On the usability of grid middleware and security mechanisms

BY STEFAN J. ZASADA, ALI N. HAIDAR AND PETER V. COVENEY*

Centre for Computational Science, University College London,
20 Gordon Street, London WC1H 0AJ, UK

Usability is an all too often neglected aspect of grid computing, although it is one of the principal factors militating against the widespread uptake of distributed computing. Many resource providers on a grid infrastructure deploy a standard middleware stack and expect users to invoke the default client tools for that middleware stack to access their resources. Unfortunately, many of these middleware client tools have been developed as an afterthought, and are widely considered difficult to use. Such tools typically require a user to interact with a machine, to stage data and launch jobs, and to use digital certificates. Our experience of working with grids over many years has led us to propose a new model of grid interaction, which we call the user–application interaction model. Similar considerations have also led us to develop environments that remove digital certificates from the user’s experience, replacing them with familiar username and password authentication credentials. In this paper, we investigate the usability of this interaction model and its security system through a series of tests, which compare the usability of our systems with commonly deployed middleware tools using five usability metrics. Our middleware and security solutions are judged to be more usable than the systems in use by most of today’s computational grids.

Keywords: grid usability; grid security; Application Hosting Environment; Audited Credential Delegation; Globus; UNICORE

1. Introduction

Grid computing is supposed to facilitate access to required resources on demand. A concise definition of the term is ‘distributed computing performed transparently over multiple administrative domains’ [1]. However, the middleware tools developed to implement the grid concept have largely failed to provide the transparency and ease of use envisaged [2]. For a scientist to be able to fully exploit the power of a computational grid, the whole environment must be usable, not just a small fraction of the tools required to use it. In recent years, grid computing has evolved along side, and influenced, the development of...
cloud computing. Indeed, cloud computing is but a special case of the foregoing
definition of grid computing, in which the services are provided according to
a business model that it is hoped will allow companies to make money from
the enterprise.

Fundamental to both grid and cloud computing is the idea of providing access
to digital resources without the user knowing where a job is running. Given
that distributed compute and data resources are expected to proliferate, usable
solutions that can transparently operate across grids and clouds will become of
increasing importance. With these aspirations in mind, the purpose of this paper
is to assess the usability of the basic tools and security mechanisms in use today
that support these infrastructures.

2. Review of existing middleware tools

The user of a grid is most interested in running applications, be they single
codes or multiple distributed codes linked together in some way, such as
in the EU-funded MAPPER project (http://www.mapper-project.eu). The
TeraGrid (http://www.teragrid.org; soon to be replaced by eXtremeDigital) and
Distributed European Infrastructure for Supercomputing Applications (DEISA;
http://www.deisa.eu; soon to be superseded by PRACE) represent two of the
major high-performance computing grids in the world. These grids run the Globus
(http://www.globus.org) and UNICORE (UNiform Interface to COmputing
REsources; http://www.unicore.org) middlewares, respectively, which feature in
many other grid infrastructures worldwide.

(a) UNiform Interface to COmputing REsources

UNICORE is a grid middleware system that has been adopted for use by
several international grid initiatives, for example the EU-funded DEISA grid. It
implements a three-tier architecture in the form of client–gateway–server. Jobs
are passed around the grid as abstract job objects (AJOs), which are serialized
Java objects. The first tier of the architecture consists of a client with which
the user prepares and submits jobs and also receives output back from the job.
The middle tier consists of a gateway which controls authentication to the target
resource. This tier also contains the Network Job Supervisor (NJS), UNICORE
User Database (UUDB) and the Incarnation Database (IDB). The NJS manages
jobs and performs authorization of a user on a target resource. The UUDB maps
user certificates to logins on the resource and the IDB translates AJOs to platform
specific commands that the resource understands. The authenticated job is then
submitted to the server tier (Target System Interface or TSI) which takes care
of running the job on the target resource. With the exception of the TSI, the
UNICORE system is implemented as a Java application programming interface
and set of related programs. Version 6 of the UNICORE middleware presents a
Web services interface based on the Open Grid Services Architecture (OG-SA)

Security is maintained in the UNICORE system though the use of X.509
certificates. Prior to using the system, the user must configure their client to
use their digital certificate, imported into the Java keystore format. The user
must also import the certificate authority root certificates for any of the resources
that they want to access. When the client is launched, the user is prompted for their password to unlock the keystore. When submitting a job to the UNICORE system, the user’s client uses the certificate to digitally sign the AJO before it is transmitted to the NJS. From here, the signature is verified using a copy of the public part of the user’s certificate maintained at the NJS site, thus establishing the identity of the submitting user.

(b) Globus

Globus is deployed by many international grid projects, constituting the backbone of both the US TeraGrid and UK National Grid Service. From Globus TOOLKIT version 3 (GT3), Globus followed a service-oriented approach, tracking to a certain extent the development of the OGSA [3]. The most widely deployed version (GT4) comprises a set of Web services and a Web services container, as well as a set of non-Web service-related tools. The services are used to manage tasks such as job launching and data transfer. Recently, Globus has moved away from Web services architecture, with GT5 reverting to the client–server system previously used in GT2. The client tools supplied with all versions of Globus are based on command line.

All versions of Globus feature a security infrastructure based on X.509 [4] public key cryptography. Users must first convert their digital certificate to the privacy enhanced mail format, and place it, with the correct file permissions, into a specified folder in their home directory. Prior to issuing a Globus command, users must create a proxy version of their certificate, a short-lived credential generated from the full grid certificate, which allows grid middleware tools to perform actions on their behalf. The client tools invoke a mutual authentication protocol to establish the user’s identity, via the proxy certificate, with the Globus job manager running on the remote resource.

(c) Typical usage scenarios

To understand the way that users interact with these systems, consider the following scenario of launching the NAMD molecular dynamics application [5] on a resource on the TeraGrid using version 4 of the Globus TOOLKIT middleware. First, the user must generate a proxy certificate as described above. Once the proxy is generated, the user can stage input data (the model used to run the simulation) to the target grid resource using the globus-url-copy command. Next, a job description file needs to be generated which specifies the number of processors to use, the location of the NAMD executable, the type of job, redirects for the standard out and standard error, and arguments to the application. This is written using the Globus job description document (JDD) XML syntax, and requires the user to be aware of specific details of the machine that they are using, such as where the NAMD binary is located, and any environment variables that need to be set in order to run the application. This JDD file is used as input to the globus-ws-run command. Once submitted, the user has to invoke further Globus commands to monitor the application and retrieve its output.

The Globus use case described above is paradigmatic of the situation faced by users of many different grids, such as those running UNICORE and GLITE (http://glite.web.cern.ch/glite/). We believe that this model of interaction renders these widely deployed grid middleware toolkits largely unusable, for

Phil. Trans. R. Soc. A (2011)
the reasons discussed in the previous section. The problems are twofold: (i) the user–machine interaction model implemented by most grid middleware solutions presents too low a level of abstraction for most users and (ii) the need to deal with digital certificates simply to access the grid presents a major hurdle to users, who just want to get on with their scientific research.

Ambitious scientific projects that seek to make coordinated use of grid resources (such as NEKTAR, SPICE and Vortonics [6]) have all required significant effort to put together the middleware infrastructure needed to support them. Indeed, the investment required is beyond the means of most grid users, as a result of which they treat the machines on a grid as single ‘island’ resources instead of using multiple resources as a unified platform. Such problems with middleware tools often lead users to resort to the tools that they are most familiar with, usually SSH (or the grid-enabled version, GSISSH). Without going any further, it is plain that the vision of a grid providing transparent access to distributed resources is still far from realized, given the mountain users are asked to climb before taking their first step upon the plains. Given that this state of affairs persists over 10 years after the lofty vision was originally articulated, it is no wonder that most scientists turned their backs on grid computing a long time ago [7].

3. Usability

One of the key purposes of developing grid-computing technologies has been to provide an infrastructure for the facilitation of large-scale scientific projects [8,9]. Conversely, one of the central problems associated with the uptake of grid technologies by scientists has been the lack of ease of using the existing toolkits [2,10–12], because current toolkits are too cumbersome for a user to rapidly build prototype distributed applications. Existing toolkits are often themselves resource-intensive and require the user to install additional software packages that are not relevant to the tasks they want to perform, but are nevertheless required for the toolkit to operate.

In addition to this, some toolkits have previously required users to install custom-patched libraries for the middleware to work. This leads to system administrators having to maintain multiple copies of libraries on machines using the middleware. These heavyweight middleware tools present a barrier to use, which must be overcome before anyone can be productive in their use of grid resources. In many cases, a research group might have one or two users accessing grid resources; these users often have to maintain their own client middleware tools, fix problems with digital certificates and apply firewall policies that allow the grid middleware tools to operate. Such heavyweight requirements lead to users abandoning the grid altogether, perhaps resorting to only one or two of the available resources to run their simulations in conventional batch mode. At a stroke, users are prevented from being ambitious and thinking about what could be achieved through coordinated access to a set of powerful resources. These problems are compounded by a lack of comprehensive documentation and tutorials for users.

Another criticism levelled at many grid middleware systems is the difficulty in using the associated security infrastructure [13], particularly with reference to user management of digital certificates. Many of the existing computational grid
environments use public key infrastructure (PKI) and X.509 digital certificates as the cornerstone for their security architectures to provide secure authentication and authorization. However, it is well documented that security solutions based on PKI lack user-friendliness for both administrators and end-users [14], which is essential for the uptake of any grid security solution. The problems stem from the lengthy, multi-step procedure for acquiring X.509 digital certificates and requesting authorization to access remote resources, which involves the creation of proxy certificates as part of the authentication process. For many users, all this rigmarole is simply too much to take. They either give up or engage in practices which weaken the security of the environment, such as the sharing of the private key of a single personal certificate to get on with their tasks.

In order to design usable grid solutions, it is fundamental to understand end-users’ and resource providers’ requirements. End-users, such as scientists who are not grid technology experts, are concerned with the results of the analyses they perform on grids rather than how to acquire and use digital certificates and to install, maintain and use middleware tools, in order to launch applications in unconventional ways.

(a) Grid application virtualization

Virtualization is a broad term used in computer science to describe the abstraction of resources. It has found specific realization in many areas of computer science and information technology. Platform virtualization [15], for example, describes the separation of operating system from physical hardware, mediated by a control program (the hypervisor), allowing multiple operating system instances to run on the same physical hardware independent of each other. Resource virtualization [16] describes the process in operating system design of creating simplified abstract interfaces to specific system hardware resources. For example, the idea of virtual memory is used in many operating systems to refer to an area of disk space that is treated as an extension of the system memory.

In order to address the grid usability issues described above, we have developed grid virtualization [17]. The principal motivation behind this approach is to simplify grid use by introducing a layer of Web services between the user and the grid which hides much of the complexity of the latter and provides an abstract interface to a given scientific application deployed on a grid. These services take care of the actual process of launching the application on a particular grid resource, and reduce the act of interacting with an application to that most relevant to the user.

Traditionally, grid computing focuses on the concept of ‘jobs’ to describe distinct workloads submitted to a batch queue. These jobs can be submitted individually by a user, or as part of a larger workflow. We purposefully focus on the concept of applications. An application is a higher level concept than a job; although an application could be realized by a single job, it could equally correspond to a coupled simulation, where two codes (launched as two high-performance computing (HPC) jobs) pass parameters between themselves, or a steered application that requires steering Web services to be initialized before the code is launched, or a complex workflow.

In Coveney et al. [18], we reported our initial implementation of a grid application virtualization tool, the Application Hosting Environment (AHE), designed to simplify computational science performed on HPC grids. Version 1 of
AHE provides a first step towards the hosting of arbitrary grid applications; we have built upon its success [19–24] to add new features and extend the grid application virtualization concept to more complex scenarios, reported in Zasada & Coveney [17].

(b) Audited Credential Delegation

In order to address the security problems caused by the use of X.509 certificates outlined above, we have developed Audited Credential Delegation (ACD), described fully in Haidar et al. [25]. ACD’s aim is to remove digital certificates from the end-user experience, by the delegation of a credential managed by a project administrator or principal investigator to a group of selected users. ACD is responsible for identity management including authentication, authorization and auditing security aspects. ACD has been successfully integrated with AHE, with each user of the system given a familiar login credential—a username and password—with which to access a proxy of the certificate. ACD facilitates the rapid establishment of virtual organizations (VOs).

The design of ACD is based on the concept of ‘wrappers’. A wrapper is a connector between a component and the outside world. It enables controlled access to the functionalities of the component. Any request by a user to perform a task is intercepted by each layer of the security wrapper to establish the identity of the requester, to check whether or not the user is allowed to perform the task, to record the results of these checks in the audit log, then to perform the task on the system and, finally, to return results to the user.

ACD consists of four components.

1. A local authentication service: the current implementation supports a username–password database specifically for ACD. To be authenticated, a user has to provide a username–password pair that matches an entry in the database. To avoid known vulnerabilities in usernames and passwords, ACD adopts Open Web Application Security Project best security practices, such as storing passwords in encrypted form, rejecting weak passwords chosen by users, forcing the password length to a minimum of eight characters including special characters, and changing the password on a regular basis. This way, if the database is compromised, the attacker will not get hold of any password. There is currently work in progress to support Shibboleth in ACD to give users more options to chose from.

2. An authorization component: this component controls all actions performed in the VO. It uses the Parametrized Role-Based Access Control (PRBAC) model in which permissions are assigned to roles. The VO policy designer associates each user in the VO with the role that best describes his/her job functions. The policy is defined at the VO setup because it depends on the VO functionalities. The tasks (permissions) assigned to roles are drawn from the VO functionality.

3. A credential repository: this component stores the certificates acquired by the VO administrator and their corresponding private keys in order to communicate with the grid. Also this component enables the creation and management of VO membership. In order to allow the members of a named VO access to grid resources, the VO is assigned a digital certificate which is used behind the scenes to authenticate requests issued by the VO at the

Phil. Trans. R. Soc. A (2011)
resource provider site. The component also maintains a list of issued proxy certificates, their corresponding private keys, and the association between users and proxies in order to trace which proxy was used by which user. At the grid resource owner’s end, all requests to access grid resources appear to come from the named VO, not individual users. Two users who submitted jobs on the same grid resource site will have different proxies issued by the same VO certificate. The resource provider will not be able to tell which individual used this proxy to run an application on its resources but ACD can provide this information.

4. An auditing component: this component records all actions within the VO including authorized and unauthorized requests to perform tasks within the VO, the username who requested them, the number of login attempts and login times. This allows the VO management to identify those ACD users responsible for having performed any tasks in the environment.

4. Usability study objectives

Our observations relating to the usability of widely deployed middleware solutions outlined above have led us to ask how the user experience can be improved. To answer these questions, we have developed AHE to make managing applications easier and ACD to make dealing with security more simple for end-users. To evaluate the usability of AHE and ACD, we conducted a comparative study, to firstly compare the usability of AHE to two of the most widely deployed grid middlewares, Globus and UNICORE, and, secondly, to compare the process of using AHE with a digital certificate to using AHE combined with ACD so as to invoke only local username/password credentials to access the grid.

Specifically, our study set out to validate the following two hypotheses.

— The application interaction model implemented in AHE is more usable than the resource interaction model implemented by many other grid middleware toolkits.
— The username/password authentication implemented by ACD is more usable than the certificate-based authentication implemented by other common grid toolkits.

To test these hypotheses, we compared five different aspects of usability: (i) whether a user was able to complete a fixed task with each tool, (ii) how quickly a task could be carried out, (iii) the user’s satisfaction with the way they performed the task that they carried out, (iv) the user’s perceived difficulty in using the tool, and (v) the user’s overall impression of the tool that they used. The methodology used to assess these usability characteristics is described in the next section, and the results are presented in §6.

5. Study methodology

Our usability study comprised two sections. Globus and UNICORE are the de facto standard middleware tools used to access contemporary production grids to which we have access. By default, Globus is accessed via command line tools

*Phil. Trans. R. Soc. A* (2011)
to transfer files and submit and monitor jobs. UNICORE has both command line and graphical clients to launch and monitor applications, as does AHE. The first part of our study compared the usability of the Globus command line clients with the usability of the AHE command line client, and the usability of the UNICORE Grid Programming Environment (GPE) graphical client (which we ourselves found easier to use than the full UNICORE Rich Client) with the usability of the AHE graphical client. The version of Globus used was 4.0.5, submitting to pre-WS GRAM; version 6.3.1 of UNICORE was used, with version 6 of the GPE client. AHE version 2.0 was used for the AHE-based tests, with a pre-release version of AHE + ACD used for the security tests.

The remaining part of our usability study set out to evaluate our second hypothesis. We compared a scenario where a user was given an X.509 certificate and had to configure it for use with AHE to a scenario where a user invoked ACD to authenticate AHE. Both sections of the study can be considered as representing ‘best-case scenarios’. Firstly, all tools were installed and preconfigured for the user. An actual user of TeraGrid or DEISA would most probably have to install and configure the tools themself. In the security section of the study, the user was given an X.509 certificate to be employed with AHE. In reality, a user would have to go through the process of obtaining a certificate from their local certificate authority, a time-consuming task that can take between two days and two weeks [25].

In passing, we note that while other middleware tools, and other interfaces to Globus and UNICORE, certainly do exist, these interfaces are often community-specific and not available to all users. Our tests evaluate the default minimum middleware solutions available to TeraGrid and DEISA users.

(a) Participants

Some usability experts (notably Nielsen [26]) maintain that five is a suitable number of subjects with which to conduct a usability study, as this number of people will typically find 80 per cent of the problems in any given interface. However, our study does not seek bugs in a single interface: it asks participants to compare the features of several middleware tools to find which is most usable. To do this, we need a sufficient number of participants to be sure that our results are statistically significant. To determine the minimum number of participants required, we conducted a power analysis [27], calculating the probability that a statistical test will reject the null hypothesis or alternatively detect an effect. In order to determine the statistical significance of our results, we used a one-tailed paired $t$-test. For a reasonable statistical power of 0.8 (i.e. the probability that the test will find a statistically significant difference between the tools), we therefore determined that a minimum of 27 participants was needed, plus a few more to allow for those who might drop out for various reasons.

We recruited a cohort of 39 participants consisting of University College London undergraduate and postgraduate students, each of whom received a £10 Amazon Voucher for taking part in the study. These participants came from a wide range of backgrounds in the humanities, sciences and engineering, but none had any previous experience in the use of computational grids. This cohort is therefore analogous to a group of new users of computational grids (e.g. a first year PhD student) in terms of educational background and experience.

*Phil. Trans. R. Soc. A* (2011)
Usability of grid middleware

As discussed, our usability study was split into two sections. In the first section participants were asked to compare Globus, UNICORE and AHE by performing the following three separate tasks.

— *Launch an application* on a grid resource using the middleware tool being tested. The application in question (pre-installed on the grid resource) sorted a list of words into alphabetical order. The user had to upload the input data from their local machine and then submit the application to the machine.

— *Monitor the application* launched in the first step until complete.

— *Download the output* of the application back to the local machine once it had completed.

The second section compared the use of X.509 certificates to ACD authentication. In this section, users were asked to perform the following two tasks.

— *Configure the AHE client to use an X.509 certificate*, and then submit a job using the graphical client.

— *Authenticate AHE using an ACD username and password*, and then submit a job using the graphical client.

In order to avoid the typical queue waiting problem when using HPC resources, all of the tests ran the application on the same server, based locally in the Centre for Computational Science at University College London, which was used solely for the purpose of running the usability test application.

*(c) Data collection*

Prior to beginning the tasks outlined above, each participant was asked a number of question related to their academic background, general IT experience and previous experience of using grid middleware tools. After each task, we asked the participants to rate the difficulty of the task and their satisfaction with their performance of the task, using a Likert scale [28] (i.e. five options from *strongly agree* to *strongly disagree*). In addition, we timed how long it took the user to complete each task. After using each tool, we asked the participant to evaluate it using the system usability scale (SUS) [29], via 10 questions about their impression of the tool giving a standard measure of usability scored out of 100, which is suitable for making comparisons between tools. After completing the two sections of the study, each participant was able to give freeform comments on impressions of the tools used, if desired. While performing each task, an observer watched each participant and recorded whether or not the task was completed successfully.

*(d) Delivery*

To ease the process of data collection and tabulation (and the timings of tasks), we developed a simple Web platform from which to deliver the usability study. The study was conducted in the Centre for Computational Science at University College London.
Table 1. Summary of statistics collected during usability trials for each tool under comparison.

<table>
<thead>
<tr>
<th></th>
<th>middleware tests</th>
<th>security tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Globus TOOLKIT</td>
<td>AHE</td>
</tr>
<tr>
<td>percentage of users who found tool difficult to use</td>
<td>45.45 75.76 30.30</td>
<td>96.97</td>
</tr>
<tr>
<td>percentage of users satisfied with tool</td>
<td>27.27 53.54 47.47</td>
<td>79.80</td>
</tr>
</tbody>
</table>

College London. Each participant in the study was assigned an ID number, which they used to log on to the delivery platform. All of the various usability metrics were then recorded against this ID in a database. Before starting the study, the delivery platform displayed a page explaining to the user the purpose of the study. The observer also explained to the participant that the observer was not able to provide any assistance or answer questions relating to the tasks being performed.

The delivery platform provided Web forms on which participants could record the answers to the questions outlined in §5. The delivery platform also described the operations that the user had to carry out. Prior to performing the task, the user had to click a start button, which set a timer running for the task, and a stop button when it was completed. When performing a task, the user was given a documentation snapshot, taken from the tool’s documentation, which instructed them how to perform the task (included as electronic supplementary material to this paper). As noted, all of the tools were preconfigured on the machine used by the participant to perform the tasks. Each of the tasks in the two sections was assigned in a random order, to minimize the risk of bias entering the study.

6. Results

Our usability tests show very clear differences between the different tools tested, based on the usability metrics defined in §4. Table 1 presents key measurements from our findings. Owing to problems with the delivery platform (such as Web browser crashes half way through a set of tests), the results from six participants have been excluded from our results, meaning that the results presented have been gathered from a cohort of 33 participants.

We applied a one-tailed, paired t-test to our results to determine the statistical significance of any differences between the tools being compared. We compared the Globus command line client with the AHE command line tool, and the UNICORE graphical client with the AHE graphical client. We also compared the AHE using a digital certificate to the AHE using ACD authentication. The p-values of these t-tests are shown in table 2, along with mean scores for the five different metrics. A p < 0.05 shows that the difference between the tools is statistically significant.
Our first usability metric looked at whether or not a user could successfully complete the set of tasks with a given tool. Table 1 summarizes the percentage of participants who were able to complete all tasks for each tool. Although the failure data are measured on an ordinal scale (success, failed, etc.), we have converted them to numerical data in order to more easily compare results. The mean failure rate is shown in Table 2, with a lower score meaning there were less failures when using the tool. Also shown in Table 2 are the \( t \)-value scores; the AHE command line was found to be less failure prone than the Globus command line \( (t(33) = 1.41, p < 0.05) \), the AHE GUI was found to be less failure prone than the UNICORE GUI \( (t(33) = 1.07, p < 0.05) \) and AHE with ACD was found to be less failure prone than AHE with X.509 certificates \( (t(33) = 1.03, p < 0.05) \).

Our second usability metric was a measure of how long it took a user to complete an application run. Figure 1a plots the mean times taken to complete the range of tasks with each tool. Again, the differences are statistically significant as shown in Table 2, with participants able to use AHE to run their applications faster than via Globus or UNICORE, and AHE with ACD faster than AHE with X.509 certificates.

Our third usability metric measured user satisfaction with the tools used. In Table 1, we have summarized the percentage of participants who reported being either satisfied or very satisfied with a tool. The Likert scale data are again ordinal, but according to common practice, we have converted them to numerical data in order to compare them [30]. The mean satisfaction level is reported
in table 2, a higher score meaning that a user was more satisfied with the tool. Again, users reported being more satisfied with the AHE than with other tools, and with ACD than with X.509 certificates, as shown by table 2.

Our fourth usability metric looked at how difficult a user perceived a tool to be. Again the percentage of users who found a tool difficult or very difficult is summarized in table 1. The mean difficulty scores are shown in table 2, with a higher score meaning that the tool was perceived as being more difficult to use. The AHE GUI client was perceived as being less difficult to use than the UNICORE GUI client \( (t(33) = 1.73, p < 0.05) \), the AHE command line interface was perceived as being less difficult to use than the Globus command line tools \( (t(33) = 2.55, p < 0.05) \), and AHE with ACD was perceived as being less difficult than AHE with digital certificates \( (t(33) = 1.52, p < 0.05) \).
Our final usability metric measured a participant’s overall impression of a tool using the SUS usability scale. The mean SUS score is shown in Table 2, with a higher score meaning the tool is more usable. Again, we found statistical significance, with AHE GUI being rated as more usable than the UNICORE GUI, AHE command line being rated higher than Globus, and AHE with ACD being rated higher than AHE with digital certificates, as summarized in Table 2.

7. Discussion of results

The results presented in §6 clearly confirm our hypotheses that the application interaction model used by the AHE is more usable than the resource interaction model implemented in the UNICORE and Globus toolkits, with AHE found to be more usable for each of our defined usability metrics. We believe the reason for this is owing to the fact that AHE hides much of the complexity of launching applications from end-users, meaning that (i) there are less things that can go wrong (hence the lower failure rate) and (ii) there are less things for a user to remember when launching an application (hence the higher satisfaction with and lower perceived difficulty of AHE tools). The fact that the AHE model encapsulates input and output data as part of an application’s instance (and stages data back and forth on the user’s behalf) means that application launching is faster via AHE.

In the case of ACD security, the familiar username and password were clearly found to be more usable than X.509 certificates, but it should also be stressed that the scenario modelled here represented the ‘best-case’ scenario, where a user was already given an X.509 certificate with which to configure their client. As previously noted, in the real world a user would have to go through the laborious process of obtaining an X.509 certificate from their certificate authority, which renders the ACD solution far more usable still.

The failure rate when using a tool is dependent on all of the subtasks being completed successfully; if one task failed, it meant that the following tasks could not be successfully completed (marked ‘failed due to previous’ by the observer). This is, however, analogous to real-world scenarios where, for example, a user will not be able to download data from a grid resource if their job is not submitted correctly.

We noted particular problems experienced by participants using the UNICORE middleware, related to staging files and configuring job input files. However, these problems were not noted by the participants themselves, because of the jobs appearing to submit properly. Figure 1b plots the percentage of users reporting satisfaction with a tool alongside the percentage of users who successfully used that tool. Curiously, more users reported satisfaction with the UNICORE client than were able to use it successfully, suggesting that many participants did not realize that their jobs had not completed successfully.

The freeform comments made by users of the individual systems also provide some illuminating insights as to their usability. Regarding the use of ACD security with AHE, one participant reported ‘To deal with security issues a user is much more at ease with a simple username–password system. The use of certificates just complicates the process unnecessarily’. Another participant...
highlighted the problems involved in learning the command line parameters required to use the Globus TOOLKIT, reporting ‘there were difficulties in accessing the outside servers i.e. adding gsiftp:// or when to input the whole path into the command line’.

8. Conclusions

Our study confirms that AHE is more usable than both UNICORE and Globus, and that familiar username–password authentication provided by ACD is more usable than X.509 certificates. While it may be argued that there are other grid interfaces that can be deployed for users (Web portals for example), such interfaces are usually non-standard, and are, in practice, not available for users on the TeraGrid or DEISA. The UNICORE and Globus tools we tested embody the user–resource interaction model, whereby users have to interact with a machine to stage files and submit jobs. AHE implements the user–application interaction model, whereby users interact with their applications, leaving the AHE to interact with resources on their behalf. AHE can be seen using hybrid Software as a Service (SaaS)/Platform as a Service (PaaS) architecture, common to many cloud computing systems; we believe that the usability findings reported in this paper will therefore be useful to both the grid and cloud communities.

By removing the digital certificate from the user’s experience, we have shown that users can begin launching applications faster, and with less chance of failure. We have shown that both AHE and ACD are important tools for the computational scientist; while it is of course possible that a junior computational scientist could spend time learning the intricacies of a particular middleware tool, the time they spend doing so is time not engaged in scientific research. The real power of our approach is that computational scientists can use a simple tool to launch applications, without worrying about the particular incantations required by a machine to run a job. The interoperable nature of AHE means that it provides a single interface to a wide range of resources, from HPC grids to departmental clusters and workstations [24]. Indeed, AHE with ACD is currently being used as the basis of a practical introduction to computational chemistry course run for undergraduate students at University College London.

The study reported in this paper focused solely on the ‘best-case’ scenario, where all middleware tools and applications were pre-deployed, and the study participants were used to evaluate and compare the tools’ interfaces. In future, we plan to extend the study by examining aspects relating to the usability of middleware deployment.

The development of AHE has been funded by the EPSRC Rapid Prototyping of Usable Grid Middleware (GR/T27488/01), RealityGrid Platform (EP/C536452/1) and RealityGrid (GR/R67699) grants, and also by OMII under the Managed Programme RAHWL project. ACD has been funded through the EPSRC User-Friendly Authentication and Authorization for Grid Environments project (EP/D051754). Current development is funded by EU FP7 VPH NoE (223920) and ContraCancrum (223979) projects. The PhD studentship of S.J.Z. is funded through the RealityGrid Platform grant. We would like to thank Sacha Brostoff and Philip Inglesant for valuable contributions to the design of the usability study. Finally, we would like to thank our many current and former colleagues at the Centre for Computational Science at UCL and the Research Computing Service at Manchester University for useful discussions regarding this work.

Phil. Trans. R. Soc. A (2011)
References


