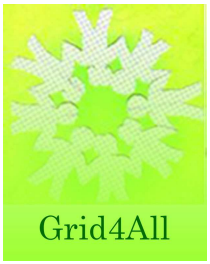




Project no. 034567

# Grid4All

Specific Targeted Research Project (STREP)

Thematic Priority 2: Information Society Technologies

# Deliverable D5.3 – Final
# Proof of concept Implementation and Evaluation Report

Due date of deliverable: 01 June 2009

Actual submission date: 27 July 2009

Start date of project: 1 June 2006 Duration: 37 months

Organisation name of lead contractor for this deliverable: FT

Contributors: Grid4All consortium

Editors: Ruby Krishnaswamy, Daniel Stern Release: 0.1

| **Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | |
| **PP** | Restricted to other programme participants (including the Commission Services) | X |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

# Table of Contents

# 1. Executive Summary

This document is part of a research project partially funded by the IST programme of the European Commission as project number IST-FP6-034567. The WP5 plans and supervises integration and evaluation tasks. While complete integration of all developed components with aim to produce a common software platform that can be deployed to execute specific scenarios is out of scope, software components that are developed within the project adhere to common vision of architecture for a democratic grid.

All Grid4all components adhere to a common vision and architecture; and Grid4All has been designed as a set of software results which may be combined and aggregated so as to produce higher-level capabilities. For example, proof-of-concept demonstration of access-control enforcement has been shown by integrating VOFS with the services provided by the Grid4All security infrastructure. However within the project (and as explained in revised DoW), all functionalities have not been combined to produce a common software platform dedicated to a specific use case or pilot study (e.g. common eLearning platform).

Consequently, it has to be pointed out that although the title of this deliverable refers to a “final integrated proof of concept implementation”, as explained in the revised DOW, we have focused on the most innovative integrations.

We have conducted two forms of evaluation of software modules produced within the project:

- qualitative: questionnaire-based evaluation; different stake-holders, end-users or developers external to the project have used and exercised the software results; feedback was retrieved by requesting the evaluators to fill questionnaires and also through informal discussions between experts and the evaluators.
- quantitative: technical evaluations provide quantitative metrics of different software; for each software result, quantitative evaluation has focused on the most meaningful metrics (in most cases performance).

This evaluation has been carried in accordance with the evaluation plans described within D5.4.

The results show that while ergonomic improvements and mature documentation are necessary in several cases and in general software results are still mainly at stage of technology pre-view, overall the results technically **fulfill their initial objectives; and when comparisons with existing solutions are possible, Grid4All, in the context of the democratized grid, provides better functionality and/or improved performance**.

# 2. Introduction

The objective of WP5 is to evaluate the software modules produced within the project. The question central to the evaluation is: “are the Grid4All results adequate to fit the stakeholder's objectives in the context of democratised grids”?

Different types of stakeholders are implied within the targeted ecosystem:

- **end-users (EU)** are the very final users of the software platform; they create virtual organisations or are members of VOs; they may play different roles such as users of applications or system administrators;
- **resource providers (RP)** own resources and contribute their resources to virtual organisations that are already formed; most resource providers are expected to be non-professional, but virtual organisations may also allocate on-demand resources provided by professional operators.
- **application and service developers (D)** use programming interfaces and functionalities provided by the Grid4All software platform to design and implement new applications or adapt existing applications.
- **platform operators (PO)** exploit the produced software to offer platform services such as the market place, identity management services, information services;
- **service providers (SP)** propose services to any end-user, e.g. e-learning, communication tools and software, digital libraries, etc.

The evaluation is constituted of two parts:

- quantitative: a set of technical evaluations assessing the adequacy of specific use cases against a set of requirements presented in D5.4. The inputs in form of user requirements as presented in D4.7 were used to determine quantitative evaluation plan.
- qualitative: questionnaire based evaluation aimed at validating the project objectives and technical solutions against needs of stakeholders. Feedback from evaluators aid the process of prioritizing future work.

As an input to evaluation, WP5 assumes that individual software modules, providing functionality necessary to realize the overall integrated capabilities, have already been tested. Functionally the software modules have been documented either in their respective deliverables or through use cases in D5.4.

All Grid4all components adhere to the common vision and a common architecture. As a consequence Grid4All has not been designed as a single software product, but as an aggregation of multiple software results, each of them providing a desired functionality. These functional modules may be combined to produce more capabilities. Within this project (and as explained in revised DoW), all functionalities have not been combined to produce a common software platform dedicated to a specific use case or pilot study (e.g. common eLearning platform). But we have shown in several scenarios how the functionalities can indeed be combined. Most innovative and meaningful integrations have been done and demonstrated

We have grouped at chapter 3 the results of qualitative evaluations and at chapter 4 **Erreur ! Source du renvoi introuvable.**the results of quantitative evaluations.

D5.4b had described the G4all modules and their principal "elementary" use cases. As a reference we recall in table below the list and a short description of the modules.

| Domain | Modules & Services | Functionality |
|---|---|---|
| **VO Management**: administrate a VO, its members, its resources | Membership Manager | Authenticate members<br>Maintain member info |
| | Resource Manager | Discover resources<br>Allocate resources<br>Donate resources<br>Monitor resources |
| | Deployment Manager | Deploy application |
| | Reservation Manager | Reserve resources |
| **Inter-VO services**: allocate or offer leases for | Market Information Service | Query information<br>Subscribe |

| Domain | Modules & Services | Functionality |
|---|---|---|
| | Market factory | Select market<br>Deploy market |
| | Currency and payment manager | Transfer currency |
| | Agreement Manager | Settle agreement<br>Distribute agreement |
| | Negotiator | Interact with markets |
| | Auction service | Register<br>Treat asks/bids |
| **Security**: manage security | Identity manager (VOMS) | Sign-on and create a proxy certificate with |
| | Policy Administration Point | Create/edit/store VO policies by VO admin |
| | Policy Repository | |
| | Policy Editor | |
| | Policy Enforcement Point | Access to a VO resource(s) protected with a PEP(s) |
| | Policy Decision Point | |
| | Policy Information Point | |
| | PEP, PDP, PR, PIP, PAP | Set/view ACL for a VO resource. e.g. VOFS directory, by a application |
| **Collaborative & Federative services**: manage VO-wide file systems; provide support for collaborative editing of shared documents | Telex | Create/read/write/delete collaborative documents |
| | VOFS Manager | Maintain a VO-wide workspace |
| | VOFS User Agent | Maintain VOFS membership |
| | DFS File Server | Serve user files |
| | VBS Storage Server | Serve User storage |
| **Execution Management** | Scheduling Service | Computes schedules |
| | XtremWeb Server | Manages distributed execution |
| | XtremWeb Worker | Performs local computation |
| **Overlay Services – DCMS** | DCMS | Executing Self-* Component-Based Applications |
| **Information Services** | Matching Service | Register service request/offers, |
| | Selection Service | |
| **Application**: support users to perform tasks within a VO | Application manager | Collaborative File Sharing<br>Collab. Netw Simulation<br>Shared Calendar<br>eMeeting<br>Gmovie |

A selective set of integration have permitted the project to demonstrate key integrated capabilities here below listed.

- Niche Distributed Component Management System uses the ADL-based deployment service for Fractal-based applications. Initial deployment of distributed applications specified (easily) using the ADL. Self-management requires writing code in Niche; deployment actions may reuse ADL scripts embedded within application packages. On top of Niche two applications have been developed and evaluated: YASS and YACS, illustrating, either for a file-system or for a computation task dispatcher, how advantageous can be a component-based model of management as Niche.
    - Within Grid4All, the compute nodes are expected to be drawn from the Internet. These may be used to execute computational jobs or even provide collaborative services or file sharing services. An important aspect is that such category of applications requires minimum manual intervention when there are churn (voluntary or failures). With demonstrators such as YACS and YASS, developed using Niche, we gain confidence that a large category of applications may also be developed using Niche. Secondly a large category of applications (gMovie, CNSE, etc) follow bag-of-task models. YACS is a service that can be used to execute and manage such applications.
- Telex and VOFS mutually serve each other. While VOFS by itself provides a simple way for collaborating peers to share their files/content, by itself it is not sufficient. Telex supports co-operative work by giving consistency guarantees that programs may rely upon. Telex implementation uses a construct specifically designed in VOFS.
    - This integration is important for two reasons. On the one hand Telex relies on the multilog abstraction provided by VOFS for its implementation and on the other hand through applications

such as CFS we show that coherency and consistency management could be moved to higher layers. VOFS by itself does not handle consistency in sharing and is hence lightweight and suitable in environments where disconnection is frequent. Telex adds this capability.

- Both the Fuse-based Linux version and the WebDav-VOFS use the Grid4All security infrastructure to implement access control (authentication and authorization).
    - The Grid4All security infrastructure, which by itself is not innovative, its XACML-based implementation allows easy plugging of access control policies for VOFS. While clearly security is pertinent to a large number of resources and services, it was essential to show that access-control could be implemented even for peer-to-peer file sharing services such as VOFS.
- Shared Calendar and CFS (collaborative file service) both use Telex. CFS uses VOFS as well to share files. The Sakura SC application allows any user to share her calendar, to create and invite users to meetings, to modify times or to cancel propositions. In contrast to systems such as Doodle, Sakura (using Telex) ensures consistency even when a user is engaged in different meetings. CFS is a graphical application providing workspaces capabilities. Using Telex, centralized servers to manage coherent workspace views are not required.
    - SC and CFS are two important class of applications required in any collaboration. Through these two applications we show how Telex may be used in applications that are executed in collaboration environments where disconnections occur.
- gMovie demonstrator has shown how different Grid4All middleware and services may be used even for legacy applications. It uses VOFS (to transfer input and output files), ADL-based deployment (to start XtremWeb worker tasks), and the scheduling service to estimate worst-case completion time.
    - This integration has permitted to understand how the different disparate set of services offered by Grid4All could in fact be used to design and implement distributed legacy applications. This shows that loose-coupling between functional modules is indeed an advantage since the usage may be driven by the needs of the application.
- A “Negotiator”, negotiation agent exercising the API of the Grid4All market place, implements the Reservation Management API. Application Managers use the Reservation Management API to provision resources on demand. Applications need not be concerned with market specific APIs or strategies; they are neither concerned with low-level protocols by which remote nodes join the buyer's overlay (VO).
    - Sophisticated implementations of negotiation agents are possible, but such implementations are specific to the requirements of the scenario; the integration aimed at showing simplicity of usage for applications.
- The Market Information Service (MIS) and Configurable Auction Server (CAS) are integrated, i.e. the CAS uses the API of the MIS to publish market information. Both these components are deployed using ADL-based deployment service.
    - The MIS and CAS were conceived early in the project as two important tools to build distributed market places. Their integration was necessary to assess that timely dissemination of market signals, which is essential for survival, durability and trust, was correctly addressed. Wrapping these modules as Fractal components rendered their integration relatively straightforward.
- Finally, two Grid4All tools, the Shared Calendar and VOFS enrich the Antares eMeeting legacy application for synchronous collaboration. This application needed redesign to use these services. Usage of VOFS benefits eMeeting users since they do not need to depend on central file services to share meeting and other collaboration data.

# 3. Qualitative Evaluation

## 3.1 Introduction

This chapter contains the report on the qualitative evaluation of selected Grid4all results. In particular this qualitative evaluation illustrates how these results match the fundamental Grid4all objectives:

- an easy setup and use,
- a reduced management and administration complexity,
- an access to collaborative tools and applications,
- the execution of applications drawing on resources available on the Internet.

Every result is first shortly described, and then its qualitative evaluation method is detailed. Then the feedback is analysed with a summary of the lessons learnt.

## 3.2 Niche[1] Distributed Component Management System

### 3.2.1 Introduction

Niche is a Distributed Component Management System (DCMS), used to develop, deploy and execute self-managing distributed component-based applications on a structured overlay network of computers. It includes (1) a set of APIs for the development of self-managing distributed applications; (2) a run-time execution environment for the deployment and execution of self-managing distributed applications developed using Niche.

Niche is a general-purpose system that can be used to develop, deploy and provide robust and scalable self-managing services with self-configuration, self-healing and self-tuning capabilities, on a network of computers donated by end-users or/and service service/resource providers.

### 3.2.2 Evaluation method

As a development environment, Niche should meet the following major requirements:

- it should be easy to use;
- it should have a convenient, flexible and expressive API that provides programming concepts, abstractions and functionality required for developing self-managing distributed applications;
- it should provide an easily managed development and execution platform (including installation, deployment, and required documentation).

In order to qualitatively evaluate Niche as a development and execution environment based on the above criteria, and to get feedback from Niche users, a development-based evaluation has been conducted at KTH and SICS by the Niche development team and a number of students of the international Master program “Software Engineering of Distributed Systems” (SEDS)[2] at the School for Information and Communication Technology (ICT) of KTH, Stockholm, Sweden. By development-based evaluation, we mean evaluation of a development environment, such as Niche, by a number of evaluator-developers, who use the environment to develop and prototype applications, for which this environment is intended to be used. Evaluation through development allows evaluators to assess strengths and weaknesses of the environment and, in particular, the API (abstraction level; the gap between the actual needs of a developer and the functionality that is provided; missing functionality, etc.).

[1] In previous Gird4All deliverables Niche is named DCMS (Distributed Component Management System)

[2] Software Engineering of Distributed Systems http://www.kth.se/studies/master/programmes/ict/2.1730?l=en

The SEDS Master students, which participated in evaluation, were (are) at their final stage of studies, and they have studied a number of advanced courses related to distributed computing. In order to prevent and avoid a possibly biased evaluation, the students, who participated in evaluation, have been given the Niche assessment questionnaire only *after* their Master theses have been examined and graded, and most of evaluators have completed their studies at KTH.

Niche has been specially developed to facilitate writing of self-management code for large-scale distributed applications where self-management (e.g. self-healing and self-configuration) takes place through one or more feedback control loops. Therefore the major goal of development-based evaluation of Niche was to see whether and to what extend the Niche programming model and corresponding API is pertinent for designing self-managing applications; and how difficult or easy it is to develop and to code the management part of applications using Niche. The second major goal of evaluation was to get feedback from developers-evaluators that would allow the Niche development team to improve Niche API and execution platform.

The role of developers-evaluators was to develop a self-managing application using Niche and in this way to text and evaluate it. Note that Niche was a completely new environment for all four evaluators when they started their projects. Developers-evaluators have been given tasks to design and implement distributed services with self-managing capabilities. One of the evaluators did not implement a service but rather extended the Niche environment with a policy-based management mechanism using a policy engine. This evaluator has used a self-managing distributed service developed using Niche, as a use case in his project.

Table 1 summarizes projects that have been performed using Niche at SICS, KTH and FT during the 2008-2009 academic year. Developers of those projects have participated in evaluation of Niche through evaluation questionnaires and open discussions.

| **Project (thesis) title** | **Performed by** | **Period and place; status** | **Short description[3]** |
|---|---|---|---|
| YACS - "Yet Another Computing Service" Using DCMS,<br><br>Master thesis project | Atli Thor Hannesson,<br>SEDS Master student,<br>KTH | Nov 2008 – June 2009;<br>Performed at SICS, Stockholm;<br>Co-supervised by KTH.<br>Completed. | YACS is a distributed computing, or execution, service enabling the use of shared and distributed computational resources for user programmed tasks, e.g. CPU intensive and time consuming movie transcoding. It shows self-healing and self-configuration capabilities to help it survive failures and adapt to changes in the environment, e.g. from membership or load changes. |
| Evaluation of Approaches to Policy-based Management in a Self-Managing Distributed System,<br>Master thesis project | Lin Bao,<br>SEDS Master student,<br>KTH | Oct 2008 – June 2009;<br>Performed at SICS, Stockholm;<br>Co-supervised by KTH.<br>Completed. | Integration of a generic policy based framework into Niche platform and demonstrate it with YASS (Yet Another Storage Service) control loop self-healing and self-configuration. |
| A Self-Managing Large-Scale File Storage With Replication<br><br>Internship, Master thesis project | Catalin Stefan,<br>SEDS Master student<br>KTH | Jan 2008 – June 2009;<br>Performed at FT, Paris;<br>Co-supervised by KTH.<br>The project is completed; the final presentation and examination at KTH is appointed on Aug 2009. | The application is an autonomic distributed storage system with adaptive replication, which aims to adjust the replication of stored data in order to provide some quality of service guarantees to the user. The system implements autonomic management to adjust to a dynamic environment. |
| File transfer in YASS, Yet Another Execution Service; and the Hello World example | Leif Lindbäck,<br>Assistant Professor,<br>KTH | Spring 2009.<br>Performed at KTH.<br>Completed. | Implementing file transfer in YASS; Niche exploration, Hello World example |

[3] As given by evaluators in their filled questionnaires.

| | | | |
|---|---|---|---|
| A project aimed to complete YASS | | | |

Table 1. Projects in which Niche was used and evaluated

The developers-evaluators have been given Niche software (including sample applications), documentation (including the Niche Programming Guide and javadoc API documentation). Each developer has been given a short introductory tutorial on Niche (installation, deployment and API). During the project work, the Niche development team at KTH and SICS also provided on-demand support, help and consultations. The Niche team was trying as soon as possible to address users' requests regarding missing functionalities in Niche, required to complete the given task.

In order to get users' feedback and assessment of Niche, we have developed a Niche evaluation questionnaire to be answered by the developers-evaluators (see Appendix 1). The Niche team has also organized open discussions with each evaluator. By means of the questionnaire we intended to assess developers' interest in Niche as well as get suggestions on improvement, in particular, on features developers would expect in future versions. Based on analysis of the feedback and suggestions, we have improved and will further improve Niche (both API and execution environment) in order to meet expectations of future users. Suggested improvements are presented below.

### 3.2.3 Analysis of feedback

Four developer-evaluators, after they have finished their projects and got their grades (in the case of students), have been asked to fill the Niche evaluation questionnaire to assess Niche and provide their feedback. For copies of the answered questionnaire see Appendix 1: Niche Evaluation Questionnaires.

The following table summarizes answers to some most essential questions of the evaluation questionnaire. The table does not include suggestions on additional features to be provided in Niche and free-text comments. We have called evaluators with symbolic names in the table. Note that evaluators have different experience of work in industry as software developers.

| **1. Evaluator (level of experience in software development)** | **Evaluator A** (several years experience in industry (at Sun Microsystems) and academia) | **Evaluator B** (4 years of work experience in industry as SW developer) | **Evaluator C** (some work experience in industry as SW develop) | **Evaluator D** (no much work experience as SW develop in industry) |
|---|---|---|---|---|
| **2. Application** | File transfer in YASS (Yet Another Storage Service); Niche exploration | YACS (Yet Another Compute Service) | Policy-based management framework for Niche | File storage with replication |
| **Initial development stage of application** | Pre-developed | Specification | Idea | Idea |
| **Management concerns of application** | Fault-tolerance | Configuration; Fault tolerance | Configuration; Fault tolerance | Fault-tolerance; optimization |
| **Self-managing properties of application to develop** | Self-healing | Self-configuration; self-healing | Self-configuration; self-healing | Self-configuration; self-healing; self-optimization |
| **Expected number and type of nodes to execute application** | - | PCs of ordinary users; number is unknown | at least two nodes | hundreds |
| **Approximate size of application (in lines of code)** | - | 14299 lines (including javadoc comments) | 800 lines for XACML implementation, 600 lines for SPL implementation | About 10000 lines, of which 30% functional code. |

| | | | | |
|---|---|---|---|---|
| **3. Main motivation for using Niche as a development environment** | To extend an existing application with self-management capabilities; To evaluate DCMS as a development environment | To develop a self-managing application from scratch; To evaluate DCMS as a development environment | To integrate a generic policy based framework into Niche, make the self-managing behavior under the government of policies. | To develop a self-managing application from scratch; To evaluate DCMS as a development environment |
| **4. Ease of learn** | | | | |
| **Was documentation sufficient to learn Niche?** | Partially | Almost | Partially | Partially |
| **Time to understand and run the YASS example** | 3 days | 5 days | 1 week | 1 week |
| **In overall, how easy or difficult it was to get acquainted with Niche?** | Acceptable | Acceptable. Conceptually, I feel DCMS is easy to understand. | Acceptable | Difficult |
| **5. Usability and Complexity** | | | | |
| **Time it took to install and configure Niche** | 3 days | A few days, closer to 1 week | 1 week | 1 week |
| **Time it took to code and test a first prototype of application** | 3 days | 2-3 days | 1 week | A month |
| **Suggested additional features to be provided in Niche** | | Component un-deployment API; Synchronous communication API between MEs; other suggestions (see answers in Appendix 1) | With subscription, I want Niche to provide one-to-all and one-to-any binding, if the subscriber is a group. | Enhanced group API; Control over the placement of (management) components (For full list of suggestions, see answers in Appendix 1) |
| **Did you find the Niche programming model pertinent for designing self-managing applications?** | | The Niche model enabled self-management | The Niche model enabled self-management | The Niche model enabled self-management |
| **Did Niche help you to design a modular application, with good separation of concerns?** | Very | Very | Very | Very |
| **How difficult or easy it was to code the application as a set of distributed components?** | Easy | Very easy | Acceptable | Easy |
| **How difficult or easy it was to develop and to code the management part of the application?** | | Easy | Acceptable | Difficult |

| | | | | |
|---|---|---|---|---|
| **Do you find the complexity of the framework justified?** | It is partially justified | It is partially justified. I feel the bindings are too complex. | Yes, it is justified | Yes, it is justified |
| **In overall, how easy or difficult it was to implement application using Niche? (very difficult, difficult, acceptable, easy, very easy)** | Acceptable | Easy | Acceptable | Acceptable |
| **6. Performance, scalability, interoperability, and stability** | | | | |
| **Performance issues** | No | Yes, component deployment occasionally block for a long, but configurable, amount of time. | No | No |
| **Scalability issues** | Sorry, I do not know | Sorry, I do not know | Sorry, I do not know | Haven't fully tested yet |
| **Interoperability issues** | No | Yes; instability when running in cross-OS setting. | Yes; Synchronized operation | Yes; an external application to interact with DCMS components |
| **Stability of Niche implementation** | Excellent | Fair | Sorry, I do not know | Excellent |
| **7. Satisfaction** | | | | |
| **Will you use Niche again?** | Yes | Yes, If un-deployment is added I could imagine trying it out for a fault-tolerant and adaptation capable data-center infrastructure project that I'm headed for. | No. I finished my thesis work. | Yes; I would probably consider the possibility of using dcms as an infrastructure for an application that requires autonomic management. |
| **Will you recommend Niche to other developers?** | Yes | Yes | Yes | Yes |
| **In overall, how you would you rate your experience with Niche?** | Rather good | Good | Good | Rather good |

In overall, according to answers, the assessment of Niche as a development and execution environment by four developers-evaluators is rather positive, even though Niche requires some efforts to study documentation, to install Niche environment, and to study the provided examples of self-managing applications. Three out of four developers-evaluators have answered that, in their opinion, it was acceptably difficult (i.e. neither difficult nor easy) to implement applications using Niche; whereas one developer (who has developed Yet Another Computing Service) has answered that it was rather easy to code his application using Niche. Some evaluators have given critical comments on API, mostly, on the group API, that must be (and has already been) addressed by the Niche development team. Below we analyze evaluators' feedback along different evaluation criteria. For each category of criteria, we summarize the feedback and present (a) our analysis of the feedback (b) possible directions of future work.

## Ease of Learning; Documentation and Examples

Three of four evaluators have answered that, in overall, it was acceptably difficult (i.e. neither difficult nor easy) to get acquainted with Niche using provided documentation and examples. One evaluator (with less work experience) found it difficult to get acquainted with Niche by studying its documentation and examples.

> **Analysis.** Like for many distributed environments and platforms, e.g. Java EE, the Niche "learning curve" is rather flat in the beginning, i.e. Niche is difficult to learn. However, according to opinions of developer-evaluators, after the initial difficulty of learning (including efforts to install, deploy, and study examples), Niche is relatively easy to use and to program with, and it offers a set of rather clear programming abstractions and the corresponding API to develop self-managing applications.

All developers-evaluators have answered that the Niche documentation (at the time of evaluation) is partially (almost) sufficient to learn Niche.

> **Analysis.** Niche documentation needs to be further improved, elaborated and clarified. Based on some suggestions from developers-evaluators regarding documentation and questions on some features of Niche (e.g. on group API), the Niche team has revised the initial version of the Niche Programmer's Guide, and, in addition, has developed the Niche Quick Start Guide in order to facilitate and speedup the initial phase of getting acquainted with Niche. Both guides are available at http://niche.sics.se.

None of the evaluators has tested the Hello World example, because the version of the example, which was available at the time of evaluation, illustrates development of only the functional part of a component-based application, whereas the management part is missing. All evaluators have studied and tested the YASS (Yet Another Storage Service) example, even though it was more complex than the Hello World example.

> **Analysis.** All evaluators-developers have found the YASS example useful and helpful for getting acquainted with Niche; and have used YASS as a sample application when developing their own applications. It took from 3 days up to 1 week for developers to understand and successfully run the YASS example. The Niche team has found this time reasonable for this rather complex example.
> In June 2009, the Niche team has revised the Hello World example by adding a management part (to illustrate self-healing) in order to make the example more illustrative but relatively simple.

**Conclusion**. In overall, for the level of research prototype, Niche documentation and provided examples are of an acceptable quality. It is a matter of polishing documentation to further improve its quality when moving Niche from research prototype to a product.

## Usability and Complexity; Ease of Use

Three out of four developer-evaluators, who participated in Niche evaluation, have answered that, in overall, it was acceptably difficult (i.e. neither difficult nor easy) to implement applications using Niche. One evaluator, who has developed Yet Another Computing Service (YACS), answered that it was easy for him to implement the service using Niche. It took from 3 days to 1 week for evaluators to install and configure Niche (including configuration and testing of the YASS example). All evaluators have requested some help from the Niche development team to properly configure Niche, and have indicated that installation and configuration of Niche, specifically in the Eclipse development environment, requires some efforts.

> **Analysis.** The Niche development team must improve the Niche installation and configuration procedure in order to facilitate it. It might be reasonable to offer Niche distribution packages with a re-configured (and pre-build) Eclipse project or a NetBeans project.

It took different time for developers-evaluators to code and test first prototypes of their applications: from a few days (YACS) to one month (Distributed Storage with Replication). Difficulty of development of the management part of applications using Niche has been assessed as "easy" (YACS), "acceptable" (policy-based management in YASS) and "difficult" (Distributed Storage with Replication) by three evaluators.

> **Analysis**. Of course, the development time (and opinion on the ease of development) depends on complexity of an application to be developed, in particular, on complexity of application-specific self-management objectives and control/optimization algorithms for self-management. In developers' opinions (expressed in answers to the questionnaire and during discussions), and in the opinion of the Niche team, the complexity of the Niche environment is justified. The Niche team has found that the amount of time it took evaluators to develop their first working prototypes is reasonable and matches the complexity of the applications and the different development skills of developers-evaluators.

Three out of four developers-evaluators has given very interesting and useful suggestions on additional features that should/could be provided in the Niche API, e.g. component un-deployment, synchronous (in addition to existing asynchronous) communication between management elements, more flexible group binding. See answers to the valuation questionnaire in Appendix 1 for detailed suggestions.

> **Analysis**. In opinion of the Niche development team, many of the evaluators' suggestions are reasonable. Some of the suggestions, e.g. the one on the interaction between management elements and on the group API, have been already implemented by the Niche development team. Other suggestions are subject for our future work.

**Conclusion**. In overall, for the level of research prototype, Niche, as a development platform is of an acceptable quality with respect to its usability, i.e. ease of use and convenient API. The complexity of the Niche environment is justified. In overall, evaluation results show that Niche provides the functionality needed to build a self-managing distributed application, and this functionality was used by evaluators-developers. In order to further facilitate development of self-managing applications using Niche, and to move Niche from the level of research prototype to the level of a product, the Niche development team should extend the Niche API with additional features suggested by evaluators. Note that some of the suggestions have been already implemented. In order to move Niche to the level of a product, the Niche team should also improve the Niche installation and configuration procedure, polish documentation, and (optionally) provide some supporting development tools, such as a tool to generate Java-skeletons based on ADL-descriptions that would reduce the risk of name errors for the same interface being described in multiple files. This is a subject for out future work.

## Performance, Scalability and Interoperability Issues

None of the developers-evaluators, but one, has given their opinion of Niche performance or performance of their own applications.

> **Analysis**. This is because, in our opinion, almost all projects were aiming at providing self-healing and self-configuration capabilities using distributed managers rather than a centralized (single) manager that might become a potential performance bottleneck, but performance evaluation was not their immediate concern. Also, management logic that were implemented were rather simple and aimed to meet management objectives in the best-effort way rather than to meet fixed service-level objectives set in SLAs that might require fast control loops (i.e. efficient monitoring and actuation, and fast controllers). Nevertheless, all evaluators (students) have done (or have participated in) performance evaluation of their systems on Grid5000 and PlanetLab testbeds. In particular, a number of evaluation experiments have been performed in order to quantitatively evaluate Niche end-user experience based on YASS (Yet Another Storage Service) and YACS (Yet Another Computing Service). Results of evaluation are presented in Section 4.2 of this document.

Some developers have indicated the overhead involved in initial deployment and that component deployment occasionally blocks for a long, but configurable, amount of time. This might cause performance penalty in performance-demanding applications.

> **Analysis**. Performance optimization of Niche requires more performance experiments and profiling in order to find (potential) performance bottlenecks. The Niche development team has made some efforts on performance optimization of Niche during the period of the projects. The Niche team understands the comments about the overhead involved in initial deployment, which partially is an effect of the less-than-desirable integration between the ADL-deployment (i.e. deployment performed based on the ADL description of application architecture) and the programmatic deployment (i.e. deployment performed programmatically in management code). For instance it is still not possible to deploy management elements through the ADL-based deployment. Future work which addresses this is integration with OZ-based deployment tools developed at INRIA Sardes. The occasional blocking during deployment, mentioned by one of the evaluators, has been already addressed by the Niche team.

Three out of four evaluators have indicated different interoperability issues, namely, some difficulties in running Niche across different OSes (Windows, Linux); and the need for interaction of an external application with Niche components.

> **Analysis**. We believe that cross-platform issue between Linux and Windows is an artifact of the specific communication package which was used at the time of evaluation. Some recent tests

(running YACS across Linux and Windows) does not expose this issue any more. Support for interoperability of Niche components with external applications is on the list of our future work.

**Conclusion.** In overall, for the level of research prototype, Niche performs and scales rather well, and **does not have interoperability problems** so far. In order to go from a research prototype to a product, Niche needs performance and memory optimization in order to identify and remove performance bottlenecks and to track down and correct memory leakage.

### Satisfaction and Overall Rating

According to answered questionnaires, three out of four developers-evaluators are willing to use Niche again in their future development work. One of the evaluators intends to try using Niche "for a fault-tolerant and adaptation capable data-center infrastructure project" that he is headed for. All four developers-evaluators will recommend Niche to other developers; and all four developers-evaluators rate, in overall, their experience (satisfaction) with Niche as (rather) good.

### Critical Comments and Suggestions on Improvement and Addition Features

Three out of four developers-evaluators has given very interesting and useful free-text comments on difficulties that they have faced when developing their applications using Niche, and suggestions on improvement and additional features that should/could be provided in Niche API.

**Analysis.** Most of comments (see Appendix 1) were about the group API, including group binding, monitoring and actuation. All evaluators have indicated that the component group abstraction (and corresponding API) is very useful, and it facilitates development of both functional and management parts. However, the group API should be improved in order to be more flexible. Most of the comments have been already addressed by the Niche team, e.g. the group creation API using group templates is now documented; bindings that time-out on communication errors are now supported; bindings with return values are working; occasional blocking during deployment is avoided; the comment about the need for return values from multiple receivers can be solved after the introduction of actuators; inconvenience in monitoring of group creation events has been recently solved.

Niche has been also used by a group of two students in a course project in the "Network Programming with Java" course at KTH. The students have developed the functional part – only – of a component-based distributed application. It took them about one week to develop a working prototype of the application. The students have been helped to install and deploy the Niche environment. The students' feedback was not documented in the form of a questionnaire, but the Niche development team have had a discussion with the students and have got their feedback and suggestions mostly related to the component group API of Niche. Students' critics and suggestions were taken into account when improving the group API. In overall, the students' opinion about Niche was rather positive; however they have not tried developing a management part of their application and could not access development of self-management code and give any critics, comments and suggestions on that.

## 3.2.4 Lessons learnt

A middleware, such as Niche, clearly reduces burden from an application developer, because it enables and supports self-management by leveraging self-organizing properties of structured P2P overlays and by providing useful overlay services such as deployment, DHT (can be used for different indexes) and name-based communication. However, it comes at a cost of self-management overhead, in particular, the cost of monitoring and replication of management; though this cost is necessary for the democratic grid (or cloud) that operates on a dynamic environment and requires self-management.

In order to better deal with dynamic environments, such as community grids, management functions should be distributed among several cooperative autonomic managers that coordinate their activities to achieve management objectives. Multiple managers are needed for scalability, robustness, and performance and they are also useful for reflecting separation of concerns. Design steps in developing the management part of a self-managing application include spatial and functional partitioning of management, assignment of

management tasks to autonomic managers, and co-ordination of multiple autonomic managers. The design space for multiple management components is large; indirect stigmergy-based interactions, hierarchical management, direct interactions. Co-ordination could use shared management elements. Further study is required. As concerns robustness, replication of management elements is a well known methodology.

### Future Work

A major concern that arises is ease of programming of management logic. Research should hence focus on high-level programming abstractions, language support and tools that facilitate development of self-managing applications. As has been reported within D6.7, we have already started to address this aspect.

There is the issue of coupled control loops, which we did not study. In our scenario multiple managers are directly or indirectly (via stigmergy) interacting with each other and it is not always clear how to avoid undesirable behavior such as rapid or large oscillations which not only can cause the system to behave non-optimally but also increase management overhead. We found, as mentioned above, that it is desirable to decentralize management as much as possible, but this probably aggravates the problems with coupled control loops. Every application (or service) programmer should not need to handle co-ordination of multiple managers (where each manager may be responsible for a specific behavior). Future work should address design of coordination protocols that could be directly used or specialized.

Although some overhead of monitoring for self-management is unavoidable, there are opportunities for research on efficient monitoring and information gathering/aggregating infrastructures to reduce this overhead. While performance is not perhaps the dominant concern of 'democratic grid' users, we believe that this should be a focus point since monitoring infrastructure itself executes on volatile resources.

Replication of management elements is a general way to achieve robustness of self-management, especially, self-healing. In fact, most evaluators assumed that management programs will be robust. They did not build-in protection from failure of management logic. Even though we have developed and validated a solution (including distributed algorithms) for replication of management elements in Niche, it is reasonable to continue research on efficient management replication mechanisms.

## 3.3 Telex

### 3.3.1 Introduction

The Telex middleware facilitates the design of peer-to-peer collaborative applications. It takes care of complex application-independent aspects, such as replication, conflict repair, and ensuring eventual commitment. Telex allows an application to access a local replica of data/information without synchronizing with peer sites. The application makes progress, executing uncommitted operations, even while peers are disconnected.

The qualitative evaluation of Telex is performed by means of a questionnaire, filled by collaborative application developers, which contains 35 questions divided into 5 main sections: Ease of learning, Complexity, Features, Control and Satisfaction. The developer is requested to answer the questions in the context of the application she/he has developed. Most sections include an overall qualitative evaluation in the range of 1 (bad) to 5 (good).

### 3.3.2 Evaluation method

The application developer is provided with the Telex library and the related documentation. (These are publicly available at Telex's web site: http://telex2.gforge.inria.fr). The developer can get support from the Telex development team by e-mail. After he has developed the application, the developer fills the questionnaire. The questionnaire begins with a brief description of the application, its development stage and

an evaluation of its size. It proceeds with the evaluation of the developer's familiarity with collaborative middleware/platform and how he got to know Telex. This will help put subsequent answers into perspective.

The *Ease of learning* section assesses how easy/difficult it is to learn Telex. Each learning resource is evaluated: documentation, code samples, support from the development team, and need for a tutorial session.

The *Complexity* section evaluates the complexity of developing an application atop Telex. The complexity is assessed through various quantitative metrics: the percentage of the application's code dedicated to interfacing with Telex, the time it took to design, code and test above Telex. The developer is also asked to rate the complexity of expressing the application's logic using the Action-Constraint Framework and the complexity of Telex's API.

The *Features* section examines to which extent Telex fulfils the application's requirements. It first determines whether Telex is missing some important features and whether those features are provided by an alternate middleware. Then it proceeds by listing which of Telex's main features the application actually uses, as some applications might use only a small subset of them. To assess Telex's novelty and usefulness, the developer is asked whether an alternate middleware provides the same combination of features and what is the benefit of using Telex, i.e. what application-level features Telex has enabled. As an indication of ACF's expressiveness, the section also determines which ACF constraints the application actually uses.

The *Control* section assesses the degree to which the developer feels in control of Telex operation. The middleware might do too much or too little of its own. This will help refine the API, either by splitting high-level methods into low-level ones, or by providing advanced methods. The section also examines whether the standardization of Telex's API or Telex's interoperability with external software is an issue in the context of the application being developed.

Finally, the *Satisfaction* section evaluates the developer's overall experience with Telex. He is asked whether the performance of Telex is sufficient for the application, and whether he has enough confidence in Telex to develop other applications, release-quality applications and to recommend Telex to other people. Finally, the developer is asked to list items that should be improved in future releases of Telex.

Three applications have been developed atop Telex as part of the Grid4All project: the Shared Calendar application, the STMBench7 benchmark and the Collaborative File Sharing application. Only one application was developed by evaluators external to Grid4All. It is a collaborative ontology editor, developed at the University of the Aegean by Giorgos Santipantakis, as part of his master thesis project under the direction of George Vouros.

We report next on this experience, on the basis of the returned questionnaire. Most answers are straightforward and need no further clarification. Some, however, need to be interpreted with the help of Giorgos Santipantakis' master thesis and the support we gave him.

### 3.3.3 Analysis of feedback

The collaborative ontology editor is a good target for the Telex middleware: users share a semantically-rich data structure (the ontology), which users need to update either on-line or off-line. The editor is a small-to-medium sized application (6 man-month, ~5000 lines of code in total). G. Santipantakis is using a collaborative middleware/platform for the first time.

#### Ease of learning

Overall rate is 4/5, 5 standing for “very easy”.

The documentation is poor: it only consists of two papers on Telex, as the tutorial on Telex is not completed yet. Examples of application code (e.g. the Telex shell) are somewhat too complex to help. The support that the Telex team provided partly mitigates this. **A tutorial session on Telex is needed**.

#### Complexity

Overall rate is 4/5, 5 standing for “very easy”.

The percentage of code dedicated to interfacing with Telex is 13%. This seems rather low, perhaps because GUI code is taken into account. Question b will be rephrased so as to consider core application code only.

The time needed to develop above Telex represents 50% of the design phases, 22% of the coding phase and 50% of the testing phases. Theses figures are in line with those reported for the Shared Calendar application. They seem reasonable considering the benefits of using Telex.

We note that although designing with Telex requires a significant effort, the complexity of translating application logic to actions and constraints is rated 4/5, 5 standing for very easy. Similarly, although testing with Telex takes a significant amount of time – most probably because this involves distributed test – **the developer feels that Telex's functions and Telex's API are not unnecessarily complex**.

### Features

Overall rate is 5/5, 5 standing for “Telex's features are essential”. However, the question should be rephrased so as to include missing features, e.g. “Telex features are essential and complete”.

Telex lacks two important features as far as the collaborative ontology editor is concerned: (i) Telex should be able to roll-back on decisions (ii) It should provide the sequence of schedules that leads to a given sate. Indeed, in the current version of Telex, making a decision implies commitment and garbage collection. These aspects should be de-coupled for users to tentatively agree on a common state and then drop it later on if not satisfactory. This would give way to distributed evaluation of *what-if* scenarios.

The ontology editor mainly uses Telex for disconnected operation, conflict detection and resolution, and reconciliation of document replicas. The developer observes that (i) no alternate middleware/platform provides this combination of features, (ii) these features are essential to the application in that they greatly ease collaborative work: users can update the ontology any time any place without coordination, Telex (and the application) will maintain the ontology's consistency.

The ontology editor only uses antagonism and causality relations. It does not use atomicity or any elementary constraint. From Giorgos Santipantakis's thesis, we know that the constraints defined so far (not-after, enables, non-commuting) cover all the editor's needs. A specific question will be added to the questionnaire, however, to address this topic.

### Control

Overall, **Telex's functionality seems satisfying**: the middleware does not do too much or too little of its own. It is suggested that Telex's API should be standardized, or an effort to standardize it should be undertaken. The questionnaire, however, does not ask any reason for this and consequently none is given. This should be added in the next version of the questionnaire. Telex's interoperability (application language, log format, etc.) is rated 4/5, 5 standing for excellent. **Standardization and interoperability does not seem to be an issue for the ontology editor**.

### Satisfaction

Overall rate is 4/5, 5 standing for “very satisfying”.

The developer has **enough confidence in Telex** to use it again for new applications and to recommend it to other people. Telex, however, cannot be used as is for release-quality applications as some bugs need to be fixed. Also, the ability to roll back on decisions taken is essential for the ontology editor. Finally, it is suggested that future releases of **Telex be improved by providing more documentation and tutorial** as well as demonstration applications.

## 3.3.4 Lessons learnt

### Evaluation process

As pointed in sub-section 3.3.3, the questionnaire needs to be improved in order to get more accurate feedback in the future (e.g. ACF's completeness, actual need for standardization, etc.). A new version of the questionnaire will be released to that end.

The questionnaire will be posted to Telex's web site and application developers will be encouraged to fill it so that we get more feedback. This will complement the feature request and technical support trackers already provided by Gforge.

### The Telex middleware

Overall, Telex meets its primary goal: it facilitates the development of collaborative applications by taking care of complex tasks such as replication, conflict detection and resolution, and eventual consistency. The middleware is fairly easy to learn and use, from both conceptual (Action-Constraint Framework) and implementation (Telex's API) perspectives.

However, Telex is not a release-quality product yet. Its documentation lacks a tutorial, streamlined application samples and more demo applications. These will be provided in next releases of Telex: INRIA Regal is about to hire a development engineer to maintain and disseminate Telex after the Grid4All project's end.

More important, Telex lacks an important feature for collaboration. While Telex allows each user to select a particular solution to a conflict, it does provide support for users to agree on a tentative solution and then roll-back if the solution turns out to be unsatisfactory later on. This feature seems of general interest and INRIA Regal will discuss its precise specification and implementation with Giorgos Santipantakis. As far as we can see, this feature does not raise any new and interesting scientific problem; it is just an engineering issue.

## 3.4 VOFS

### 3.4.1 Introduction

VOFS is a personal tool that can be used by both ordinary users and expert administrators to create a simple to use and straightforward peer-to-peer file-sharing environment for collaboration, over the traditional file system. A number of users were introduced to the VOFS prototype and their feedback was analyzed in order to validate the ideas and solutions behind VOFS and some aspects of usability and engineering nature. The next sections present the evaluation method, an analysis of the feedback and general conclusions.

### 3.4.2 Evaluation method

The evaluation plan for VOFS featured a workshop session organised by a VOFS developer (the *introducer*) where users (the *evaluators*) would be introduced to VOFS, allowed to experiment with it and would provide feedback in the form of pre-prepared Questionnaires. The actual evaluation was performed in two separate workshop sessions where 5 people attended and provided formal feedback (i.e. filled a Questionnaire). Additional feedback was received by personal communication of the evaluator with another 2 individuals. The individuals received a similar introduction of the VOFS as in the workshop but over the Internet.

For the workshop, two laptops were configured to be ready for use with VOFS as public laptops were users would experiment and users also brought their own laptops to participate in the VOFS network. All computers were connected to a LAN.

The introduction to VOFS started with basic concepts about what VOFS is and progressively offered more specific information along with a practical introduction to the software. The next paragraphs summarize the introduction given both during the workshop sessions and to the individuals over the internet.

VOFS provides a file server for everyone. This file server preferably remains always online and serves a unique file system with the user's files. Users access this file system with their clients, which may or may not be located in the same machine. The client presents this file system as a normal file system in their local Operating System. Users were asked to perform a new VOFS installation and were directed to launch their server and client as in the package documentation.

VOFS appears as a conventional disk file system but it is not. To facilitate the extra controls needed for VOFS, a naming convention introduces a number of special “virtual” files per real file. The users were shown what virtual files are and were given examples of their usage.

Each file server in the VOFS network is actually a peer accessible by other peers. User's clients are such peers and users were shown how to navigate other user's file systems, using the special naming conventions introduced by VOFS. Using virtual files, users were directed to create symbolic links from a given file system to another one. The introducer then explained how this feature can be used to federate files into a common directory, creating a shared workspace. Scripts that automated the workspace creation and management were introduced to the users and they were encouraged to experiment.

VOFS clients are designed to expect and deal with loss of network connectivity with the minimum cost for the user. Both file system structure and data are cached locally and this cache is used to immediately serve user requests. Users were invited to disconnect their computers and try to access remote files. It was demonstrated that responsiveness was immune to network unavailability. Users were encouraged to experiment with different types of network failure and combinations of failing machines.

Finally, the VOFS network has a third kind of peer, the storage provider which is the peer that actually hosts file data. By default, the storage provider is not separate from the file server. Users were introduced to launching separate storage providers, assemble storage pools and control the storage on a file-to-file or whole directory basis. The introducer explained the role of the storage pool in the workspace created by the automated scripts and invited the users to experiment.

### 3.4.3 Analysis of feedback

***Evaluator profiles***

The evaluators will be anonymously referred to by the numbers 1 to 7. Evaluators 6 and 7 were the two that were introduced by personal communication over the internet. Evaluators 1 to 5 were computer experts. Evaluators 6 and 7 were comfortable with computers and the Internet beyond office work but were not computer experts.

Although questionnaires were prepared to collect feedback from evaluators, during the workshop sessions there was a lot of discussion and eventually most of the feedback was collected personally by the introducer by keeping notes on the discussed issues and opinions. A significant portion of the feedback was offered in conversation by evaluators that were discussing questionnaire questions they were trying to answer. Questionnaires were filled just for formality as the information collected otherwise made them irrelevant, especially since all evaluators had time and direct communication with the introducer.

Two general issues were taken into account when interpreting the evaluator opinions and reactions. First, users might raise issues regarding VOFS that are either of engineering or conceptual nature. Engineering aspect is not the main concern of the evaluation, since our objective was to understand the impact of the concepts behind VOFS and not its maturity as a product. Second, there was a noticeable distinction between two types of evaluators. The first type, which we will code-name *the enthusiasts,* care about their efficiency and that of the tools they use, have well formed opinions and preferences, and are curious and willing to experiment in order to personalize and optimize their workflows. The second type, which we will code-name *the conventional*, conform to the standard workflows of their environment and prefer to just get the job done the way they have been shown to and they do not invest time in experimentation. To extract more insight, evaluator feedback was always interpreted with a certain bias considering the two types of evaluators.

***Specific feedback analysis***

This section presents an analysis of specific evaluator feedback according to the structure of the questionnaire that ends with feedback from discussion that came up independently.

*Software set-up and launch*

The evaluators found the set up and launch of VOFS software straightforward. Concerns about setting up the prerequisite software were deemed unimportant as engineering issues. One issue raised was whether setting up and launch would remain simple after VOFS had matured into a full-featured system. However, this is no concern since by design, all additional complex features and services that are added to VOFS have to present themselves through the same simple and standard environment as the basic functions are.

*New and interesting features*

By popularity, the evaluators reported as the most interesting VOFS features, the *links across file systems*, *disconnected operation*, and *storage pools and controlling storage*. However, the evaluators reported that the control VOFS offers on file storage is not convenient. VOFS requires the user to decide the storage provider before file creation whereas users found it cumbersome to care about storage beforehand. Instead they would prefer to be able to adjust storage details afterwards and only if needed. From a technical point of view this would be inefficient but from a usability point of view it is an issue to be addressed.

*How VOFS was used by evaluators*

An unexpected realization was that most evaluators thought *media sharing* and *sharing for collaboration* as different tasks. We speculate that this is due to the different contexts active and different tools that are used to accomplish these tasks. The enthusiast evaluators were intrigued by the realization that VOFS as a personal tool could unify the way both flavors of sharing is achieved.

*Local network performance*

The evaluators when asked reported that performance was lower than the local file system but was well accepted for small files and was only a concern with large files or directories. However, all evaluators were content with the consistency of the performance and the responsiveness of the system. From a technical point of view this performance was expected as a constant overhead that can be eliminated by further engineering. The focus of the VOFS prototype is its responsiveness and handling of the unique environment VOFS creates.

*The biggest flaw*

The conclusion about the flaws that were reported was that the evaluators tried to use or at least considered VOFS for their every day needs but found that it was not mature enough and sufficiently integrated with their software environment. However, such integration is out of scope of the prototype and the fact that the evaluators tried to incorporate VOFS in their workflows validates its design.

*The nicest thing about VOFS*

The reported nicest practical thing about VOFS was by far its responsiveness and tolerance for connectivity problems. The evaluators reported that frustration with software responsiveness due to network problems was a recurring issue in their every day work and were relieved to discover the extent VOFS was unaffected. From a conceptual point of view, all evaluators recognized the need for simplified file sharing the way VOFS promotes. The enthusiasts were also positive about the peer-to-peer character of the system in contrast to similar solutions that always require some central management out of their control.

*Independent discussion*

Due to the technical nature and expertise of the *enthusiasts*, there was a lot of general discussion about VOFS and the ideas that puts forward. Besides integration, which is an engineering issue, other interesting issues were raised.

The new idea that everyone has their own server and serve shared files on the Internet was well accepted. Inside a LAN there is no issue, but on the Internet traditionally home users do not have the connectivity required to maintain servers. However, emerging computing technologies, such as the ones collectively labeled as *cloud computing*, will soon offer the capability to easily maintain such servers not in the home but on a provider's infrastructure. The VOFS design anticipates that development, especially in the context of the Grid4All goal for democratization of computing.

Security is a complex issue, both technically and socially. An important realization was that users trust impersonal corporations (e.g. Google) with their files even if they know there are security issues. On the other hand, the same users are unwilling to risk exposure to their socially close peers through an environment such as VOFS, even if security is technically better than the impersonal corporation case.

### 3.4.4 Lessons learnt

Overall, the evaluation justified the VOFS proposition to add simple new primitives for sharing into the traditional file system. Users were found to indeed need new tools towards that direction. From a technical point of view, most engineering issues were already anticipated. The most constructive results of the evaluation concern the way the technical features of VOFS have to be presented and exploited in the context of the users and their environment.

Regarding direction of future development, VOFS has two alternatives: evolve more towards being a personal tool or being an administrator's tool for infrastructure. The prototype followed the former approach and the evaluation showed that it would be more appropriate. User's frustration came from the complexity and inefficiency to use tools and solutions and not from the lack of them.

Another realisation was that VOFS introduces new concepts and new ways of thinking and working that is not easy for all users. Therefore, VOFS should focus first on a receptive audience, such as the enthusiasts described earlier. They that will help both in better shaping VOFS and in introducing it to new users and supporting them.

# 3.5 CFS

## 3.5.1 Introduction

The Collaborative File Sharing Application allows all participants in the system to share files in a collaborative way. It is related to all scenarios with needs in sharing information among users. This application will provide mechanisms for collaboratively uploading, downloading and updating files among the participants. All operations are executed in an optimistic way, i.e. there will be no locks in any file when being used by other user. In order to maintain a consistent view for all users, a mechanism for solving any possible conflict will be provided.

CFS is used to facilitate collaborative learning activities as part of courses, at the UPC campus, which usually last for 5 to 10 weeks. Students work in projects to develop Internet based applications for a fictitious or real organization. They work in groups, in the classroom using laptops with support from the teacher, and at different times from other places during the course of the project. The result is an “Educational Portfolio”: a collection of materials (documents and folders) that demonstrate the work done.

## 3.5.2 Evaluation method

Based on the description in D5.4 on the planning for the experiment, the evaluation was performed as follows.

The evaluation team was a mix of selected and voluntary participants from UPC: 4 lecturers, an 6 students were involved. The evaluators were end-users from university, students and teachers, who regularly use computers in the classroom. They participated at a seminar describing CFS and its features that support collaborative learning activities. Background Evaluators are lecturers and students at UPC and UOC. They are well versed in IT technologies. They evaluated CFS in their corresponding roles of teacher and students.

The seminar was performed as a hands-on session where participants edited shared files and organized them in folders. CFS was tested considering its use in collaborative learning (portfolio development for project-based learning projects), in the second week of May 2009. The session was followed by a brief discussion. The questionnaires were then completed by the evaluators.

## 3.5.3 Analysis of feedback

The feedback was obtained from the resulting questionnaires already presented in D5.4c[4] and the discussion at the end of the evaluation activity.

These lessons can be divided as:

- A qualitative part describing the overall evaluation of the result: What went particularly well or badly,

lessons learned paying attention to comments that highlight key aspects.

- Identification of new usage scenarios as expressed by evaluators.
- Ideas for improvement of CFS and new features based on needs expressed by evaluators.

The 10 questionnaires contained questions with answers in the range 1-5 (lowest to highest rating), yes/no questions, and questions with three options (yes, no, a third response), and free-form text fields to collect comments. The non textual responses have been summarized in the following table showing the average values in each response.

The summary of the evaluation from the moderator of the discussion at the end of the session is as follows:

> *“Users do not clearly perceive the benefits of CFS compared to other applications (they also tested and use TortoiseSVN and BSCW). They are not clearly aware of the decisions made in conflict resolution. Instead, they see it not very user friendly and lacking in certain basic features. Some shortcomings or deficiencies found: the application does not automatically detect the type of file to download (the user must manually set the extension), no information messages are displayed to*

[4] Deliverable D5.4c – Addendum to Evaluation Criteria and Test Plan

*users to know the status of others, or actions that are being conducted on different items, the functions for copy / paste do not work properly, it does not allow the "rating" of the different versions, how to upload and download files is not very intuitive, does not have instant messaging and file preview options, the interface is not attractive.*

*As for the resolution of conflicts, they have only been able to observe the behavior of CFS with 2 different situations: when uploading at the same time 2 versions of the same document the application recognizes both versions as "top versions, but when a new version is uploaded, the previous 2 versions appear as completely different. On the other hand, if a user changes the name of a folder, while another user deletes it, the application always delete the folder and does not notify the conflict nor how it has been resolved.*

*Ultimately, as we had anticipated, the general assessment is not very favorable, but "the concept" and possibilities of CFS are perceived as valuable, although CFS would not be used in its current stage of development."*

The most relevant results from the quantitative results from the evaluation questionnaire is average (blue) or high (green) for most results. The most negative comments are: they would not consider using CFS instead of other applications they know (5a: 0.3 with 0=No, 1=Yes). To the question if they would recommend CFS, the answer is close to "not in the current form", and they suggest several shortcomings that must be solved before the application could be recommended (as we could expect since CFS is a prototype, not a release quality product). As it is, and based on their experience with other tools, they consider that the features CFS provides do not really improve additional features to support collaborative work (4d: 0.4) although they consider CFS very useful (4e: 1).

A few users also mentioned that CFS could be applicable not only in collaborative learning activities but also for other activities that require sharing a collection of files that are created and modified over time, as it occurs in research groups that produce reports and papers with several contributors potentially located at different places and working at different times.

| # | Q | Values | AVG |
|---|---|---|---|
| 2a | Activity | | |
| 2b | Dimensions | | |
| 2c | Status | | |
| **Ease of learning** | | | |
| 3a | Documentation | | **2,9** |
| 3b | Use | | **3,4** |
| 3c | Overall | | **2,9** |
| **Functionality** | | | |
| | Version mgmt | | |
| 4a | Ease of use | - 1..5 + | **3,3** |
| | Recovering top version | - 1..5 + | **3,9** |
| | Recovering old version | - 1..5 + | **3,6** |
| | Folders | | |
| 4b | Ease of use | - 1..5 + | **2,7** |
| | Correct automatic resolution of conflicts | - 1..5 + | **3,1** |
| | Event logs | | |
| 4c | Ease of use | - 1..5 + | **3,7** |
| | Info on items read | - 1..5 + | **4** |
| | Info on relevance of an item | - 1..5 + | **2,7** |
| 4d | Features that improve collaborative support | N 0..1 Y | **0,4** |
| 4e | Useful features for supporting the intended collaborative activity | N 0..1 Y | **1** |
| 4f | Missing features | N 0..1 Y | **0,8** |
| 4g | Functionality already provided by alternative application | N 0, ? 1, 2 Y | **1,2** |
| 4h | Overall experience | - 1..5 + | **2,9** |
| **Satisfaction and trust** | | | |
| 5a | Consider using CFS in future similar activities | N 0..1 Y | **0,3** |
| 5b | Recommend it | N 0, ? 1, 2 Y | **1,7** |
| 5c | Overall experience | - 1..5 + | **2,8** |

### 3.5.4 Lessons learnt

This application is a prototype and users have evaluated the application in comparison with stable commercial products with similar but also some different features. The most positive and distinctive or innovative aspects perceived are:

- The application provides a user-friendly graphical interface that facilitates access, organization and work with a repository of files combined with discussion forums and folders shared by a group of people.
- The application provides mechanisms supporting the resolution of certain conflicts that can occur when multiple participants share a collection of files that can be modified at the same time by multiple participants.

However, the potential of the application cannot be exploited in full, as it is currently, due to the following reasons:

- The application has several limitations in the usability of the user interface, and on the user-friendliness in providing sufficient feedback or notification of changes (e.g. Awareness of what other group members are doing). This was known in advance but the development effort was limited in this part.
- The resolution of conflicts is not sufficiently sophisticated: users could see the application was resolving conflicts but they found that way they were solved was not so friendly. Again, the effort in this part was limited to simple resolution, not more.
- One great potential for CFS was the support for disconnected mode (where users could work at different places, different times and disconnected from other co-workers). As Telex did not support this mode by the time CFS was finished, this part is not currently supported by the application.
- In settings where work is collocated (face-to-face: performed in small groups at the same place and the same time) conflicts do not happen so frequently. In case they occur they can be more easily solved by social interaction. The automatic resolution may also be confusing to the participants as the current implementation does not ask or notify the user. It simply takes a default action.

The answers obtained from the evaluation show which are the main issues perceived by users that must be resolved and improved before it can constitute a better alternative than related products (even with less functionality). Once these aforementioned issues have been solved, CFS has the potential to effectively support collaborative learning (or work) particularly in activities that expand over time and distance, and where users may be partially disconnected (one Telex supports disconnected mode). Another aspect, not that visible to users is that CFS only works in a now old version of the Firefox environment: updating CFS to work on the latest Firefox engine (3.5 at the time of writing) is another need. As a result, CFS could become a useful addition to one of the most widespread web browsers in the world. Based on that, the plans of the UPC group are to spend some effort within the research group to try to solve these issues and then release to the community an open source version of a usable CFS application during the following year.

## 3.6 eMeeting

### 3.6.1 Introduction

eMeeting is an online synchronous collaborative tool allowing users to share among several remote participants not only voice and video but also documents, charts and perform polls during meetings, and can be seen as a perfect companion for the rest of the G4A applications as it provides people with a synchronous environment supporting rich interaction, giving the possibility to immediately share information or discuss and decide on an activity or project without the delays of asynchronous communication.

The evaluation session was expected to be performed with members of the “Atlas de la diversidad” network. Due to some delay on the release of the new “Atlas de la diversidad” website, it’s been not possible to perform it.

In order to evaluate the eMeeting results related to the Grid4All project, Antares prepared an evaluation session with members of the e-learning company 3iMultimedia. 3iMultimedia is already using the e-Tutor application, developed by Antares which is the basis of the eMeeting application. As an e-learning company,

3iMultimedia makes an intensive use of the application to run on-line synchronous tutoring sessions, enabling and supporting collaborative work through a widespread audience (at different locations). Its participation as evaluators was useful due to its knowledge on the usage of the application.

## 3.6.2 Evaluation method

Evaluation has been planned as a comparison between e-Tutor and eMeeting applications, having both the same functionality but running on different environments (at least partially).

The session took place the 26th May at Antares offices and was lead by Gabriel Belvedere (responsible of e-Tools at Antares) and Alicia Bou (Project Manager) with the participation of Lluis Recolons (CEO Manager), Beatriz Muntaner and José Luis Riera from 3i Multimedia.

To evaluate the eMeeting adaptation to the infrastructure of Grid4All we asked the participants to make a reservation through the room reservation module. As the integration with the rest of the services wasn't 100% complete, the request was successfully created via a web service call, and forwarded to Sakura's (shared calendar) confirmation.

Once the reservation was scheduled, we asked the users to identify with VOManager in order to log on. The first user who logged in became the session host (and this gives permission to control the application features) and the other evaluators are simple participants.

During the meeting, we tested the audio and video, and basically checked if the documents stored in VOFS were available for the session and the users were able to correctly access these documents.

With the integration tests in mind we asked evaluators to assess the experience compared to using the application in a centralized environment. To do this we modified the questionnaire by eliminating questions 1, 4, 5, 6, 13 and 14.

## 3.6.3 Analysis of feedback

| Question | Evaluator 1 | Evaluator 2 | Evaluator 3 | Comment |
|---|---|---|---|---|
| **Will the number of participants be the same for each session or will they vary from session to session?** | No, from session to session the number of users is going to be different. | No, from session to session the number of users is going to be different. | No, from session to session the number of users is going to be different. | |
| **Are your sessions intensive in direct interaction among participants?** | Some users are very active, others not so much. | Some users are very active, others not so much. | Some users are very active, others not so much. | |
| **Is the eMeeting user manual useful?** | Documentation is right, and I have been able to find all the information in the manual. | Documentation is right, and I have been able to find all the information in the manual. | Documentation is right, and I have been able to find all the information in the manual. | |
| **Could you rank from 1 to 5 how easy the use of the eMeeting's Tools is?** | File system: 2; Video: 2; calendar: 3; Chat; 1 | File system: 3; Video: 3; calendar: 3; Chat; 1 | File system: 3; Video: 2; calendar: 3; Chat; 1 | *Note: very easy (1), easy (2), not diffcult (3), difficult (4), very difficult (5)* |
| **Could you rank from 1 to 5 the utility of the eMeeting's Tools?** | File system: 2; Video: 1; Collaborative calendar: 2; Chat; 1 | File system: 2; Video: 1; Collaborative calendar: 3; Chat; 1 | File system: 2; Video: 1; Collaborative calendar: 2; Chat; 1 | *Note: very useful (1) (2) (3) (4) (5) not useful* |
| **Please, can you rate the video quality?** | 3 | 3 | 3 | *Note: poor (1) (2) (3) (4) (5) very good* |

| | | | | |
|---|---|---|---|---|
| **Do you think it may be useful to keep the files used in an eMeeting session available for other users once the session has already finished?** | Yes | Yes | Yes | |
| **While you were showing the file slides, has anyone noticed any delay?** | 3 | 3 | 3 | *Note: often (1) (2) (3) (4) (5) never* |
| **Do you find the calendar tool useful?** | 3 | 4 | 2 | *Note: very useful (1) (2) (3) (4) (5) not useful* |
| **Are the functionalities of calendar tool sufficient? If not, what would you like?** | Yes | Yes | Yes | |
| **Do you think you could use some of the eMeeting features, such as file Management or Shared Calendar independently of its use in eMeeting for other purposes in your organization?** | Yes, Shared calendar | Yes, Shared calendar | Yes, Shared calendar | |
| **What other Tools do you think would be interesting to be offered by eMeeting?** | Collaborative whiteboard | Document exporting feature | Collaborative whiteboard | |
| **How do you rank your experience with eMeeting?** | 4 | 3 | 4 | *Note: bad (1) (2) (3) (4) (5) very satisfying* |

As for the conclusions drawn from the questionnaire, we can state that:

The number of users and the interaction between them will always depend on the topic of the meeting and therefore the application should be prepared to manage any of the conditions that may occur.

The questionnaires revealed a high degree of satisfaction using the application in questions 6, 9, and 13 (How would you rate the video Quality, Do you find the calendar tool useful? How do you rank your experience with eMeeting)?

Even when the tests were satisfactory and the use of the tool was not being harmed by it integration into the environment Grid4All, the evaluators found no clear benefits in the two tools that were added (SC and VOFS), but they could see the potential of the eventual integration in a decentralized environment as Grid4All.

We conclude however that while peer-to-peer sharing through VOFS may not be of great interest to these participants, decentralization of eMeeting could be advantageous from an economic point of view since it would reduce the costs of a Flash Media Server and increase the scalability and the capacity to work when the main server degrades or fails. Re-engineering the centralized version of the collaboration tool will nevertheless incur a high initial cost that as mid-size business, they cannot afford.

### 3.6.4 Lessons learnt

Based on the perception of the evaluators as previous users of the centralized version of eMeeting, it appears that the advantages of integration into Grid4All are not clear to the end user although it should be pointed out that the assessment was not conducted in real conditions for integration, but in a simulation.

One possible advantage of integrating eMeeting with Grid4All would be the ability to access files outside the eMeeting application (by using VOFS) and the potential of using a group calendar applications, in addition, being able to distribute video and audio in a decentralized manner will also be a relevant opportunity.

## 3.7 Synthesis of the Qualitative Evaluation

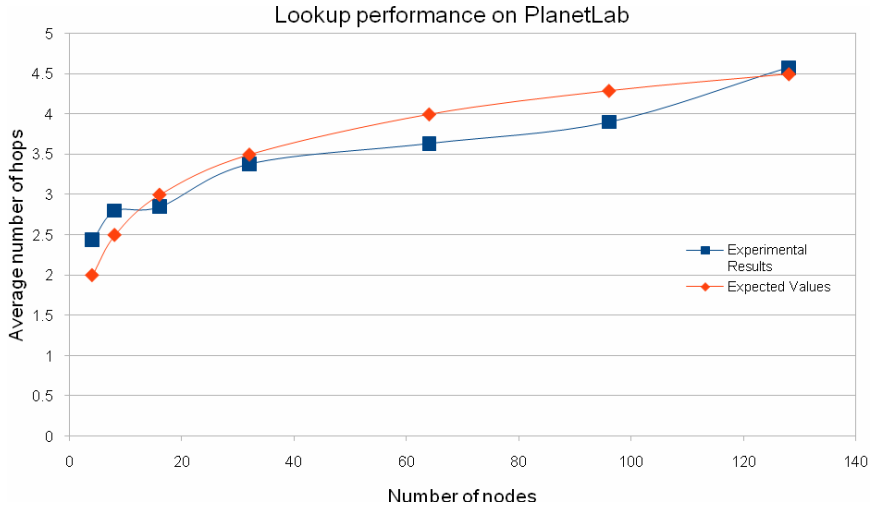
Below we have summarized in a table the main pros, cons for each analyzed result.

| Grid4all result | Stakeholder | Strengths | Weaknesses | Other | Comment |
|---|---|---|---|---|---|
| **Niche (DCMS)** | D | Performance (in decentralized architecture)<br>Tolerance to churn | Interoperability (Linux-Windows)<br>Installation proc.<br>Documentation | | |
| **Telex** | EU, D | Simplicity | No roll-back | No alternate middleware/platform provides this combination of features | Questionnaire will be posted to Telex's web site |
| **VOFS** | EU, D | Responsiveness<br>Tolerance to disconnections | Low maturity (integration to OS environment) | Easier accessible to the more technical users | |
| **CFS** | EU | Combines shared documents with versions, files and folders<br>Automatic conflict resolution | Poor user-friendliness<br>Lack of support for disconnected mode | | Could become an addition to Firefox |
| **eMeeting** | EU | Wide scope: synchronous share of voice, video, documents, charts | Integration within Grid4all is partial | | Opinions depend on the topic of the meeting |

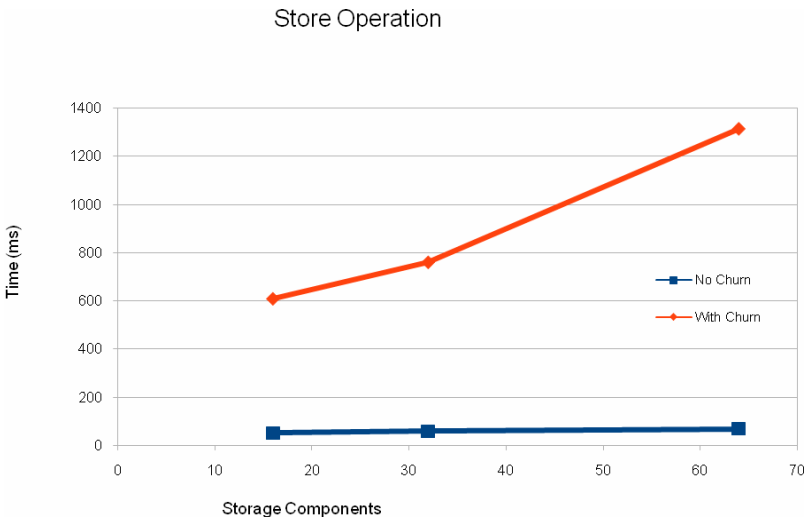
# 4. Quantitative Evaluation

## 4.1 Niche (a Distributed Component Management System, DCMS)

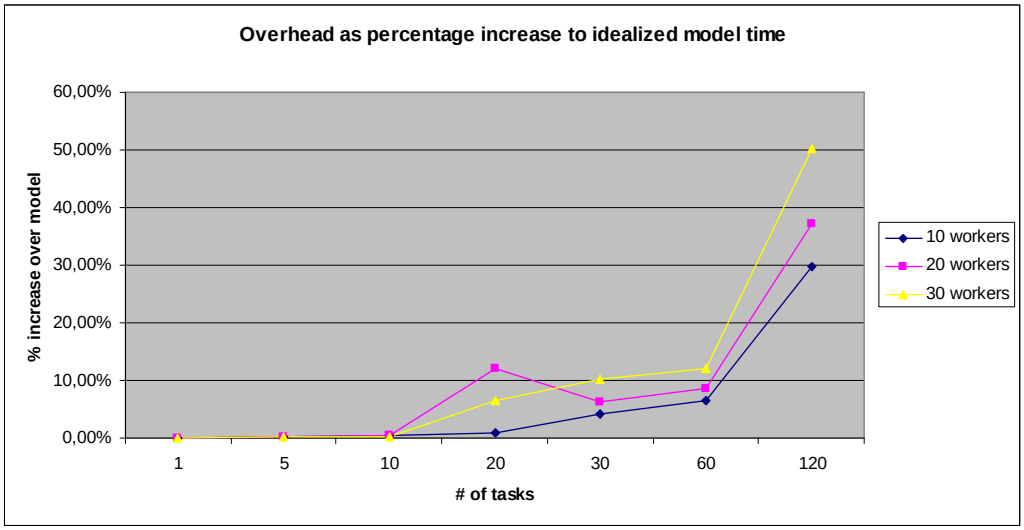
| Name | Niche structured overlay network infrastructure performance |
|---|---|
| **Test objective** | Assess the performance of underlying Niche for application deployment |
| **Testing steps** | Communication components perform series of lookups on systems (overlay networks) of varying size.<br>Measuring the average lookup time and the average number of hops. |
| **Comparative** | Average number of hops to do a lookup in varying sized overlays are compared to the expected O(log n) number of hops |
| **RESULTS** | Our performance experiments on Grid5000 show that the lookup in Niche is scalable as O(log n) and follows the expected number of hops.<br>Figure below depicts the number of hops to do a lookup in the Niche overlay with different number of nodes; and the expected theoretical number of hops, O(log n), to do lookup in a structured overlay network. It is easy to see that experimental values almost coincide with expected theoretical values.<br>The performance evaluation experiments have been performed on PlanetLab (128 nodes) and on Grid5000 (512 nodes) testbeds. |



| Name | Niche user experience based on YASS (Yet Another Storage Service) |
|---|---|
| **Test objective** | Evaluate the scalability and robustness of applications developed and managed using Niche based on the user experience under different conditions and scale.<br>The application used for evaluation is Yet Another Storage Service, YASS, which is a robust storage service that allows a client to store, read and delete files on a set of computers. The service transparently replicates files in order to achieve high availability of files and to improve access time. The current version of YASS maintains the specified number of file replicas despite of nodes leaving or failing, and it can scale (i.e. increase available storage space) when the total free storage is below a specified threshold. |
| **Testing steps** | Simulate user file store operation and measure the average time needed to store a file under different network sizes (16 – 64 nodes) and dynamic environments (node churn following the shifted Pareto distribution).<br>YASS has been configured to have a number of storage components (each on its own computer) and one front-end (YASS client) issuing store requests with configurable intensity. YASS was deployed on Grid5000 nodes. YASS performance was estimated as the average time it takes to store a file with replicas (file transfer was not included). In order to evaluate how performance of YASS depends on the system scale (number of storage elements) and the level of churn (intensity of nodes leaves and joins), we performed a number of experiments running YASS with different number of storage components (16, 32, or 64) and different rates of churn: no churn, low churn (mean node lifetime of 60 min following Pareto distribution), medium churn (mean node lifetime of 40 min following Pareto distribution) and high churn (mean node lifetime of 20 min following Pareto distribution). Note that when YASS is running under churn, the self-healing control loop restores the files and replicas, which are lost due to node failures or leaves; whereas the self-configuration control loop allocates and deploys new storage components to maintain the required size of the storage space. Thus, when YASS operates under churn, we expect that the access performance will decrease. Some results of performance experiments are presented below (see RESULTS) |
| **Comparative** | N/A |
| **RESULTS** | Figure below depicts file-store operation time versus the size of the overlay network in which YASS operates (the number of storage components, each component on its own node). Figure shows two cases: no churn, and low churn (average node lifetime of one hour following shifted Pareto distribution).<br>In the case of no churn, Niche/YASS scales with increase of the network size (number of storage components). In case of low churn, the average store operation time increases because of timeout/retry that can happen when nodes leave or fail, and also when nodes join and by this make overlay routing tables temporarily inconsistent. The results are sensitive to the timeout values that can be tuned for different environments.<br>The performance evaluation experiments have been performed on the Grid5000 test bed using up to 64 nodes. |



| Name | Niche user experience based on YACS (Yet Another Computing Service) |
|---|---|
| **Test objective** | Measure the management overheads for job management in a self-managing computing service called Yet Another Computing Service (YACS).<br>YACS is a robust distributed computing service implemented using Niche. YACS allows a client to submit and execute jobs, which are bags of independent tasks, on a network of nodes (computers). YACS guarantees execution of jobs despite of nodes leaving or failing. Furthermore, YACS scales, i.e. changes the number of execution components, when the number of jobs/tasks changes. YACS supports check pointing that allows restarting execution from the last checkpoint when a worker component fails or leaves.<br>The goal of these performance experiments is to estimate YASS self-management efficiency (namely self-configuration efficiency) measured as self-management overhead in executing jobs (bag of tasks) of different size and granularity by YACS with different amount of computing resources (masters and workers, each on its own computer). In evaluation experiments, YACS operates without resource churn (node leaves, joins, failures). |
| **Testing steps** | Divide a job to different number of tasks (of different granularity), assign the job to different number of workers and measure the time to complete the job.<br>The jobs execution time is measured as the period of time between submitting a job by the YACS front-end (client) and receiving execution results by the client. Management overhead includes the time of interaction of the front-end with YACS to find a free master for the given job, time of interaction of the master with the resource service to find free workers to execute tasks in the job; time to assign the tasks to the workers; time to communicate results from workers to its master; time to deploy new masters and workers in the case of high load.<br>The performance experiments have been performed on the Grid5000 test bed using up to 32 nodes.<br>Experiment scenario: Create a job with different number of tasks (of different granularity) and assign the job to a master that uses different number of workers to execute the job; measure the execution time it takes for the master and workers to complete all tasks in the job. Compare the execution time with the time of an idealised execution (with no overhead) that includes only useful work of performing tasks. Comparison of real execution time with the ideal one should show the management overhead. |
| **Comparative** | Ideal-case job execution without management overheads |
| **RESULTS** | The figure below shows the management overhead versus the number of tasks (load) for YACS with different number of works.<br>According to results (see figure below), the management overhead can be very low if using the optimum number of workers and task granularity. The high overhead can be accepted for critical jobs where the finish time is more important than the overhead.<br>In the performance experiments, the functional part of YACS was configured to have 1 front-end, 1 resource service (that keeps track and allocates workers and masters), and a number of workers (up to 30).<br>The performance experiments have been performed on the Grid5000 test bed using up to 32 nodes. |



| Name | YACS (Yet Another Computing Service) tolerance to churn |
|---|---|
| **Test objective** | YACS is a robust distributed computing service implemented using Niche. YACS allows a client to submit and execute jobs, which are bags of independent tasks, on a network of nodes (computers). Job execution in YACS follows the master-worker paradigm. YACS guarantees execution of jobs despite of nodes leaving or failing, i.e. it can tolerate node churn (leaves, joins, failures).<br>The goal of these evaluation experiments was to test how YACS/Niche can tolerate resource churn, i.e. how it can operate when nodes (running masters and workers) can unpredictably leave the service, fail, and new nodes can join the service. Note that when a new node joins the Niche overlay, YACS worker or master can be deployed on that node and put to service. |
| **Testing steps** | Apply different increasing intensities of churn until the system breaks, i.e. until it fails to complete tasks of submitted jobs.<br>Experiment scenario: Assume some intensity of node churn (joins, leaves and failures) in YACS. Submit a job (with configurable number and granularity of tasks) to YACS and observe whether the service is able to complete all tasks in the job. If YACS manages to complete the job given the churn intensity, it is said to be able to successfully tolerate churn of that intensity.<br>YACS was tested under the following three types of churn: (1) alternating leaves and joins, beginning with a leave; (2) only leaves; (3) random leaves and joins. Load: Jobs of 30 tasks of 6 min each with checkpoints every 10 seconds.<br>Churn intensity was chosen in the range from 1 leave/join each 240 sec (4 min) to 1 leave/join each 5 seconds. The churn intensity of 1 failure each 4 min was meant to have one failure during the lifetime of a 6-min task (of the job performed by 30 workers). The churn intensity of 1 failure each 5 sec was meant to test YACS configured with timeout of 10 sec, under the churn intensity higher than the churn intensity that YACS was expected to tolerate. |
| **RESULTS** | Churn-tolerance test experiments on Grid5000 have shown that the service is consistently successful to tolerate churn until the churn rate with nodes joining and leaving around every 10000 ms (10 sec), which is the configured value of the internal Niche timeout causing restart of stalled operations.<br>YACS churn tolerance was much better than YASS (Yet Another Storage Service). This is due to application design. Because in YASS a failure requires much more self management actions to restore files than in YACS to restart failed jobs/tasks. This shows that churn tolerance is highly application specific.<br>The churn-tolerance experiments have been performed on the Grid5000 test bed for different number of nodes (up to 192 nodes) |

## 4.2 Telex

Due to lack of resources, quantitative evaluation has not been performed. In this last period, the Telex team has focused on qualitative evaluation, to improve Telex documentation and correct bugs.

## 4.3 VOFS

Since the VOFS prototype was tailored to be a personal tool rather than an infrastructure tool, simple and portable back-ends were developed without focus on performance. Therefore, the quantitative evaluation of the prototype aimed at a usability minimum rather than high performance.

| Name | Basic file system access |
|---|---|
| **Test objective** | Test the performance of creating/writing and reading a number of files across different peers. |
| **Testing steps** | 256 files, sized 400KB each were copied into a workspace and then read from another peer. The files were distributed across 1, 2 and 4 machines in a LAN in different tests. |
| **RESULTS** | • Create/write 256 files of 400KB each: ~1.6 seconds per file<br>Performance was the same for 1, 2 or 4 machines |

| | |
|---|---|
| | • Read 256 files of 400KB each (not cached): ~0.34 seconds per file<br>Performance was the same for 1, 2, or 4 machines<br>The results indicate that the prototype is usable for lesser rates of I/O.<br>The constant performance scaling indicates that there is a constant overhead that can be eliminated with further engineering. This is very likely due to the generic and portable back-ends used, especially for storage (sqlite3) |
| **Future Work** | Performance has to be improved by removing the constant overhead from back-ends or develop high-performing new ones. |

## 4.4 SIS

| **Name** | **Query-Answering performance** |
|---|---|
| **Test objective** | Test the performance of query execution within the SIS |
| **Testing steps** | Measure response times to queries against various collections and numbers of advertisements. The evaluation process of the G4A-SIS Web Service consisted of query submissions on a variety of collections of resource advertisements. Initially, twenty five collection variations were defined, based on the five complexity levels. For each complexity level, five collections of advertisements were created, comprising 50, 100, 200, 300 and 500 advertisements respectively. For each of the twenty five collections, ten queries were submitted and the response times were measured for each one. Each entry in Table 1 corresponds to the average response time for each distinct case. The response times reported are in seconds. As shown in Table 1 (see **Erreur ! Source du renvoi introuvable.**), level 1 is for collections containing only compute node offers, level 2 is for collections containing an equal amount of compute node offers and complex compute node offers, level 3 is for collections with complex compute node offers, level 4 is for collections with cluster offers, and level 5 is for collection with complex cluster offers. The bottom right cell of the table (queries on collections of 500 complex cluster offers) does not contain any results because the client which was used to issue queries shut down the connection to the service before any results were retrieved (timeout for the client was set to ten minutes, which means the service would need more time to return results) |
| **RESULTS** | Another setup was created in order to observe the responsiveness of the G4A-SIS Web Service in more realistic cases, as far as complexity of advertisements and queries is concerned. For example, it is generally expected that most of the agents interacting with the G4A-SIS will either provide or search for compute nodes, and only few of them will offer or request clusters combined with XOR/AND operators. Therefore, an attempt was made to capture an “average” state of the G4A-SIS registry. This was accomplished by creating collections of advertisements which contain the four basic kinds of advertisements (not level 2), distributed according to the Zipfian distribution. Table 2 shows the obtained results after testing the G4A-SIS Web Service by submitting queries to registries that contained advertisements distributed using the Zipfian distribution. Ten measurements were taken for each collection, and the values in the table are the average values. In “Zipf 1” collections, the distribution exponent is equal to 1, while in “Zipf 2” collections the exponent is equal to 2. |

Table 1. G4A-SIS Response times

| Level \ #Ads | 50 | 100 | 200 | 300 | 500 |
|---|---|---|---|---|---|
| 1 | 0.654 | 1.772 | 5.572 | 13.087 | 33.899 |
| 2 | 1.219 | 4.472 | 14.318 | 28.931 | 80.498 |
| 3 | 1.981 | 6.581 | 25.765 | 56.996 | 154.878 |
| 4 | 3.200 | 11.200 | 42.649 | 98.242 | 269.048 |
| 5 | 12.237 | 49.024 | 189.731 | 450.335 | N/A |

Table 2. G4A-SIS Response times (Zipfian distribution of advertisements)

| Distribution \ #Ads | 50 | 100 | 200 | 300 | 500 |
|---|---|---|---|---|---|
| Zipf 1 | 1.988 | 6.740 | 25.565 | 58.724 | 153.182 |
| Zipf 2 | 1.297 | 4.100 | 15.284 | 35.143 | 84.905 |

| **Name** | **Automated Semantic Annotation of WSDL Services Accuracy** |
|---|---|
| **Test objective** | Measure the accuracy of USDS method (synthesis of state-of-the-art mapping methods) – a method to discover matches between WSDL part names and OWL domain ontology classes using textual descriptions of WSDL elements |
| **Testing steps** | Measure accuracy of USDS method using Semantic Service Selection (S3) Contest test bed (250 selected services of 7 domains) and compare results with state-of the art mapping methods (string-based and vector-based ones) |
| **RESULTS** | Accuracy is quite satisfactory for most of the tests (6 out of 7 domains). In Figure 1 we present accuracy results from individual mapping methods as well as from the USDS. USDS exploits information from the "name" attribute of each operation, the "name" attribute of messages and the "name" attribute of part elements of messages. Text that may have been introduced in the documentation element of operations and messages' parts, contribute further information in the process. The USDS system performs equally well to the corpus, with or without exploiting documentation (textual descriptions). This mainly happens since, in the specific corpus we have used in these experiments, all the significant terms that describe the intended meaning of the service's part name are used as components of the value of the "name" attribute of part, messages, and operation elements. Special attention should be given to domain 5 where the SimpleString method achieves accuracy equal to 0.59 (in average), while the other individual methods perform lower than 0.60, and the USDS method performs impressively well, achieving an accuracy of 0.93. In domain 7 the USDS method performs equally well with the SimpleString (base-line) method. The vector-based methods perform mildly for this domain. It should be stated that although the SimpleString method performs well in our corpus, in real life services this is unlikely to happen due to the fact that it is very rare to develop services whose part names are syntactically so much close to the labels of classes in any ontology, even in highly technical domains. |
| **FUTURE WORK** | *Improve accuracy using different synthesis operators (e.g. union, geometric mean, etc) of the USDS method* |

| | Domain 1 | Domain 2 | Domain 3 | Domain 4 | Domain 5 | Domain 6 | Domain 7 |
|---|---|---|---|---|---|---|---|
| SimpleString | 0.96 | 0.94 | 0.71 | 1.00 | 0.59 | 0.93 | 1.00 |
| VSMM | 0.44 | 0.24 | 0.53 | 0.30 | 0.48 | 0.49 | 0.44 |
| COCLU | 0.97 | 0.95 | 0.98 | 0.80 | 0.48 | 0.89 | 0.97 |
| LSA | 0.42 | 0.30 | 0.53 | 0.65 | 0.44 | 0.44 | 0.72 |
| LEVEN | 0.56 | 0.92 | 0.93 | 0.43 | 0.59 | 0.95 | 1.00 |
| USDS | 0.54 | 0.78 | 0.84 | 1.00 | 0.93 | 1.00 | 1.00 |

Fig. 1. Average accuracy results for the corpus, per method and domain

| Name | Ranking and Selection Efficiency |
|---|---|
| Test objective | Measure the efficiency of the internal Selection module to satisfy buyers' and sellers' preferences as well as to balance query load among sellers. |
| Testing steps | Measure the satisfaction and query load of agents (buyers and sellers) for different workloads and in the following environments: when agents are (i) only interested in their preferences, (ii) only interested in their load, and (iii) interested in both their load and preferences. |

| | |
|---|---|
| **RESULTS** | Our main objective during all our experiments was: to study how well this module adapts to different agents' preferences. With this in mind, we consider agents (i) that are only interested in their preferences (the preference-based case), (ii) that are only interested in their load (the utilization-based case), and (iii) that are interested in both their load and their preferences (the normal case).<br><br>Figure 2 shows the satisfaction results of these experiments with a workload range from 30 to 100 percent of the total system capacity. Values above 1 mean that agents are satisfied with query allocation and values under 1 means that they are not satisfied. We can observe in such a figure that, as expected, agents are more satisfied in the preference-based case than in the utilization-based case. But, contrary to the expected, providers are less satisfied in the preference-based case than in the normal case. During our experimentations, we observed that those providers with highly preferences tend to monopolize the queries, which causes dissatisfaction to those with lowly preferences. This phenomenon does not occur in the normal case because the ranking internal service also considers the agents' utilization. This is why agents are in average less satisfied in the preference-based case than in the normal case. However, since in the normal case providers pay more attention to their utilization as the workload increases, providers have the same degree of satisfaction, for high workloads, in both preference-based and normal cases. Concerning query load balancing (see Figure 3), the Selection module performs well in the utilization-based and normal cases. But, in the preference-based case, the Selection module significantly degrades the agents' utilization because agents have no consideration for their load. On the other side, observe that, in the utilization-based case, the Selection module follows the behavior of a load balancing algorithm, but it is much better from a satisfaction point of view. These results allow us to conclude that the Selection module allows agents to obtain from the system what they want and not what the system considers relevant for them. In other words, our results demonstrate that the Selection module ensures good levels of satisfaction as far as the system is adequate to agents and vice versa. Thus, if the agents correctly express their preferences, the Selection module allows them to reach their expectations while ensuring good query load balancing as well. |
| FUTURE WORK | Improve performance of the Selection module by optimizing the agents' preferences manager. |

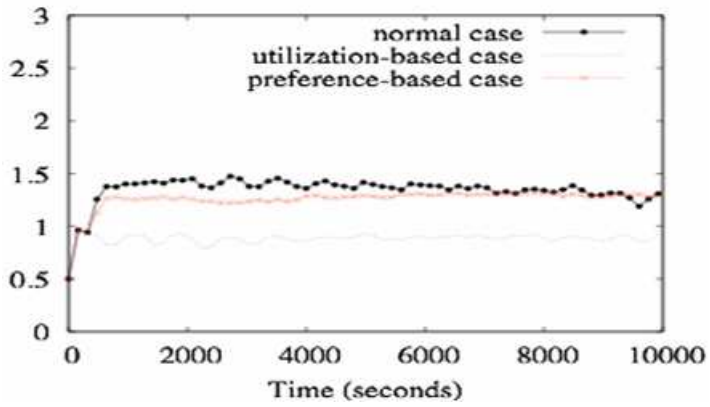



Fig. 2. Satisfaction results for a workload range from 30 to 100% of the total system capacity when participants are captive and for three kinds of providers: (i) when they are interested only in their preferences (the preference-based case), (ii) when they are just interested in their utilization (the utilization-based case), and (iii) when their utilization is as important as their preferences (the normal case).

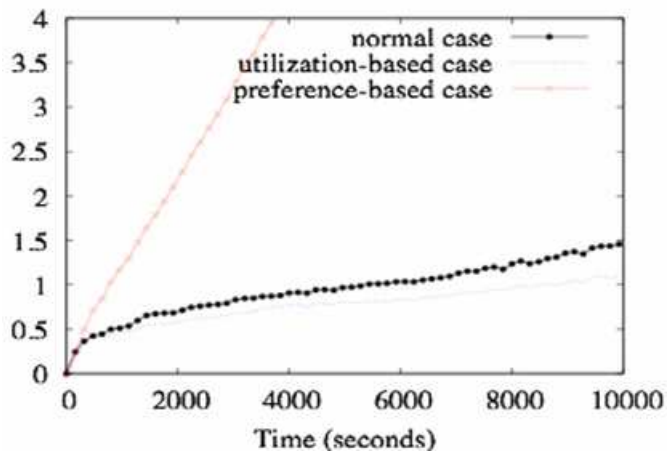



Fig. 3. Query load balancing results for a workload range from 30 to 100% of the total system capacity when participants are captive and for three kinds of providers: (i) when they are interested only in their preferences (the preference-based case), (ii) when they are just interested in their utilization (the utilization-based case), and (iii) when their utilization is as important as their preferences (the normal case).

## 4.5 MIS

| **Name** | **Evaluation within decentralised markets** |
|---|---|
| **Test objective** | Evaluate the influence of the MIS to decentralised markets |
| **Testing steps** | The bidding agents are trading in separated equilibriums prices at one of the 8 markets. Afterwards, the MIS provides the agents with a global information and the influence is measured |
| **Comparative** | -Centralised Grid markets like Tycoon. We took the central market as optimal price to reach, however the central market is limited in its scalability. Our decentralized market using the MIS performs close to the prices obtained by the central market.<br>-Decentralised markets without MIS. Our system with decentralized markets using the MIS leads to a higher social welfare and to fairer prices. Without the MIS, the system shows a lower social welfare at each market and a high variety among the prices. |
| **FUTURE WORK** | Intelligent bidding tools and strategies exploiting market information. |

## 4.6 CAS – CA

| **Name** | **Comparison of heuristics developed to solve the Winner Determination Problem for Grid4ALL CA model** |
|---|---|
| **Test objective** | Compare the allocative efficiency (surplus) and time to compute WDP with that of an exact resolution using CPLEX solver. |
| **Testing steps** | Same set of input instances (bids representing jobs from buyers and bids representing bundle offer from sellers) are resolved using the following methods:<br>● CPLEX solver<br>● Greedy methods<br>● Gradient descent method |
| **Comparative** | Qualitative comparison with other published combinatorial auction mechanisms specifically adapted for computational resources have been done. Comparable solutions for the most part do not satisfy one or more of the following criteria (a) provide heuristics-based resolutions (b) provide means for suppliers to indicate the minimum quantity of allocation (c) preferentially allocate at the earliest possible times.<br>The GreedyX heuristic proposed by SORMA project (www.ist-sorma.eu) presents 70% average efficiency. |
| **PREVIOUS STATUS (D5.2)** | When instance sizes (bids) exceed 100, CPLEX either stops (out of memory) or takes many hours to achieve reasonable gaps from best estimated solution.<br>The different greedy methods achieve different levels of efficiency. The best achieved allocative efficiency (closeness to optimal solution) reached 85% of optimality.<br>The gradient descent method has shown to reach 82% of optimality over the compared input instances. It also improves the quality of solution by increasing the number of allocated jobs and preferentially allocates (maintaining equivalent efficiency) the closest (to current time) time slots. |
| **FUTURE WORK** | The pricing scheme has been completed (below) and this module is executed once optimal allocations have been computed.<br>The combinatorial auction has not been integrated within the auction framework as planned (lack of resources).<br>It is recommended that the following tools complement the current design:<br>• Selection of appropriate heuristic (or exact method) based on input instances and other parameters (such as time to resolve, availability of licence)<br>• Weighted multi-attribute matching with appropriate similarity functions<br>• Intelligent tools for consumers to specify bids based on knowledge of price and supply/demand trends and local policies. |

| Name | Evaluation of the pricing scheme developed for Grid4ALL Combinatorial auction |
|---|---|
| **Test objective** | Study the quality of hourly commodity (resource) prices that is computed by the auction |
| **Testing steps** | Different set of input instances (bids representing jobs from buyers and bids representing bundle offer from sellers) are resolved to compute the allocations.<br>Based on the computed allocations, generate prices:<br>• Per resource type and as function of time<br>• Per resource type and as function of resource quality<br>• Per resource type |
| **Comparative** | Comparable pricing schemes (reported in literature) do not furnish either commodity prices or commodity prices as a function of time. Instead they compute prices of bundled resources. Whereas this is computationally efficient, this approach does not allow participants to understand the weight associated to each type of resource (storage, cpu, network etc).<br>Further more the only incentive-compatible pricing schemes that have been reported basically follow the Vickrey-Clarkes-Grove method. This method is not budget-balanced and more over computationally very intensive.<br>One of the methods (MACE) proposed by the Sorma project (www.ist-sorma.eu) is to generate bundle prices and distribute the surplus amongst the suppliers inversely proportional to their contribution to the welfare. This cannot be explained. The second method they propose (GreedyEx) distributes the surplus based only on the quantity of CPU that is contributed. This is not acceptable since the importance of each type of resource is dependent on the nature of the application. Furthermore by this method, suppliers who furnish only storage do not receive any surplus. |
| **RESULTS** | Not applicable |
| **FUTURE WORK** | The method that we propose determines commodity (per resource type) prices as a function of time. The computed price satisfies (a) market clearing property, i.e. losing jobs (bundles) have prices that are inferior (superior) (b) is budget balanced (c) individually rational. It is however not incentive compatible (strategy proof).<br>There is however a limitation in the quality of prices at different time-slots. If the different times are insufficiently bid, then price complementarities (between times) may not be computable, e.g., some slots may have prices set to zero. This should not be a reason to discard commodity pricing schemes; however adaptive interpretations methods should be studied and proposed. |

# 5. Conclusion

An evaluation has been performed to several innovative integrations of Grid4all results. This evaluation had two forms:

- qualitative: questionnaire-based evaluation; different stake-holders, end-users or developers external to the project have used and exercised the software results; feedback was retrieved by requesting the evaluators to fill questionnaires and also through informal discussions between experts and the evaluators.
- quantitative: technical evaluations provide quantitative metrics of different software; for each software result, quantitative evaluation has focused on the most meaningful metrics (in most cases performance).

Individual software results evaluated quantitatively to measure metrics relevant to the nature of the offered functionality. Quantitative evaluation is important since it has helped identify the weak points and the areas that require improvement. For example performance figures give thresholds that could help improve operational exploitation.

Qualitative evaluation allowed us to assess the adequacy of these Grid4all results against the users needs and more generally the stakeholders needs. The most mature set of results were identified for qualitative evaluation. Qualitative evaluation was conducted by the stake-holders according to the nature of the result. Telex and Niche were evaluated through development projects where applications (or servers) were designed and developed using these software platforms. VOFS, CFS and eMeeting were evaluated by end-users of different degrees of technical (at least in this domain) awareness.

Developers (evaluating Telex and Niche) evaluated the suitability of the APIs, the programming complexity, the feature set. End-users (evaluating VOFS, CFS and eMeeting) focused on the ergonomic aspects, the usefulness with respect to their needs and the efficiency of the products.

Finally these qualitative evaluation results show that:

- ergonomic improvements and more documentation are necessary in several cases;
- generally the software results are not in a release-quality stage yet.

Overall, the results technically **fulfill their initial objectives; and when comparisons with existing solutions are possible, Grid4All, in the context of the democratized grid, provides better functionality and/or improved performance**.

# Appendix 1: Niche Evaluation Questionnaires

## Evaluator A

**DCMS: Distributed Component Management System Evaluation Questionnaire**

We are interested to get a feedback on using the DCMS development environment for development and implementation of distributed applications with or without self-managing capabilities.

**1. About you (you can remain anonymous if you want)**

Your name: Leif Lindbäck Email: leifl@kth.se

Institution: SCS, ICT, KTH Your position: Lecturer

**2. About your application**

(a) Please, give a short description of your application.

YASS file transfer, Niche exploration

(b) What was the development stage of your application, when you considered using DCMS?

□ Idea □ Specification □ Design

X Developed □ Released □ Other:

(c) What design methodology and software engineering methodology have you used?

(d) What are the management concerns of your application?

□ Configuration X Fault-tolerance □ Optimization

□ Protection □ None □ Other:

(e) Which of the following self-managing properties should your application have?

□ Self-configuration X Self-healing □ Self-optimization

□ Self-protection □ None □ Other:

(f) Have you developed the *functional* part of your application using DCMS?

X Yes

□ No. What alternative development environment/platform have you used to develop the functional part of your application?

(g) Have you developed the *management* part (for self-management) of your application using DCMS?

X Yes

□ No. Why not?

(h) What is the expected number and type of nodes, on which your application is expected to execute?

(i) What is the approximate size of your application (in lines of code)?

**3. Your knowledge about DCMS**

(a) How did you hear about DCMS?

□ Searching the Web □ Research papers □ Friend X Colleague □ Other:

(b) Have you considered using another development environment(s) for developing of a self-managing application or a management part of your application?

X No

□ Yes. Which?

(c) What was the main motivation for considering DCMS as a development environment?

X To extend an existing application with self-management capabilities

□ To develop a self-managing application from scratch

□ To re-write my application in order to achieve self-management capabilities

X To evaluate DCMS as a development environment

□ Others:

**4. Ease of learn (documentation, examples, time to learn)**

(a) Was the DCMS documentation sufficient to learn DCMS?

□ Not at all X Partially □ Almost □ Fully

(b) How was the DCMS documentation pedagogically?

□ Excellent □ GoodX Rather good □ Rather bad □ Bad

(c) What is your opinion about the provided examples (“hello world”, YASS)?

□ Excellent □ Good□ Rather good X Rather bad □ Bad

(Short) comment:

When i say “Rather bad” i mean as an example for learning Niche, i find that YASS is good in itself. The problem is that YASS is a bit confusing as an example program since there are so many versions and it is not really clear which class does what. I do not think i would have understood it fully without help. I did not try the “hello world” program.

(d) How much time did it take you to understand and to successfully run the "hello world" example?

X I did not try it □ 1 day□ 3 days □ 1 week □ Other:

(e) How much time did it take you to understand and to successfully run the YASS example?

□ I did not try it □ 1 day□ 3 days □ 1 week X Other: About three days, but it would have been considerably more without help.

(f) In overall, how easy or difficult it was to get acquainted with DCMS in order to develop your own application?

□ Very difficult □ Difficult X Acceptable □ Easy □ Very easy

(Short) comment:

(g) Did you have to explain your DCMS-based application to other people who are not familiar with DCMS (e.g. to present its details to your colleagues or an examiner), and, if yes, how difficult it was?

X No, I didn't have to

□ Yes, I had to, but failed miserably. They need to read DCMS documentation first

□ Yes. It was somewhat complicated to introduce DCMS

□ Yes. It was rather straightforward

**5. Usability and Complexity**

(a) How much time did it take to install and to configure DCMS?

□ 1 dayX 3 days □ 1 week □ other:

(b) How much time (excluding the learning and installation phases) did it take to code and test a first prototype of your application developed using DCMS?

□ 1 dayX 3 days □ 1 week □ other:

(c) What of the following DCMS features have you used in the functional part of your application?

□ Deployment X Groups X One-to-any bindings X One-to-all bindings

□ I did not use DCMS in the functional part

(d) What of the following DCMS features have you used in the management part of your application?

□ Deployment □ Watchers □ Aggregators □ Managers

□ Sensors □ Actuation API X I did not develop the management part

(e) Which of the following DCMS features were useful in development your application?

□ Deployment X Groups X One-to-any bindings X One-to-all bindings

□ Watchers □ Aggregators □ Managers □ Sensors □ Actuation API

□ Others:

(f) Would you like to have any additional features (e.g. APIs, refinement of APIs, higher-level tools or services) in DCMS?

Said in words

(g) Did you find the DCMS programming model pertinent for designing self-managing applications?

□ No, the DCMS model did not help

□ Yes, the DCMS model did help

□ The DCMS model enabled self-management

(h) What alternative self-management programming models do you prefer?

(i) Did DCMS help you to design a modular application, with good separation of concerns?

□ I don't think DCMS matters here

□ It should have helped, but it didn't

□ Somewhat

X Very

□ Perfectly

(j) If you developed the *functional* part of your application using DCMS, how difficult or easy it was to code the application as a set of distributed components?

□ Very difficult □ Difficult □ Acceptable X Easy □ Very easy

(k) How difficult or easy it was to develop and to code the *management* part of the application?

□ Very difficult □ Difficult □ Acceptable □ Easy □ Very easy

(l) Do you find the complexity of the framework justified?

□ It is not complex at all

□ Yes, it is justified

X It is partially justified

□ No, it is too complicated (please indicate which parts of the framework or/and APIs are unnecessary complex)

(m) In overall, how easy or difficult it was to implement your application in the DCMS framework?

□ Very difficult □ Difficult X Acceptable □ Easy □ Very easy

**6. Performance, scalability, interoperability, and stability**

(a) Did your DCMS application perform according to your expectations?

X Sorry, I do not know

□ Yes

□ No; why

(b) Did your DCMS application scale according to your expectations?

X Sorry, I do not know

□ Yes

□ No; why:

(c) Have you faced performance issues with using DCMS for your application?

X No

□ Yes; what issues:

(d) Have you faced interoperability issues with using DCMS for your application?

X No

□ Yes; what issues:

(e) How would you rate the stability of the DCMS implementation?

X Excellent □ Very good □ Fair □ Not good □ Bad □ Sorry, I don't know

**7. Satisfaction**

(a) Will you use DCMS again?

□ No, why

X Yes, in which area

(b) Will you recommend DCMS to other developers?

X Yes

□ No, why:

(c) In overall, how you would you rate your experience with DCMS?

□ Excellent □ GoodX Rather good □ Rather bad □ Bad

**8. Any other comments:**

Told in words

**9. Any suggestions on improvement of DCMS:**

Told in words

## Evaluator B

**DCMS: Distributed Component Management System Evaluation Questionnaire**

We are interested to get a feedback on using the DCMS development environment for development and implementation of distributed applications with or without self-managing capabilities.

**1. About you (you can remain anonymous if you want)**

Your name: Atli Thor Hannesson Email: athan@kth.se

Institution: KTH Your position: Master student

**2. About your application**

(a) Please, give a short description of your application.

Yet Another Computing Service (YACS) is the name of the application. It is a distributed computing, or execution, service that enables the use of shared and distributed computational resources for user programmed tasks, e.g. CPU intensive and time consuming movie transcoding. It shows self-healing and self-configuration capabilities to help it survive failures and adapt to changes in the environment, e.g. from membership or load changes.

(b) What was the development stage of your application, when you considered using DCMS?

☐ Idea **X** Specification ☐ Design

☐ Developed ☐ Released ☐ Other:

(c) What design methodology and software engineering methodology have you used?

Iterative development. Initial requirements specification and preliminary design were created. These requirements and preliminary design were then divided up into three iterations. These iterations were processed sequentially, with each being designed, implemented and tested.

(d) What are the management concerns of your application?

**X** Configuration **X** Fault-tolerance ☐ Optimization

☐ Protection ☐ None ☐ Other:

Preferably all should be of concern, but only configuration and fault-tolerance are seriously considered in current YACS implementation.

(e) Which of the following self-managing properties should your application have?

**X** Self-configuration **X** Self-healing ☐ Self-optimization

☐ Self-protection ☐ None ☐ Other:

Preferably all, but only configuration and fault-tolerance are implemented in current YACS.

(f) Have you developed the *functional* part of your application using DCMS?

**X** Yes

☐ No. What alternative development environment/platform have you used to develop the functional part of your application?

(g) Have you developed the *management* part (for self-management) of your application using DCMS?

**X** Yes

☐ No. Why not?

(h) What is the expected number and type of nodes, on which your application is expected to execute?

Community grid nodes, i.e. in general ordinary users, like domestic users, small schools and institutions, with unstable and highly varied resources and connectivity. Number unknown.

(i) What is the approximate size of your application (in lines of code)?

Total number of lines is 14299. This includes every single line; e.g. source code, Javadoc and comment lines.

**3. Your knowledge about DCMS**

(a) How did you hear about DCMS?

☐ Searching the Web ☐ Research papers ☐ Friend ☐ Colleague **X** Other: school project

(b) Have you considered using another development environment(s) for developing of a self-managing application or a management part of your application?

**X** No ☐ Yes. Which?

(c) What was the main motivation for considering DCMS as a development environment?

☐ To extend an existing application with self-management capabilities

**X** To develop a self-managing application from scratch

☐ To re-write my application in order to achieve self-management capabilities

**X** To evaluate DCMS as a development environment

☐ Others:

**4. Ease of learn (documentation, examples, time to learn)**

(a) Was the DCMS documentation sufficient to learn DCMS?

☐ Not at all ☐ Partially **X** Almost ☐ Fully

(b) How was the DCMS documentation pedagogically?

☐ Excellent **X** Good ☐ Rather good ☐ Rather bad ☐ Bad

(c) What is your opinion about the provided examples (“hello world”, YASS)?

☐ Excellent **X** Good ☐ Rather good ☐ Rather bad ☐ Bad

(Short) comment: YASS shows most, if not all, major properties; i.e. groups, the group communication patterns, fault detection, application specific sensing, all management element types and their hierarchy. Therefore, I feel it is a good example. Unfamiliar with the “hello world” example.

(d) How much time did it take you to understand and to successfully run the "hello world" example?

**X** I did not try it ☐ 1 day ☐ 3 days ☐ 1 week ☐ Other:

(e) How much time did it take you to understand and to successfully run the YASS example?

☐ I did not try it ☐ 1 day ☐ 3 days ☐ 1 week **X** Other: approx. 5 days.

(f) In overall, how easy or difficult it was to get acquainted with DCMS in order to develop your own application?

☐ Very difficult ☐ Difficult **X** Acceptable ☐ Easy ☐ Very easy

(Short) comment:

Conceptually, I feel DCMS is easy to understand. In practice it is a bit more difficult:

- Development perspective: I feel the bindings complicate things considerably. There are multiple places where they are defined or specified in some way, in Fractal files and throughout code in binding calls and group templates. Multiple ways of binding also complicate, to-all, to-any, coupled with return-value bindings and without. It gets easy to lose track and make mistakes that only become apparent in runtime.
- Deployment perspective. Getting YASS to deploy and start was a time consuming and frustrating process. Getting the YACS system to deploy and start for the first time was very time consuming and very frustrating. There are configurations, scripts and files in multiple different locations that must all be correct so the different systems are able work together, build-src.xml, build.xml, beanshell scripts, build.properties the manifest files are examples of places that must be modified so that Jade properly deploys and starts. These are many places and its easy to make mistakes. Any errors can be very cryptic and hard to understand.

(g) Did you have to explain your DCMS-based application to other people who are not familiar with DCMS (e.g. to present its details to your colleagues or an examiner), and, if yes, how difficult it was?

☐ No, I didn't have to

☐ Yes, I had to, but failed miserably. They need to read DCMS documentation first

☐ Yes. It was somewhat complicated to introduce DCMS

**X** Yes. It was rather straightforward

I have not seen any comments on my thesis report regarding confusion with respect to its discussion of DCMS/Niche per se.

**5. Usability and Complexity**

(a) How much time did it take to install and to configure DCMS?

☐ 1 day ☐ 3 days ☐ 1 week **X** other: a few days, closer to 1 week than three days, for both DCMS and YASS. All initial DCMS work was done in relation to getting YASS running so I can't really distinguish time between them.

(b) How much time (excluding the learning and installation phases) did it take to code and test a first prototype of your application developed using DCMS?

☐ 1 day ☐ 3 days ☐ 1 week **X** other: Getting things done once you've solved deployment and startup issues is easy. I probably spent 2-3 days on deployment issues, but after that it was only a matter of hours to define very basic components and get messages flowing between them.

(c) What of the following DCMS features have you used in the functional part of your application?

**X** Deployment **X** Groups **X** One-to-any bindings **X** One-to-all bindings

☐ I did not use DCMS in the functional part

(d) What of the following DCMS features have you used in the management part of your application?

**X** Deployment **X** Watchers **X** Aggregators **X** Managers

**X** Sensors **X** Actuation API ☐ I did not develop the management part

(e) Which of the following DCMS features were useful in development your application?

**X** Deployment **X** Groups **X** One-to-any bindings **X** One-to-all bindings

**X** Watchers **X** Aggregators **X** Managers **X** Sensors **X** Actuation API

**X** Others: non-group bindings, i.e. to individual components.

(f) Would you like to have any additional features (e.g. APIs, refinement of APIs, higher-level tools or services) in DCMS?

- Component un-deployment API.
- Synchronous communication API between management elements. Or rather that the current binding API (and underlying system, of course) supports it.
- Bindings which support both return-value and void functions at the same time.
- Binding the same interface to many different components or groups at the same time. That is multiple instances of the same interface, bound to different components/groups.
- Difficulty in invoking the self-management. Now I have to listen for group creation events to get things moving and have the appropriate management elements deployed. I guess the goal is to have it transparent to the user but it shouldn't either cause significant "acrobatics" in the self-management part. Also, having a special component subscribed to ALL components for group creation events seems shaky.

(g) Did you find the DCMS programming model pertinent for designing self-managing applications?

☐ No, the DCMS model did not help

☐ Yes, the DCMS model did help

**X** The DCMS model enabled self-management

(h) What alternative self-management programming models do you prefer?

(i) Did DCMS help you to design a modular application, with good separation of concerns?

☐ I don't think DCMS matters here

☐ It should have helped, but it didn't

☐ Somewhat

**X** Very

☐ Perfectly

(j) If you developed the *functional* part of your application using DCMS, how difficult or easy it was to code the application as a set of distributed components?

☐ Very difficult ☐ Difficult ☐ Acceptable ☐ Easy **X** Very easy

(k) How difficult or easy it was to develop and to code the *management* part of the application?

☐ Very difficult ☐ Difficult ☐ Acceptable **X** Easy ☐ Very easy

(l) Do you find the complexity of the framework justified?

☐ It is not complex at all

☐ Yes, it is justified

**X** It is partially justified

☐ No, it is too complicated (please indicate which parts of the framework or/and APIs are unnecessary complex)

I feel the bindings are too complex and to spread out. They are defined in multiple places, in fractal files and in code. Easy to make mistakes.

Group binding templates are a source of mystery.

(m) In overall, how easy or difficult it was to implement your application in the DCMS framework?

☐ Very difficult ☐ Difficult ☐ Acceptable **X** Easy ☐ Very easy

**6. Performance, scalability, interoperability, and stability**

(a) Did your DCMS application perform according to your expectations?

**X** Sorry, I do not know

☐ Yes

☐ No; why:

(b) Did your DCMS application scale according to your expectations?

**X** Sorry, I do not know

☐ Yes ☐ No; why:

(c) Have you faced performance issues with using DCMS for your application?

☐ No

**X** Yes; what issues: component deployment occasionally block for a long, but configurable, amount of time.

(d) Have you faced interoperability issues with using DCMS for your application?

☐ No

**X** Yes; what issues: instability when running in cross-OS setting. When running between Linux and Win deployments there were multiple false failure suspicions, which wreaked havoc with YACS.

(e) How would you rate the stability of the DCMS implementation?

☐ Excellent ☐ Very good X Fair ☐ Not good ☐ Bad ☐ Sorry, I don't know

**7. Satisfaction**

(a) Will you use DCMS again?

☐ No, why

**X** Yes, in which area

If un-deployment is added I could imagine trying it out for a fault-tolerant and adaptation capable data-center infrastructure project that I'm headed for.

(b) Will you recommend DCMS to other developers?

**X** Yes

☐ No, why:

(c) In overall, how you would you rate your experience with DCMS?

☐ Excellent **X** Good ☐ Rather good ☐ Rather bad ☐ Bad

**8. Any other comments:**

No comments

**9. Any suggestions on improvement of DCMS:**

I've mentioned many of these things earlier in the questionnaire. But I'll summarize them here:

- Un-deploy support is essential for production use.
- Difficulty in invoking self-management. Having to listen for group creation events is somewhat awkward. As is having to bind to the group in order to find out what kind of group it is, which is needed to be able to deploy appropriate management elements. I would say its unlikely that systems will only have one type of group that needs management, in fact YACS has different ones.

- Bindings and interfaces are complicated in general. They are defined and named in fractal files, bound in different places in code and listed in mysterious group templates. To further complicate there are the different binding possibilities of one-to-any and one-to-all, coupled with return or no-return value bindings. It is very easy to make mistakes and this is only discovered in runtime. Some of these factors definitely have a very valid “existence”, e.g. the one-to-any and all possibilities, but I wonder how it could be simplified.
- Binding the “same” interface to multiple components/groups at the same time, i.e. interface instances. For example, the Job Master component of YACS has to bind to Workers individually and instruct each to perform a particular task. If it later wants to query the status of the task it has to rebind to each Worker component that is of interest.
- Bindings/function invocations that time out, instead of blocking indefinitely. I believe this has been implemented by now, but I haven’t had the chance to test.
- The redundant event delivery is an easy problem to solve, but would be nice to be without.
- Guaranteed consistency among management element replicas would be extremely beneficial.
- Synchronous cross management element communication
- In addition to the now possible asynchronous event communication.
- Removing the stable boot node requirement.
- Forcing NON-co-location deployment. If I want to be “absolutely” sure that an element is not on the same node as the component, or group.
- IF I recall correctly:
    - NO_SEND_TO_SENDER doesn’t work
    - Parameter-less functions don’t work
- Initial deployment and startup is a frustrating experience. Configuration parameters, scripts and file deployments are ubiquitous. It is very easy to make mistakes and the results are often very cryptic error messages. The number of systems, flow of information and sequence of events is complicated; Jade, Beanshell, Oscar etc.

## Evaluator C

**Niche (A Distributed Component Management System) Evaluation Questionnaire**

We are interested to get a feedback on using the Niche development environment for development and implementation of distributed applications with or without self-managing capabilities.

**1. About you (you can remain anonymous if you want)**

Your name: *Lin Bao* Email: *linb@kth.se*

Institution: *KTH* Your position: *Master student*

**2. About your application**

(a) Please, give a short description of your application.

*I integrate a generic policy based framework into Niche platform and demonstrate it with YASS control loop self-healing and self-configuration.*

(b) What was the development stage of your application, when you considered using Niche?

☒ Idea ☐ Specification ☐ Design

☐ Developed ☐ Released ☐ Other:

(c) What design methodology and software engineering methodology have you used?

*Design methodology, I drew the class diagram for each class, and remembered that they will be deployed on different nodes*

(d) What are the management concerns of your application?

☒ Configuration √ ☒ Fault-tolerance √ ☐ Optimization

☐ Protection ☐ None ☐ Other:

(e) Which of the following self-managing properties should your application have?

☒ Self-configuration√ ☒ Self-healing√ ☐ Self-optimization

☐ Self-protection ☐ None ☐ Other:

(f) Have you developed the *functional* part of your application using Niche?

☐ Yes

☒ No. What alternative development environment/platform have you used to develop the functional part of your application? √

*I did not develop any application on the platform. All things I was developing is nonfunctional requirements. The functional part of YASS used as a case study was given.*

(g) Have you developed the *management* part (for self-management) of your application using Niche?

☒ Yes√

☐ No. Why not?

(h) What is the expected number and type of nodes, on which your application is expected to execute?

*At least two nodes.*

(i) What is the approximate size of your application (in lines of code)? *800 lines for XACML implementation, 600 lines for SPL implementation*

**3. Your knowledge about Niche**

(a) How did you hear about Niche?

☐ Searching the Web ☐ Research papers ☐ Friend ☐ Colleague ☒ Other: √*Supervisors*

(b) Have you considered using another development environment(s) for developing of a self-managing application or a management part of your application?

☒ No√

☐ Yes. Which?

(c) What was the main motivation for considering Niche as a development environment?

☐ To extend an existing application with self-management capabilities

☐ To develop a self-managing application from scratch

☐ To re-write my application in order to achieve self-management capabilities

☐ To evaluate Niche as a development environment

☒ Others: √ *to integrate a generic policy based framework into Niche, make the self-managing behavior under the government of policies.*

**4. Ease of learn (documentation, examples, time to learn)**

(a) Was the Niche documentation sufficient to learn Niche?

☐ Not at all ☒ Partially√ ☐ Almost ☐ Fully

(b) How was the Niche documentation pedagogically?

☐ Excellent ☒ Good√ ☐ Rather good ☐ Rather bad ☐ Bad

(c) What is your opinion about the provided examples (“hello world”, YASS)?

☒ Excellent√ ☐ Good ☐ Rather good ☐ Rather bad ☐ Bad

(Short) comment: *I did not use the “hello world” example. YASS is excellent. Small application with self-managing control loops, is quite suitable to my work*

(d) How much time did it take you to understand and to successfully run the "hello world" example?

☒ I did not try it√ ☐ 1 day ☐ 3 days ☐ 1 week ☐ Other:

(e) How much time did it take you to understand and to successfully run the YASS example?

☐ I did not try it ☐ 1 day ☐ 3 days ☒ 1 week√ ☐ Other:

(f) In overall, how easy or difficult it was to get acquainted with Niche in order to develop your own application?

☐ Very difficult ☐ Difficult ☒ Acceptable√ ☐ Easy ☐ Very easy

(Short) comment:

(g) Did you have to explain your Niche-based application to other people who are not familiar with Niche (e.g. to present its details to your colleagues or an examiner), and, if yes, how difficult it was?

☒ No, I didn't have to √

☐ Yes, I had to, but failed miserably. They need to read Niche documentation first

☐ Yes. It was somewhat complicated to introduce Niche

☐ Yes. It was rather straightforward

**5. Usability and Complexity**

(a) How much time did it take to install and to configure Niche?

☐ 1 day ☐ 3 days ☒ 1 week√ ☐ other:

(b) How much time (excluding the learning and installation phases) did it take to code and test a first prototype of your application developed using Niche?

☐ 1 day ☐ 3 days ☒ 1 week√ ☐ other:

(c) What of the following Niche features have you used in the functional part of your application?

☐ Deployment ☐ Groups ☐ One-to-any bindings ☐ One-to-all bindings

☒ I did not use Niche in the functional part√

(d) What of the following Niche features have you used in the management part of your application?

☐ Deployment ☒ Watchers√ ☒ Aggregators√ ☒ Managers√

☐ Sensors ☒ Actuation API√ ☐ I did not develop the management part

(e) Which of the following Niche features were useful in development your application?

☐ Deployment ☐ Groups ☒ One-to-any bindings√ ☒ One-to-all bindings√

☒ Watchers√ ☒ Aggregators√ ☒ Managers√ ☐ Sensors ☒ Actuation API√

☐ Others:

(f) Would you like to have any additional features (e.g. APIs, refinement of APIs, higher-level tools or services) in Niche?

*With subscription, I want Niche to provide one-to-all and one-to-any binding, if the subscriber is a group.*

(g) Did you find the Niche programming model pertinent for designing self-managing applications?

☐ No, the Niche model did not help

☐ Yes, the Niche model did help

☒ The Niche model enabled self-management√

(h) What alternative self-management programming models do you prefer?

(i) Did Niche help you to design a modular application, with good separation of concerns?

☐ I don't think Niche matters here

☐ It should have helped, but it didn't

☐ Somewhat

☒ Very√

☐ Perfectly

(j) If you developed the *functional* part of your application using Niche, how difficult or easy it was to code the application as a set of distributed components?

☐ Very difficult ☐ Difficult ☒ Acceptable√ ☐ Easy ☐ Very easy

(k) How difficult or easy it was to develop and to code the *management* part of the application?

☐ Very difficult ☐ Difficult ☒ Acceptable√ ☐ Easy ☐ Very easy

(l) Do you find the complexity of the framework justified?

☐ It is not complex at all

☒ Yes, it is justified√

☐ It is partially justified

☐ No, it is too complicated (please indicate which parts of the framework or/and APIs are unnecessary complex)

(m) In overall, how easy or difficult it was to implement your application in the Niche framework?

☐ Very difficult ☐ Difficult ☒ Acceptable√ ☐ Easy ☐ Very easy

**6. Performance, scalability, interoperability, and stability**

(a) Did your Niche application perform according to your expectations?

☒ Sorry, I do not know√

☐ Yes

☐ No; why:

(b) Did your Niche application scale according to your expectations?

☒ Sorry, I do not know√

☐ Yes

☐ No; why:

(c) Have you faced performance issues with using Niche for your application?

☒ No√

☐ Yes; what issues:

(d) Have you faced interoperability issues with using Niche for your application?

☐ No

☒ Yes; what issues: √ *Synchronized operation*

(e) How would you rate the stability of the Niche implementation?

☐ Excellent ☐ Very good ☐ Fair ☐ Not good ☐ Bad ☒ Sorry, I don't know √

**7. Satisfaction**

(a) Will you use Niche again?

☒ No, why √ *I finished my thesis work.*

☐ Yes, in which area

(b) Will you recommend Niche to other developers?

☒ Yes√

☐ No, why:

(c) In overall, how you would you rate your experience with Niche?

☐ Excellent ☒ Good√ ☐ Rather good ☐ Rather bad ☐ Bad

**8. Any other comments:**

*Niche does not provide a unified development environment. Some parts are programmed in Fractal model and some parts are hard coded in Java Classes. Each MEs contains long, almost the same java code of "Fractal Stuff". I think only different codes need to be present here. Furthermore, those "Fractal Stuff" make the code less readable.*

**9. Any suggestions on improvement of Niche:**

*Subscription can support one-to-any, one-to-all binding. Maybe it has already provided, but I never used. StartManager in YASS application should be implemented in Fractal model.*

## Evaluator D

**DCMS: Distributed Component Management System Evaluation Questionnaire**

We are interested to get a feedback on using the DCMS development environment for development and implementation of distributed applications with or without self-managing capabilities.

**1. About you (you can remain anonymous if you want)**

Your name: Catalin Stefan Email: stefan.catalin@orange-ftgroup.com

Institution: France Telecom Your position: stagiaire/intern

**2. About your application**

(a) Please, give a short description of your application.

The application is an autonomic distributed storage system with adaptive replication, which aims to adjust the replication of stored data in order to provide some quality of service guarantees to the user. The system implements autonomic management to adjust to a dynamic environment.

(b) What was the development stage of your application, when you considered using DCMS?

☒ Idea ☐ Specification ☐ Design

☐ Developed ☐ Released ☐ Other:

(c) What design methodology and software engineering methodology have you used?

Rational Unified Process engineering model

(d) What are the management concerns of your application?

☐ Configuration ☒ Fault-tolerance ☒ Optimization

☐ Protection ☐ None ☐ Other:

(e) Which of the following self-managing properties should your application have?

☒ Self-configuration ☒ Self-healing ☒ Self-optimization

☐ Self-protection ☐ None ☐ Other:

(f) Have you developed the *functional* part of your application using DCMS?

☒ Yes

☐ No. What alternative development environment/platform have you used to develop the functional part of your application?

(g) Have you developed the *management* part (for self-management) of your application using DCMS?

☒ Yes

☐ No. Why not?

(h) What is the expected number and type of nodes, on which your application is expected to execute?

Estimated: hundreds

(i) What is the approximate size of your application (in lines of code)?

Abt. 10000 lines, of which 30% functional code.

**3. Your knowledge about DCMS**

(a) How did you hear about DCMS?

☐ Searching the Web ☐ Research papers ☐ Friend ☐ Colleague ☒ Other: University professor, work supervisor

(b) Have you considered using another development environment(s) for developing of a self-managing application or a management part of your application?

☒ No

☐ Yes. Which?

(c) What was the main motivation for considering DCMS as a development environment?

☐ To extend an existing application with self-management capabilities

☒ To develop a self-managing application from scratch

☐ To re-write my application in order to achieve self-management capabilities

☒ To evaluate DCMS as a development environment

☐ Others:

**4. Ease of learn (documentation, examples, time to learn)**

(a) Was the DCMS documentation sufficient to learn DCMS?

☐ Not at all ☒ Partially ☐ Almost ☐ Fully

(b) How was the DCMS documentation pedagogically?

☐ Excellent ☐ Good ☒ Rather good ☐ Rather bad ☐ Bad

(c) What is your opinion about the provided examples (“hello world”, YASS)?

☐ Excellent ☒ Good ☐ Rather good ☐ Rather bad ☐ Bad

(Short) comment:

(d) How much time did it take you to understand and to successfully run the "hello world" example?

☐ I did not try it ☒ 1 day ☐ 3 days ☐ 1 week ☐ Other:

(e) How much time did it take you to understand and to successfully run the YASS example?

☐ I did not try it ☐ 1 day ☐ 3 days ☒ 1 week ☐ Other:

(f) In overall, how easy or difficult it was to get acquainted with DCMS in order to develop your own application?

☐ Very difficult ☒ Difficult ☐ Acceptable ☐ Easy ☐ Very easy

(Short) comment:

(g) Did you have to explain your DCMS-based application to other people who are not familiar with DCMS (e.g. to present its details to your colleagues or an examiner), and, if yes, how difficult it was?

☒ No, I didn't have to

☐ Yes, I had to, but failed miserably. They need to read DCMS documentation first

☐ Yes. It was somewhat complicated to introduce DCMS

☐ Yes. It was rather straightforward

**5. Usability and Complexity**

(a) How much time did it take to install and to configure DCMS?

☐ 1 day ☐ 3 days ☒ 1 week ☐ other:

(b) How much time (excluding the learning and installation phases) did it take to code and test a first prototype of your application developed using DCMS?

☐ 1 day ☐ 3 days ☐ 1 week ☒ other: a month

Comment:

A first prototype of the application was lengthy to develop and was somewhat complex in that in included several dcms features that we were testing: statically deployed components, groups and bindings, experimentation with monitoring stacks etc. There were several problems with dynamic component deployment and dynamic bindings and obtaining information (e.g. id of manager) for statically deployed components; which took a lot of time to resolve.

Also, note that development of a first prototype was very much linked with developing an initial architecture of the application.

A hello world application was also developed simply to test that the dcms was installed correctly and to verify the way to start creating applications. This step took only 2-3 days.

(c) What of the following DCMS features have you used in the functional part of your application?

☐ Deployment ☒ Groups ☒ One-to-any bindings ☒ One-to-all bindings

☐ I did not use DCMS in the functional part

(d) What of the following DCMS features have you used in the management part of your application?

☒ Deployment ☒ Watchers ☒ Aggregators ☒ Managers

☒ Sensors ☒ Actuation API ☐ I did not develop the management part

(e) Which of the following DCMS features were useful in development your application?

☒ Deployment ☒ Groups ☒ One-to-any bindings ☒ One-to-all bindings

☒ Watchers ☒ Aggregators ☒ Managers ☒ Sensors ☐ Actuation API

☐ Others:

Comment:

deployment – initial deployment of application is done both statically, for regular components, and dynamically for management elements in a start manager.

Monitoring stacks (sensors, watchers and aggregators) are widely used in my application both to send events occurring in the system, and to send acknowledgements from regular component to manager. Clarification: a manager issues batch operations to different regular components, regular components perform operations locally and need to send back the result. Since they cannot directly connect to a manager, a monitoring stack is in place to send the result as an event.

Bindings – both types are used to send actuation commands. There are both statically assigned bindings at deploy time, and dynamic bindings (a manager binds to the required group at run time).

(f) Would you like to have any additional features (e.g. APIs, refinement of APIs, higher-level tools or services) in DCMS?

See section 9

(g) Did you find the DCMS programming model pertinent for designing self-managing applications?

☐ No, the DCMS model did not help

☐ Yes, the DCMS model did help

☒ The DCMS model enabled self-management

(h) What alternative self-management programming models do you prefer?

(i) Did DCMS help you to design a modular application, with good separation of concerns?

☐ I don't think DCMS matters here

☐ It should have helped, but it didn't

☐ Somewhat

☒ Very

☐ Perfectly

(j) If you developed the *functional* part of your application using DCMS, how difficult or easy it was to code the application as a set of distributed components?

☐ Very difficult ☐ Difficult ☐ Acceptable ☒ Easy ☐ Very easy

(k) How difficult or easy it was to develop and to code the *management* part of the application?

☐ Very difficult ☒ Difficult ☐ Acceptable ☐ Easy ☐ Very easy

(l) Do you find the complexity of the framework justified?

☐ It is not complex at all

☒ Yes, it is justified

☐ It is partially justified

☐ No, it is too complicated (please indicate which parts of the framework or/and APIs are unnecessary complex)

(m) In overall, how easy or difficult it was to implement your application in the DCMS framework?

☐ Very difficult ☐ Difficult ☒ Acceptable ☐ Easy ☐ Very easy

**6. Performance, scalability, interoperability, and stability**

(a) Did your DCMS application perform according to your expectations?

☐ Sorry, I do not know

☒ Yes

☐ No; why:

(b) Did your DCMS application scale according to your expectations?

☒ Sorry, I do not know. I have not yet fully tested the scalability of the application

☐ Yes

☐ No; why:

(c) Have you faced performance issues with using DCMS for your application?

☒ No

☐ Yes; what issues:

(d) Have you faced interoperability issues with using DCMS for your application?

☐ No

☒ Yes; what issues:

Comment:

We would like to have a way for a local application (non-dcms) to be able to connect to a dcms component and issue commands.

If two or more dcms instances are started independently, there is no way to merge them or communicate between components of different instances.

(e) How would you rate the stability of the DCMS implementation?

☒ Excellent ☐ Very good ☐ Fair ☐ Not good ☐ Bad ☐ Sorry, I don't know

**7. Satisfaction**

(a) Will you use DCMS again?

☐ No, why

☒ Yes, in which area

Comment: I would probably consider the possibility of using dcms as an infrastructure for an application that requires autonomic management.

(b) Will you recommend DCMS to other developers?

☒ Yes

☐ No, why:

(c) In overall, how you would you rate your experience with DCMS?

☐ Excellent ☐ Good ☒ Rather good ☐ Rather bad ☐ Bad

**8. Any other comments:**

There were a few problems during development that took a long time to solve, or we needed to find a way around.

Project development and testing - very long testing cycle - have to restart whole system each time

Resource management - difficult to use, unclear. example: discovery and allocation of resources the way done in yass is not

implementation of a resource manager unclear

allocation of resources to nodes - currently we can allocate one resource for a node (ex storage space)

Creation of components - can only dynamically deploy a regular component if we have a statically deployed component from which we can use the parameters

Cannot obtain id of statically deployed management component

Group creation - two ways to create it, depending on the type of component (management/regular). Should templates always be used? - unclear.

Binding problems - component binding to group works only when a group has been created using a template for the bound interface

If the group is created in regular component, and cannot add template,

If the required template is added to a supergroup, an invocation to the subgroup will work - not sure if this is intended

ONE_TO_ANY_WITH_RETURN_VALUE binding does not work

Cannot bind from regular component to management element - understandable - but we may need reply values from a one-to-any or one-to-all invocation

E.g. if a manager needs to issue multiple commands to multiple components, and the commands require processing on each component, we would not want the manager to connect to each component and wait for a return value before issuing the next command. Instead, we would like the manager to issue all commands, and each component upon completion of its computation, should connect back to the manager and provide a reply value. Note that this can be resolved currently in several ways: a) using threads on the manager to communicate with multiple components at one – this method introduces complexity for the developer, would prefer to be avoided; b) using a monitoring stack (sensor, watcher, aggr) to pass back the result. However, this is not the intended use of a monitoring stack, and also it introduces delays and potential desynchronisations.

Do not have a way for an external component to connect to a DCMS component.

E.g. we would like to have a client application – basically any local or remote application – to be able to find an (one or more) entry point to the DCMS application (a component) and be able to issue commands.

**9. Any suggestions on improvement of DCMS:**

Filters for groups- to be able to do a one to any invocation and select a member with a specified probability

Extend groups to handle management elements: would be useful for implementing an inter-ME protocol

Control over the placement of (management) components e.g. close to other frequently accessed components

Return path (bindings) from regular components to ME for simple return values (that cannot be provided immediately with a function

# Appendix 2: Telex Questionnaire

## "Telex questionnaire"

We are interested to get a feedback on your usage of the Telex middleware in developing your collaborative application.

**1- About you**

Name: X.Y Email:
Institution: Aegean University Role: Postgraduate student
Country: Greece

**2- About your collaborative application**

*a- Please give a short description of your collaborative application and its execution environment. In particular, indicate the type and number of devices involved (cell-phone, computer, ...), the network used (Internet, corporate, ...), and usage profile (disconnected operation, mobile users, ...).*

The aim of our project was to develop a software system to assist the collaborative ontology development and evolution. The operation of the system was based on the semantics of the actions performed by the knowledge engineers and workers, acting on a single ontology, as these were realized and exploited by Telex. The developed system aims to support the HCOME methodology for the collaborative engineering of ontologies, as it has been presented by the Artificial Intelligence Laboratory of the University of the Aegean, although being generic enough for supporting other methodologies and systems. Currently, our system is developed for computer devices of unlimited number. It has been tested only on local area network but we aim to have results over wider network settings soon. Both disconnected and online operation is supported.

*b- What is the development stage of your application?*
[ ] Idea [] Specified [✓] Developed [ ] Released [ ] Other:

*c- Please indicate the overall development time of your application (if applicable)*
.6. man-month

*d- Please indicate the number of lines of code of your application (if applicable)*
.4961 lines of java and 467 lines of prolog code (as of June 10, 2009)

**3- About your knowledge of collaborative middleware/platforms**

*a- How did you find about Telex?*
[ ] Searching the Web [✓] Recommendation [ ] Technical papers [ ] Other:

*b- Have you developed above a collaborative middleware/platform before?*
[✓] no [ ] yes, which:

*c- Have you considered using an alternate middleware/platform to develop this application?*
[✓] no [ ] yes, which:

**4- Ease of learning**

*a- How would you rate Telex's documentation (papers, tutorial, FAQ, etc.)?*
(bad) [ ] 1 [✓] 2 [ ] 3 [ ] 4 [ ] 5 (excellent)

*b- How would you rate the examples of application code (Telex shell, Sakura-sc, etc.)?*
(bad) [ ] 1 [ ] 2 [✓] 3 [ ] 4 [ ] 5 (excellent)

*c- How would you rate the support you got from the development team?*
(bad) [ ] 1 [ ] 2 [ ] 3 [ ] 4 [✓] 5 (excellent)

*d- Would you recommend that we organize a tutorial on Telex?*
[✓] yes [ ] no

*e- Overall, you would say that getting to learn Telex is*
(very difficult) [ ] 1 [ ] 2 [ ] 3 [✓] 4 [ ] 5 (very easy)

**5- Complexity**

*a- Is your application developed from scratch or is it a port above Telex?*
[✓] scratch [ ] port

*b- Please indicate the percentage of your application's code dedicated to interfacing with Telex*
12.98 %

*c- Please estimate the time it took to develop above Telex for the following development phases*

| | | |
|---|---|---|
| design: | .30. days | 50 % of design phase |
| coding: | .20. days | 22.22% of coding phase |
| testing: | .15. days | 50 % of testing phase |

*d- Starting from the application's specification, you would say that translating application logic to actions and constraints is*
(very difficult) [ ] 1 [ ] 2 [ ] 3 [✓] 4 [ ] 5 (very easy)

*e- Do you feel that some Telex's functions or some API's methods are unnecessarily complex?*
[✓] no [ ] yes, which:

*f- Overall, you would say that developing an application above Telex is*
(very difficult) [ ] 1 [ ] 2 [ ] 3 [✓] 4 [ ] 5 (very easy)

**6- Features**

*a- Do you think that Telex misses features that are important to your application?*
[ ] no [✓] yes, which:
i. provide the ability to roll back on decisions taken if members agree to do so.
ii. provide the sequence of schedules taken. It can be useful to define the path that led to a schedule not satisfying members, as well as the schedule to roll back instead of starting over.

*b- Are those features provided by an alternate middleware/platform?*
[✓] no [ ] don't know [ ] yes, which:

*c- Please check the three Telex features that are the most important to your application*
[✓] replication / disconnected operation
[✓] conflict detection and resolution
[✓] reconciliation of document replicas
[ ] multi-document / multi-application support
[ ] partial replication support
[ ] other, which:

*d- Are those features provided by an alternate middleware/platform?*
[✓] no [ ] don't know [ ] yes, which:

*e- Please check the type of constraints that your application uses*
high-level constraints: [ ] atomicity [✓] antagonism [✓] causality
low-level constraints[5]: [ ] enables [ ] not-after [ ] non-commuting

*f- Have Telex enabled some of your application's functions that could hardly be provided otherwise?*
[ ] no [✓] yes, which:
i. document replication and reconciliation enables ontology versions' distribution in the group
ii. conflict detection and resolution provides alternative ontology versions preserving consistency
iii. off-line operation allows members modify their ontology versions anytime, knowing that collaborative members will be informed automatically by the time a network connection is present.

*g- Overall, you would say that Telex's features are*
(useless) [ ] 1 [ ] 2 [ ] 3 [ ] 4 [✓] 5 (essential)

5not used in high-level constraints

**7- Control**

*a- Would you like to have more control on Telex operation?*
[✓] no [ ] yes, in which area:

*b- Would you like Telex to automate some of your application's tasks?*
[✓] no [ ] yes, which:

*c- Would you suggest that Telex's API should be standardized, or an effort to standardize should be undertaken?*
[ ] no [✓] yes

*d- How would you rate Telex's interoperability (application language, log format, etc.) according to the needs of your application?*
(bad) [ ] 1 [ ] 2 [ ] 3 [✓] 4 [ ] 5 (excellent)

**8- Satisfaction**

*a- Is the performance of Telex sufficient for your application needs?*
[ ] yes [✓] no, in which area:
the ability to roll back on decisions taken is an important issue

*b- Will you use Telex to develop other applications?*
[✓] yes, in which area: distributed ontologies, agents' knowledge alignment (currently under disucssion)
[ ] no, because:

*c- Will you use Telex for release-quality applications?*
[ ] yes
[✓] not in its current form, since: bugs detected. Until solutions are provided we are not able to do so.
[ ] no, because:

*d- Which aspects should be improved in future releases of Telex?*
i. documentation and tutorials
ii. demonstrating applications

*e- Will you recommend Telex to other people?*
[✓] yes [ ] no, why not?:

*f- Overall, you would say that your experience of Telex is*
(bad) [ ] 1 [ ] 2 [ ] 3 [✓] 4 [ ] 5 (very satisfying)