A SOFTWARE ENGINEERING PERSPECTIVE FOR HIGHER QUALITY DISTRIBUTED DEVELOPMENT

Bojana Koteska
Faculty of Computer Science and Engineering
University Ss. Cyril and Methodius
Skopje, Macedonia

Anastas Mishev
Faculty of Computer Science and Engineering
University Ss. Cyril and Methodius
Skopje, Macedonia

ABSTRACT

The goal of this paper is to investigate the deficiencies of a distributed development process, especially a grid middleware software development and to propose some useful software engineering practices that will improve its quality. Considering that the distributed development approach is very useful for developing complex grid systems and scientific software, reducing the errors in the development process and improving its quality is a challenging problem for software engineers. Concerning to show the possible deficiencies which usually result in delay of the entire project or in the unsuccessful integration of the processes we consider the European Middleware Initiative (EMI) as our case study. Also, we found some specific problems that occur during the development of the two versions of the EMI products. That helped us to discover the reasons for those problems and to suggest some formal and standardized software engineering practices that will enhance the distributed and the grid middleware development process.

Keywords Grid Middleware, Distributed Development, Software Engineering, Higher Quality Applications.

I. INTRODUCTION

Most of the problems in different scientific areas are usually solved by partitioning the problem into smaller parts and by using grid computing and distributed development approach. The trend of using such a development approach grows parallel with the need to achieve a successful collaboration and coordination of the activities in an environment that allows high scalability and access to heterogeneous resources.

The distributed development becomes a more common approach because it provides a modern development environment and it enables coordination and collaboration between the participants in a project located at different places. Some of its biggest advantages are: a large and qualified resource pool, mixing of ideas and different cultures between the developers, reduced development costs, etc. [2]

Since the grid technology and grid computing which is an advance in distributed computing provide an effective infrastructure for sharing computing and data in a dynamic way its popularity has been growing rapidly [3]. The grid middleware is a mediation layer between the network and the applications in the grid infrastructure. It also manages the information exchange, security, and access. The development of the grid middleware software should also consider the complexity of the developing and managing grids [4].

To sum up, the development of the distributed software requires establishing a rich interaction and efficient communication which is an additional challenge for software engineers [1].

In [5], the authors proposed a research agenda for distributed software development, but it affects only several areas such as testing, software development tools, application knowledge management and process and metrics issues. Some strategies for the successful distributed development of physics grids are presented in [6]. In [7], the authors suggest new development paradigms for distributed systems which lead to the creation of new middleware platforms.

Our paper is focused on finding the appropriate software engineering practices for improving the quality of distributed software development, especially grid middleware development. Considering that the grid middleware software is a core for making successful collaborating in a distributed development environment, the development process should be organized and performed using some standards and formal principles.

Concerning to find the grid middleware quality issues in a real project scenario, we considered the EMI (European Middleware Initiative) as our case study. We found the problems that appeared in the two EMI released versions and that helped us to propose some methods and practices to solve them. The issues found in the distributed environments and the grids are also important for the grid middleware development because it follows the paradigm of the distributed development.

The paper is organized as follows: The second Section describes the specific characteristics of the distributed development and the grid which are helpful for discovering the possible development issues. These issues are presented in the third Section. The fourth Section refers to the grid middleware development problems that decrease the quality of the grid. The results from our case study (EMI project) are shown in the fifth Section. The software engineering practices for improving the grid middleware development are presented in the Section 6 and the conclusion and future work are given in the last Section.

II. SPECIFICATIONS OF DISTRIBUTED DEVELOPMENT AND GRID

The distributed software development process is different from the traditional centralized software development. The main difference between these two development approaches is the spatial organization of the development resources, including computers, people, network resources, etc.

Besides the good organization of resources, the distributed development requires a coordinated process of interaction between the members in the project. Also, the

purpose of the project, the size of the development team and software processes should be taken into account [8].

The distribution of the software can be performed in several ways. It is determined by the geographic location and project control. More in detail, the software project can be distributed in the same country where the headquarters is located or abroad. The project control can be organized in two ways: the company can buy external software or it can offer its own resources [9].

For the sake of implementation of the distributed development process efficiently, it is necessary to establish an effective communication, collaboration and control. Communication is defined as a process of exchanging information between the participants in the software project; collaboration is an organization of activities and tasks to achieve a single goal and control is a process of adhering to the quality standards, policies, formal principles, etc. [1].

The grid concept is a special kind of the distributed development concept. Some of the main grid characteristics [10] are presented in Table 1.

Table 1: Main grid characteristics.

Characteristic	Description
Large Scale	The grid must be able to deal with millions of resources.
Geographical distribution	The grid must provide a good concept for distribution of the resources at different locations.
Heterogeneity	Different kinds of software (files, programs, data, etc.) and hardware resources (computers, networks, scientific instruments, etc.) must be supported.
Resource sharing	Different organizations must allow access to their resources in the grid.
Multiple administrations	Each of the organizations must define specific access privileges to their own resources.
Resource coordination	To provide aggregated computing capabilities, there must be a coordination of the resources in the grid.
Transparent access	The accessing control should cause the whole grid to be seen as a single virtual computer.
Dependable access	The grid users must receive a high level of performance under the established QoS (Quality of Service) requirements.
Consistent access	To provide scalability and hiding of the resources heterogeneity, the grid must be built using standard protocols and services.
Pervasive access	The grid must provide maximum performance from the resources available to it.

III. Grid and Distributed Development issues

One of the biggest problems related to the grid concept is the coordination of resources which are usually heterogeneous and geographically distributed. Moreover, the resources can belong to different enterprises that have no knowledge of each other. The grid resources can be managed in different administrative domains that have dissimilar access and specific security policies. Also, the differences in the development environments such as processor architectures and operating systems exist [11].

Issues in the distributed development could be divided into several categories [9] [1] [8] [12]:

Table 2: Issues in distributed development.

Issues	Description
Cultural issues	Differences exist in the national (spoken language, values, practices), professional, organizational, technical, and team culture. These issues are the reason for a difficult communication between the participants in the project.
Communication Issues	The changes in a complex distributed infrastructure and different time zones often lead to decreasing the quality of the communication over the network.
Organizational issues	Geographically dispersed developers must be integrated efficiently into the project. It means that the organization of the development process and the management activities must be performed on time.
Knowledge management issues	Sharing the knowledge and experiences between the participants in projects is important for a successful development because it reduces the costs and the redundant work (Web based software such as Wikis are useful for work progress tracking and publishing).
Development process management issues	The requirements and specifications changing, the bad organization, and the informal communication between the members have a negative impact on the development process and productivity. The control of the overall system becomes more complex and it means that there must be a good management process.
Technical issues	There can be a lack of tasks synchronization due to the dispersed locations. The transmission of data could be critical because of the slow and unreliable network connections.
Risk management issues	The risk management of the distributed development process is more difficult because defects appear more frequently. Thus, additional effort for the risk management and control is required.
Bug and change	Bug and change tracking process is a

tracking issues	difficult process because of the distributed environment. It means that the need of a quality tracking system is essential.
End to end ownership issues	It can be unclear whose the responsibility is when any of the entities in the grid fails. This can happen because sometimes people from different organizations are responsible for the development of the same entity. The entity should be modified and tested again.

The resource heterogeneity and dynamism can result in faults and failures. Transporting the large amount of data across the grid network is risky and it requires additional effort for establishing a high-bandwidth connection for long periods [11].

Taking into account these requirements, it can be concluded that the grid infrastructure and the distributed development environment are different from the traditional software development environments. Concerning to provide more efficient and more quality development process, additional activities must be performed.

If most of these issues are solved, the grid infrastructure will provide collaborative engineering, distributed computing, high-throughput and data exploration [13].

IV. GRID MIDDLEWARE DEVELOPMENT

A. Characteristics

Middleware is software that provides a homogenized infrastructure and uniform access to all resources in the heterogeneous and geographically distributed grid environment [7].

Having in mind the complexity of the overall organization and the diversity of the entities involved in the development process, the need for management of the development activities and resources is essential. Thus, the main idea behind the development of grid middleware software is to enable consistent and effective computing environment.

The tasks of the grid middleware can be divided into five categories: job execution tasks, security, information and file transfer tasks, information tasks and file management [4]. The grid middleware is part of the typical grid architecture. The grid layered architecture is consisted of four layers: a fabric layer (computers, storage devices, networks, etc.); a core middleware layer (co-allocation of resources, remote process management, security and QoS); a user-level middleware layer (high-level abstraction provided by development environments and programming tools, scheduling tasks); an application level (applications and portals) [14].

B. Development problems

The development of the grid middleware software is a challenging problem for the software engineers because the software should be shared, reused, and extended by others in the future [4].

Primarily, the problems in the grid middleware development process are oriented to deployability and core functionality. These problems can result in job failures also in controlled and middle grid environments [16].

An additional challenge for the grid middleware developers is the system scalability and the quality requirements [17]. Problems can arise as a result of non implementation of quality standards, especially when the grid is intended for public use.

Other problems when developing the grid middleware are the ability to check the validity of the source code and the possibility to deal with faults and errors [18]. Checking the source code validity requires good process of testing, but it is not easy to be done because the grid middleware must be developed to support heterogeneous and complex environments.

Providing system interoperability and service autonomy can cause a number of unexpected errors because it depends on factors such as: enabling different deployment scenarios within the existing grid infrastructure, possibility to use the services as stand alone entities in different context, etc. [19].

To sum up, developing the grid middleware software requires knowledge about: grid infrastructure, services it provides, used development technologies, standards that must be taken into consideration when coding, testing, etc. Additionally, the developers must take care of the available resources and performance of the system.

V. EUROPEAN MIDDLEWARE INITIATIVE (EMI): A CASE STUDY OF DEVELOPMENT ISSUES

A. The EMI project

The European Middleware Initiative (EMI) is one of the leading software platforms for a distributed scientific computing which supports different middleware experts, engineers, and developers to work together to produce an interoperable middleware solution. The EMI project arose as a result of the close collaboration between the three major middleware providers (ARC, gLite and UNICORE) and other software providers like dCache. There are total 24 partners from 18 different countries. The duration of the project is three years. Its main idea is to deliver a set of middleware components for deployment in the EGI (European Grid Infrastructure) and the other distributed infrastructures. There are three major releases (Kebnekaise, Matterhorn, and Monte Bianco) which are delivered once a year and provide services to satisfy the needs of different scientific communities such as: Computational Physics and Chemistry, Astronomy, Earth Sciences, Geography, Life Sciences, Geophysics, Astrophysics, Material Sciences, Multimedia, etc. [20] [21] [22].

B. Quality issues

The major releases of the EMI project offer a balance between conflicting requirements and innovations. Every release and updates include new services and functionalities which are based on the new user requirements or the existing functionalities and services are being improved. There is

always a trade between new research and system stability [20] [21].

The development of the middleware software is not easy due to the different requirements, technologies and time limitations. The primary goal of the updates is to reduce the errors and faults found in the previous software versions. To deal with them, the development of the EMI middleware software follows quality standards and software engineering practices. However, the complexity of the system, resource and time limitations cause some deficiencies in the development process. Some of the found deficiencies are shown in the Table 3.

Table 3: EMI quality issues

Issue	Quality problem description
Poor documentation	No documentation was available for the installation of the external dependencies in ETICS or for the installation of the external dependencies for the deployment of the EMI components [23].
No definition of requirements for some EMI services	- monitoring interactive jobs - new features for Unicore Target System - support for VO (Virtual Organization) renaming and migration - forcing users to choose a group during VO registration - CEs (Computing Elements) should support a set of LRMS (Local Resource Management System) - Publish mw service version [24]
Late delivery of documents	EMI-0 Software Release Schedule was released with 2 months of delay [25] -Filling User Requirements and Technical Objectives -EMI 1 Defining Release Schedule
Late delivery of software	-Developing features, implementing bug fixes [25]
Late testing	-Testing and certifying developed components [25]
No test plan for several EMI components	A-Rex; ARC Compute Elements; ARC Data Clientlibs; ARC Infosys; ARC Security utilities; ARC Container; DGAS; StoRM [25]
No components regression test suite	ARC components do not provide a regression test suite. Only libarcdata provides some regression testing using CPPUnit [26]
No complete validation of A-REX EMI service	-No complete validation of the activity description -No schema validation -Partial service capabilities validation [27]
No test plan complaint with template	Not all the product teams succeeded in providing a test plan compliant with the template defined in the SA Testing Policy [15].
No unit testing and code coverage	Most of the ARC and gLite components do not provide unit testing. Otherwise, when unit testing is performed, the code coverage is not measured [28].
No difference between regression, unit and functional testing	UNICORE component test suite does not distinguish between the regression, unit and functional test [28].

VI. Improving the Quality of Distributed and Grid Middleware Development

The research showed that the development process of distributed software, compared to centralized software, requires improvement of additional activities such as: communication, collaboration, planning, organization, security, testing, etc. Analysing the results from the EMI project showed that most of the problems during the grid middleware development arise as a result of the late and bad organization, no proper testing plans, and insufficient documentation.

To improve the quality of the distributed development and the grid, we propose some software engineering practices that meet the problems described in the previous sections. SE practices can be split into three categories: organization, development, and testing.

The organization part is composed of several activities: organization of the available resources, making development plan, and communication and collaboration activities. This is an important step in the distributed environment primarily because of the geographically distributed resources and project size and complexity. One way to improve the organization of the development process is to increase the communication and collaboration between the participants. Videoconferencing and online interactions are suitable for establishing successful communication and collaboration. Creation of the activity plan should be based on standards that are generally accepted and also for those who will be established internally within the project.

Software development in a distributed environment requires increased interaction between the developers, documenting every part of the development process, and integration of software components. Integration of the parts of the software is a process that needs proper development of each of the parts individually. It is necessary to make templates for the documents, particularly those where changes of the source code and software requirements are described. Despite this, the available resources must be used efficiently and optimally. The resource limits and nonfunctional requirements must be taken into account.

The grid middleware is the most critical part in the development section where core services are created to connect applications and hardware resources. After the development of each component, there must be adequate documentation for its installation and use.

There must be a test plan and test reports must be made after each performed test. Testing the distributed software requires additional validation activities. It is recommended tests to be made before the source code is written. In addition, unit testing must be performed and all requirements should be satisfied. It is necessary to perform integration testing which

will verify the validity of the system as a whole. After each made change of the source code, especially during the development of scientific software, regression testing must be applied. Testing of the grid middleware software must be made often and in accordance with the testing plan within the project.

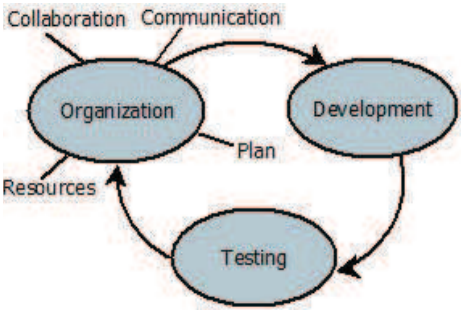

Figure 1: Iterative distributed development process.

An iterative development is the best solution for successful distributed development process. The global steps of the development process are shown on Fig.1

The quality of the distributed development depends on factors that directly or indirectly take part in the development process. Besides the quality standards for software development, the development experience in this field is essential.

VII. Conclusion

This paper outlines the most specific grid and distributed development characteristics and some of the problems that usually arise during the development process. The research is also oriented to the grid middleware developing deficiencies. To confirm the grid middleware development issues in practice, the most common problems in the development of the two versions of the EMI project are presented. Taking into account the results from the EMI project, some software engineering practices to improve the quality of grid middleware and distributed development are proposed.

It will be useful as future work to find out how and which of the detected issues will be solved in the third EMI version, how these solutions can be improved and what the benefits of those improvements will be.

References

[1] F. Lanubile, "*Collaboration in Distributed Software Developmen*". In *Software Engineering*, Andrea Lucia and Filomena Ferrucci (Eds.). Lecture Notes In Computer Science, Vol. 5413. Springer-Verlag, Berlin, Heidelberg pp.174-193, 2009.

[2] R. Kommeren and P. Parviainen, "Philips experiences in global distributed software development". *Empirical Softw. Engg.* 12, pp. 647-660, December 2007.

[3] A. Kyriakidou-Zacharoudiou, "Distributed development of large-scale distributed systems: the case of the particle physics grid". *PhD thesis*, The London School of Economics and Political Science (LSE), 2011.

[4] G. Von Laszewski and K. Amin, in *Architecture*, (Wiley, 2004), pp. 109-130.

[5] B.Sengupta, S. Chandra, and V. Sinha, "A research agenda for distributed software development". In *Proceedings of the 28th international conference on Software engineering* (ICSE '06). ACM, New York, NY, USA, 731-740.

[6] A.Kyriakidou and W.Venters, "Distributed large-scale systems Development: Exploring the collaborative development of the particle physics Grid", *5th International conference on e-Social Science*, Cologne, 2009.

[7] V. Issarny, M. Caporuscio, and N. Georgantas, "A Perspective on the Future of Middleware-based Software Engineering". In *2007 Future of Software Engineering* (FOSE '07). IEEE Computer Society, Washington, DC, USA, 244-258.

[8] L. Keil and P. Eng," Experiences in distributed development: a case study". *In: The InternationalWorkshop on Global Software Development, Portland*, OR USA, pp. 44-47, 2003.

[9] Journal of Computer Science and Engineering,Volume 1, Issue 1, p10-17, May 2010

[10] M. L. Bote-Lorenzo, Y. A. Imitriadis And E. Gómez-Sánchez, "Grid Characteristics And Uses: A Grid Definition", In First European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2004, pp 291-298

[11] C. Dabrowski, "Reliability in grid computing systems". *Concurr. Comput. : Pract. Exper.* 21, vol. 8, 927-959, June 2009.

[12] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. "Does distributed development affect software quality? An empirical case study of Windows Vista". In *Proceedings of the 31st International Conference on Software Engineering*(ICSE '09). IEEE Computer Society, Washington, DC, USA, 518-528, 2009.

[13] M. Baker, Rajkumar Buyya, and D. Laforenza, "Grids and grid technologies for wide-area distributed computing". *Softw. Pract. Exper.* 32, 15 1437-1466, December 2002.

[14] P. Asadzadeh et al.,"Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies", High performance computing: paradigm and infrastructure, laurence yang and minyi guo (eds), ISBN: 0-471-65471-X,2005.

[15] A. Ceccanti, Djra1.7.2 – Software Development Quality Control Report V1.0 Eu Deliverable: D5.7.2, 24/05/2011, http://cds.cern.ch/record/1277534/files/EMI-DJRA1.7.2-1277534-Software_Development_Quality_Control_Report_M12-v1.0.pdf

[16] A. Iosup and D. Epema, "Build-and-Test Workloads for Grid Middleware: Problem, Analysis, and Applications". In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid* (CCGRID '07). IEEE Computer Society, Washington, DC, USA, pp. 205-213

[17] S. Gorlatch and J. Dünnweber, "From Grid Middleware to Grid Applications: Bridging the Gap with Hocs". In *Future Generation Grids: Proceedings of the Workshop on Future Generation Grids, November 1-5, 2004, Dagstuhl, Germany*. pp. 241-261, Springer, 2004

[18] P. Collet, F. Křikava, J. Montagnat, M. Blay-Fornarino and D. Manset, " Issues and scenarios for self-managing grid middleware". In *Proceedings of the 2nd workshop on Grids meets autonomic computing* (GMAC '10). ACM, New York, NY, USA, pp. 1-10

[19] F. Hemmer et al., "Middleware for the Next Generation Grid Infrastructure", In: Proceedings of CHEP 2004, Intelaken, Switzerland

[20] EMI European Middleware Initiative, *http://www.eu-emi.eu/*

[21] A.Di Meglio (CERN), "European Middleware Initiative (EMI)", https://register.nordu.net/speakers/files/AlbertoDiMeglio.pdf

[22] e-Science city, *http://www.gridcafe.org/*

[23] G. Fiameni, Review of Software Release Plan v1.0, 29/10/2010, https://twiki.cern.ch/twiki/pub/EMI/SA1QCPM6/EMI-PM6-Review_Software_Release_Plan_v1.0.pdf

[24] B. Kónya, Technical Development Plan V1.0 Eu Deliverable: D1.3.2, 23/5/2011, https://twiki.cern.ch/twiki/pub/EMI/DeliverableDNA132/EMI-DNA1.3.2-1277543-Technical_Development_Plan-v1.0.pdf

[25] Software Maintenance Quality Control Report v1.0 Eu Deliverable: D3.3.2, https://twiki.cern.ch/twiki/pub/EMI/DeliverableDSA132/EMI-D3.3.2-Software_Maintenance_Quality_Control_Report_v1.0.pdf

[26] A. Ceccanti, Djra1.7.1 - Software Development Quality Control Report V1.0 Eu Deliverable: D5.7.1, 19/10/2010, https://twiki.cern.ch/twiki/pub/EMI/DeliverableDJRA171/EMI-DJRA1.7.1-1277533-Quality_Control_Report-v1.0.pdf

[27] M. Cecchi, Djra1.1.3 - Compute Area Work Plan And Status Report V1.0 Eu Deliverable: D5.1.3, 24/04/12, https://twiki.cern.ch/twiki/pub/EMI/DeliverableDJRA113/EMI-DJRA1.1.3-1277612-Compute_Area_Work_Plan-M24-v1.0.pdf

[28] A. Ceccanti, - Software Development Quality Control Report, http://cds.cern.ch/record/1277533/files/EMI-DJRA1.7.1-1277533-Quality_Control_Report-v1.0.pdf