



EurValve Model Execution Environment in Operation

Marian Bubak^{1,2}, Daniel Haręźlak¹, Tomasz Bartyński¹, Tomasz Gubala¹,
Marek Kasztelnik¹, Maciej Malawski^{1,2}, Jan Meizner¹, Piotr Nowakowski¹

¹ACC Cyfronet AGH, Krakow, Poland



²Department of Computer Science, AGH, Krakow, Poland

<http://dice.cyfronet.pl/>



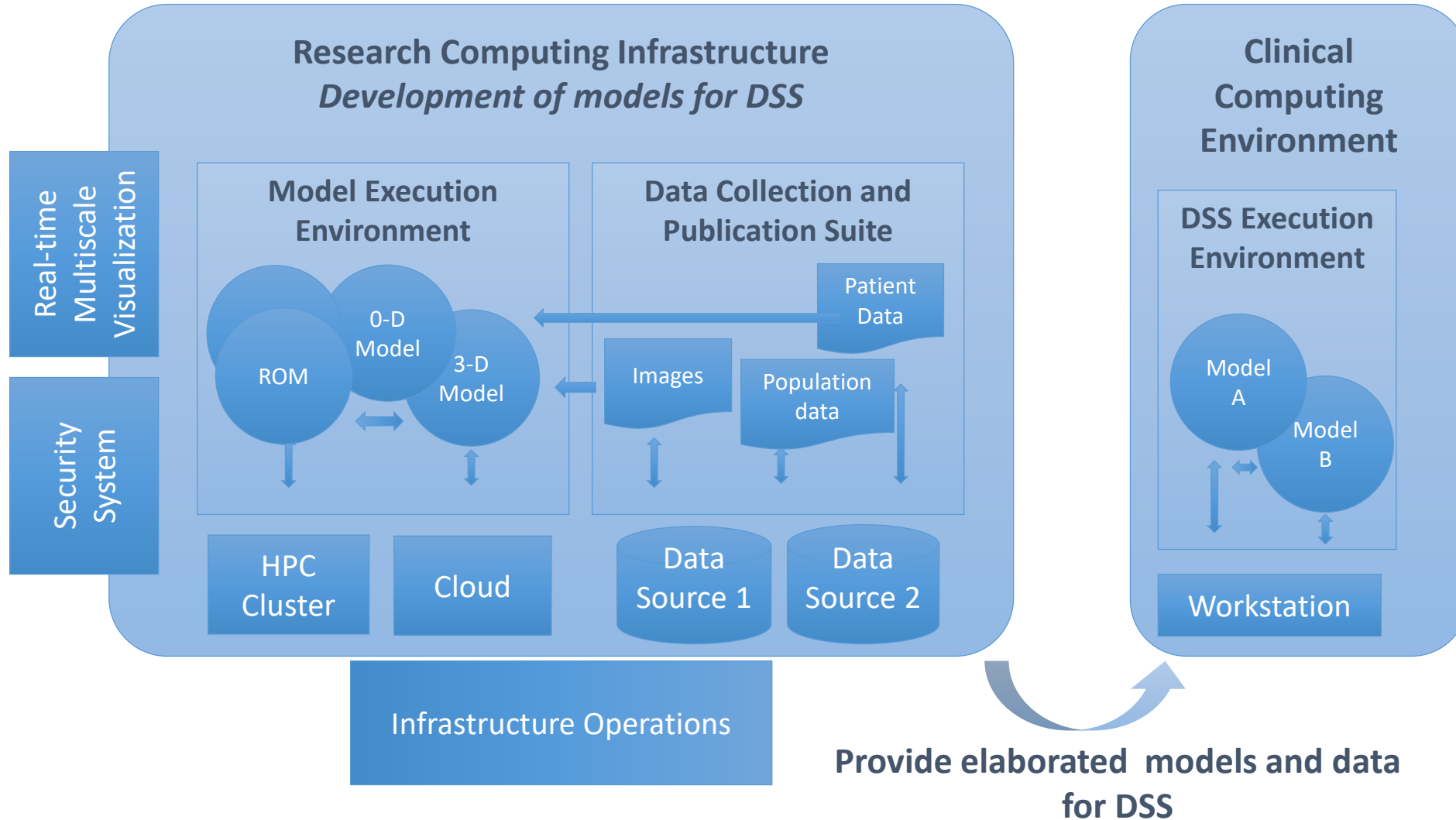
Outline



- Motivation: towards a Decision Support Systems for valvular diseases
- Requirements for computing and storage
- Architecture of the Model Execution Environment
- Implementation of the Model Execution Environment
- MEE in operation: a patient pipeline
- Recorded demos of MEE operation
- Summary and future work



From Research Environment to DSS



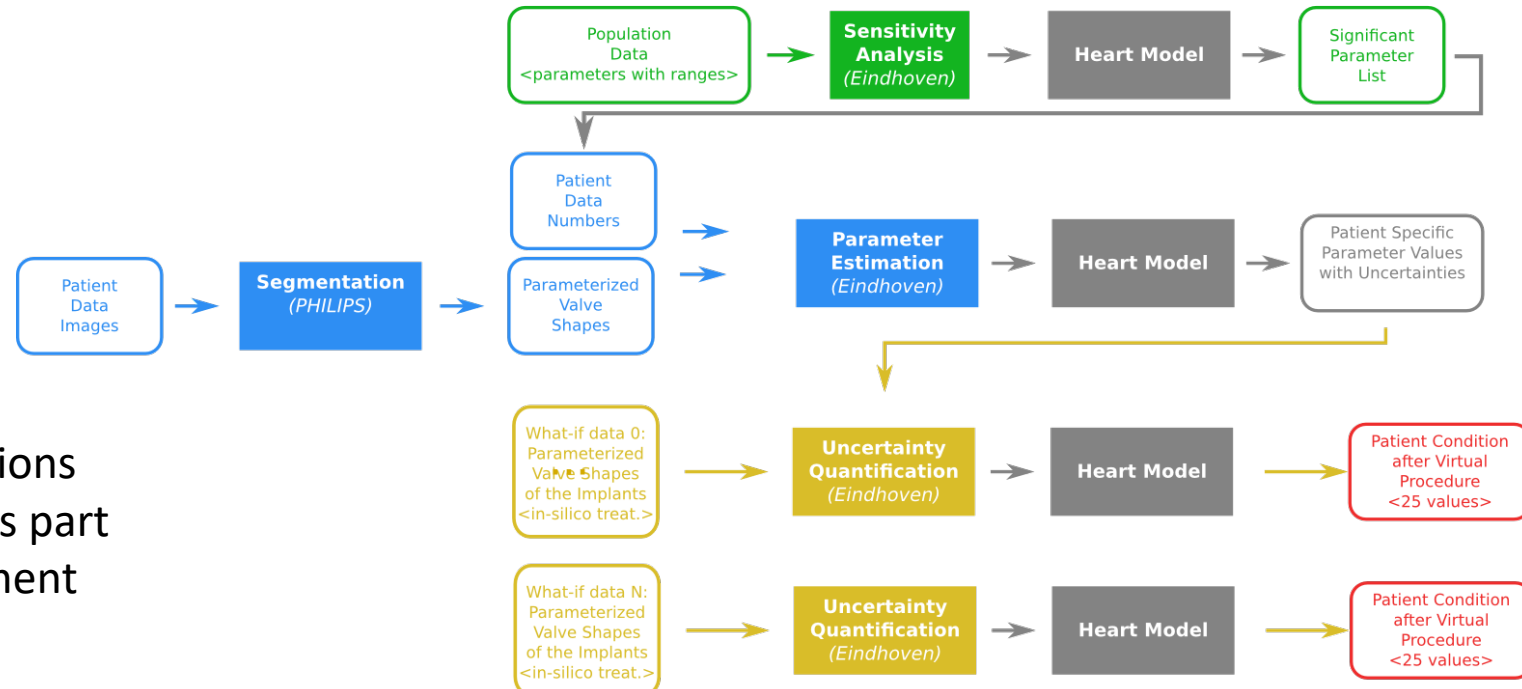


Pipelines for ROM and sensitivity analysis



Data and action flow consists of:

- full CFD simulations
- sensitivity analysis to acquire significant parameters
- parameter estimation based on patient data
- uncertainty quantification of various procedures



The flow of CFD simulations and sensitivity analysis is part of clinical patient treatment



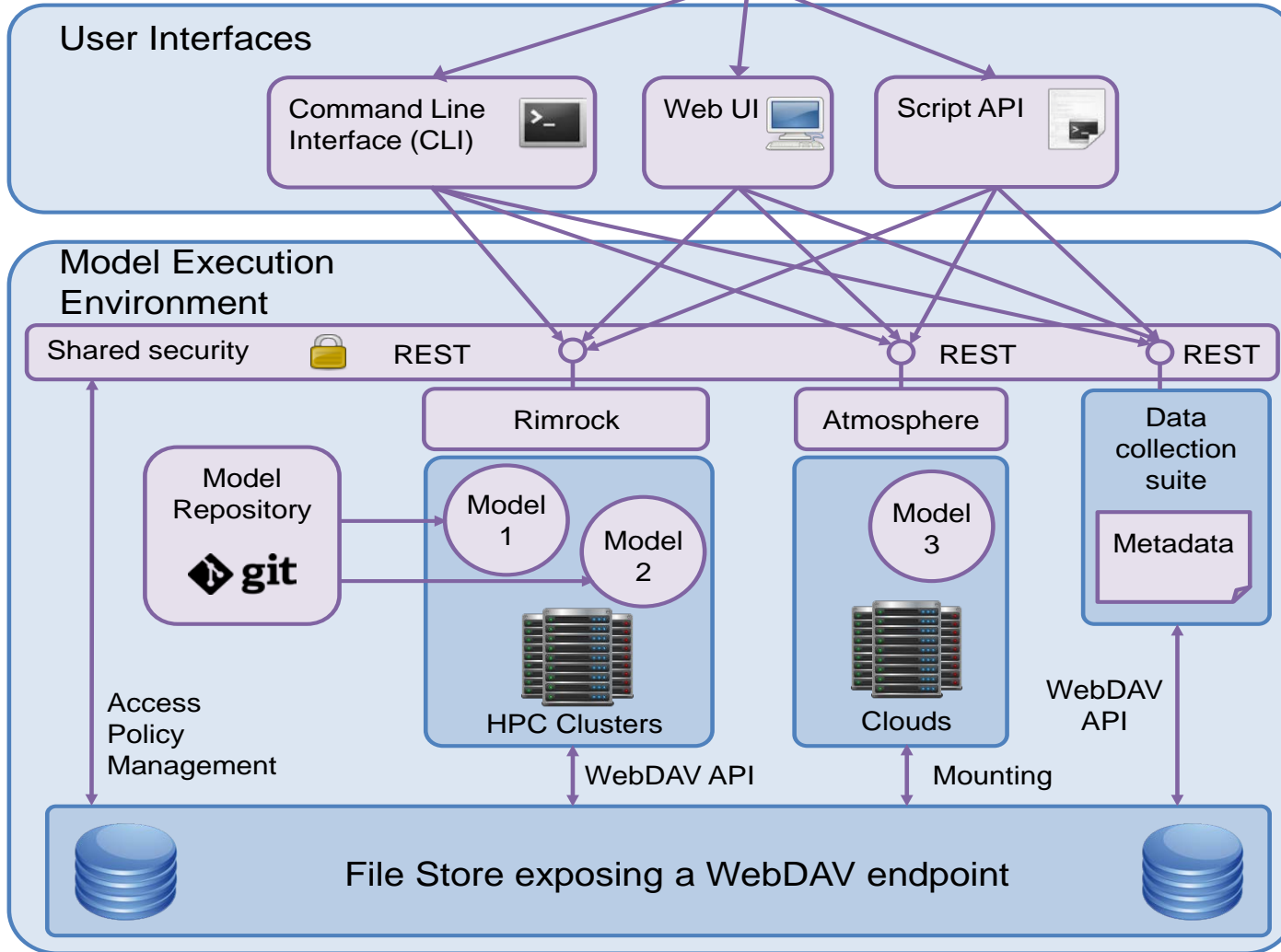
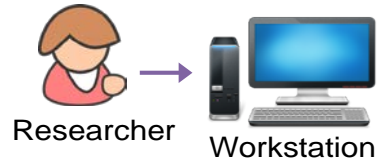
Functionality of Model Execution Environment



- Reproducibility, versioning, documentation of the pipeline
- Automation of the simulation pipeline with human in the loop for:
 - new models, new versions of models,
 - new users
- Data preservation
- Basic provenance features
- Helpful visualization of simulation flow and obtained results
- Generation of some components of publications
- Portability



Model Execution Environment - structure



- The MEE can be interfaced from a dedicated GUI (the EurValve Portal), through a RESTful API or through a command-line interface, depending on the researcher's preferences.
- Computational tasks can be run on HPC resources or in a cloud environment, as appropriate.
- A uniform security layer is provided.

API – Application Programming Interface

REST – Representational state transfer

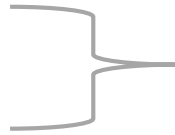
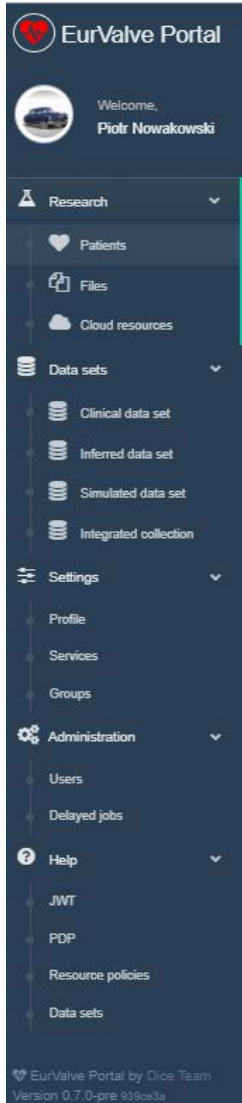
Rimrock – service used to submit jobs to HPC cluster

Atmosphere – provides access to cloud resources

git – a distributed revision control system

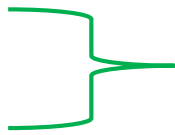


Implementation of MEE



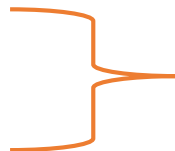
Model Execution Environment:

- Patient case pipeline integrated with File Store and Prometheus supercomputer
- *File Store for data management*
- Cloud resources based on Atmosphere cloud platform



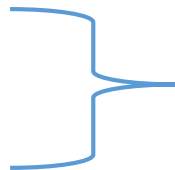
Data sets

- Structural access to patient databases
- Query interfaces for real, simulated and inferred data



Security configuration

- Service management – for every service dedicated set of policy rules can be defined
- User Groups – can be used to define security constraints



REST API

- Creating a new user session – as a result, new JWT (JSON Web Token) tokens are generated for credential delegation
- PDP – Policy Decision Point: check if user has access to concrete resource
- Resource policies – add/remove/edit service security policies



MEE extensibility – support for additional computational modules



Additional modules can be implemented:

- As applications deployed on the Prometheus supercomputer
- As external services communicating with the platform via its REST interfaces
- As virtual machines deployable directly in the Cyfronet cloud via the Atmosphere extension of the MEE

Encapsulating pipeline steps as HPC tasks:

- Scripts are run on the Prometheus supercomputer via the Rimrock extension
- Files uploaded to the FileStore (e.g. using MEE GUIs) can be accessed on Prometheus nodes via **curl**, leveraging the WebDAV interface provided by FileStore
- Any result files can also be uploaded directly to FileStore from the Prometheus computational nodes
- External tools can be used to monitor job completion status e.g. by periodically scanning FileStore content



Generic MEE tools



The screenshot shows the 'Cloud resources' page in the EurValve Portal. The left sidebar contains navigation options: Research, Patients, Files, Cloud resources, and Data sets. The main content area is titled 'Cloud resources' and includes tabs for 'Instance Management', 'instances', and 'Workflows'. A 'Start development instance' button is visible. Below, a table lists 'Running Development Instances' with columns for Name, IP, Location, Status, Charge, and Actions.

Name	IP	Location	Status	Charge	Actions
drs	10.100.20.55	Cyfronet new site	active	38.00	[stop] [start] [suspend] [reboot] [delete]
my_vm	10.100.20.53	Cyfronet new site	active	30.00	[stop] [start] [suspend] [reboot] [delete]

The screenshot shows the 'Files' page in the EurValve Portal. The left sidebar is the same as in the previous screenshot. The main content area is titled 'Files' and includes an 'Upload' button, a 'New directory name' input field, and a '+ Create directory' button. Below, a list of files and directories is shown with columns for Name, Location, and created time. Each item has a set of action icons (upload, download, delete, share).

Name	Location	created
codas		created 1/18/17, 6:57 PM
danstest		created 6/29/17, 8:45 AM
demo		created 6/29/17, 12:54 PM
development		created 9/18/17, 8:41 AM
EurValveODModule		created 1/10/17, 6:57 PM
mmstest		created 6/23/17, 11:05 AM
production		created 5/23/17, 10:33 AM
ProspectiveImaging		created 9/22/17, 8:28 AM

Cloud resources

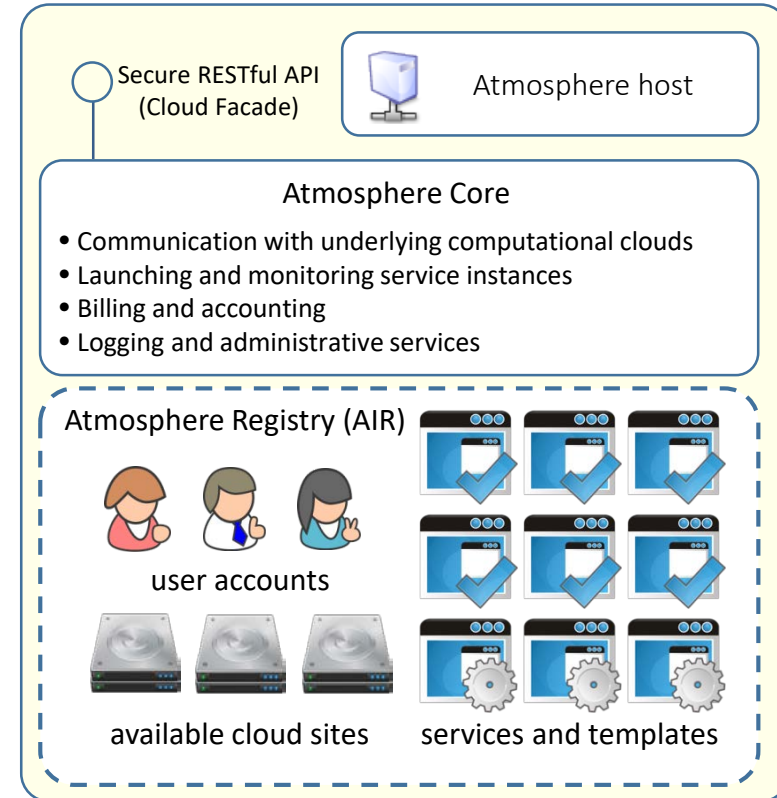
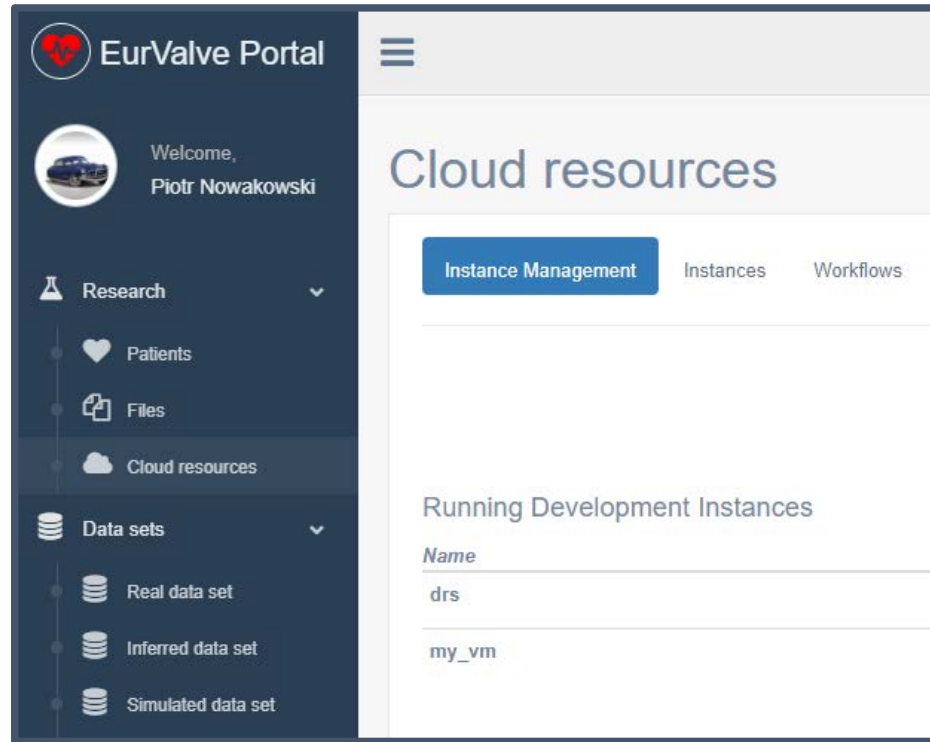
- Based on Atmosphere cloud platform
- Can start/stop/suspend virtual machine on cloud infrastructure
- Can save running existing machine as template
- (Future) can share templates with other users

File Store

- Basic file storage for the project
- Ability to create new directories and upload/download files
- Can share directories with other users or groups of users
- Can be mounted locally using WebDav clients
- The File Browser GUI can also be embedded in other views



MEE - cloud access via Atmosphere



Access to cloud resources

- The Atmosphere extension provides access to cloud resources in the EurValve MEE
- Applications can be developed as virtual machines, saved as templates and instantiated in the cloud
- The extension is available directly in the MEE GUI and through a dedicated API
- Atmosphere is integrated with EurValve authentication and authorization mechanisms



MEE functionality via REST API



EurValve Portal

Welcome, Piotr Nowakowski

Policy management API

Access to the policy management API is authorized by delegating user credentials (using Bearer Authorization header) and providing a service token via the `X-SERVICE-TOKEN` header with each of the requests. The path attribute can contain a wildcard character at the end (and only there) to match any path part (e.g. `http://host.com/path*`). The API exposes the following REST methods:

GET /api/policies[?path=...]

path A coma-separated list of paths or a single path.

Returns a list of policies for a given path (or paths). The matching of paths is exact without any regular expression processing.

Response body:

```
{
  "policies": [
    {
      "path": "...",
      "managers": {
        "users": ["..."],
        "groups": ["..."]
      },
      "permissions": [
        {
          "type": "user_permission|group_permission",
          "entity_name": "...",
          "access_methods": ["..."]
        }
      ]
    }
  ]
}
```

Example using cURL:

Generate user JWT Token

- User (or other service) can retrieve new JWT token by passing username and password
- JWT token can be used for user credential delegations by external EurValve services

PDP API

- Check if user has right to access a specific resource

Resource policy management

- Create/edit/delete local policies by external EurValve service on user behalf
- Currently integrated with File Store
- Initial ArQ integration tests underway



MEE security management UIs



Name	URI	
Eurvalve_Pro prospective_Data	https://eurvalve.shef.ac.uk	🔗 ✖
File Store	https://files.valve.cyfronet.pl	🔗 ✖
pdp-test	http://valve.cyfronet.pl:8080	🔗 ✖
Testing	http://tom.ek/path	🔗 ✖

Name	
admin	🔗 ✖
Cyfronet	🔗 ✖
demo	🔗 ✖
Eurvalve_clinical	🔗 ✖
Eurvalve_research	🔗 ✖
mkaasztehnk-group	🔗 ✖
supervisor	🔗 ✖
webdav	🔗 ✖

Services

- Basic security unit where dedicated security constraints can be defined
- Two types of security policies:
 - Global – can be defined only by service owner
 - Local – can be created by the service on the user's behalf

Groups

- Group users
- Dedicated portal groups:
 - Admin
 - Supervisor – users who can approve other users in the portal
- Generic groups:
 - Everyone can create a group
 - Groups can be used to define security constraints



Recorded demos of MEE



- Logging in to EurValve and PLGrid systems – <https://youtu.be/4I907aAOCvU>
- File Store Browser – <https://youtu.be/-6fXG0am6iE>
- Distributed Cloud File Store – <https://youtu.be/FTF-QaI5ZZQ>
- Services, security, restricted access – <https://youtu.be/-6fXG0am6iE>
- Cloud Resource Access – <https://youtu.be/A4wkxFCRLak>
- Patient case
 - <https://youtu.be/SlwpxdoQYWw> (patient case)
 - <https://youtu.be/j0Nu-E-0eIE> (pipeline diff)
- Integration of computational services – <https://youtu.be/SlwpxdoQYWw>



The Patient Case Pipeline



1. Segmentation – to start this calculation, a zip archive with a dedicated structure needs to be created and transferred into the OwnCloud input directory. Next, the output directory needs to be monitored for computation output.
2. Reduced Order Model analysis – based on the results of the segmentation step, a ROM simulation is executed and its results uploaded to the File Store.
3. Parameter Optimization – a technical step which prepares suitable parameters for the OD model sequence
4. OD model sequence – runs four versions of the OD model analysis for various input datasets
5. Uncertainty Quantification – Matlab script which can include the OD Heart Model. It will be executed on the Prometheus supercomputer, where input files will be transferred automatically from the File Store. Results are transferred back from Prometheus to the File Store.
6. Output visualization – produces actionable visualization of the OD model output data based on File Store contents.

Patient Case Pipeline high-level building blocks:

- File-driven computation (such as Segmentation) – use case: upload file to remote input directory, monitor remote output directory for results
- Scripts started on Prometheus supercomputer – use case: transfer script and input files from File Store to the cluster, run job, monitor job status, once the job has completed – transfer results from the cluster to File Store (examples: OD Heart Model, Uncertainty Quantification, CFD simulation)



Future work



1. Further improvement of the MEE based on user feedback
2. Implementation of computation quality validation against retrospective patient data in order to compare pipeline results with retrospective patient data measured *in vivo* following intervention
3. Implementation of additional automation features for existing pipelines, enabling fine-grained control over the configuration and execution of individual pipeline steps in the context of specific patient data.
4. Development of advanced accounting mechanisms:
 - logging events, such as data access or attempts to do so, with appropriate metadata
 - analysis of events to detect anomalies (such as suspected access from new IP/location, device etc.) to potentially alert administrators, data owners, users etc. about suspicious activities.



More at

<http://dice.cyfronet.pl>

<http://www.eurvalve.eu>



EurValve H2020 Project 689617