

CONTRACT NUMBER 508830

## DEISA

### DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR SUPERCOMPUTING APPLICATIONS

#### European Community Sixth Framework Programme

#### RESEARCH INFRASTRUCTURES

#### Integrated Infrastructure Initiative

#### Design of the Materials Science Portal

Deliverable ID: DE ISA-DJRA1-2

Due date: April 30th, 2005

Actual delivery date: May 15, 2005

Lead contractor for this deliverable: RZG, Germany

Project start date: May 1<sup>st</sup>, 2004

Duration: 5 years

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

## Table of Contents

Table of Contents.....	1
1. Introduction.....	1
1.1 Executive Summary.....	1
1.2 References and Applicable Documents.....	1
1.3 List of Acronyms and Abbreviations.....	2
2. Design of the Portal.....	3
2.1 Motivation.....	3
2.2 Existing Solutions and ongoing developments.....	4
2.3 The Web-Portal approach.....	6
2.4 Portal Design Guidelines.....	7
2.5 The Picture: Portal, UNICORE Client, and DESHL.....	7
2.6 Phases of the portal development.....	10
2.7 The general Portal Structure.....	11
3. Enhancements to CPMD usage.....	13
3.1 CPMD plugin version 2.5.....	13
3.2 Enhancements of the CPMD plugin.....	14
3.3 CPMD plugin portability.....	16
4. Further relevant materials science applications.....	17
4.1 WIEN2K.....	17
4.2 ESPRESSO.....	20
4.3 NWCHEM.....	20
4.4 CP2K.....	21
5. Scientific Community and DEISA Extreme Computing Initiative.....	21

## 1. Introduction

### 1.1 Executive Summary

This document is the 12-month deliverable of Joint Research Activity 1 (materials science). Activities were mainly focused on the following categories: portal design and application plugins, further selection of and work on relevant simulation codes, initiating discussions with leading computational scientists with respect to the planned DEISA Extreme Computing Initiative. Details are presented about the design of the Materials Science Portal and the improvement of CPMD application usability by enhancement of the CPMD plugin.

### 1.2 References and Applicable Documents

- [1] Deliverable SA3-1a
- [2] UNICORE, <http://unicore.sourceforge.net/>
- [3] Parallab UNICORE WebStart client, <http://portal.bccs.no/webapps/unicore/>
- [4] Java, <http://java.sun.com/>
- [5] Java WebStart, <http://java.sun.com/products/javawebstart/index.jsp>
- [6] HPC-Europa, JRA2, <http://www.hpc-europa.org/ra2.html>
- [7] UniGrids, <http://www.unigrids.org/>
- [8] OpenMolGRID, <http://www.openmolgrid.org/>
- [9] Job Submission Description Language (JSDL), <http://forge.gridforum.org/projects/jsdl-wg>
- [10] Arcon Client Library  
[https://sourceforge.net/project/showfiles.php?group\\_id=102081&package\\_id=127938](https://sourceforge.net/project/showfiles.php?group_id=102081&package_id=127938)
- [11] D. Szejnfeld, "The HPC-Europa project and GridSphere",  
[http://www.nesc.ac.uk/talks/549/Day\\_2\\_0900\\_Szejnfeld.ppt](http://www.nesc.ac.uk/talks/549/Day_2_0900_Szejnfeld.ppt)
- [12] Portlet Specification, JSR168,  
<http://www.jcp.org/aboutJava/communityprocess/review/jsr168/>
- [13] GridSphere, <http://www.gridsphere.org/>
- [14] The Spring Component framework, <http://www.springframework.org/>
- [15] The Cocoon Publishing framework, <http://cocoon.apache.org/>
- [16] Java 2 Enterprise Edition, J2EE, <http://java.sun.com/j2ee/>
- [17] PKCS#11, <http://www.rsasecurity.com/rsalabs/node.asp?id=2133>
- [18] PKCS#15, <http://www.rsasecurity.com/rsalabs/node.asp?id=2141>
- [19] PKCS#12, <http://www.rsasecurity.com/rsalabs/node.asp?id=2138>

---

### 1.3 List of Acronyms and Abbreviations

AFS	Andrew File System
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CPMD	Car-Parrinello Molecular Dynamics
DESHL	DEisa Services for Heterogeneous Management Layer
DN	Distinguished Name
EJB	Enterprise Java Bean
GGF	Global Grid Forum
GPFS	General Parallel File System
GUI	Graphical User Interface
HLRS	Höchst-Leistungs-Rechenzentrum Stuttgart
HPC	High Performance Computing
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	secured version of HTTP
IETF	Internet Engineering Task Force
J2EE	Java 2 Enterprise Edition
JCP	Java Community Process
JINI	Java network component architecture
JRA	Joint Research Activity
JSDL	Job Services Description Language
JSR	Java Specification Request
KDC	Key Distribution Centre (Kerberos)
MVC	Model-View-Controller pattern
NJS	Network Job Supervisor
O/R Mapping	Object/Relational Mapping
OASIS	Organisation for the Advancement of Structured Information Standards
OpenMolGRID	Open Computing GRID for Molecular Science and Engineering
PDA	Personal Digital Agent
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
POJO	Plain Old Java Object
RMIS	Resource Management Information System
SA	Service Activity
SOA	Service Oriented Architecture
SPRING	Java network component model
SSH	Secure Shell
SSO	Single Sign On
TSI	Target System Interface
UNICORE	UNiform Interface to COmputing REsources
UniGrids	UNICORE extension project [7]
UNIX	operating system
W3C	World Wide Web Consortium
WebDAV	Web-based Distributed Authoring and Versioning
WS	Web Service
XLST	XML Style SheeT
XML	eXtensible Markup Language
XSD	XML Schema Definition
WAN	Wide Area Network

## 2. Design of the Portal

### 2.1 Motivation

JRA1 (in collaboration with JRA3) has taken the responsibility to develop and provide means of easy access ("Portals") for users in material sciences and plasma physics to a grid enabled infrastructure (DEISA). The server side middleware for this infrastructure is based in the UNICORE server landscape (Gateway, NJS, and TSI) [1]. Hence it would be straight forward to employ the UNICORE client component as the preferred means of access to DEISA as pursued in the initial approach (see previous JRA1 and JRA3 deliverables). But it turned out that the classical UNICORE client leaves little flexibility for enhancements and expansions. Building a portal solution solely based on this client would imply many restrictions on the integration of new applications.

In the following we list requirements from a user's point of view on such a portal:

- *Easy access to computing resources*

The access to computing resources needs to be as easy as possible. All supercomputing users are familiar with secure shell login (SSH) and command line tools for accessing a single computing site. Many of them would like to skip even this, if they had other more comfortable ways of controlling their jobs.

Dealing with a Grid infrastructure, the SSH approach is even more complicated. That is why there exist client tools such as the classical UNICORE client.

- *First time contact*

It seems important that a user does not need to go through many hassles until he/she is set up to use these resources.

For example, to contact a UNICORE Grid Infrastructure, a user currently needs to

1. download and install Java
2. download and install the classical UNICORE client
3. download and install the appropriate UNICORE plugins
4. set up the PKI infrastructure (key and trust stores)
5. configure the classical UNICORE client

The trickiest parts are certainly steps 4 and 5 since those are the ones most prone to mistakes, even when carried out by experienced users.

A Web-Portal solution, on the other hand, radically reduces the efforts of first time contact. The client application is the web browser. It is, of course, still dependent on a correctly configured browser for the PKI part, but that is all.

- *An easy way to submit job, check job results and perform analysis*

- *the unsophisticated and intuitively usable GUI*

It is certainly of great help to a user to be able to use a single tool with an intuitive graphical user interface in order to submit jobs to a Grid infrastructure, check results and perform parts of the result analysis.

Rich clients, as well as Web-Portals, are in principle able to fulfil this task.

Retrieving information from a job, such as job status or preliminary results should take a matter of seconds and involve a minimum of user actions.

- *Help with the creation of input files for complex material science applications*

One of the things which is cumbersome and error-prone is the creation of input files. Any user wishes to be able to employ tools which support him/her by performing this task. A graphical user interface to a Grid Environment is perfectly suited to fulfil this wish. At present the UNICORE client provides an extension point by supporting so-called plugins. These integrate external applications seamlessly into the client and can, depending on the level of sophistication, realize almost every user community's wishes.

- *Concentrate on science rather than becoming a Grid Computing Expert*

Most of the users do not want to have to think differently about a grid system than they think about a local cluster. It is desirable to hide from the user as much of the complexity as possible that a Grid infrastructure naturally has.

Next is to analyse what solutions are already present or in development (as far as we do know) or are being developed in order to fulfil those requirements.

## **2.2 Existing Solutions and ongoing developments**

First of all there are several solutions and/ or new developments to be considered:

- *Classical UNICORE Client [2]*
- *Parallab Web Start version of the UNICORE Client [3]*
- *HPC Europa JRA2 [6]*
- *UniGrids [7]*
- *OpenMolGRID [8]*

### **The classical UNICORE client**

The classical UNICORE client is a Java [4] implementation of a rich client. It was initially developed along with the first version of UNICORE. Its design is based on the decisions made at that time. In the late nineties, when the original version of the client had been developed, it was usual to combine code for viewing windows and widgets together with control functionality and model operations. Among others this design decision is still present in the code base we are dealing with today.<sup>1</sup>

- Although the concept of plugins was a revolutionary idea in those days, the inherent mixture due to the design decisions of the original authors makes it complicated to follow simple, but extremely useful, patterns such as Mode-View-Controller (MVC). The client's code does in large parts not cleanly follow the principle of separation of concerns. Code bases for showing graphical elements are mixed with code for evaluating events (e.g. when a user presses a button) and actions which have to be carried out (e.g. sending Job Objects to the server side). This makes it extremely cumbersome and error-prone to develop

---

<sup>1</sup> The Unigrids project is attacking these design disadvantages and is planning to develop new client code bases; see below.

sophisticated plugins as wished by the DEISA user communities represented by JRA1 and JRA3.

One of the most prominent examples is the CPMD plugin. It was developed in the natural fashion of the UNICORE code base. Now, as some years have passed, new versions of CPMD need to be supported and new features are wished to be implemented by users. Due to the UNICORE client's design weaknesses, extending the plugin has become a by far bigger task than expected.

- Currently, the UniGrids project [7] is developing a new set of UNICORE clients. It is not clear if the old plugins will be supported by the new client(s) or need to be refactored.
- A growing number of users are employing cryptographic devices, so-called crypto-tokens and smart cards, to store their private key and certificates. While all kinds of browsers are able to communicate with those devices by using the PKCS#11 [17] (and sometimes in addition PKCS#15 [18]) standard, the UNICORE client can only deal with a more insecure version of key repositories, the JAVA and PKCS#12 [19] software key stores.

### **The Parallab Web Start Solution**

Parallab support both Globus and UNICORE for their Grid endeavours. In this context they provide a Java WebStart [5] version of the UNICORE client. The aim is to make it easy for the user to work with the UNICORE client by simply clicking on a special icon on an ordinary web page. However, their solution has several disadvantages which need to be overcome.

- The UNICORE client code is not signed. Signing is essential in order to trust the Java code transmitted over an insecure line and giving it all permissions on the target system (the client system).

Solution: Sign the UNICORE client code. The user needs to trust the signer: this is feasible and a minimum measure to ensure unaltered client code is installed on the client.

- Plugins cannot be transferred in the same session.

Solution: Use an extra plugin installer which could be deployed by again using the WebStart technology or incorporated in a modified version of the UNICORE client.

Naturally this approach has all the known disadvantages of the UNICORE client itself, since Parallab did not change anything internally.

As a summary, deploying a rich client using web start is certainly a good idea that should be communicated to the UniGrids project. However, some work in addition to that already been done by Parallab would be necessary.

### **HPC Europa JRA2**

HPC Europa's JRA2 is developing a Single Point of Access to complex computing systems like computing Grids. Their objectives seem to be similar to what JRA1 and JRA3 have in vision: The development of a *uniform, flexible and intuitive user access to Grid resources from anywhere* [6].

Dawid Szejnfeld from the Poznan Supercomputing and Networking Centre presented a talk at the "Grid Sphere and Portlets workshop" held at the escience Institute, Edinburgh in March 2005 [11]. During this presentation he described the objectives and implementation basics of the HPC-Europa JRA2 Portal (HPC-Portal in the

following). The HPC-Portal is based on two frameworks: the Portlet Specification (JSR168) based GridSphere framework, which mainly has the task to present the view, and the Spring component framework. Their approach is indeed very similar to the one presented in the next section. Unfortunately, DEISA's JRA1 and JRA3 cannot yet benefit from this project since HPC Europa's JRA2 is unfortunately still far from the point where they are going to release their product.

### **OpenMolGRID**

OpenMolGRID is a suite of plugins serving a specific purpose: providing a Grid enabled infrastructure for bioinformatics applications. Hence, OpenMolGRID does not change or improve the UNICORE client itself, but extends it to provide the functionality needed to do Bioinformatics research.

### **UniGrids**

The aim of the UniGrids project is to equip UNICORE with support of standards (or currently emerging standards) such as the Job Submission Description Language (JSDL) [9]. One of the benefits of the outcome of this ambitious project will be that clients will be able to talk to any Grid Server side implementing the GGF's and other standards, among them naturally UNICORE, but also the Globus Toolkit (version 4). Hence, the UniGrids project is extremely important for DEISA, especially for SA3, SA5, and all the JRAs which employ UNICORE in one way or another.

Although it is going to provide new ways of accessing a UNICORE based infrastructure, again DEISA unfortunately cannot yet benefit from the UniGrids project at this early stage.

## **2.3 The Web-Portal approach**

The UNICORE client's deficiencies have been pointed out. Other promising solutions are under development, but far from being ready to be used in production within 2005.<sup>2</sup> It cannot be neglected that the demand is there and there is a growing number of UNICORE client users.

So JRA1 and JRA3 are faced with four options:

1. Investing in aged code base, which is soon (by the end of 2006) to be replaced by a new product without knowing if the JRAs' developments can be integrated.
2. Waiting for a JRA7 or UniGrids solution
3. Waiting for HPC-Europa's JRA2 developments
4. Creating an application which is different from the current approaches and hence fills a niche, an application which is extendable to future changes which are sure to come and which is developed in collaboration with JRA7.

Since the classical UNICORE client is going to be replaced anyway within the next 18 months, we are not going to invest in it.

JRA7 is working hard on providing a Web Service based front end to UNICORE. That involves most of their resources, so that they are not able to directly assist JRA1 and JRA3 to build up a portal base for them in time.

HPC-Europa efforts are promising but it is not clear when a version of their Portal solution will be available. Rather, since none of the members can make a statement

---

about the availability at this point of time, it has to be assumed, that is going to be too late to be of use for JRA1 and JRA3 directly.

In order to be able to provide their deliverables in time, JRA1 and JRA3 have chosen to start with the development of a Web Portal (based on a web application front-end and model component backend) on their own. Aims are to integrate JRA7 work as soon as it is available and moving work from JRA1 and JRA3 to JRA7 as soon as JRA7 has free resources, and to benefit from the outcome of the UniGrids and HPC Europa JRA2 projects on the longer terms. Also, at least of the outcome of this development will presumably be of interest to HPC-Europa's JRA2.

## **2.4 Portal Design Guidelines**

The Portal application has been designed on several main principles

- *Security*  
A secure system is essential in a Grid Infrastructure.
- *Easy to use GUI*  
An easy to use GUI is the key to success in terms of user acceptance.
- *Intuitive GUI*  
An intuitive GUI provides functionality in the way it is expected even by the inexperienced user
- *Extensibility*  
Easy and comfortable extensibility is the key to success in terms of developer acceptance
- *Built on freely available, enterprise quality components*  
Components which are freely available, as well as having enterprise quality are the key to success over a long period of time.  
And those components speed up the development process since they can be used "out of the box".
- *Separation of concerns*  
By honouring basic software design patterns like the separation of concerns developers can work in parallel on different aspects of the code (which speeds up development time and also supports team development).  
Since clear interfaces have to be defined between components, the number of errors and bugs is naturally reduced (speeds up time to delivery).

The direct benefit of honouring the above principles is that new and emerging technologies can seamlessly be integrated without producing unwanted effects on the behaviour of the rest of the software.

For example, the portal design and its implementation directly allow

- Easy field testing of new technologies (e.g. Instant Messaging or SMS messaging upon Job Status changes)
- Content Management
- Integrated File Staging

All of these will be described in more detail in section 3.

## 2.5 The Picture: Portal, UNICORE Client, and DESHL

How does a Portal Application fit into the picture of available solutions for DEISA? Especially in the context of JRA7, which is developing a Web Service based access to UNICORE, it is important to ask how the Portal will integrate into JRA7s work.

The following figures address these questions.

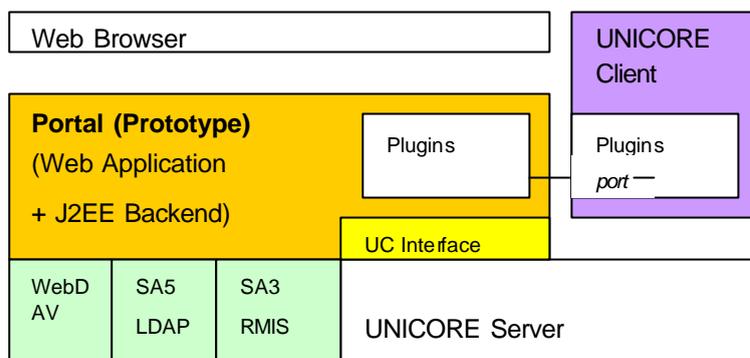


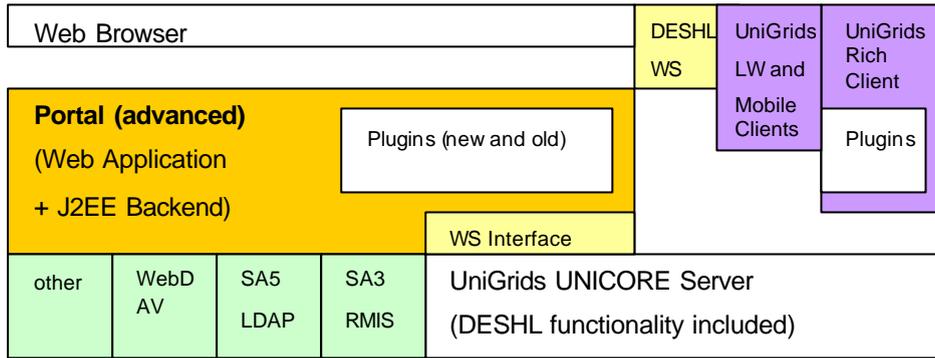
Figure 1: Stage 1, ca. May 2005

As pointed out before, the Portal application will be accessed by the means of an ordinary web-browser. So the Portal Application which runs on the server side has to implement much of the logic currently present in the classical UNICORE client. The most important parts, as JRA1 and JRA3 are concerned, are the UNICORE plugins, which are the GUI components used to build job requests for special scientific applications such as CPMD. In order to be able to communicate with UNICORE, the Portal application needs an interface to the UNICORE server side. Unfortunately, this cannot be achieved by simply employing the Arcon library [10] alone. This library is designed to meet the requirement of a single user application. The Portal Application, as well as the DESHL server side, is naturally an application which has to handle multiple requests from different users at the same time.

An interface has to be developed which can run in such an environment. Currently, DESHL development is concentrating on providing a correct Web Services based implementation to the client and chooses to postpone its work on the Interface to the UNICORE server side (by accepting the single user constraint for now, and hence breaking the end to end security). The Portal implementers were forced to also include the UNICORE interface development, with the intention to hand over their interface to the JRA7 developers, when they are ready to integrate it.

In Stage 1 we will have a picture where the main interface to UNICORE is still the classical UNICORE client but we can already present a Prototype of the Portal Application which already embeds some of the ported plugins, as well as the necessary functionality to interface DEISA's UNICORE server environment and some of DEISA's other infrastructure components. In addition to normal UNICORE client functionality, the Prototype of the Portal application is able to interconnect to WebDAV repositories, the SA5 user database, and the SA3 RMIS. The





**Figure 3: Stage 3, options for solutions expected for late 2006**

Although the components developed will certainly mature, it is conceivable that some will have been replaced in addition to what is depicted in Figure 3. For example, the web publishing framework used (cocoon for the prototype and the first release) could be replaced by the HPC-Europa’s JRA2’s choice. The decision process will occur in close cooperation with all other DEISA partners, with some also being engaged in the ongoing UniGrids and / or the HPC Europa JRA2 projects, with the aim to coordinate efforts and bundle forces to work together for an optimal Grid access solution for supercomputing applications.

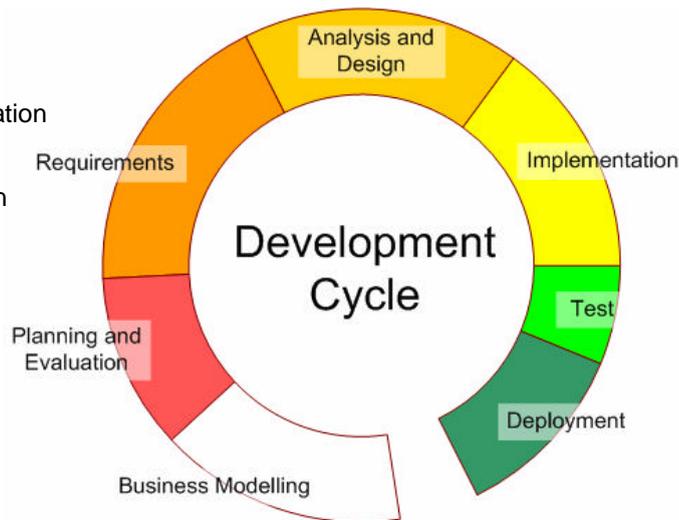
**2.6 Phases of the portal development**

This section describes the design of the portal application as well as some implementation details valid for the Portal Application Prototype.

The development of the Portal has undergone several phases. Other phases are planned to follow and will be described in deliverable DEISA-DJRA1-3.

A software project usually follows seven phases:

1. Business Modelling
2. Planning and evaluation
3. Requirements
4. Analysis and Design
5. Implementation
6. Test
7. Deployment

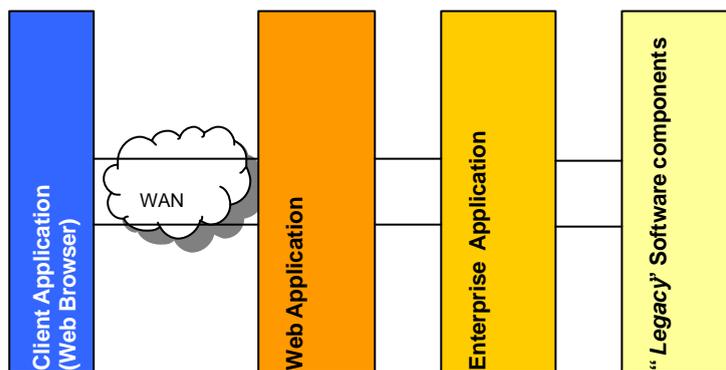


JRA1's and JRA3's portal project has just finished phase 4 (Analysis and Design) with the provision of a prototype Portal Application. Hence, the major Portal components have been identified. The interfaces to external component are known and have been defined, as well as the interfaces between the internal components. A Proto-Type application has been developed in order to prove that the principle design is working.

In the following, all major components are being described. Then the reader will be given a closer look at the security solution employed. Finally, implementation details of the prototype application will be revealed.

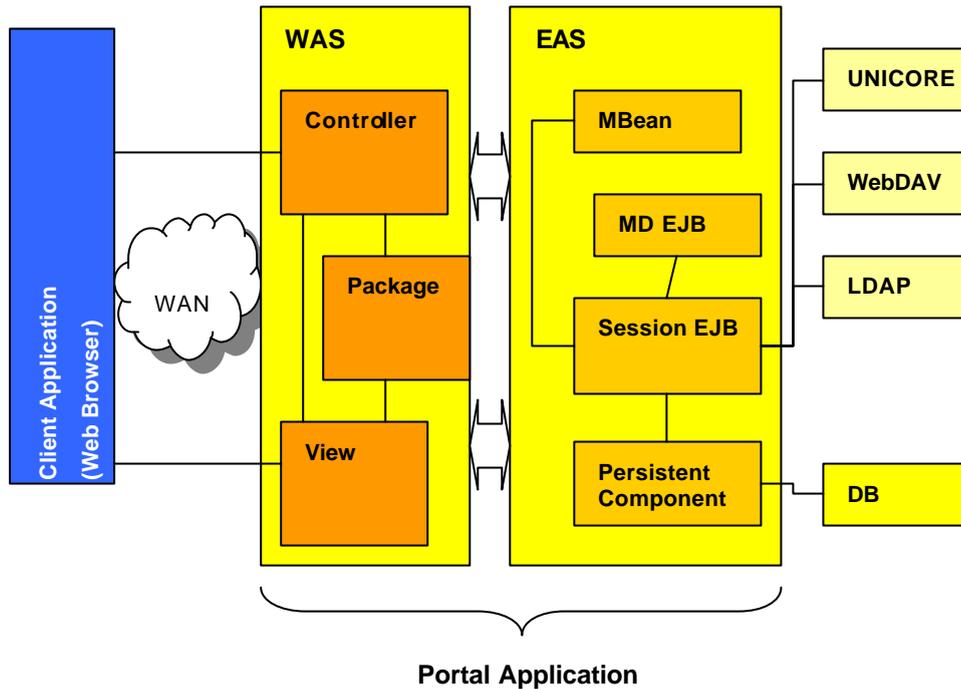
## 2.7 The general Portal Structure

The portal application utilizes the so-called Four-Tier Pattern, where the tiers are the client (the web browser), the web-application, the Component Container, and the outside components (UNICORE, WebDAV, etc). The general structure of such a Four-Tier Pattern is depicted in Figure 4. The client (e.g. the Web Browser) sees and interacts only with the Web Application Layer. The Web Application utilizes external components (from the Enterprise Server Application Layer) to perform the tasks requested by the client. These components deal with legacy software and other packages which have a fixed external interface such as UNICORE.



**Figure 4: In the Four-Tier Model, the client interfaces, the enterprise application, and the legacy software components are separated and communicate via a few, well-defined interface**

Figure 5 shows a more detailed view of the Portal Application tiers. The Portal Application is divided into two functional parts, the Web Application (hosted by a Web Application Server – WAS) and the Enterprise Application (hosted by an Enterprise Application Server – EAS). Components in the Web Application handle client connection (WAS + Controller), initiate actions upon client requests (Controller with the help of the Packages), assemble responses (View), and interact with EAS hosted components (Controller, Packages and View, if appropriate).



**Figure 5: Overview of the fundamental design and architecture of the JRA1/JRA3 Portal Application. Interaction of the different components is indicated by arrows.**

The components hosted by the EAS perform actions requested by WAS and other components (synchronously with Session EJBs, stateless or stateful depending on the context, or asynchronously by the help of Message Driven EJBs – MD EJBs), implement the persistent layer (Entity EJBs and Hibernate Objects) and perform management tasks (Management Beans – MBeans).

Although Figure 5 implicitly implies that the implementation model is J2EE, the portal application and its components could be implemented using alternative component models such as CORBA or in general any service oriented architecture (SOA),..

Note that components can be made of clusters of components. Here EJBs are seen as the atomic components.

### 3. Enhancements to CPMD usage

The CPMD code is a parallelized plane wave/pseudopotential implementation of Density Functional Theory, particularly designed for ab-initio molecular dynamics [<http://www.cpmid.org>].

The previous deliverable [DEISA-D-JRA1-1] described the grid-enabling of CPMD, i.e. the CPMD application version 3.9.1 was deployed and tested at different DEISA sites, and we chose UNICORE [<http://www.unicore.org>] as the underlying middleware that allows the submission of CPMD task chains seamlessly by means of the UNICORE client. In order to configure the CPMD tasks we employed the CPMD plugin 2.5 (build 0 from 24/06/2002) developed at the Research Center Jülich FZJ [cf. <http://www.fz-juelich.de/unicoreplus/download/cpmid/>].

We now report new developments on the CPMD plugin for UNICORE, based on a version originally written by V. Huber [V. Huber, "UNICORE: A Grid Computing Environment for Distributed and Parallel Computing", in *Parallel Computing Technologies*, Proceedings of the 6th PaCT Conference 2001, pp. 258, Springer-Verlag LNCS 2127, 2001]. The source code of version 2.5 which is the starting point of the new developments, has been kindly provided by FZJ via sourceforge [[www.sourceforge.org](http://www.sourceforge.org)]. We like to thank V. Huber, B. Schuller, M. Krack and A. Seitsonen for helpful comments on possible benefits of an implementation of a backfill functionality.

#### 3.1 CPMD plugin version 2.5

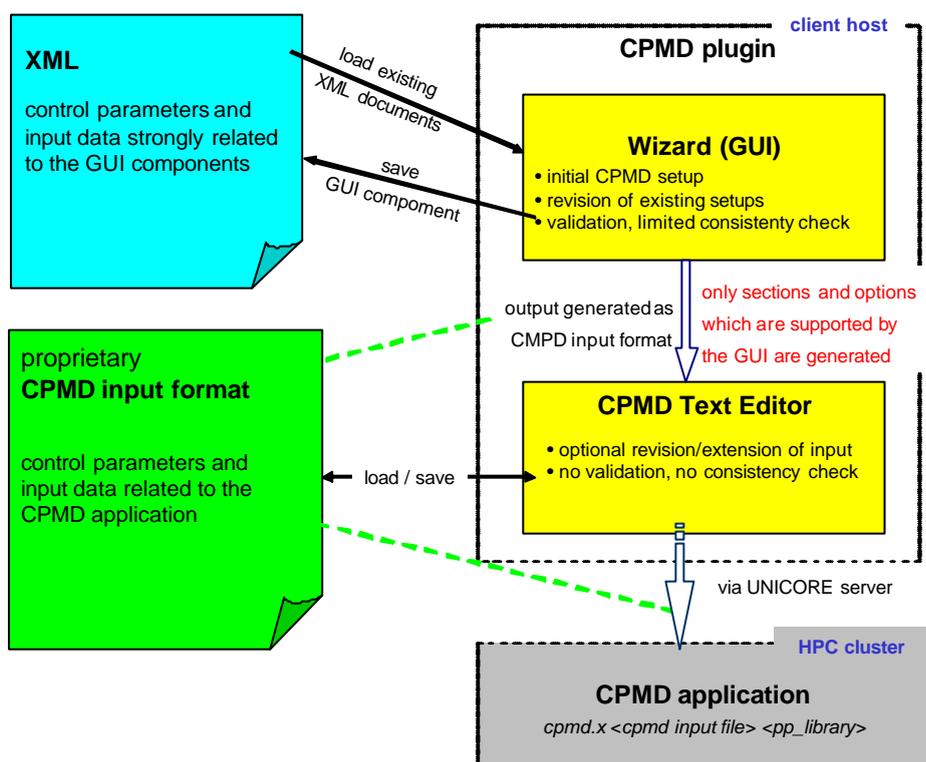
The CPMD plugin version 2.5, mainly a complex graphical user interface (Wizard) combined with a text editor which allows the revision of CPMD control parameter and data items textually, facilitates the configuration of CPMD jobs significantly. Options can be selected, context specific input fields can be edited and a tooltip informs about the meaning of every GUI element. The users input is validated immediately and consistency checks are performed up to a limited extent before the CPMD input is generated and delivered to the text editor.

The workflow of the CPMD plugin version 2.5 is schematically depicted by figure 6. Since the configuration of CPMD tasks often means that hundreds of parameters and data items must be defined, the Wizard possesses the capability to load pre-configured CPMD parameter and data files represented in XML. As far as available these XML setup files can be used as the basis for the work with the wizard. The content of these XML files is closely related to the GUI components, i.e. many of the JAVA Swing classes such as *JCheckBox* or *JRadioButton* which are used by the GUI, are itemized as well.

Since the corresponding XML file scheme is not widely used, a CPMD user must create his own set of XML files by using the Wizard interactively in order to accomplish a sufficient basis for an efficient work. It is not possible to reuse and load the widely-used proprietary CPMD input files into the wizard (the green sheet on fig. 6), and also not to load the textual edited CPMD input from the text editor to the wizard. Although the task-building is facilitated by means of the CPMD plugin, its acceptance by the CPMD user community could be much more widespread if it allows re-use of the widely-used CPMD input files.

The model and view components are not adequately separated and therefore the portability of the software into a portal environment is rather limited.

To summarize, the last major release of the CPMD plugin in June 2002 provides significant facilitation for the setup of CPMD jobs in a GRID environment. But new developments of the CPMD, e.g. the TDDFT section (CPMD version 3.9.1), new requirements such as the separation of concerns and the request by the user community to re-use widely-used CPMD input files as a basis for the CPMD wizard all require some revisions - not only of the implementation but also of the design. In any case, the design of the existing complex CPMD wizard is extremely useful, even if the view components will have to be implemented in a web application.



**Figure 6**

Schematic view of the workflow with the present CPMD plugin version 2.5

### 3.2 Enhancements of the CPMD plugin

The JRA1 enhancements of the CPMD plugin address and realize the following important features after feedback from CPMD developers and users:

- ? Implementation of a backfill functionality based on CPMD input files (mostly implemented)

- ? Separation of model and view components
- ? JAVA implementation of consistency checks based on the routines provided by the CPMD code.

Our main concern was to study the portability of the CPMD plugin and to investigate how important design elements of the plugin could be reused in a portal environment. Therefore the adaption of the CPMD plugin to new features (sections) of CPMD version 3.9.1 has not been carried out.

As mentioned above, our work based on the source code of the CPMD plugin version 2.5 that has been developed at the Research Center Jülich. We also referred to the FORTRAN source code of the CPMD 3.9.1 to figure out the relevant variables and initial values for the model component. The consistency check routines had also been derived from the Fortran code.

New functionality of the CPMD plugin is indicated by red arrows and text elements in figure 2, which describes the new workflow schematically: The CPMD input files can now be imported by the CPMD wizard. Consistency checks based on the original code of CPMD 3.9.1 can now be performed independently from the view components either triggered by the CPMD wizard or even after the CPMD input has been revised in the CPMD text editor. This feature is important in order to avoid the situation where a supplementary revision of the CPMD input generated by the CPMD wizard may introduce new syntactical errors that would not be detected before runtime of the CPMD job on the HPC machine.

The separation of concerns has been realized by the enhanced plugin by the introduction of a CPMD parameter object (the model component) which contains the CPMD control parameter and input data. This parameter object corresponds to the information contained in the CPMD input file. The binding of the GUI elements (i.e. check boxes, data fields) with the numerous attributes of this parameter object (more than 300 attributes) was one of the more extensive parts of the work.

Note that the CPMD input which is generated by the CPMD wizard only includes those sections and options currently supported by the wizard. Additional sections, such as the TDDFT section for instance, can not be generated by the wizard. Therefore, information related to these sections is not contained in the XML files which are exported by the CPMD wizard. However, the enhanced plugin allows sections (and options) which are not supported by the wizard to be bypassed directly to the text editor so that the complete CPMD input according to CPMD version 3.9.1 can finally be checked and then submitted to the target application via a UNICORE NJS server.

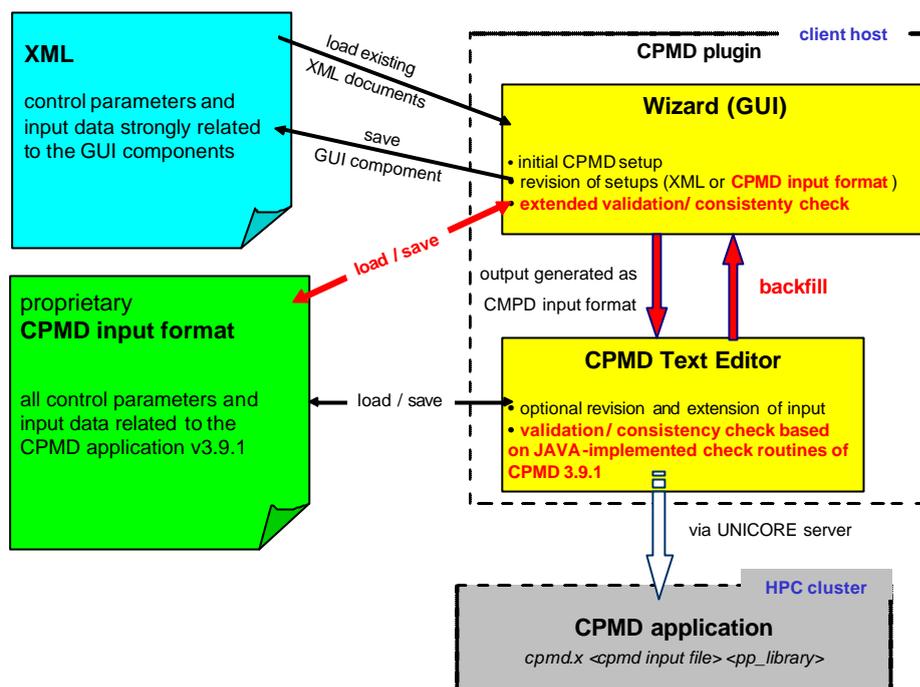


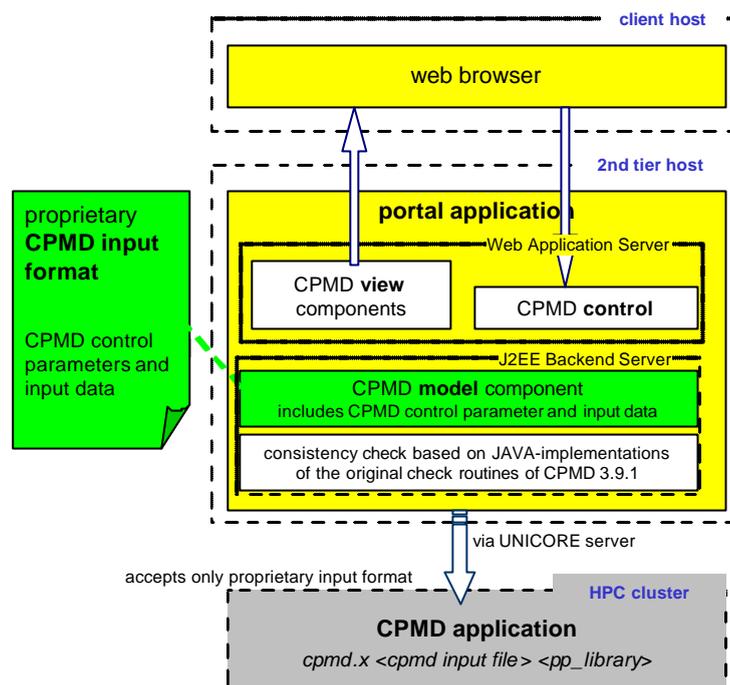
Figure 7

Schematic view of the workflow of the enhanced CPMD plugin. New features are indicated in red.

### 3.3 CPMD plugin portability

Since the CPMD plugin version 2.5 has been closely designed according to the plugin framework provided with the UNICORE client, the portability to a DEISA portal as discussed in the previous chapter is rather limited. A portal requires a design according to the Model-View-Control (MVC) paradigm, particularly a strict separation between view and model components (separation of concerns). Regarding the current design of the CPMD plugin, the view components (the CPMD wizard) also contain all model information (the CPMD control parameter and data items). In addition only the GUI components provide the logic that is needed for a validation and consistency check of the CPMD input. According to the MVC paradigm, the model components must provide the logic for consistency checks. A portal would require that the model components operate separately from the view components, for instance in a J2EE backend container.

In order to facilitate an integration of the new enhanced CPMD plugin for future deployment in the newly designed general portal, changes to the original design were unavoidable. Following the separation of concerns paradigm the CPMD model component has been extracted to be conformant with the Model-View-Control pattern, as illustrated in fig. 8.



**Figure 8**

Separation of the CPMD model component according to the Model-View-Control paradigm in order to support a future deployment in the new general portal.

## 4. Further relevant materials science applications

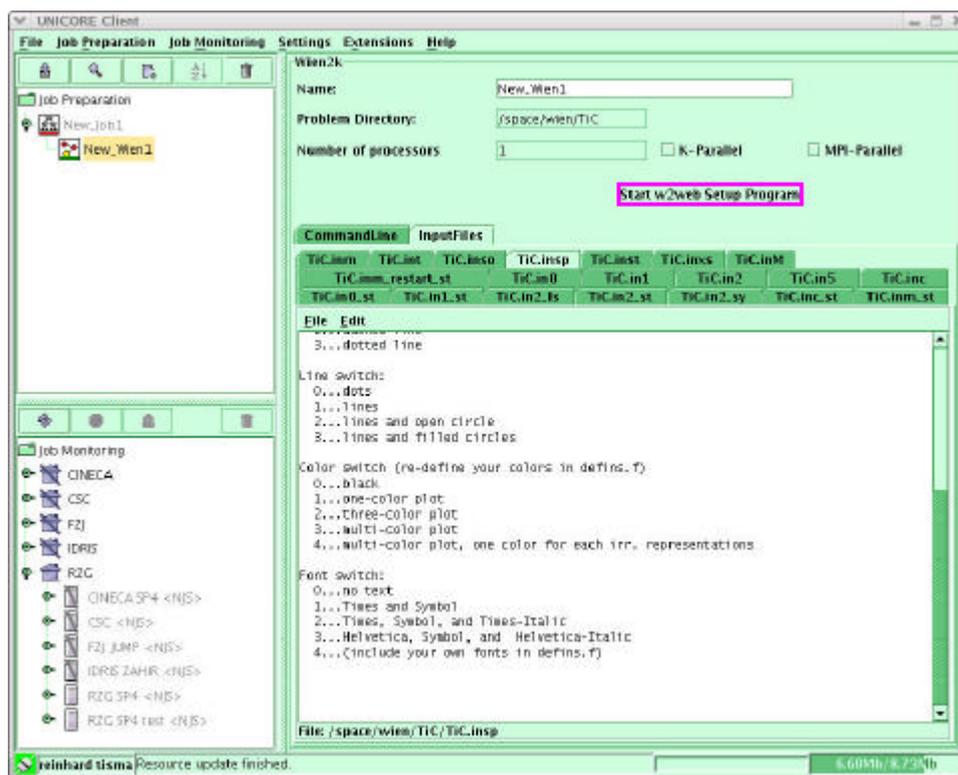
In addition to work on CPMD, further important materials science simulation codes to be supported within DEISA have been selected. Work in progress is described in the following sub-chapters.

### 4.1 WIEN2K

Work on the WIEN2K simulation code ([www.wien2k.at](http://www.wien2k.at)) and especially the corresponding plugin for UNICORE has been continued. Different options were considered. A graphical user interface to WIEN2k (called w2web) has been provided by the (WIEN2K) authors. Detailed inspection of the tool convinced us to integrate it into the planned DEISA portal/plugin.

W2web is a perl script based web server which administrates complete WIEN2K sessions. It serves to prepare and set up the necessary input files and files for the specification of run time parameters.

The application is started from within the session which also provides the framework for post processing. This w2web tool has been modified and adapted for use within the UNICORE client. A WIEN2K plugin was developed by this JRA which allows for two different usage options. As a first option the plugin was designed to be able to edit existing input parameter files (e.g. from an earlier run).



**Figure 9**  
WIEN2K plugin for the UNICORE client

As a second option one can choose to directly use the w2web tool. The corresponding process is then started via a newly created interface between plugin and w2web server. All parameter files can be configured according to the original design.

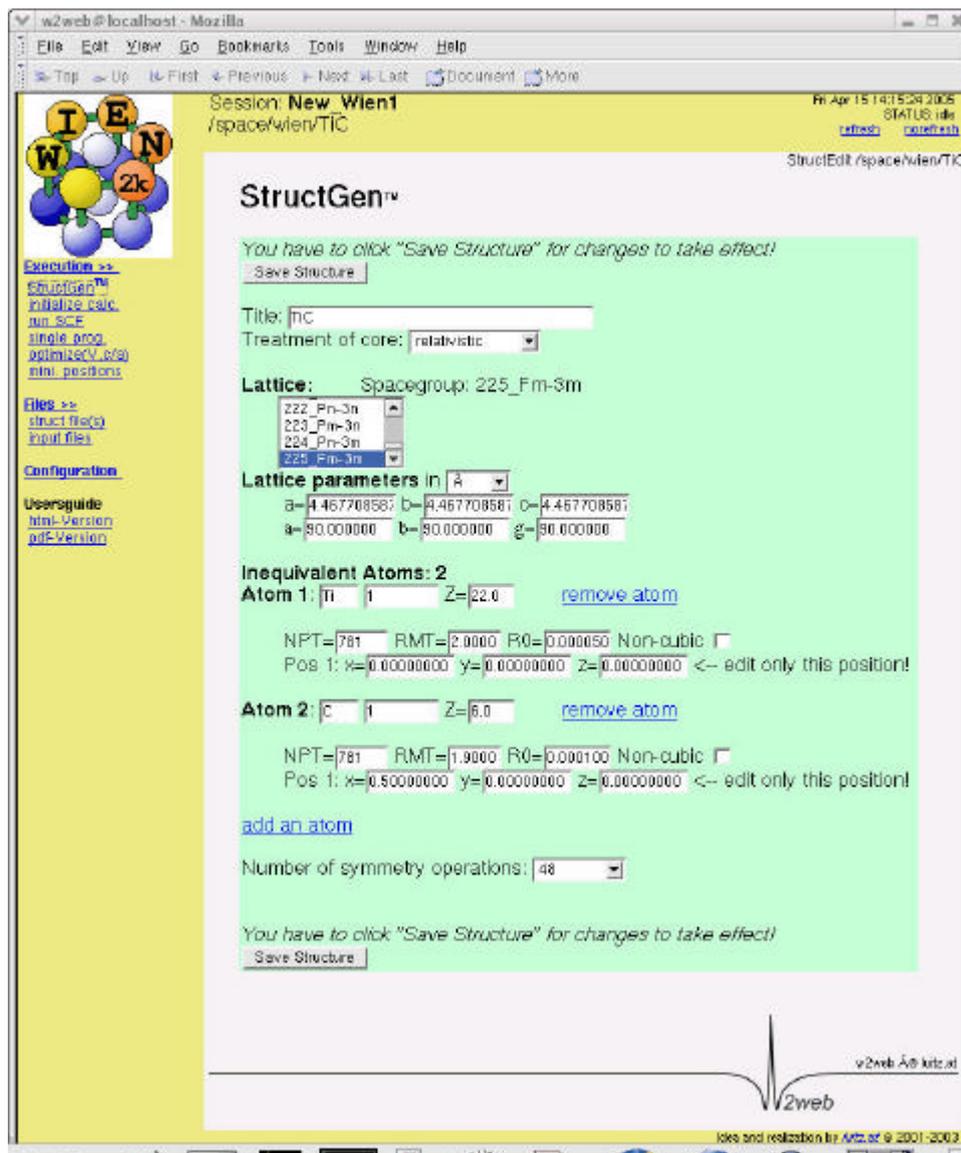
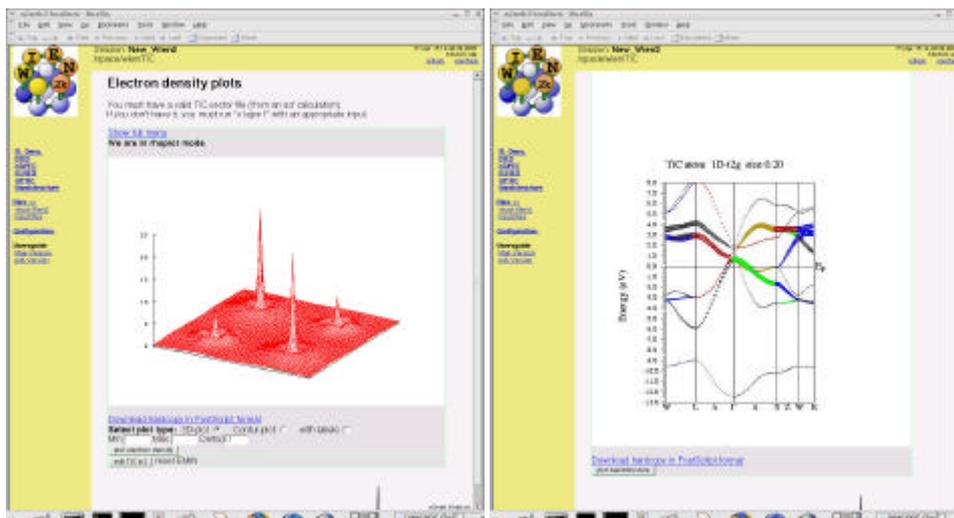


Figure 10

W2Web page started via UNICORE-plugin

The complete command for WIEN2K execution is transferred back to the UNICORE session.

After execution termination, results can be inspected with post-processing routines provided by w2web which can now be invoked via the new UNICORE plugin.



**Figure 11**

Visualization of different aspects of WIEN2K simulation results

## 4.2 ESPRESSO

ESPResSo stands for **Extensible Simulation Package for Research on Soft matter** and is a newly written program package which has been developed at the Max Planck Institute for Polymer Research in Mainz, Germany (see [www.espresso.mpg.de](http://www.espresso.mpg.de)). The package has been designed to perform numerical MD/MC simulations for a broad class of soft matter systems in a parallel computing environment. The main concept in developing ESPResSo was to provide an easy to use simulation tool which serves at the same time as a research platform capable of rapidly incorporating the latest algorithmic developments in the field of soft matter sciences. The strength of the present version lies in its efficient treatment of long range interactions in various geometries in a parallel computing environment. The source code relies on simple ANSI-C, is Tcl-script driven, and possesses easily modifiable interfaces, for example for real-time visualizations, or a graphical interface. The distribution of the source code adheres to the open source standards.

The ESPResSO package has been selected for exploration of the technical options and solutions for computational steering in the context of JRA1 (materials science) within DEISA, and discussions with the authors have started.

## 4.3 NWCHEM

NWChem (see [www.emsl.pnl.gov/docs/nwchem](http://www.emsl.pnl.gov/docs/nwchem)) is a computational chemistry package that is designed to run on high-performance parallel supercomputers as well as conventional workstation clusters. It aims to be scalable both in its ability to treat large problems efficiently, and in its usage of available parallel computing resources. NWChem has been developed by the Molecular Sciences Software group of the

Environmental Molecular Sciences Laboratory (EMSL) at the Pacific Northwest National Laboratory (PNNL). Most of the implementation has been funded by the EMSL Construction Project.

The charge-free academic license for the newest version NWCHEM 4.6 has been applied for and received for RZG. The executable has been installed and is ready for usage at RZG. A module for the DEISA CPE will be provided soon. For NWCHEM usage at other DEISA sites, separate licenses will have to be applied for.

#### **4.4 CP2K**

CP2K is a freely available (GPL) program (see [cp2k.berlios.de](http://cp2k.berlios.de)), written in Fortran 95, to perform atomistic and molecular simulations of solid state, liquid, molecular and biological systems. It provides a general framework for different methods such as density functional theory (DFT) using a mixed Gaussian and plane waves approach (GPW), and classical pair and many-body potentials.

CP2K provides state-of-the-art methods for efficient and accurate atomistic simulations, sources are freely available and actively improved.

A work plan for providing the CP2K package and especially the new QUICKSTEP code also via portal has been initiated in collaboration with a respective author from the Parrinello group, ETHZ.

### **5. Scientific Community and DEISA Extreme Computing Initiative**

In order to maintain the relevance of this JRA for the computational scientists of the European materials science community, Prof. Parrinello from ETHZ has been visited in late 2004 to also discuss guidelines for the DEISA Extreme Computing Initiative which was already in the planning stage. Though CPMD is expected to remain the ab-initio molecular dynamics workhorse for still quite some years, the new MD simulation code package CP2K has now been recommended for integration into DEISA, especially with its new component QUICKSTEP which will be implemented for DEISA in close collaboration with the Parrinello group.

Further groups have also already been contacted and consulted with respect to the Extreme Computing Initiative, and efforts have been undertaken to help structuring and setting up the interface to the scientific community.