTC(10^{10}) = \begin{pmatrix} 0 & 51282 & 32258 \\ 14897 & 0 & 25236 \\ 23809 & 24464 & 0 \\ 23255 & 24691 & 917.58 \end{pmatrix}$$

We will present distance computation which is based on graph algorithm, in addition of that, we will implement the classical algorithm to solve the k-median problem. The k-median problem (its implementation is designed in the subsequent Algorithm) is simply stated as:"Given a graph $G=(\upsilon, \varepsilon)$, find $\upsilon_k \subseteq \upsilon$ such that $|\upsilon_k|=k$, where k may either be variable or fixed, and that the sum of the shortest distances from the vertices in $\{\upsilon/\upsilon_k\}$ to their nearest vertex in υ_k is minimized".

***Algorithm** kmedians (k, σ, δ, d): best_solution*

```
Input data
    $k$: integer (number of hosts)
    $\sigma$: set of source vertices
    $\delta$: set of destination vertices
    $P_k(\delta)$: sub-set of source vertices such that $|P_k(\delta)| = k$
    $d$: matrix of $|\sigma|\times|\delta|$ real (DTC in our case)
Ouput data
    best_solution: set of vertices ($k$ best locations)
Method
    best_criterion $\leftarrow \infty$
    for all solution $\in P_k(\delta)$ do
        criterion $\leftarrow 0$
        for all $h_s = 1$ to $|\sigma|$ do
            min_dist $\leftarrow \infty$
            for all $h_d = 1$ to $k$ do
                if $d(h_s, solution(h_d)) <$ min_dist then
                    min_dist $\leftarrow d(h_s, solution(h_d))$
                end if
            end for
            criterion $\leftarrow$ criterion + min_dist
        end for
        if criterion < best_criterion then
            best_criterion $\leftarrow$ criterion
            best_solution $\leftarrow$ solution
        end if
    end for
    return best_solution
```

E. Synthesis

The optimal values of DTC related to component deployment are computed using the k-median algorithm, then we forecast their variations according component locations (see Figure 4).

Here are some observations based on these graphs:

- The performances should be as expected improved with more instances of components.
- If we want to limit the network to a single host, H3 appears to be the best location for components.
- The DTC with k=2 is roughly the half of the DTC for k=1. But the value with k=3 corresponds to a real improvement.
- If we consider all the sizes together, a real impact of DTC appears from 107. This is obvious because the cost of the transfer of very small data is negligible face to the cost of a large data transfer.

Therefore, we decide to place the component on the three hosts H_1, H_2 and H_3, since it is the best solution to ensure good performances of the generic deployment platform.

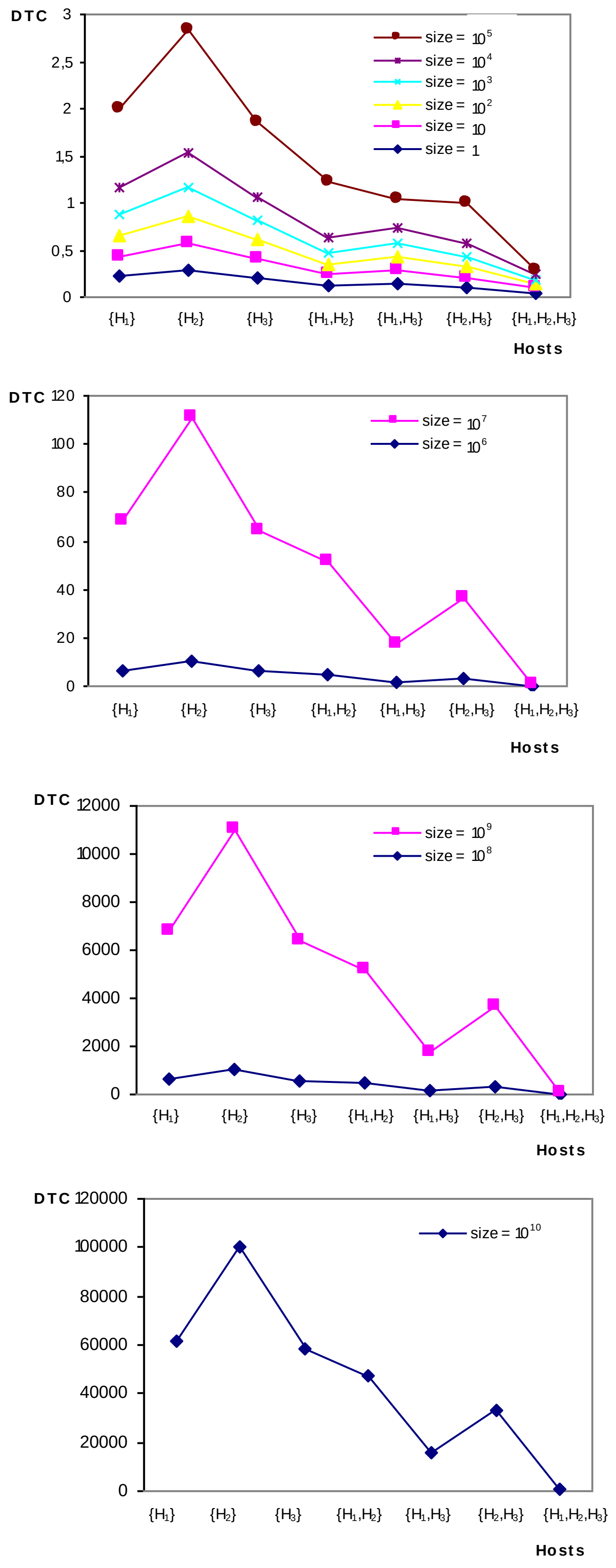


Figure 4. Variation of Component *DTC* According to their Locations.

VI. CONCLUSION

With the evolution of networks and software component, the deployment process becomes more complex and must cover the classical deployment activities (e.g., release, install, activate, update, adapt, de-install, de-release). Many component systems (e.g., EJB, .Net and CCM) currently exist. Therefore, a generic deployment model that wraps all these component systems would be indispensable. The main contributions of this study are twofold:

- Proposing a generic deployment infrastructure based on D&C specification and MDA approach. The proposed approach is tested with EJB model, but it can be obviously extended to other specific model.
- Applying a method designed to define made-to-measure distances for any given transaction network. The relevance of this provided distance is clearly enhanced by using graph algorithm.

Actually, experimentation is made by testing and evaluating the performance of EJB model deployment. Future research can be performed in various viewpoints.

We selected the most interesting ones:

- Integration of new component software and application architectures such as (e.g., CCM, service oriented architecture…).
- A
- Extending to others performance parameters such as Computation Task Cost (CTC) which take into account the complexity of the computation according to the request data size, the provider capacity and load.

REFERENCES

[1] D. H. Akehurst and S. J. H. Kent, "A Relational Approach to Defining Transformations in a Metamodel", Proc. Unified Modelling Language (UML 05), Springer Berlin/Heidelberg, 2005, pp. 243-258.

[2] F. Berman, "Adaptive Computing on the Grid using AppLes, " IEEE Transactions on parallel and distributed systems, vol. 14, 2003, pp. 369-82.

[3] A. Benamar, N. Belkhatir, and F. T. Bendimerad, "A Proposition of Generic Deployment Platform for Component-based Applications, " Journal of Software Engineering, Academic Journals Inc, vol. 2, 2008, pp. 23-38.

[4] J. Bézivin and F. Jouault, "Using ATL for checking models," Proc. Graph and Model Transformation (GraMoT 06), 2006, pp. 69-81.

[5] J. Bézivin, F. Jouault, P. Rosenthal, and P. Valduriez, "Modelling in the Large and Modelling in the Small," Proc. MDA Workshops Foundations and Applications (MDAFA 04), Springer Berlin/Heidelberg, 2004, pp. 33-46.

[6] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. Van der Hoek, and A. L. Wolf, "A Characterization Framework for Software Deployment Technologies, " Technical Report CU-CS-857-98, University of Colorado, 1998.

[7] G. Deng, J. Balasubramanian, W. Otte, D. C. Schmidt, and A. Gokhale, "DAnCE: A QoS-enabled Component Deployment and Configuration Engine, " Proc. Component Deployment (CD 05), Springer Berlin/Heidelberg, 2005, pp. 67-82.

[8] M. Dibo and N. Belkhatir, "Defining an Unified Meta Modeling Architecture for Deployment of Distributed Components-based Software Applications, " Proc. International Conference on Enterprise Information Systems, (ICEIS 10), SciTePress, vol. 1, 2010, pp. 316-321.

[9] M. Faerman, A. Su, R. Wolski, and F. Berman, "Adaptive Performance Prediction for Distributed Data-Intensive Applications, " Proc. High Performance Networking and Computing (HPNC 99), ACM/IEEE, 1999.

[10] A. Gerber, M. Lawley, K. Raymond, J. Steel, and A. Wood, " Transformation, the Missing Link of MDA, " Proc. Graph Transformation (GT 02), Springer Berlin/Heidelberg, 2002, pp. 90-105.

[11] J. Gossa and Jean-Marc Pierson, "End-To-End Distance Computation In Grid Environment by NDS, the Network Distance Service, " Proc. European Conference on Universal Multiservice Networks (ECUMN 07), IEEE Computer Society, 2007, pp. 210-222.

[12] J. Gray, Y. Lin, and J. Zhang, "Automating Change Evolution in Model-Driven Engineering, " Special issue on Model-Driven Engineering, IEEE Computer Society, vol. 39, 2006, pp. 51-58.

[13] P. Hnětynka, "Making Deployment of Distributed Component-based Software Unified, " Proc. Automated Software Engineering (ASE 04), Computer Society, 2004, pp. 157-161.

[14] F. Jouault and I. Kurtev, "Transforming Models with ATL, " Proc. Model-Driven Engineering Languages and Systems (MoDELS 05), Springer Berlin/Heidelberg, 2005, pp. 128-138.

[15] M. Karlsson and M. Mahalingam, "We Need Replica Placement Algorithms in Content Delivery Networks? " Proc. Web Content Caching and Distribution Workshop (WCW 02). Boulder Editions, 2002, pp. 117-128.

[16] B. Lowekamp, B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann, and M. Swany, "A Hierarchy of Network Performance Characteristics for Grid Applications and Services, " Proposed Recommendation Global Grid Forum (GGF), Network Measurement Working Group (NMWG), 2004.

[17] N. Merle and N. Belkhatir, "Open Architecture for Building Large Scale Deployment Systems, " Proc. Software Engineering Research and Practice (SERP 04), 2004, pp. 930-936.

[18] R. Monson-Haefel and B. Burke, Enterprise JavaBeans 3.0, O'Reilly Media, Inc, 5th Edition, USA, 2006

[19] OMG, "Deployment and Configuration of Component-based Distributed Applications Specification, " 2004, http://www.omg.org/docs/ptc/04-08-02.pdf

[20] OMG, "CORBA Component Model: CCM version 4.0, " 2006, http://www.omg.org/spec/CCM/4.0/PDF

[21] J.S. Qiang Xu, "Automatic Clustering of Grid Nodes, " Proc. Grid Computing (GC 05), IEEE/ACM, 2005, pp. 227-233.

[22] K. Seymour, A. YarKhan, S. Agrawal, and J. Dongarra, NetSolve: Grid Enabling Scientific Computing Environments, Grid Computing and New Frontiers of High Performance Processing, Lucio Grandinetti eds., Elsevier, Advances in Parallel Computing, vol. 14, 2005.

[23] D. Varró, G. Varró, and A. Pataricza, "Designing the automatic transformation of visual languages, " Science Computing Programming, vol. 44, 2002, pp. 205-227.

[24] A. J. A. Wang and K. Qian, Component-oriented Programming, 1st edition, John Wiley and Sons Inc., Chichester, UK, 2005.

[25] R. Wolski, N. T. Spring, and J. Hayes, "The Network Weather Service: a Distributed Resource Performance Forecasting Service for Meta-computing, Future Generation, " Computer Systems, vol. 15, 1999, pp.757-768.