

Simplified Grid Implementation of Medical Image Processing Algorithms using a Workflow Management System

Dagmar Krefting, Michal Vossberg and Thomas Tolxdorff

Institute of Medical Informatics, Charité - Universitätsmedizin Berlin, Germany, dagmar.krefting@charite.de

Abstract

Complex analysis of large amounts of medical image data quickly exceeds storage capacity and computing power of single workstations or small local networks. When limited hardware resources impede full utilization of medical image analysis, a possible solution is the usage of computing grids, the collaboration of distributed resources across institutional borders. Many existing image processing problems would benefit from parallel processing, e.g. on single image or volume slice level. Such coarse-grained parallelization can easily be achieved by implementation into a grid infrastructure. Furthermore, grids allow distributed users to share their code, promoting collaborative projects. In this paper, we describe the grid implementation of existing code using grid workflows. The workflow management system is able to execute all tasks related to grid communication, such as authorization, scheduling and monitoring. It remains to the developer to make the code accessible for the workflow manager, and to define, where the code is found on the grid and what to do with it. We describe the procedure how to bring the code to the grid and show exemplarily the implementation of segmentation and registration algorithms for transrectal ultrasound guided prostate biopsies.

1 Introduction

Many scenarios in medical image processing demand high computing power and storage capacity. Increasing usage of high resolution images and multidimensional data, like volume sequences or multi-modality data, amplify hardware requirements. When results are required within a certain time, compromises between accuracy and computing time are unavoidable on limited resources. Furthermore, new algorithms developed by research groups are often hardly available or adaptable for related research problems. A promising solution to overcome these barriers are modern grid networks. A grid infrastructure is defined as a collaboration of possibly inhomogeneous, distributed hardware resources across administrative borders [4]. The network is virtualized as a single resource providing scalable hardware resources and a variety of applications. The management of the network is realized by the middleware, an abstraction software layer between the grid network and the applications.

Many medical image analysis tools process consecutively image series, volume slices or tiles, or loop over a range of input parameters, e.g. multiscale analysis. The computing time of such processing steps scales with $1/n$ (n = number of frames, slices, or tiles), when coarse-grained parallelized and implemented to a distributed computing system with more than n CPUs. Due to transfer times, when reuniting intermediate results for integrated analysis, the resulting compute time savings are reduced, but in most cases notable. Especially in the development state of image processing algorithms, where large parameter scans are realized, distributed

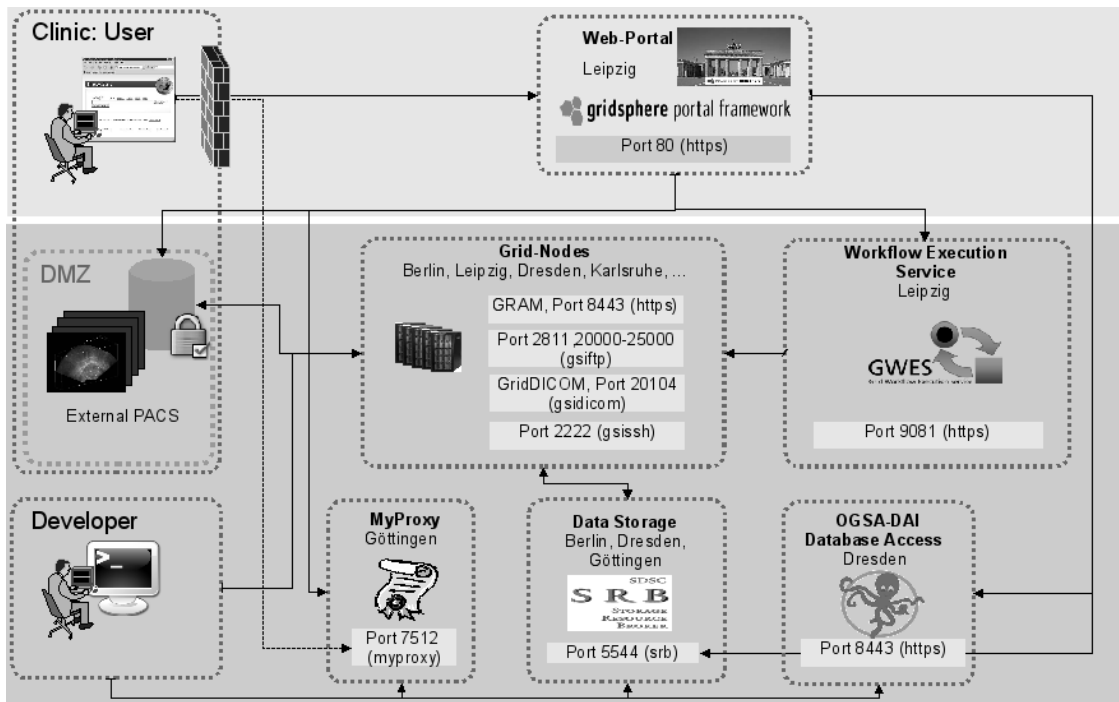


Figure 1: Implemented MediGRID system architecture.

computing may save days of waiting for results. But not only time saving aspects may be relevant when considering to use grids for medical image applications. Access to distributed image databases or distributed software with particular hardware requirements can easily be achieved. If a grid portal - a webbased user interface - is set up, users may start, stop and check grid jobs from every place with internet access. However, if the expected effort for the grid implementation is estimated to outbalance the (near-term) benefit, interested researchers are discouraged to use grid resources. The method proposed in this paper allows the grid implementation without deep knowledge of the underlying grid infrastructure. Quick results can be achieved by basic implementation using XML-based workflow descriptions. The implementation can later be enhanced regarding user friendliness by integration into a grid portal.

2 Methods

2.1 Grid architecture and middleware solutions

The implementation is realized within the German MediGRID infrastructure, which is part of D-Grid[15, 8]. D-Grid offers generic middleware components for a variety of user communities, from high energy physics to linguistics. MediGRID uses basic services from D-Grid, e.g. user management, but has extended the hardware and software to satisfy community specific requirements. It is a *Globus* (GTK4) based grid infrastructure, which is the de facto standard for modern grid architectures[7]. It provides built-in security mechanisms, data transfer (gridftp) and generic job submission (GRAM). A grid portal, based on *GridSphere*, allows for easy access from different sites even behind strict firewalls[19]. The main extensions above the core infrastructure are the Grid Workflow Execution Service (GWES)[10], the Storage Resource Broker (SRB) for distributed data storage[21], and gridDICOM for medical image transfer[23]. The implemented system architecture of MediGRID is given in Fig. 1.

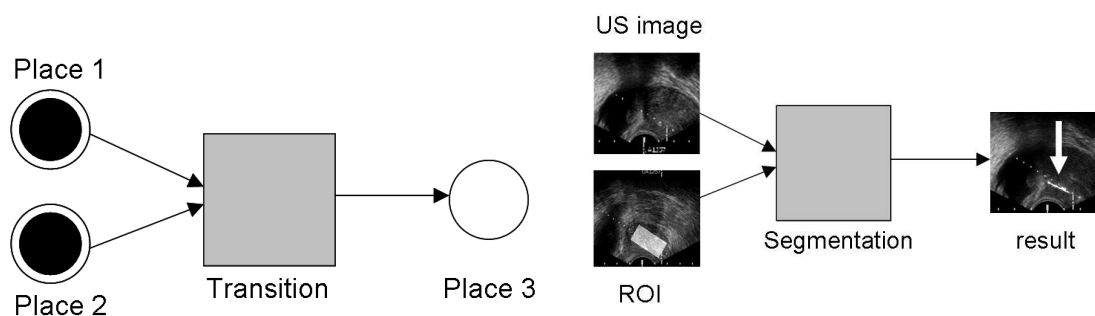


Figure 2: A simple petri net (left), modelling the execution with two inputs and one output, e.g. biopsy needle segmentation (right).

2.2 GWES

GWES is a workflow manager established within the K-WF-grid[5]. In contrast to other scientific workflow systems, such as Condor's DAGMan tool[6] or LONI[22], it is specifically targeted for WSRF-standardized grid usage and allows to create complex workflows without detailed knowledge of the grid infrastructure. The core of the GWES is the Grid Workflow Description Language (GWorkflowDL), which is an XML based standard for describing workflows as a Petri net. A Petri net is a mathematical formalism to describe discrete distributed systems and allows to model the workflow with a few basic graph elements. As such, Petri nets are often easier to use and more intuitive than other workflow languages, such as the widely known BPEL (Business Process Execution Language)[2], which has the disadvantage of possessing complex and rather informal semantics and an extensive syntax. Other graph-based languages mostly base on directed acyclic graphs (DAGs) and offer only a limited expressiveness so that it is often hard to describe complex workflows (e.g., loops cannot be expressed directly). For the actual workflow descriptions, GWES uses an extension, high level Petri nets (HLPN). They can be directly used to model transfer, execution and storage of any kind of input and output data as well as control data (e.g. the exit status of a workflow step). Data is modeled as *tokens*, located at a *place*, and program execution is represented as a *transition*. In contrast to control-centric languages, like BPEL, GWES is data driven and the Petri net representation corresponds to the pure data flow. Additionally, the flow can be controlled through user constraints to the processor, which is a benefit over other data-flow languages like Scuff/XScuff[13].

The basic workflow description provides the definition of transitions and their input- and output places. Figure 2 gives a simple example of a Petri net. It consists of one transition, represented by a square, two input places (Place 1 and Place 2) and an output place (Place 3). Places are represented by open circles. Directed arcs connect places and transition and define the flow relation. Tokens are represented as black dots. This Petri net might - for example - model the segmentation of a biopsy needle in an ultrasound image. Then Place 1 models the US image, Place 2 models a predefined region of interest, the transition models the execution of the segmentation algorithm and Place 3 models the result. At the current initial state of the processing, tokens are located at the input places. Within runtime of the workflow, tokens on input places are consumed and sent to output places. Highlevel Petri nets can do anything that can be defined in terms of an algorithm [12]. We found them perfect to map the image pipelines of our applications. GWES descriptions can be realized on several abstraction levels, which are then concretized during runtime. It provides basic resource brokering and scheduling. The information about available hard- and software is provided in the XML-based D-Grid Resource Description Language (D-GRDL), and is stored in the resource database[1]. GWES also offers fault-tolerance strategies for reliable process execution. If an execution step fails, the error is reported and the transition is rescheduled to another resource up to an adjustable number of retries.

The generic GWES web interface (GWUI) allows to upload of workflow descriptions and monitoring of running workflows. An automated visualisation of the workflow layout and the live status are also offered. This is particularly helpful in the development and testing phase, as workflow layout and implementation errors immediatly become visible.

3 From command line program to a grid application

In this section we describe the different steps that have to be processed by the developer to bring an existing image processing program onto the grid. We will focus here on automated image processing steps to demonstrate the basic implementation method. Moderate user interaction like workflow suspension and restarting is provided by the GWES webinterface. Further usability can be realized completely webbased by integration of the workflow into the grid portal. The grid implementation encompasses the following steps:

1. Deployment of the software to the gridnodes
2. Generation of a wrapper script
3. Registration of the software
4. Creation of a workflow description
5. Optional: Integration of the workflow into the user portal

3.1 Deployment of the software

The easiest way to make software available on the grid is to deploy it as appropriately compiled code. All software related to the specific application is stored at a separate directory on each frontend of the connected clusters in the used infrastructure. The frontends are accessible via *gsissh*, a grid enabled version of ssh. Developers can log on to these machines and manage their application directories. As of today, there is no effective automatic deployment of the software. We use a version manager (subversion) to ease the update process of the grid nodes[3].

3.2 Writing GWES wrapper scripts for the software

During workflow execution, the GWES workflow manager calls the programs designated by the transitions of the workflow's Petri net. In the call, the name of the transition denotes the executable and the connected input tokens specify the input parameters. As the tokens in the net adhere in no particular order, GWES passes the parameters as key/value pairs and uses the edge-expression of the token's arc as the key (i.e. *program-name -edge_expr1 token1 -edge_expr2 token2 ...*). Since most existing programs do not conform to this plugin convention, it proved good practice for the developer to provide a small wrapper script which is called instead of the actual program. In the wrapper script the parameter values are extracted from the key/value pairs and mapped to the application specific program call. This also adds an additional layer of abstraction, as the script can be changed more easily than the workflow or may contain auxiliary commands.

3.3 Registration to the resource database

To publish the new software to the grid, a resource description file is generated and stored in the database. The resource description contains the internal GRDL name of the software and the path of the wrapperscript

```
<resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.gridworkflow.org/kwfguid/src/xsd/resource-d-grdl.xsd"
  uri="software:medigrid-wrapperscript">
<ofClass uri="urn:dgrdl:software:medigrid-wrapperscript"/>
  <name>wrapperscript</name>
  <description>Medical image application: Example of a wrapper script</description>
  <simpleProperty ident="executable" type="string" unit="">
    /wpath/wrapperscript.sh
  </simpleProperty>
</resource>
```

Figure 3: Example of a GRDL resource description.

on the gridnode. An example for the script *wrapperscript.sh*, located at */wpath* is given in fig. 3. An XML-element, containing the URI of the software, is added to the hardware resource description files to let the system know where the software is available.

3.4 Creation of a workflow description

The workflow description is also an XML document, using GWorkflowDL. This part of the implementation might need some familiarization for user who are inexperienced with XML. The workflow definition consists of the following parts

1. Header: provides properties and execution information for the GWES.
2. Place elements: defines places and initial assignment with tokens.
3. Transition elements: defines the software to be executed and its input and output places.

An example can be found in MediGRID project's deployment guide[11].

3.5 Visualization and execution of a workflow

The workflow description can now be uploaded to GWES using the web interface, where the underlying Petri net is visualized. If the resulting layout is as expected, the workflow can be started. The actual state of the places (occupying tokens) and transitions (available and chosen hardware resource, state) as well as possible error messages or warnings are monitored. The upload of the workflow description is the default for development and testing. The disadvantage is, that the initial tokens have to be defined directly in the workflow description. This is fine for quick implementation, if the programs are used by the developer. A more userfriendly method is to integrate the generation of the workflow into a graphical user interface.

3.6 Using workflow templates

Graphical user interfaces are implemented mainly as grid portlets within the used grid infrastructure[14]. When integrating the workflow into such a GUI, users can upload data or select them from available data storage (PACS and SRB). Additional parameters can be set or modified. The desired image processing workflow can be started at the push of a button. A workflow template stored within the portal is then automatically complemented and transferred to the GWES. This is the most convenient solution for the end user, but requires of course more development time for writing the grid portlets[20].

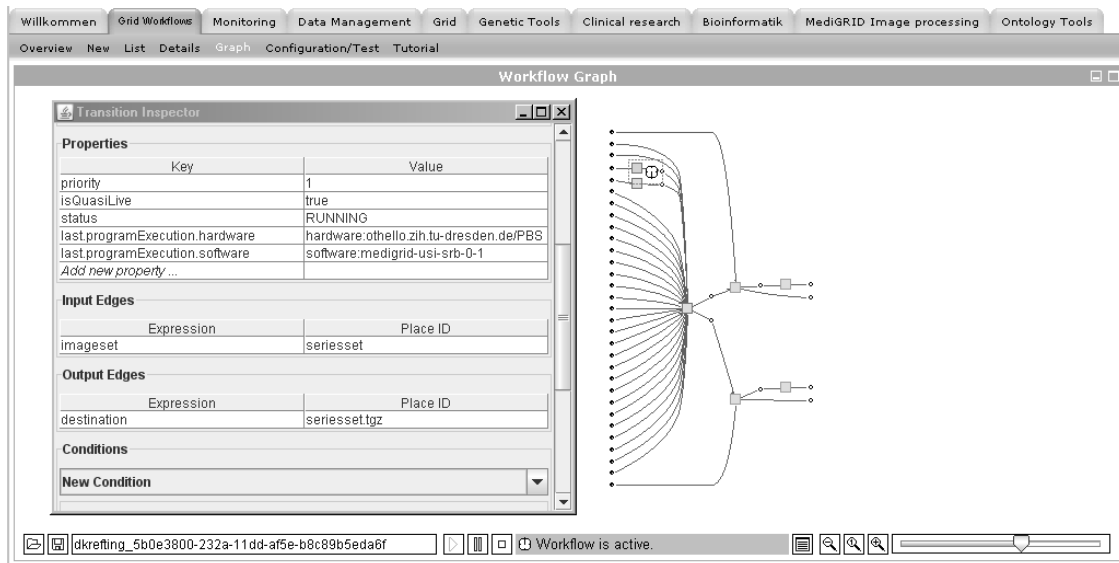


Figure 4: Single registration workflow with status control.

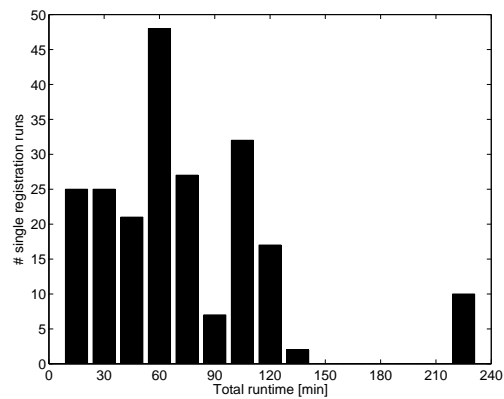


Figure 5: Runtime distribution of 214 single registration workflows.

4 Implementation of image processing algorithms

To date, the described method was used to successfully implement completely different applications to the grid, as parallelized preprocessing of functional MRI data, hemodynamic simulations, gene prediction or biosignal analysis of polysomnographies. Here, we present the implementation of algorithms to be used for analysis of transrectal ultrasound images taken during prostate biopsies. Prostate cancer is the most common cancer in men. Current goldstandard for prostate cancer diagnosis is ultrasound guided prostate biopsy[9]. Monitoring the prostate by transrectal ultrasound (TRUS), tissue probes are taken from different parts of the prostate. The present application determines and visualizes the position of the tissue probes within the prostate volume. The localization is done by automated segmentation of the biopsy needle in the guiding 2D ultrasound images and subsequent registration of the 2D images into a previously taken 3D ultrasound volume[16]. The algorithms are developed in *Matlab* and *c/c++* using *ITK*, *VTK* and *MITK*[17, 18]. While segmentation and registration are mainly independent, the respective workflows are modeled separately. 2D-3D image registration of the transrectal ultrasound images turned out to be a difficult task, as many local minima exist in the used costfunctions (normalized correlation and mutual information). Furthermore,

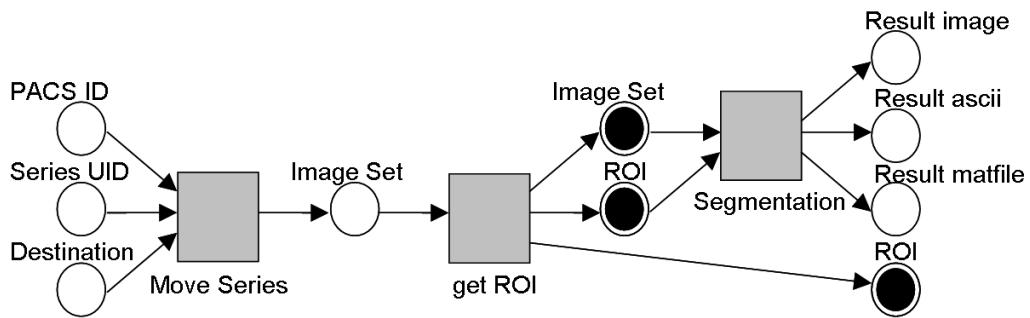


Figure 6: Petri net of the segmentation workflow. Image transfer and ROI calculation are already finished, the segmentation will be executed next.

the optimization algorithms implemented in *ITK* show poor performance in optimization of rotation angles. Therefore a parameter scan of up to 600 single registration runs with different initial rotation angle values is performed. A single registration run for maximum allowed 1000 iteration steps needs in average 20 minutes. As an example, our workstation (AMD 64 X2 Dual Core Processor 4200+, 2Gb RAM) requires a processing time of about 4 days to complete the scan. The single registration run on the grid can be divided in the following processing steps:

1. Volume and 2D image transfer from SRB.
2. Registration execution.
3. Storage of the results in SRB.
4. Download of the results.

Fig. 4 shows the workflow layout and the status information for the registration workflow provided by the GWES web interface. The many input places of the central transition, which denotes the registration execution, are the variety of additional parameters which are passed to the algorithm, e.g. the costfunction and optimizer used for registration. With grid implementation, we were able to reduce the total runtime down to 12 hrs, which is approximately 8 times faster than using the workstation. A detailed look to a scan consisting of 214 initial parameter sets gives more insight into the performance. Fig. 5 shows the distribution of total runtimes of the registration runs. Total runtime includes the waiting time before scheduled, the queue time on the selected gridnode and the CPU time. The total runtime reaches from 20 minutes up to 220 minutes (3.7 hrs), which determines the total runtime of the complete scan. The difference between the "fastest" and the "slowest" single registration run is mainly waiting time due to high load on the gridnodes, which can be accessed by all users of the D-Grid and is extensively used for jobs running approximately 12 hrs. The average load on the sites used, encompassing 580 cpus, was between 80% and 100% during the runtime of the workflows. The completion time was more than 9 times faster than the approximate runtime of 35 hrs on our workstation, but still a factor of 10 of the "optimum" of 20 minutes, which would be the pure cpu-time with vanishing queue times. Current expansion of the grid resources and implementation of advanced local scheduling algorithms are let us expect further reduction of the overall runtime. At the current state of the infrastructure, failure rates of up to 5% are experienced. The relatively high number of failures is mainly due to the fact, that the infrastructure itself is still under development and middleware components are restarted frequently.

The needle segmentation consists of the following steps:

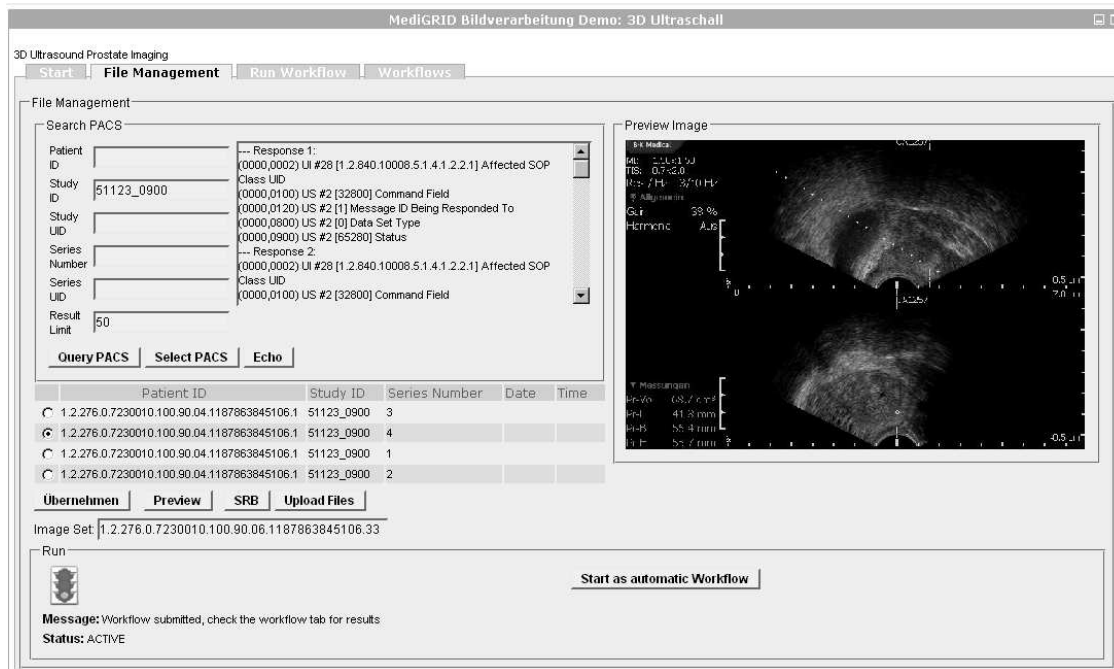


Figure 7: Screenshot of the segmentation portlet. Input image sequence can be selected from the PACS, the SRB, or can be uploaded. After image selection, the workflow can be initiated.

1. Image retrieval from a PACS.
2. Calculation of the Region of Interest (ROI) on the first image of the series.
3. Segmentation of the image series.

The workflow layout is given in Fig. 6. The code was written in *Matlab* and was compiled using *Matlab's Compiler Toolbox*. The time saving of grid usage is here on image sequence level. Each biopsy procedure encompasses 10 individual tissue probes at different prostate locations which are then sent to different grid nodes, depending to the actual load. Since the segmentation itself only needs a few minutes and transfer load would be quite high (each single image needs the ROI and the statistical model), parallelization on image level was abandoned in the segmentation. An application specific portlet was developed. The image series can now be selected interactively and the results are visualized on the grid portal. A screenshot of the portlet is shown in Fig.7.

5 Discussion

The use of the workflow manager makes it easy to integrate existing code without deeper knowledge of grid computing and the underlying system architecture. Automated medical image processing problems with high coarse grained parallelization potential (parameter scans, image sequences) can profit from grid use with low implementation offset. Basic implementation of new algorithms can be performed from developers even with little experience in less than a day. If the applications are used repeatedly or shall be available to end users, integration of workflow templates into application specific grid portlets is strongly recommended. A portlet with basic functionality may be developed within 2 days, but advanced user interaction may need development time up to a few weeks. The additional effort is rewarded by ease of use and full application control from all sites with internet access, even from institutions with strict firewall settings,

e.g. clinical environments. The presented grid infrastructure is mainly used for research purposes. Besides implementation of further applications, current developments focus on security, reliability and usability to allow grid use for clinical applications.

6 Acknowledgements

The work is funded by the German Federal Ministry of Education and Research (BMBF), grant number 01AK803F.

References

- [1] Martin Alt, Andreas Hoheisel, Hans-Werner Pohl, and Sergei Gorlatch. A grid workflow language using high-level petri nets. In *Proceedings of the 6th international conference on Parallel processing and applied mathematics. PPAM 2005. Poznan*, pages 715–722, 2005.
- [2] T. Andrews, F Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services (BPEL4WS). Specification Version 1.1. Technical report, Microsoft, BEA, and IBM, 2003.
- [3] B. W. Fitzpatrick B. Collins-Sussman and C. M. Pilato, editors. *Version Control with Subversion*. O'Reilly, 2004. <http://svnbook.red-bean.com/>.
- [4] V. Breton, A.E. Solomonides, and R.H. McClatchey. A perspective on the Healthgrid initiative. In *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, Illinois, USA*, pages 434–439. IEEE Computer Society, 2004.
- [5] M. Bubak, T. Fahringer, L. Hluchy, A. Hoheisel, J. Kitowski, S. Unger, G. Viano, K. Votis, and K-WfGrid Consortium. K-Wf Grid - Knowledge based Workflow system for Grid Applications. In *Proceedings of the Cracow Grid Workshop 2004*, page 39. Academic Computer Centre CYFRONET AGH, 2005.
- [6] Condor Team. DAGMan: A Directed Acyclic Graph Manager, July 2005. <http://www.cs.wisc.edu/condor/dagman>.
- [7] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2005.
- [8] W. Gentsch. D-grid, an e-science framework for german scientists. In *Proceedings of The Fifth International Symposium on Parallel and Distributed Computing (ISPDC 2006)*, pages 12–13. IEEE Computer Society, 2006.
- [9] A. Heidenreich, G. Aus, C. C. Abbou, M. Bolla, S. Joniau, V. Matveev, H-F. Schmid, and F. Zattoni. Guidelines on prostate cancer, Update March 2007. http://www.uroweb.org/fileadmin/user_upload/Guidelines/Prostate%20Cancer.pdf.
- [10] A. Hoheisel. Grid workflow execution service - dynamic and interactive execution and visualization of distributed workflows. In *Proceedings of the Cracow Grid Workshop 2006*, pages 13–24. Academic Computer Centre CYFRONET AGH, 2007.

- [11] A. Hoheisel, D. Sommerfeld, and K. Peter. *Guidelines for the Deployment of Applications in MediGRID*, December 2007. http://www.medigrid.de/downloads/MediGRID_application_deployment_1-2.pdf.
- [12] Andreas Hoheisel and Martin Alt. Petri nets. In Ian J. Taylor, Dennis Gannon, Ewa Deelman, and Matthew S. Shields, editors, *Workflows for e-Science - Scientific Workflows for Grids*. Springer, 2006.
- [13] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock1, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Res*, 34:729–732, 2006.
- [14] JavaCommunityProcess. JSR-000168 Portlet Specification, 2005. <http://jcp.org/aboutJava/communityprocess/review/jsr168/index.html>.
- [15] D. Krefting, J. Bart, K. Beronov, O. Dzhimova, J. Falkner, M. Hartung, A. Hoheisel, T. A. Knoch, T. Lingner, Y. Mohammed, K. Peter, E. Rahm, U. Sax, D. Sommerfeld, T. Steinke, T. Tolxdorff, M. Vossberg, F. Viezens, and A. Weisbecker. Medigrid: Towards a user friendly secured grid infrastructure. *Future Generation of Computer Systems*, doi:10.1016/j.future.2008.05.005, 2008.
- [16] D. Krefting, B. Haupt, and T. Tolxdorff. Segmentation of prostate biopsy needles in transrectal ultrasound images. In *Proc. SPIE 6512*, pages 6512–105, 2007.
- [17] Mathworks, Inc. MATLAB - The Language of Technical Computing, 2006. <http://www.mathworks.com>.
- [18] NLM. Insight Segmentation and Registration Toolkit (ITK), 2006. <http://www.itk.org>.
- [19] Jason Novotny, Michael Russell, and Oliver Wehrens. Gridsphere: An advanced portal framework. In *Proceedings of the 30th Euromicro Conference*, pages 412–419, 2004. <http://www.gridisphere.org>.
- [20] GridSphere Project. *Grid portlets development guide*. <http://docs.gridisphere.org>.
- [21] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, Sheau-Yen Chen, and R. Olschanowsky. Storage Resource Broker - managing distributed data in a grid. *Computer Society of India Journal*, 33(4):42–54, Oct 2003.
- [22] David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga. The LONI pipeline processing environment. *NeuroImage*, 19:1033–1048, July 2003.
- [23] Michal Vossberg, Thomas Tolxdorff, and Dagmar Krefting. DICOM Image Communication in Globus-Based Medical Grids. *IEEE Trans. Inf. Technol. Biomed.*, 12:145–153, 2008.