

# Use Case: Medical Image Reconstruction

## The GEMSS Project

### Grid-Enabled Medical Simulation Services



Siegfried Benkner, Gerhard Engelbrecht, Rainer Schmidt

Institute for Software Science

University of Vienna

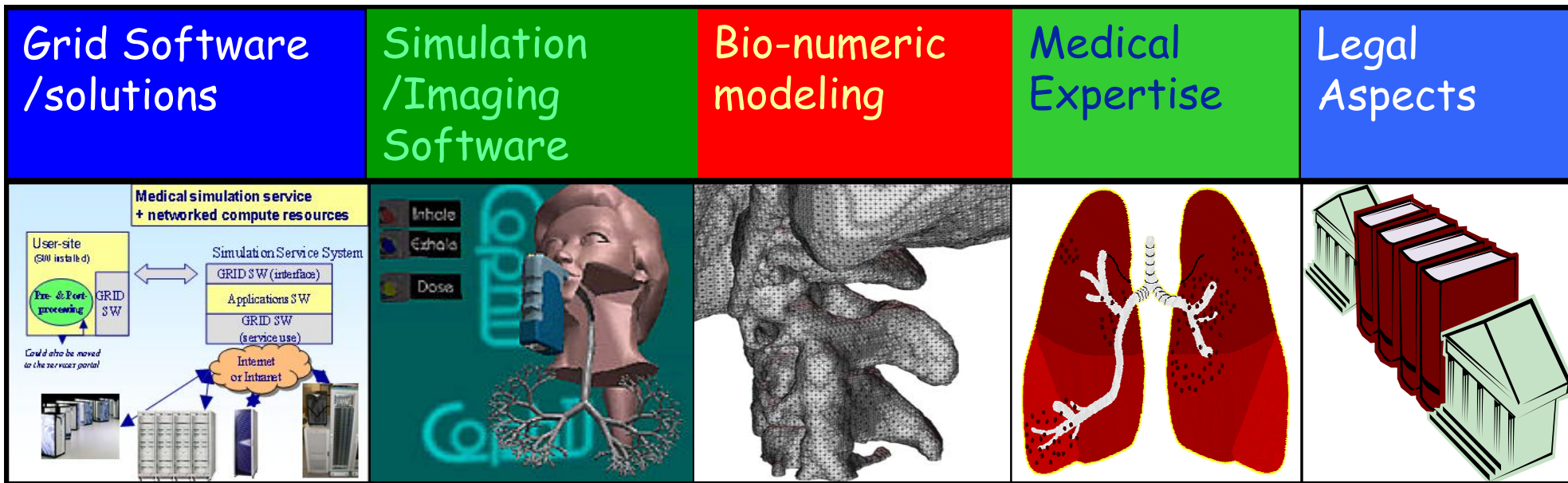


# Outline

---

- *GEMSS Overview*
- *GEMSS High-Level Architecture & Infrastructure*
- *Use Case: Medical Image Reconstruction*
- *Conclusions*

# The GEMSS Project



- ➔ **Secure & lawful** Grid provision of medical simulation services
- ➔ Build 6 Grid-enabled **medical prototype applications**
- ➔ Build suitable **middleware on top of common standards**
- ➔ Install and evaluate a **GEMSS test-bed**
- ➔ Anticipate **privacy, security** and other legal concerns

# GEMSS Applications

Name	Domain	Class
Maxillo-facial surgery simulation	Medicine - pre-surgical planning	On demand / distributed supercomputing
Neurosurgery support	Medicine - intra-operative planning	On demand
Radiotherapy planning	Medicine - Monte Carlo treatment simulation	On demand / distributed supercomputing
Inhaled drug delivery planning	Medicine - air flow dynamics	On demand / distributed supercomputing
Cardio-vascular system simulation	Medicine - blood flow dynamics	On demand
Advanced image reconstruction	Medicine - nuclear / in vivo diagnostics	On demand

# GEMSS Applications - Summary

---

- ❑ **Compute-intensive** numerical methods (parallel MPI codes)
  - Finite Element Method (FEM)
  - Computational Fluid Dynamics (CFD)
  - Monte Carlo Simulation (MC)
  - Iterative Image Reconstruction Method (ML-EM)
- ❑ **Data Transfers** (few MBs to few GBs)
- ❑ **Services** composed of **multiple application components** (workflow)
- ❑ **Flexibility - User Interactions**
- ❑ **Near-realtime requirements** (QoS support essential)

# GEMSS Architecture

---

## □ Service oriented architecture

- Based on **standard Web Services** technology (WSDL, SOAP, WSLA, WS-Security).
- Possible integration with other hosting platforms via OGSA/WSRF

## □ Client-side Framework

- Pluggable component framework.
- **High-Level client-side API** for invoking services in a protocol-independent manner.

## □ Service-Provision Framework

- **Generic Application Services**
- Transformation of HPC application components into Services

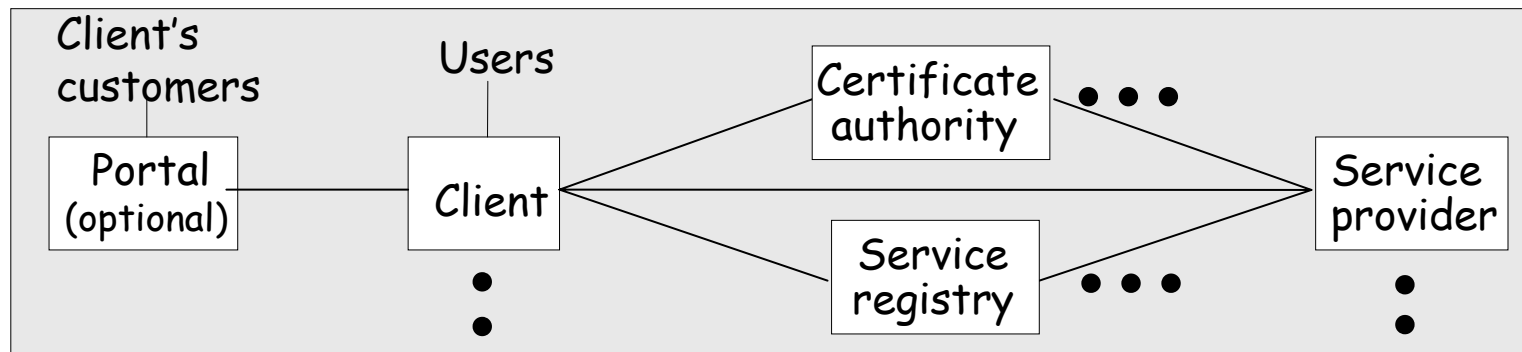
## □ Support of complex process models

- business workflow
- dynamic **QoS negotiation**
- application workflow

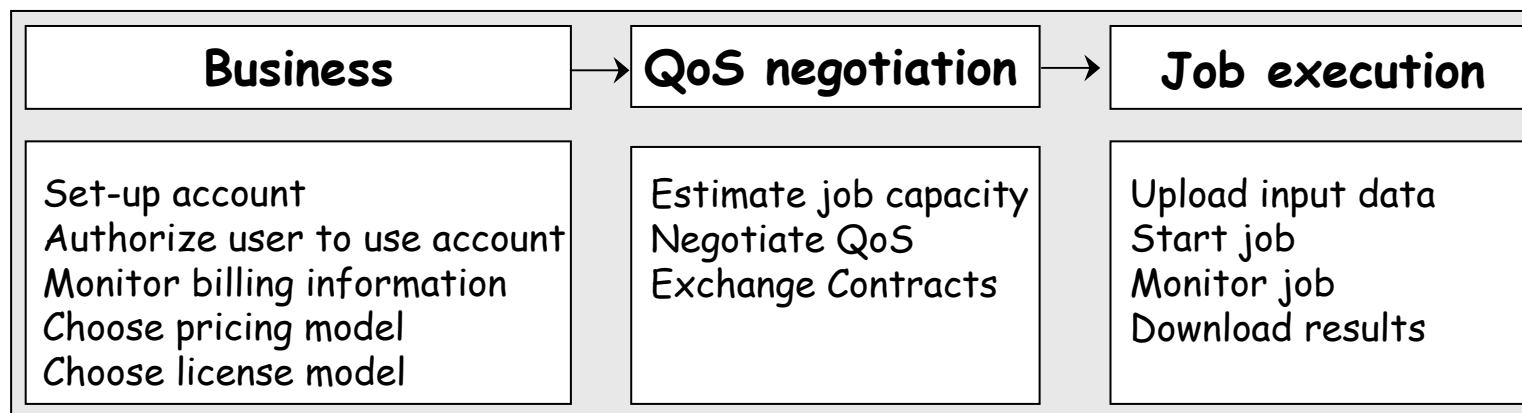


# High-level Grid Architecture

## Service-oriented architecture

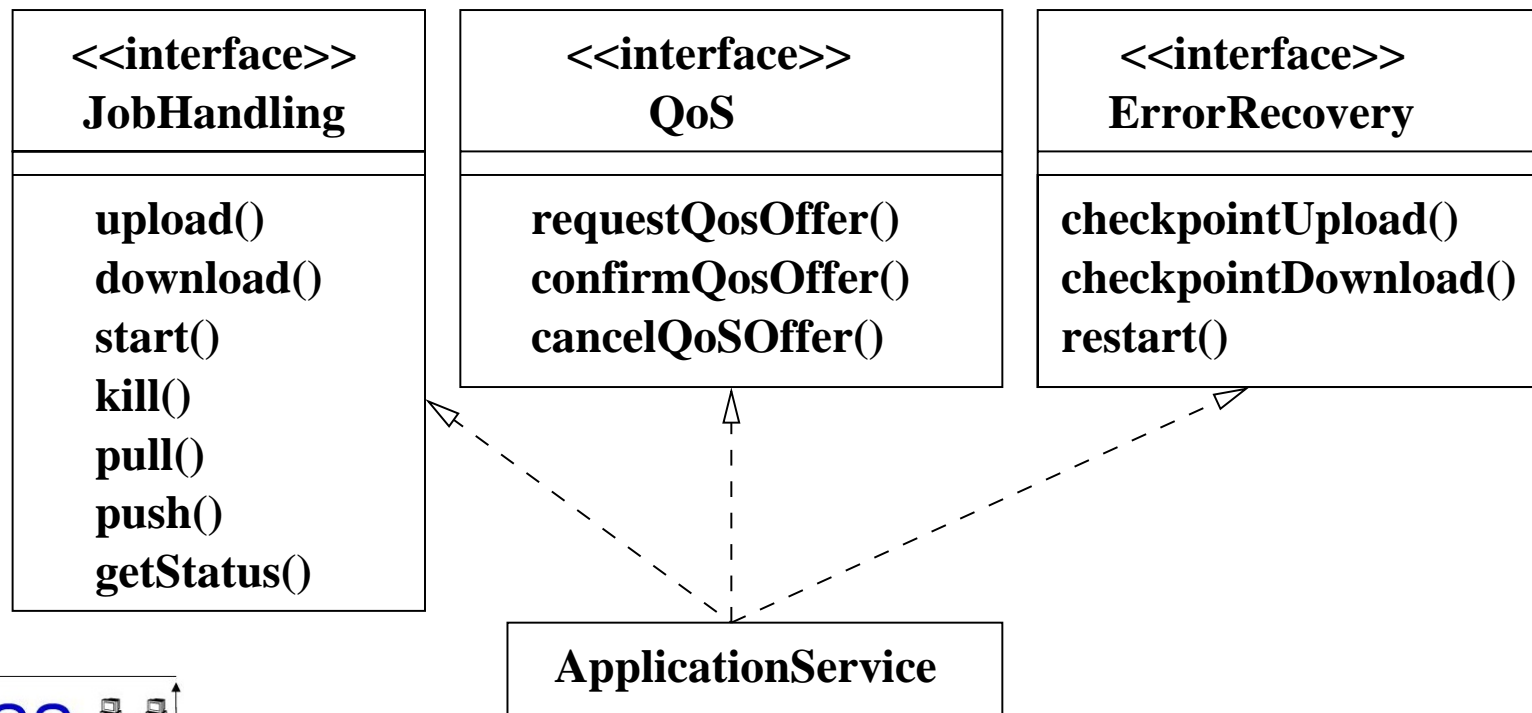


## 3-step process model for job execution



# Generic Application Services

- ❑ common methods for data staging, job management, error recovery, QoS, ...
- ❑ customized for a specific application by means of XML application descriptor
- ❑ accessed by clients through a high-level Client API
- ❑ automatic deployment on different hosts with varying QoS characteristics



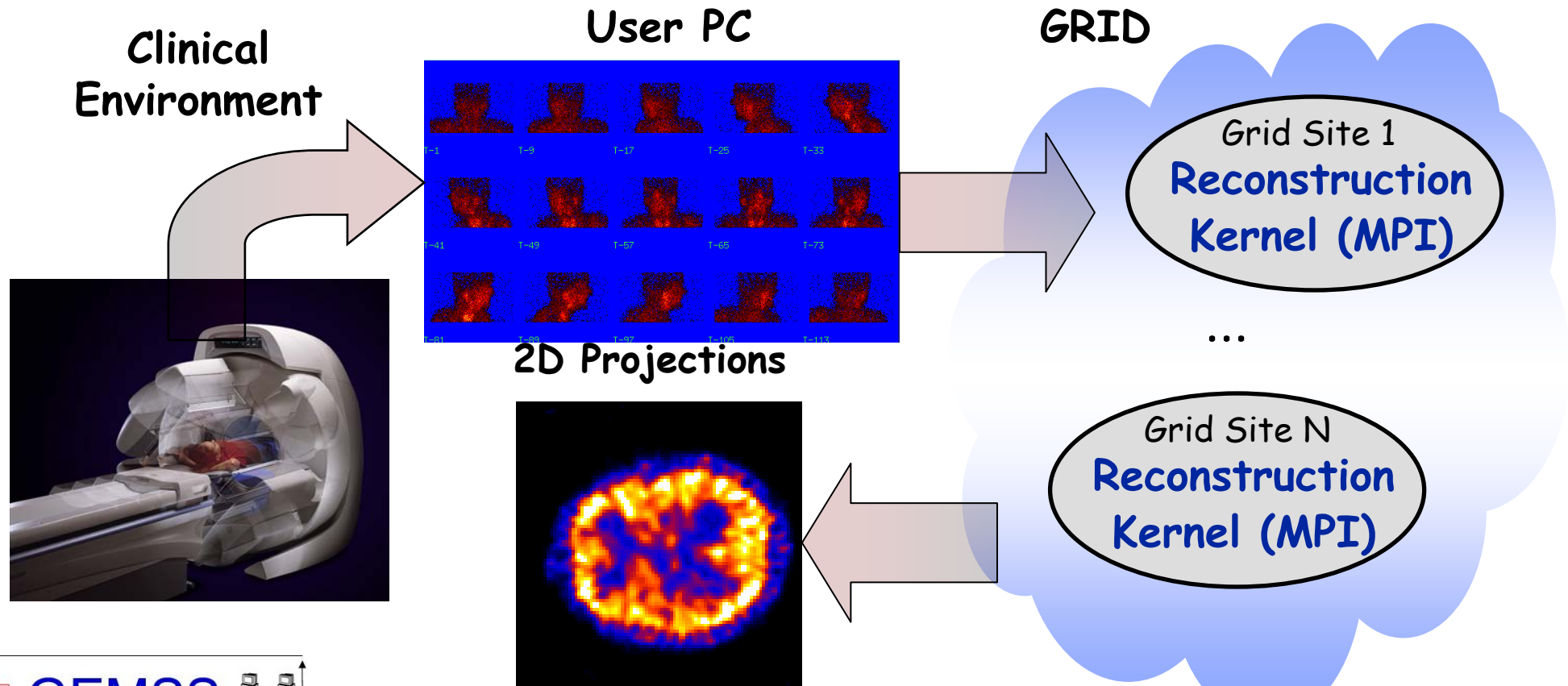
# GEMSS Client-side programming

```
...
public class Client {
...
    public static void main(String[] args) {
        ...
        SPECT spect = (SPECT) GEMSS.getService
            ("SPECT", reqDsc, qosDsc);

        spect.upload("input.zip");
        spect.start();
        while (!finished) {
            spect.getStatus();
            ...
        }
        spect.download("output.zip");
    }
}
```

# Image Reconstruction Service (SPECT)

- ❑ Single photon emission computed tomography (SPECT) is an effective & robust method for diagnosis of cancer, heart diseases and functional pathologies.
- ❑ our method takes into account: scanner geometry, scatter, attenuation, etc.  
→ large computational requirements

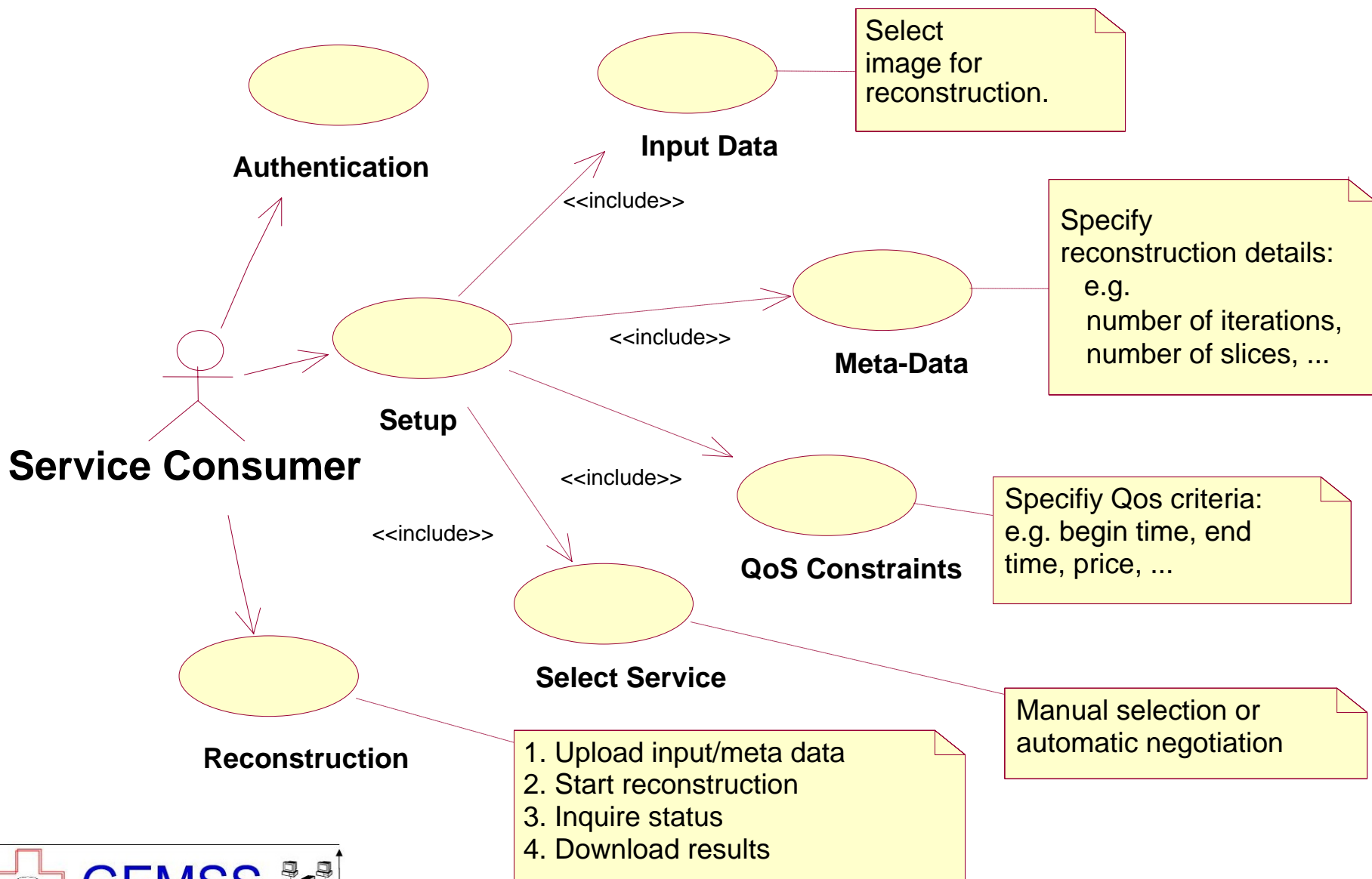


# Using a GEMSS Service

---

1. Authentication (once)
2. Acquisition of 2D projection data (e.g. from a SPECT scanner)
3. Specification of reconstruction **meta-data** (ROI, required accuracy, ... )
4. Specification of **QoS requirements** (e.g. execution time) [optional].
- ~~5. Service selection [optional or done automatically]~~
- ~~6. Upload of 2D input data (approx. 4 MBs)~~
7. **Start of reconstruction** (parallel ML-EM reconstruction kernel)
8. **Query status** information [optional]
9. **Download of results**
10. Visualization of reconstructed 3D image

# Using a GEMSS Service



# SPECT Client - Import Projection Data

**GEMSS SPECT Client** [min] [max] [close]

Definition File Import: Jas.def

**Data File:**

**Data File Type:**

**Data File Format:**  **Dynamic Acquisition:**

**Number of Voxels:** X:  Y:  Z:

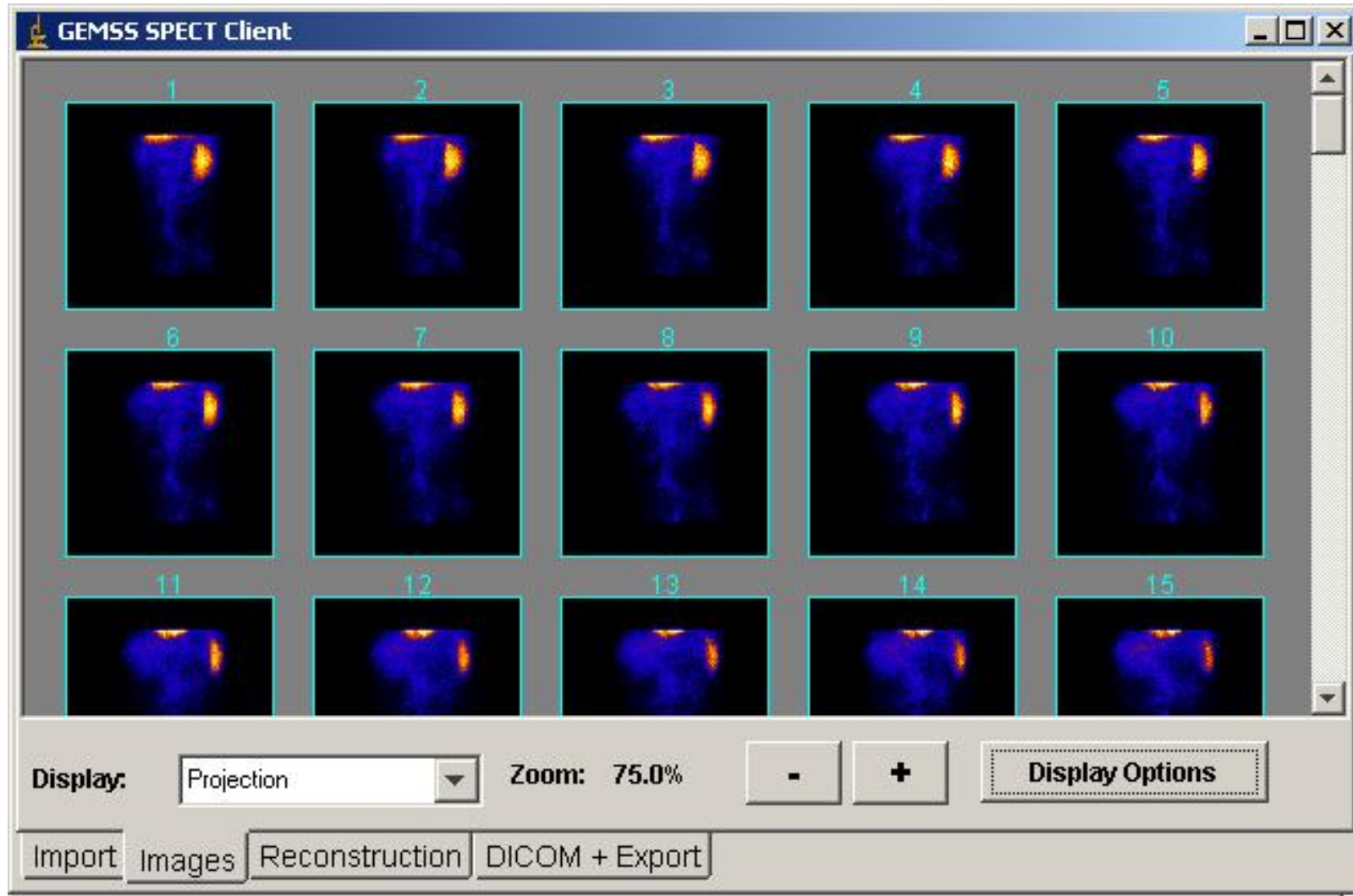
**Voxel Size (mm):** X:  Y:  Z:

**Projection Range (°):**  **Rotation Direction:**

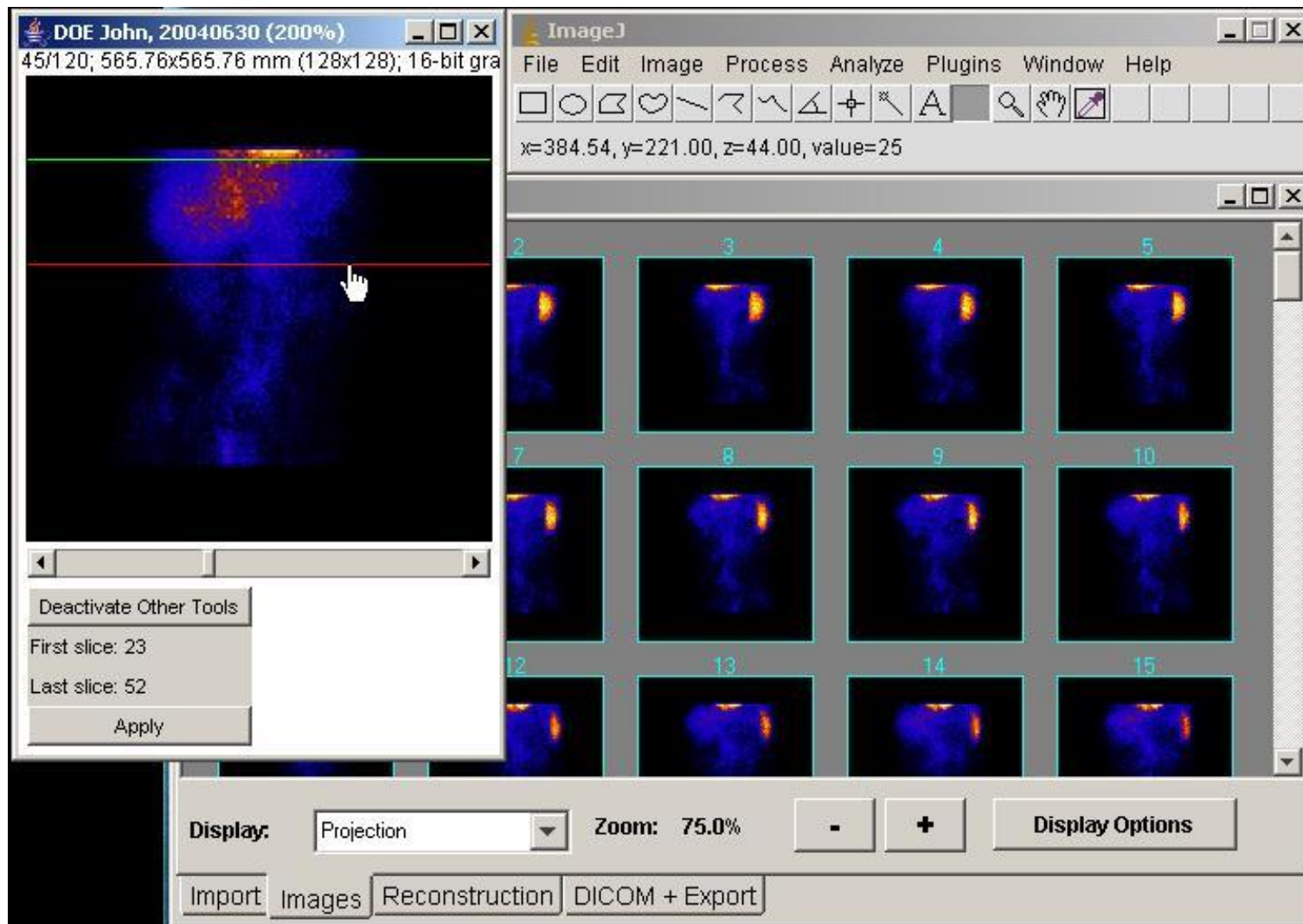
**Import Type:**  **Import As:**

Import Images Reconstruction DICOM + Export

# SPECT Client - View Projection Data



# SPECT Client - Detail Selection



# SPECT Client - Reconstruction

**Reconstruction Meta-Data**

Options: **Edit Reconstr. Opt.**

es: 128  
els XYZ: 128 128 128

First slice: 1  
Last Slice: 128  
Rotation direction: counter-clockwise  
Projection range (degrees): 360.0  
Rotation Increment (degrees): 3.0  
Radius of detector rotation (mm): 325.6  
Voxel size X,Y,Z (mm): 4.42 4.42 4.42

**QoS Parameters**

Service Options:

Begin Time: 05:39  
Stop Time: 06:34  
Max Price: 1000  
Service(s):  
https://unicore.ccril-nece.de:4430/gemss/services/SPECT/apex?wsdl  
https://bridge.vcpc.univie.ac.at/gemss/services/SPECT/apex?wsdl  
https://bridge.vcpc.univie.ac.at/gemss/services>HelloWorldUnix/apex?wsdl

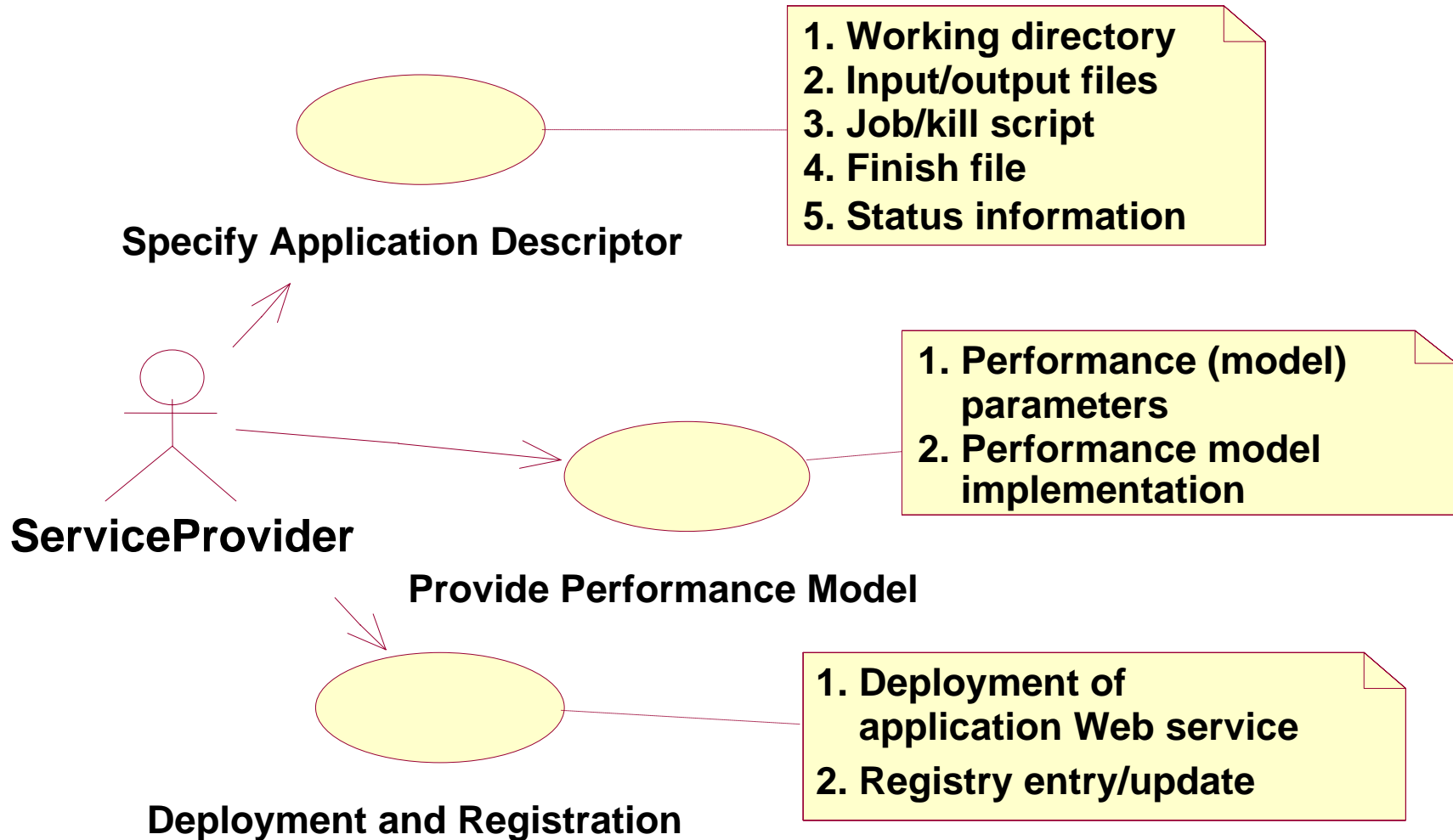
**Service Selection and Reconstruction**

Reconstruction Status: **Start** **Stop**

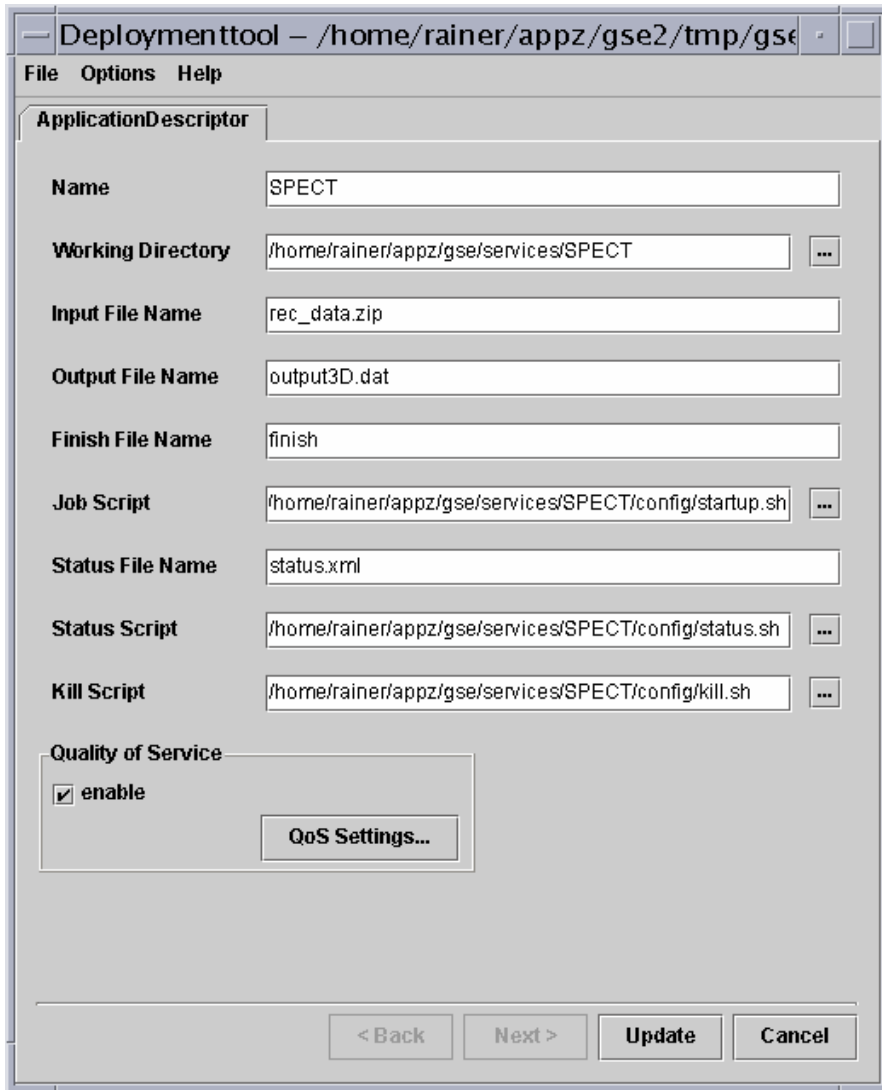
14.07.2004 05:41:21: Input archive file = C:\Projects\GEMSS\_SPECT\_Client\data\input1089844873313.zip  
14.07.2004 05:41:21: Output archive file = C:\Projects\GEMSS\_SPECT\_Client\data\input1089844873313.zip  
14.07.2004 05:41:21: Fast retry delay [negotiation] = 1000 ms  
14.07.2004 05:41:21: Slow retry delay [job run] = 5000 ms  
14.07.2004 05:41:21: Timeout delay for negotiation = 30000 ms  
14.07.2004 05:41:21: Timeout delay for operation invocation = -1 ms  
14.07.2004 05:41:21: Timeout delay for job runs = -1 ms  
14.07.2004 05:41:21: Starting the component manager  
14.07.2004 05:41:25: Asking for a GEMSSNegotiator to run the job  
14.07.2004 05:41:25: Waiting for negotiation to finish  
14.07.2004 05:41:31: Reconstruction stopped - Negotiation failed. Unable to connect to

Import Images Reconstruction DICOM + Export

# GEMSS Service Provision



# Application Configuration



The screenshot shows a window titled "Deploymenttool" with a menu bar containing "File", "Options", and "Help". The main area is titled "ApplicationDescriptor" and contains several fields for configuration:

- Name: SPECT
- Working Directory: /home/rainer/appz/gse/services/SPECT
- Input File Name: rec\_data.zip
- Output File Name: output3D.dat
- Finish File Name: finish
- Job Script: /home/rainer/appz/gse/services/SPECT/config/startup.sh
- Status File Name: status.xml
- Status Script: /home/rainer/appz/gse/services/SPECT/config/status.sh
- Kill Script: /home/rainer/appz/gse/services/SPECT/config/kill.sh

At the bottom, there is a "Quality of Service" section with a checked "enable" checkbox and a "QoS Settings..." button. Navigation buttons at the very bottom are "< Back", "Next >", "Update", and "Cancel".

## Application Configuration:

- ⇒ Working Directory
- ⇒ Job/Kill Script
- ⇒ Input/Output Files
- ⇒ Status File/Script
- ⇒ Finish File

## ➤ Application Descriptor

# Application Configuration

Deploymenttool – /home/rainer/appz/gse2/tmp/gse

File Options Help

ApplicationDescriptor

Name SPECT

Working Directory /home/rainer/appz/gse/services/SPECT

Input File Name rec\_data.zip

Output File Name output3D.dat

Finish File Name finish

Job Script /home/rainer/appz/gse/services/SPECT/config/startup.sh

Status File Name status.xml

Status Script /home/rainer/appz/gse/services/SPECT/config/status.sh

Kill Script /home/rainer/appz/gse/services/SPECT/config/kill.sh

Quality of Service

enable

QoS Settings...

< Back Next > Update Cancel

## Application Configuration:

- ⇒ Working Directory
- ⇒ Job/Kill Script
- ⇒ Input/Output Files
- ⇒ Status File/Script
- ⇒ Finish File

## ➤ Application Descriptor

# Application QoS Support (optional)

The screenshot shows the 'Deploymenttool' window with the 'QoS Settings' tab selected. It is divided into four main sections: 'Provider Parameters', 'Application Performance Model', 'Machine Parameters', and 'Performance Parameters'. Each section contains fields for 'Implementation Class', 'Java Archive', and 'Configuration (optional)'. The 'Machine Parameters' section includes a 'Number of Nodes' field with a list of options (1, 2, 4, 8, 12, 16) and a note about node numbering. The 'Performance Parameters' section has a table with columns for 'name' and 'Edit', containing 'NumberOfSlices' and 'ImageResolutionX'.

name	Edit
NumberOfSlices	
ImageResolutionX	

## QoS Configuration:

- ⇒ Business Model  
(implementation, configuration)
- ⇒ Performance Model  
(implementation, parameters, configuration)
- ⇒ Compute Resource Manager  
(implementation, parallelization)

# Application Descriptor

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application>
  <info>
    <name>SPECT</name>
  </info>
  <configuration>
    <working-directory>
      <path>/home/rainer/appz/gse2/tmp/gsel/gsel/services/SPECT</path>
    </working-directory>
    <input-files>
      ...
    </configuration>
    <performance>
      <performance-parameters>
        <performance-parameter>
          <name>NumberOfSlices</name>
          ...
        </performance-parameter>
      </performance-parameters>
      <machine-parameters>
        <number-of-nodes>
          <number>1</number>
          ...
        </machine-parameters>
        <provider-parameters>
          <compute-resource-manager-class>at.ac.univie.iss.CManager</compute-resource-manager-class>
          <business-model-class>at.ac.univie.iss.bm.BusinessModel</business-model-class>
          <business-model-configuration>/home/rainer/appz/gse/conf/payperuse.xml</business-model-configuration>
          <performance-model-class>at.ac.univie.iss.apm.PerformanceModel</performance-model-class>
          <performance-model-configuration>/home/rainer/appz/gse/conf/SPECT.xml</performance-model-configuration>
        </provider-parameters>
      </performance>
    </application>
```

# Application Deployment + Registration

Deploymenttool - /home/rainer/appz/gse/services/!

File Options Help

ApplicationDescriptor Configuration

Context Name SPECT

ApplicationDescriptor /home/rainer/appz/gse/services/SPECT/config/SPECT.xml ...

Tomcat

Directory /home/rainer/appz/jakarta-tomcat-4.1.27 ...

URI http://carry.par.univie.ac.at

Username manager

Password \*\*\*\*\*

publish to registry

enable

Registry URI http://carry.par.univie.ac.at/registry/regservice?wsdl

name	value
serviceclass	SPECT

enable WS-Security

< Back Next > Save Cancel

## Deployment:

- ⇒ Application Descriptor Location
- ⇒ Hosting Environment Settings (Tomcat)

## Registration:

- ⇒ Registry Service URI
- ⇒ Service Parameters

# Conclusion

---

GEMSS is about **secure & lawful** grid provision of medical simulation services!

- ❑ “processor” of patient data must be known
- ❑ restrictions on data management
- ❑ most available Grid technologies of limited use to GEMSS

**Application-level QoS is a key issue** in medical context!

- NEC Europe Ltd. (St. Augustin, Germany; Co-ordinator)
- IT Innovation (UK)
- Institute for Software Science (Austria)
- Max-Planck Institute of Cognitive Neuroscience (Leipzig)
- University of Sheffield (UK)
- AEA Technology (UK)
- CRID, Research Centre for Computer and Law (Belgium)
- Advanced Simulation & Design GmbH (Rostock, Germany)
- IDAC Ltd (Ireland)
- Institute of Biomedical Engineering and Physics (Austria)