



HAL
open science

Coolemall D2.6 Final release of the simulation and visualisation toolkit

Eugen Volk, Yosandra Sandoval, Nico Eichhorn, Georges da Costa, Thomas Zilio, Micha Vor Dem Berge, Dirk Michels, Wojciech Piatek, Piotr Grabowski, Enric Pages

► **To cite this version:**

Eugen Volk, Yosandra Sandoval, Nico Eichhorn, Georges da Costa, Thomas Zilio, et al.. Coolemall D2.6 Final release of the simulation and visualisation toolkit. [Research Report] IRIT-Institut de recherche en informatique de Toulouse. 2014. hal-01818161

HAL Id: hal-01818161

<https://hal.science/hal-01818161>

Submitted on 18 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Project acronym: **CoolEmAll**

Project full title: **Platform for optimising the design and operation of modular configurable IT infrastructures and facilities with resource-efficient cooling**



D2.6 Final release of the simulation and visualisation toolkit

Author: Eugen Volk (HLRS)

Version: 1.11

Date: 31/03/2014

Deliverable Number:	D2.6
Contractual Date of Delivery:	31/03/2014
Actual Date of Delivery:	31/03/2014
Title of Deliverable:	Final release of the simulation and visualisation toolkit
Dissemination Level:	Public
WP contributing to the Deliverable:	WP 2
Authors:	Eugen Volk (HLRS)
Co-Authors:	Yosandra Sandoval (HLRS) Nico Eichhorn (HLRS) Georges da Costa (IRIT) Thomas Zilio (IRIT) Micha vor dem Berge (Christmann) Dirk Michels (Christmann) Wojciech Piatek (PSNC) Piotr Grabowski (PSNC) Enric Pages (ATOS)

History			
Version	Date	Author	Comments
1.0	03.03.14	Eugen Volk (HLRS)	Structure and content update based on D2.5
1.1	06.03.14	Yosandra Sandoval (HLRS)	Update of Sections 3.2.3, 4.2.3 and 2.3
1.2	06.03.14	Thomas Zilio (IRIT)	Update of Sections 2.2.7, 2.3.6, 3.1.7 and 4.1.6
1.3	11.03.14	Micha vor dem Berge (Christmann)	Update of Section 2.4
1.4	12.03.14	Thomas Zilio (IRIT)	Update
1.5	13.03.14	Wojciech Piatek (PSNC)	Updating DCworms-related sections.
1.6	17.03.2014	Yosandra Sandoval (HLRS)	Merging contributions of the partners (PSNC, ATOS,

			Christmann and IRIT)
1.7	26.03.2014	Enric Pages	Update Annex A
1.8	26.03.2014	Piotr Grabowski (PSNC)	UpdateSections
1.9	26.03.2014	Wojciech Piatek (PSNC)	Updating DCworms-related sections.
1.10	27.03.2014	Nico Eichhorn (HLRS)	Updating COVISE-related sections.
1.11	27.03.2014	Yosandra Sandoval (HLRS)	Merging contributions of the partners (PSNC, HLRS, ATOS, Christmann and IRIT)

Approval		
Date	Name	Signature
31/03/2014	Andrew Donoghue (451G)	
31/03/2014	Lara López (ATOS)	

Abstract

This deliverable describes the realisation of the final prototype of the Simulation, Visualisation and Decision (SVD) support toolkit and the interaction of its components. It further describes the usage and the tests of the components of the final prototype of the SVD-Toolkit, via the CoolEmAll-Web-GUI interface. Another focus of this deliverable is describing the heterogeneous deployment architecture of the SVD-Toolkit and the use of the different components for performing an automatic simulation.

Special focus is put on the distributed deployment architecture, realization, usage and tests of this final prototype of SVD-Toolkit via the CoolEmAll-Web-GUI interface.

Keywords

SVD-Toolkit, CFD, Workload simulator, DCworms, database, deployment, Repository, Simulation, Visualization, COVISE, OpenFOAM

Table of Contents

1	Introduction	12
2	Realisation of the final Prototype of SVD-Toolkit	14
2.1	Deployment architecture	14
2.2	Detailed description of the components	21
2.2.1	Application Profiler	21
2.2.2	Repository	21
2.2.3	Database	23
2.2.4	DCworms	24
2.2.5	Open-FOAM based CFD-Solver	26
2.2.5.1	Naming convention for PLMXML-file	28
2.2.5.2	Path to data stored in database	29
2.2.5.3	Orientation of velocity at inlet	29
2.2.6	COVISE and CFX based CFD Solver	30
2.2.7	Metric Calculator	31
2.3	Detailed description of the CoolEmAll-Web-GUI	32
2.3.1	ExperimentConfigurator GUI	34
2.3.2	DEBBConfigurator GUI	38
2.3.3	DCworms GUI	49
2.3.4	COVISE GUI	52
2.3.5	MOP GUI	53
2.3.6	Metric Calculator and report-GUI	54
2.3.7	User interface	57
2.4	Addressing the EcoDesign Directive	59
3	Usage of the CoolEmAll-Web-GUI and SVD-Toolkit	61
3.1	Usage of the CoolEmAll-Web-GUI	61
3.1.1	General Flow	61
3.1.2	ExperimentConfigurator selection dialogue	62
3.1.3	DEBBConfigurator GUI	64
3.1.4	DCworms GUI	65
3.1.5	COVISE GUI	68

3.1.6	MOP GUI.....	69
3.1.7	Metric Calculator and Report-GUI.....	70
3.2	Expert usage of SVD-Toolkit components.....	71
3.2.1	Application Profiler.....	71
3.2.2	SVN Repository.....	71
3.2.3	Database.....	72
3.2.4	DCworms.....	73
3.2.5	CFD using OpenFOAM.....	74
3.2.6	CFD using COVISE with Ansys CFX.....	75
3.2.7	Metric Calculator.....	76
4	Test of the CoolEmAll-Web-GUI and SVD-Toolkit.....	80
4.1	Test of the CoolEmAll-Web-GUI.....	80
4.1.1	ExperimentConfigurator GUI.....	80
4.1.2	DEBBConfigurator GUI.....	84
4.1.3	DCworms GUI.....	88
4.1.4	MOP-GUI.....	91
4.1.5	COVISE-GUI.....	93
4.1.6	Metric Calculator and Report-GUI.....	93
4.2	Test of the SVD-Toolkit components.....	96
4.2.1	Application Profiler.....	96
4.2.2	SVN Repository.....	98
4.2.3	Database.....	99
4.2.4	DCworms.....	101
4.2.5	CFD simulation using OpenFOAM.....	102
4.2.5.1	Flow through RECS.....	103
4.2.5.2	Flow through Compute Room.....	104
4.2.6	CFD simulation using COVISE with Ansys CFX.....	105
4.2.7	Metric Calculator.....	105
5	Summary.....	109
6	Annex A. Description of test implemented to assess CoolEmAll methodology	
	111	
7	References.....	121

List of Figures

Figure 2-1: SVD-Toolkit Architecture overview – 1 st Prototype	15
Figure 2-2: SVD-Toolkit final prototype	17
Figure 2-3: Database Table Structure.....	23
Figure 2-4: Login page	33
Figure 2-5: Welcome page.....	33
Figure 2-6: ExperimentConfigurator–GUI Menu	35
Figure 2-7: Experiment configuration GUI	35
Figure 2-8: New experiment interface.....	36
Figure 2-9: Trial information	37
Figure 2-10: DEBB-menu.....	39
Figure 2-11: DEBBConfigurator GUI – Processor tab (components level)	40
Figure 2-12: DEBBConfigurator GUI – Node creation	43
Figure 2-13: DEBB configuration GUI – Node group creation	45
Figure 2-14: DEBBConfigurator GUI – Rack view with costs	46
Figure 2-15: DEBB configuration GUI – server room.....	47
Figure 2-16: DCworms GUI.....	51
Figure 2-17: Covise-GUI parameters and visualisation	52
Figure 2-18: MOP GUI Visualization	54
Figure 2-19: Simulation Report from Metric Calculator report GUI	57
Figure 2-20: User menu	58
Figure 2-21: Register interface	58
Figure 2-22: Register interface	59
Figure 3-1: Sequence diagram of ExperimentConfigurator GUI.	63
Figure 3-2: Sequence diagram of DEBBConfigurator GUI.	65
Figure 3-3: Sequence diagram of DCworms GUI	67
Figure 3-4: Sequence diagram of COVISE GUI.	69
Figure 3-5: Sequence diagram of MOP GUI.....	70
Figure 3-6: COVISE visualisation environment and outcome of the heat-flow simulation.....	75
Figure 4-1: Overview of Experiments.....	80
Figure 4-2: Experiment with two trials.....	82

Figure 4-3: List of trials of a particular experiment.....	83
Figure 4-4: Trail on the context.....	83
Figure 4-5: node group creation tab with adopted node	85
Figure 4-6: Exported room as ZIP file	86
Figure 4-7: Positioned heatsink in the chassis.....	88
Figure 4-8: Preview of application profile within DCworms GUI	89
Figure 4-9: DCworms GUI – load specification window.....	90
Figure 4-10: DCworms GUI - energy efficiency metrics.....	91
Figure 4-11: MOP-GUI - Standard mode view	92
Figure 4-12: MOP-GUI - Comparison mode view	92
Figure 4-13: COVISE GUI.....	93
Figure 4-14: Report-GUI experiment selection error.....	94
Figure 4-15: Report-GUI missing report error	94
Figure 4-16: Report-GUI currently computing error	94
Figure 4-17: Report-GUI normal report.....	95
Figure 4-18: Power usage chart generated for the DCworms simulation	102
Figure 4-19: velocity and temperature distribution inside RECS	103
Figure 4-20: velocity and temperature distribution inside a compute room	104
Figure 4-21: Visualisation of the heat-flow distribution within a room using COVISE renderer	105
Figure 22: DEEB configurator menu	111
Figure 23: SVN upload path.....	112
Figure 24: Node Group break-down menu	114
Figure 25: Room graph window	115
Figure 26: Adding data series to MOP	117
Figure 27: MOP - setting time for CFD simulation	119
Figure 28: CFD heat and air flow simulations	120

List of Tables

Table 2-1: Components overview 1 st Prototype	15
Table 2-2: Components of the final prototype of SVD Toolkit.....	18
Table 2-3: Software dependency list for SVN	22

Table 2-4: Software dependency list for Python Wrapper 24

Table 2-5: Software dependency list for DCworms..... 25

Table 2-6: Software dependency list for OpenFOAM based CFD 26

Table 2-7: Software dependency list for Metric Calculator 32

Table 2-8: Workload and resource management policies available within DCworms 49

Table 2-9: Frameworks, libraries and licenses used for implementation of the COVISE-GUI..... 53

Table 4-1: Workload characteristics..... 101

Table 6-1: Node Groups parameters 111

Table 6-2: RACK parameters..... 113

Table 6-3: Room parameters 114

Table 6-4: Experiment parameters 116

Table 6-5: Room parameters 116

Table 6-6: DCWorms result comparison 118

List of abbreviations

API	Application Programming Interface
CFD	Computational Fluid Dynamics
COVISE	Collaborative Visualisation and Simulation Environment
DCworms	Data Center Workload and Resource Management Simulator
DEBB	Data Centre Efficiency Building Block
GPL	General Public License
LGPL	GNU Lesser General Public License
GSSIM	Grid Scheduling Simulator
GUI	Graphical User Interface
GWF	Grid Workload Format
IP	Internet Protocol
MOP	Module Operation Platform
PLMXML	eXtensible Markup Language for Product Lifecycle Management
RPC	Remote Procedure Call
SVD	Simulation Visualisation and Decision support toolkit
SVN	Apache Subversion software versioning and revision control system
SWF	Standard Workload Format
STL	Surface Tessellation Language
TDP	Thermal Design Power
TIMaCS	Tools for Intelligent System Management of Very Large Computing Systems
URL	Uniform Resource Locator
VRML	Virtual Reality Modelling Language

1 Introduction

In scope of “D2.5 Second Release of the SVD-Toolkit” [D2.5] the realization of the 2nd prototype of the Visualization and Decision Support Toolkit (SVD-Toolkit) was described and delivered in PM 28. Within the last two project months, the focus within the WP2 was on the refinement of the implementation of the SVD-Toolkit, removing bugs to make it more stable and simplifying usage of the SVD-Toolkit. In this deliverable the realization of the final prototype of SVD-Toolkit is described, updating D2.5 final description and interaction of its components and CoolEmAll-Web-GUI interfaces. Another focus of this deliverable is to describe the heterogeneous deployment architecture of the SVD-Toolkit and the use of the different components for performing an automatic simulation with and without web based interfaces. It further describes the usage and the tests of the components of the final Prototype of the SVD-Toolkit via CoolEmAll-Web-GUI interfaces.

As noted in D2.4, the SVD-Toolkit is a tool to help design more energy efficient data centres and optimize existing data centres to operate in a more energy efficient manner. It allows assessment of energy- & cooling efficiency and facilitates optimization of data centre building blocks, reflecting various configurations of a data centre and its components on various scale levels, by means of coupled workload and thermal-airflow simulation. This is done in several different consecutive steps. First, different application profiles are calculated. These application profiles resemble the requirements normal applications usually have. With these application profiles synthetic workloads are generated and used by the workload simulator to evaluate various scheduling policies and determine power usage of the individual hardware components reflected in configuration of the data centre. These results are used as inputs for CFD (Computational Fluid Dynamics) simulation to calculate thermal-airflow distribution within compute-room, in order to identify hot spots and assess cooling efficiency. Furthermore, thermal-heat provided by CFD simulation is used to calculate power-usage of the cooling-devices. All results are stored in a central database to be used for calculation of the assessment metrics. Additional results are obtained by conducting several characteristic trials, so that all results can be verified.

In order to simplify the usage of the SVD-Toolkit, in scope of the final prototype a web based GUI interface was developed, allowing users to interact with each component of the SVD-Toolkit to execute and evaluate trials with various configurations in a single web based environment guided by corresponding WEB-GUI pages. Using CoolEmAll-Web-GUI users are capable easily of: (i) defining experiment configuration parameters, (ii) designing data centre efficiency building blocks (reflecting design of data centre components and layout), (iii) selecting application-profiles, workload-profiles and simulating various scheduling policies, (iv) getting visualisation of the workload simulation results and compare these against real measurements, (v) evaluating interactively various data-centre layouts and visualizing heat-flow distribution

within the server-room for various arrangements, and (vi) getting report assessing energy-efficiency and cooling-efficiency of experiment-trials. Furthermore, all components of the SVD-Toolkit have been improved and refined, adding: (i) new scheduling policies on room level, (ii) integrating DEBBs with COVISE and CFD simulation enabling interactive arrangement, and, (iii) adding new assessment metrics on room level. Overview of new components and extended functionality is presented in section 2.

The SVD-Toolkit components along with the CoolEmAll-Web-GUI described in this deliverable can be downloaded from the project-website [SVD-Toolkit].

This deliverable is split into five major parts. Special focus is put on the distributed deployment architecture, realization, usage and tests of this final prototype of SVD-Toolkit via CoolEmAll-Web-GUI interfaces. After the short introduction presented in this section, this deliverable describes the architecture and properties of the individual components of the final prototype of SVD-Toolkit presented in section 2, usage of these components in section 3, and their tests in section 4. Finally, section 5 summarizes this deliverable.

2 Realisation of the final Prototype of SVD-Toolkit

The final prototype of SVD-Toolkit consists of several different components including the web based GUI, called CoolEmAll-Web-GUI, allowing the interaction with the SVD-Toolkit and its components. The development of each component was done individually, while the interaction between all components in a seamless workflow is controlled via scripts and web based GUIs, integrated into a single environment – a portal solution based on Symfony2™ [SF2]. This approach grants the user the possibility of: (i) using each of the components of SVD-Toolkit on its own in expert mode using command line interfaces, and (ii) interacting seamlessly with all components integrated in the SVD-Toolkit via the web based GUI interfaces, allowing changing interactively simulation parameters, executing workload- and CFD- simulations, and visualising simulation results and assessment metrics.

This section is structured as follows: in Section 2.1 we describe the deployment architecture of the SVD-Toolkit, Section 2.2 provides detailed description of the core components of the SVD-Toolkit and Section 2.3 provides detailed description of the CoolEmAll-Web-GUI, consisting of several GUIs enabling interaction with the components.

2.1 Deployment architecture

In this section we describe deployment architecture of the 1st prototype of SVD-Toolkit, based on D2.4, and present architecture of the final prototype of SVD-Toolkit, extending the 1st prototype by integrating CFD simulation with COVISE and development of the CoolEmAll-Web-GUI.

The deployment of the SVD-Toolkit components and interaction between components of the 1st prototype is shown in Figure 2-1, presented in D2.4. At the beginning of the experiment DEBB-files are created. In DEBBs (Data centre Efficiency Building Blocks) all information, which is relevant for the individual simulation or trial, is stored. This is especially true for the underlying geometry. The DEBBs are stored in Apache Subversion™ (SVN) [ApSu] repository [CoolEmAll-SVN], along with the experiment description file specifying experiment setting, containing references to DEBB and workload (along with application-profile) used within the experiment. Workload specified in the experiment configuration file is used by the workload simulator DCworms, being executed on hardware represented by power-profiles stored in DEBB. The results, represented by several workload cases with specific power consumption, are then stored into the database. The CFD-simulator then retrieves the data from the database to perform its simulation on it and write the results again to the database where it is the input for the metric calculator. The metric calculator writes back into the database, after the calculation of metrics, where the MOP GUI can retrieve it. With this workflow the experiment conductor has the full feedback about his conducted experiment [D2.4].

For the physical deployment of the individual components the following was implemented in scope of the 1st prototype: Repository, Data Center Workload and Resource Management Simulator (DCworms), and Database are deployed at PSNC location. The Application Profiler and Metric calculator are located at IRIT. The CFD Solver is located at HLRS on a cluster environment. The detailed interaction between SVD-Toolkit components was explained in D2.2.

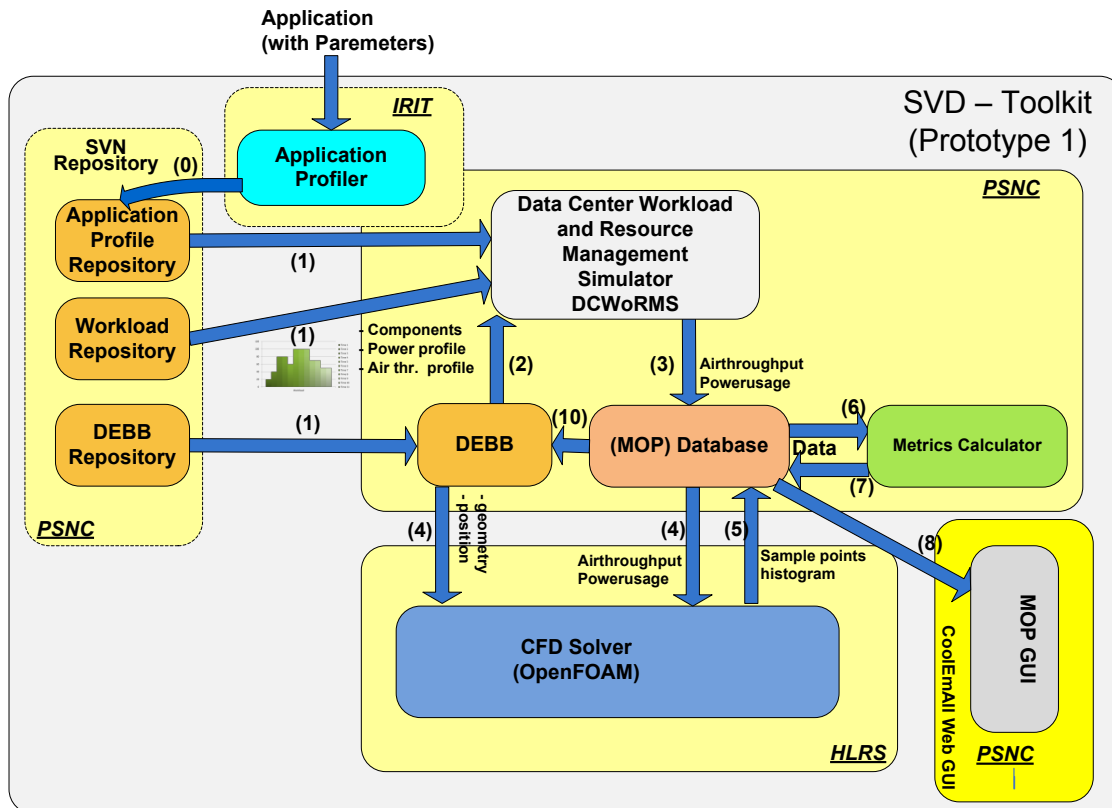


Figure 2-1: SVD-Toolkit Architecture overview – 1st Prototype

Table 2-1 summarizes components of the SVD-Toolkit, specifying components' license, description and functionality.

Table 2-1: Components overview 1st Prototype

Component name	License / Website	Description	Provided functionality for CoolEmAll
Database	GPL License. Version 2 LGPL License for RPC client and RPC server	MySQL Database for storing experimental	Storing dynamic data, interconnection

	Download: [SVD Toolkit]	data and outcome	point
CFD-simulator	GNU General Public License Version 3 MPL2 The MIT License Download: [SVD Toolkit]	Automated CFD-calculation environment for decision making in thermal management questions	Performing flow and temperature calculations
SVN-Repository	http://subversion.apache.org/ Apache License 2.0	Software versioning and revision control system	Repository with input parameters required: DEBBs, Profiles, Workloads
DCworms	GPL License Download: [SVD Toolkit]	Simulator for workload and resource management policies	Creates boundary and initial values for CFD-simulation
Metric calculator	Open Source Download: [SVD Toolkit]	Correlates energy consumption to work done	Evaluates experiment for energy efficiency
Application profiler	Open Source Download: [SVD Toolkit]	Simulation hardware requirements of different applications	Creates application profiles

Figure 2-2 provides an overview of the architecture of the final SVD-Toolkit prototype, extending 1st Prototype by integrating CFD with COVISE and especially implementing CoolEmAll-WEB-GUI, allowing easy interaction with the SVD-Toolkit and its components.

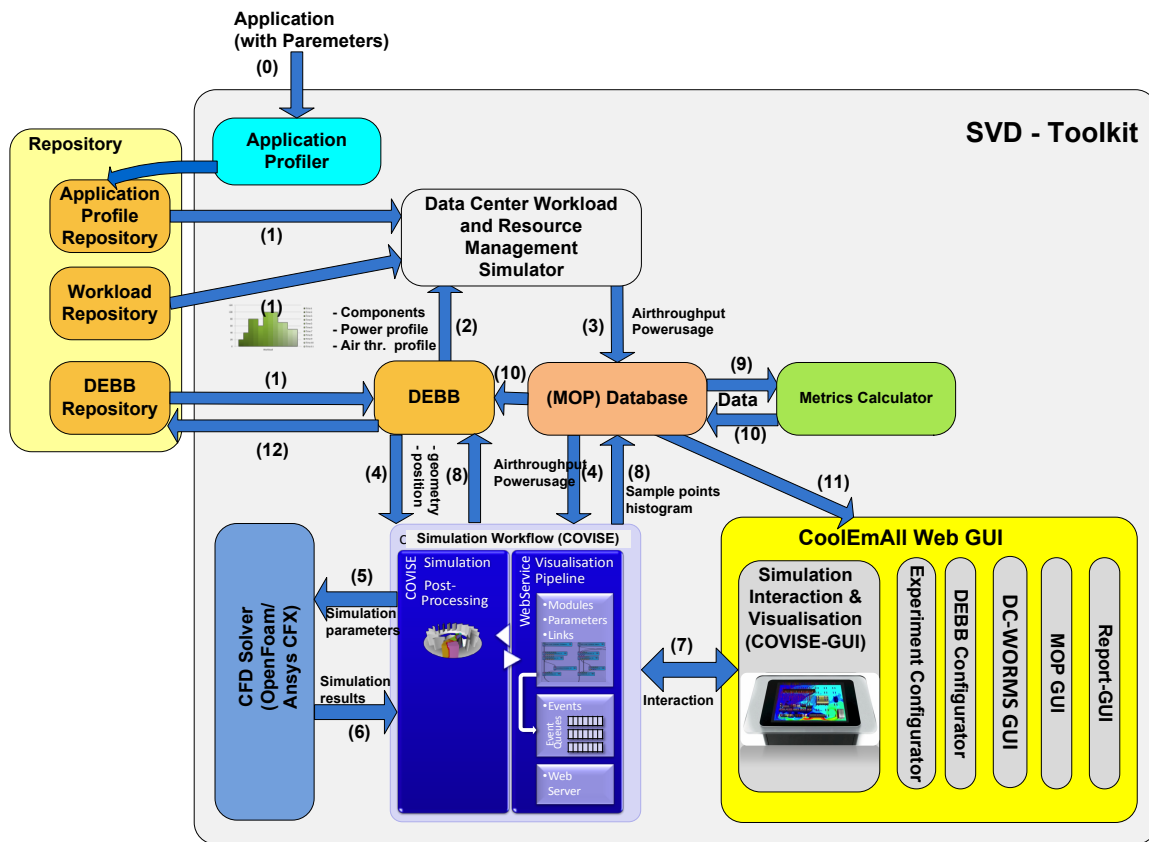


Figure 2-2: SVD-Toolkit final prototype

As noted in D2.2 Design of the CoolEmAll simulation and visualisation environment [D2.2], the Simulation Workflow COVISE (COLlaborative Visualization and Simulation Environment) is an extensible distributed software environment capable to integrate simulations, post-processing and visualisation functionalities in a seamless manner. The CFD Solver performing CFD simulation is directly integrated into the COVISE workflow, including all necessary pre- and post-processing tasks. The interaction with the CFD solver in the second prototype is done via COVISE. Both COVISE and CFD solver are deployed at HLRS.

The CoolEmAll-Web-GUI provides a web based GUI environment allowing interaction with the SVD-Toolkit and visualising its results. As noted, in D2.2 [D2.2] it comprises several GUIs integrated into a common web based GUI environment and consists of:

- Experiment configuration GUI enabling users to configure required experiment and trial parameters,
- DEBB configuration GUI allowing users to define DEBBs on various granularity level
- DCworms GUI allowing selection of workloads, applications and

- scheduling policies and showing the results of workload execution
- COVISE GUI presenting entire simulation results of the heat-flow (CFD) simulation and allowing interaction with the simulation by changing position of objects (racks) within the simulated server-room
- MOP-GUI allowing retrieving, visualizing and comparing test bed (real measurements) and simulation based results to validate models
- Report page providing assessment metrics evaluating energy- and cooling efficiency on various granularity levels of the simulation results.

Table 2-2 provides overview of the components developed and integrated within the final SVD-Toolkit prototype. As noted, all components of the 1st prototype have been refined, and new components were added, highlighted by bold/italic font style.

Table 2-2: Components of the final prototype of SVD Toolkit

Component name	License / Website	Description	Provided functionality for CoolEmAll
Database	GPL License Version 2.0 LGPL License for RPC client and RPC server Download: [SVD Toolkit]	MySQL Database for storing experimental data and outcome	Storing dynamic data, interconnection point
CFD-simulator (OpenFOAM)	GNU General Public License Version 2.0 MPL2 The MIT License Download: [SVD Toolkit]	Automated CFD-calculation environment for decision making in thermal management questions	Performing flow and temperature calculations
<i>COVISE</i>	<i>Dual license:</i> <ul style="list-style-type: none"> - <i>Academic license</i> - <i>Commercial license</i> 	<i>Scientific Simulation and Visualisation program</i>	<i>Steering of CFD simulation and visualisation of results</i>
<i>CFD solver (Ansys CFX)</i>	<i>Commercial Licence of ANSYS</i> www.ansys.com	<i>CFD solver for calculating heat-flow distribution field</i>	<i>Calculating heat-flow distribution to identify hot-spots</i>

SVN-Repository	http://subversion.apache.org/ Apache License 2.0	Software versioning and revision control system	Repository with input parameters required: DEBBs, Profiles, Workloads
DCworms	GPL License Download: [SVD Toolkit]	Simulator for workload and resource management policies	Creates boundary and initial values for CFD-simulation, delivers data for Metric Calculator and MOP-GUI
Metric calculator	Open Source Download: [SVD Toolkit]	Correlates energy consumption to work done	Evaluates experiment for energy efficiency
Application profiler	Open Source Download: [SVD Toolkit]	Simulation hardware requirements of different applications	Creates application profiles
CoolEmAll-Web-GUI	<i>Symfony2 Open Source PHP Web application</i> <i>MIT license</i> http://symfony.com/	<i>Web based php framework for integration of GUIs of SVD-Toolkit components</i>	<i>Web based GUI to SVD Toolkit</i>
Experiment Configurator GUI	<i>Symfony2 Open Source PHP Web application</i> <i>MIT license</i> http://symfony.com/	<i>GUI for ExperimentConfigurator</i>	<i>Configuration of Experiments and Trials for the simulation process</i>
DEBB Configurator GUI	<i>Open source license</i> <i>MIT license</i>	<i>GUI for selection, definition and design of DEBBs</i>	<i>GUI Definition of DEBBs</i>
DCworms-GUI	<i>Open source license,</i> <i>MIT license</i>	<i>GUI for DCworms</i>	<i>GUI for selecting workloads, workload and resource</i>

			<i>management policies and running DCworms simulations</i>
MOP-GUI	Apache License 2.0 for MOP-GUI website and Chart Generator, GNU GPL v3 for Vitral 3D Visualisation component	Web site presenting metrics in form of 3D visualisation and 2D line charts.	Metrics data visualisation
COVISE-GUI	<p><i>Misc licenses:</i></p> <ul style="list-style-type: none"> - <i>Apache CXF JavaScript Clients (v2.6.1) - Apache license</i> - <i>bootstrap (v2.3.2) Apache license</i> - <i>Jquery (v2.0.3) MIT license</i> - <i>log4Javascript (v1.4.6) Apache License</i> - <i>fabric (v1.2.0) http://fabricjs.com</i> - <i>bootstrap-slider (v2.0) Apache License</i> 	<i>Web based GUI to COVISE renderer, enabling interaction with CFD simulation and visualisation of results</i>	<i>Interaction and visualisation</i>
Modules for COVISE	<p><i>LGPL</i></p> <p>http://www.gnu.org/licenses/lgpl.html</p>	<i>Special COVISE modules enabling extraction of data from CFD solver results</i>	<i>Extracting and storing CFD data into database</i>
Report GUI	<p><i>Symfony2 Open Source PHP Web application</i></p> <p><i>MIT license</i></p> <p>http://symfony.com/</p>	<i>Web site presenting a summary of calculated metrics for a selected experiment</i>	<i>GUI for Metric-Calculator reports</i>

2.2 Detailed description of the components

This chapter gives a description of the individual components of SVD-Toolkit, based on the description provided in D2.2 [D2.2] and D2.4 [D2.4].

2.2.1 Application Profiler

For simulations in CoolEmAll, the focus is on power-, energy- and thermal-impact of decisions on the system. In order to have realistic simulations, a precise evaluation of resource consumption is necessary. The *Application Profiler* is used to create profiles of applications that can be read by DCworms for simulation purpose. It uses data obtained during runtime and stored in TIMaCS by the monitoring infrastructure. Using these data, it creates a description of applications based on their phases. For instance, an application following two phases (one CPU-intensive and one Network intensive) would have the following description:

```
<resourceConsumptionProfile>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT4S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>2</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT93S</duration>
    <behaviour name="cpu">
      <value>77</value>
    </behaviour>
    <behaviour name="network">
      <value>96</value>
    </behaviour>
  </resourceConsumption>
</resourceConsumptionProfile>
```

A more detail explanation is available in D2.3 [D2.3] and D5.4 [D5.4].

2.2.2 Repository

The repository is the central point in the SVD system architecture. It allows storing, editing and accessing files used by SVD-Toolkit components remotely, while ensuring their consistency. The repository contains:

- **Application-profiles**, describing resource usage of applications at different application phase
- **DEBBs**, describing data centre building blocks and models used by SVD-Toolkit

- **Workload-profiles**, workload characteristics in terms of used application-profiles and resource requirements used for workload simulation

For the realization of the repository we use Apache Subversion, short SVN [SVN]. The project repository is located at [CoolEmAll-SVN].

Table 2-3: Software dependency list for SVN

Software name	License / Website	Description
Apache Subversion	http://subversion.apache.org/ Apache License. Version 2.0	Subversion is an open source version control system.

The repository is structured in common and user spaces. Common space contains well-defined application-profiles, DEBBs and workload-profiles, each of them stored in a dedicated repository folder. User space contains files changed/added by each user. Files (particularly PLMXML files of DEBB) in user space can contain "links" to files in both spaces. Files in common space can contain links only to files in common space. The structure of repository is shown below:

```

repository
├── common
│   ├── applications
│   ├── workloads
│   └── debbs
└── users
  
```

The structure of DEBBs repository-folder is defined as follow:

```

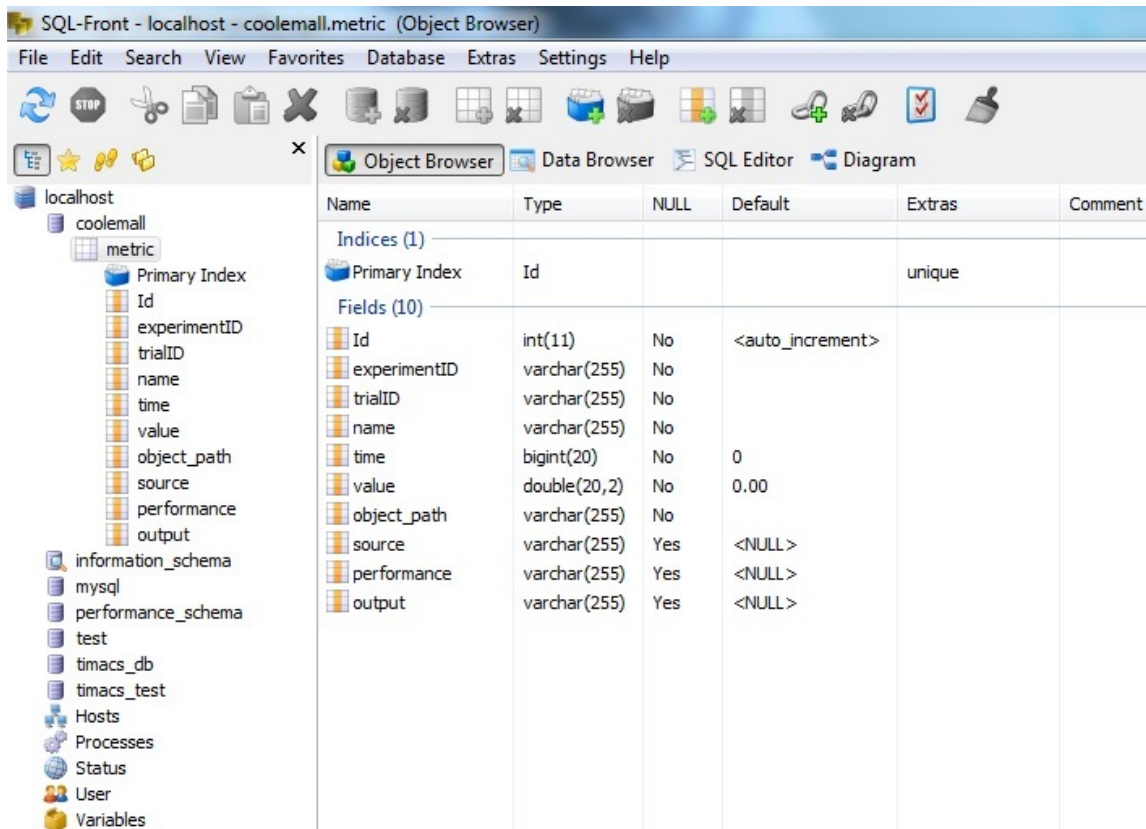
debbs
├── <location> (PSNC, HLRS, IRIT)
│   ├── [objects]
│   │   ├── <STL files>
│   │   ├── <VRML files>
│   ├── <mainPLMXML>.xml
│   └── <DEBBComponent_X>.xml
  
```

The DEBBS top-folder contains for each test bed site dedicated folder <location>, named according to location of the test bed: PSNC, HLRS, IRIT. The <location> folder contains DEBBS that are characteristic for particular test beds located at PSNC, HLRS and IRIT. Within the <location folder>, there is “objects” folder which contains geometrical objects of DEBB, in STL and VRML format. The main PLMXML file and DEBBComponent.xml files are located within the location folder.

2.2.3 Database

For saving simulations data a MySQL database has been designed. In this first version, the database contains the table “*metric*” with all collected information related to experiments and trials. In Figure 2-3 we can observe the fields of the table. For communication with the database we have created the following component:

- **Python Wrapper:** to insert and access the data in the MySQL database. Methods defined on the wrapper can be executed, both locally and remotely. For remote executing we have to use the stand-alone Remote Procedure Call (RPC) client available.



Name	Type	NULL	Default	Extras	Comment
Indices (1)					
Primary Index	Id			unique	
Fields (10)					
Id	int(11)	No	<auto_increment>		
experimentID	varchar(255)	No			
trialID	varchar(255)	No			
name	varchar(255)	No			
time	bigint(20)	No	0		
value	double(20,2)	No	0.00		
object_path	varchar(255)	No			
source	varchar(255)	Yes	<NULL>		
performance	varchar(255)	Yes	<NULL>		
output	varchar(255)	Yes	<NULL>		

Figure 2-3: Database Table Structure

The following table provides an overview on software and libraries used for implementation of the developed component.

Table 2-4: Software dependency list for Python Wrapper

Software name	License / Website	Description
Mysql 5.1	GPLv2 http://dev.mysql.com/	Used for creating the DB.
MySQLdb module	GPLv2, CNRI Python License, Zope Public License http://mysql-python.sourceforge.net/	MySQLdb is a thread-compatible interface to the MySQL database server that provides the Python database API.
Stand-alone RPC client	GNU Lesser General Public License www.timacs.de [MOP-package]	RPC based client allowing to insert and to retrieve data in DB.
Python 2.6	http://www.python.org/ Open Source, GPL compatible	Used by Python Wrapper

2.2.4 DCworms

Data Center Workload and Resource Management Simulator (DCworms) supports studies of dynamic states of IT infrastructures, like power consumption and air throughput distribution, with respect to the various workload and application profiles, resource models and energy-aware resource management policies. Details concerning DCworms can be found in D2.2 [D2.2] and in [DCworms2012].

As described in D2.2 and in D2.4 [D2.4], DCworms is the main component of workload simulation phase, which refers to the specific workload and application

characteristics as well as to the detailed resource parameters. Based on these models and taking into account applied resource management policy, DCworms is able to provide data including a distribution of power usage and air throughput for the models specified within the SVD-Toolkit. These values may be then analysed directly (using MOP-GUI) and/or provided as an input to the CFD-solver.

Referring to the functionality and characteristics presented in the aforementioned papers and reports ([D2.2], [D2.4] and [DCworms2012]), DCworms has been extended with several new features. Their aim is to support the user performing more comprehensive studies on energy-usage optimization.

Firstly, resource description can now be extended with load calendars that describe general load distribution for particular resources. For each resource, user can specify its utilisation levels within different periods of time. This information is then passed to DCworms and can be used by the user to estimate power consumption of the system. By these means, real-world statistics/measurements defining, for instance, background or initial load levels can be easily applied.

Moreover, resource description schema was modified in order to support definition of other, non-IT resources. In this way devices, like fans, cooling system, power supplies, etc. can be described. For each of them, the user can specify their detailed characteristics, including power profiles and power consumption models. These data can be then easily accessed by the user within the DCworms plugins and used, for instance in a scheduling process. DCworms provides also the implementation of power consumption models for cooling equipment following the ones presented in D2.3.1 [D2.3.1].

Finally, the outcome of the workload simulation phase was extended. Apart from a set of performance stats and a distribution of power usage and air throughput, a collection of energy efficiency metrics including PUE, productivity and energy waste level is calculated at the end of each simulation.

With respect to the information provided by D2.4, some new software dependencies come with the recent update of DCworms. Table 2-5 presents dependencies added since the ones specified in D2.4.

Table 2-5: Software dependency list for DCworms

Software name	License / Website	Description
Apache Xalan	Apache License, Version 2.0 / http://xml.apache.org/xalan-j/	An XSLT processor for transforming XML documents into HTML, text, or other XML document types
Guava	Apache License, Version 2.0 / http://code.google.com/p/guava-libraries/	Google's core libraries related to: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, etc.

2.2.5 Open-FOAM based CFD-Solver

In this section we describe Open-FOAM based CFD-Solver, used to simulate heat-flow simulation within the servers (RECS). The description provided here is originated from D2.4 [D2.4].

The CFD-Solver does the CFD-simulation and creates the flow field and values on which other components rely on. For its work it needs input from various other components. First it needs the geometry input from the DEBB in .PLMXML-format, retrieved from the DEBB repository. This is then transformed in a simulation region. Additionally there are boundary conditions and initial conditions needed. These values are supplied by the DCworms workload simulator and automatically retrieved from the MOP database. With these starting values the CFD-toolkit performs the flow and temperature simulation automatically. Therefore it first reads the relevant geometry files. These files are then meshed automatically and supplied to the CFD-calculation tool, which performs the CFD-calculation automatically. After the simulation has finished links to the flow and temperature field are stored in the central database and mean values for all relevant values, e.g. velocity and temperature for the interesting geometry, which are especially inlet and outlet are created and stored to the database.

To perform these calculations different tools of OpenFOAM and specifically developed software is used. The following table summarizes software used by CFD solver.

Table 2-6: Software dependency list for OpenFOAM based CFD

Software name	License	Description
CFD-simulator	GNU General Public License	Automated CFD-calculation environment for decision making in

	Version 2.0 MLP2 The MIT License	thermal management questions
OpenFOAM	GNU General Public Licence Version 2.0.	OpenFOAM is a free, open source CFD software package
Database	MySQL based implementation GNU Public License (GPL). Version 2	Data interchange, correct path specification
Eigen	MPL2	Library for linear algebra operation written in C++ programming language
RapidXML	The MIT License	Fast XML-Parser written in C++ programming language

At the beginning, the setup of the case is done by a script. Then the simulation environment is set up for “blockMesh”. This is done by parsing a XML-file and the rest of the setup for blockMesh is then done automatically. BlockMesh then creates a rectangular mesh. This is the basis for the work of “snappyHexMesh”. But before snappyHexMesh can start its work the .PLMXML-file needs to be parsed and the necessary transformations for the .STL-files, which are the mandatory geometry representations are made. These .STL-files have to be supplied to the toolkit by the user and need to represent all used geometry, especially inlets and outlets, individually. These geometry files are then used by snappyHexMesh to create the computational mesh. After the geometry is transformed into a computational mesh the boundary conditions are set up automatically by invoking the governing scripts and specially developed programs. For all different geometry representations this setup is performed individually. After these introducing steps the decomposition of the mesh is done and the actual calculation is performed in parallel mode to speed up the process. The solver to perform the calculation is “bouyantBoussinesqSimpleFoam”. It is capable of calculating incompressible flow for stationary conditions in conjunction with heat transfer. After the parallel solver has finished the decomposed computational mesh is reassembled and converted to EnSight and VTK format. EnSight-format is the preferred format for COVISE and VTK is the preferred format for Open Source applications such as ParaView [PaVi]. The next step is to calculate the mean values for the relevant geometry and store the data directly to the database. For the flow fields, only links are stored to the database to save

space inside the database. The utility to perform the final calculation is “swak4foam” and the governing script invokes it automatically.

Dependencies exist especially on the input site of the CFD-Solver. Here are to name the geometry files, which have to be .STL-files. These .STL-files need to be referenced correctly by the .PLMXML-file which represents the DEBB. To ensure consistency between all the invoked applications a naming convention was made.

A second very important convention is set up to find the data stored in the database. This is done by a convention for the data storage path in the database.

Another dependency is on the site of the boundary conditions. To create the correct boundary conditions the CFD-solver needs the output of the DCworms workload simulator in the units and for the correct values. The values monitored for this stage power and airflow is used.

For the output only one dependency is obvious. The data, which has to be stored in the database, needs to be put on the right place. Therefore the path where the input data is located is reused and the according data is added.

As noted, each functional surface, e.g. inlet and outlet has their own .STL-files and need to be referenced in .PLMXML-file. This is necessary for setting up the boundary conditions for CFD-simulation. For this purpose the next sections describe naming conventions used in .PLMXML-file and path.

2.2.5.1 Naming convention for PLMXML-file

For now this convention needs to be applied for the “ProductInstances”, as these are the parts, which matter for the CFD-simulation. This naming convention is used for the name of the ProductInstance, e.g. the name specified in the first line of the product instance.

As pointed out, the generation of the geometry data is extracted from the DEBB (main PLMXML file), containing references to geometric objects specified in .stl files, as described in D3.2 ([D3.2]). The geometric objects are composed of faces¹. There are four (4) significant faces for CFD, that are handled in simulation in different way:

- inlet (source of airflow)
- outlet (exhausting airflow)
- heatsink (source of heat)
- wall (surface reflecting the airflow)

¹ Sides of a geometric object are called faces.

For specification of the boundary patch, an inlet, name of ProductInstance-Element within the PLMXML file should consist of the keyword, specifying face-type (for inlet this keyword is “inlet”). Next there is a “\$” as a separator followed by the name of the corresponding geometry-object the according boundary patch belongs to.

< face-type>\$<object-name >

- **<face-type>** is element of {“inlet”, “outlet”, “heatsink”}, in case of absence of face-type, “wall” face-type is presumed.
- **<object-name>** is the name of the geometry-object and might contain “@”, that is converted to ‘/’ path-separator used to access object-path.

An example for this is: inlet\$RackNECWC_01@inlet_01, specifying face-type inlet, object RackNECWC_01 and its part inlet_01.

2.2.5.2 Path to data stored in database

The database stores different input parameters for CFD simulations (such as power and airflow), that belongs to particular surfaces of objects, used within simulation. In order to setup simulation with right parameters (boundary conditions) belonging to corresponding geometry-object, such as *airspeed* at “inlet” of a rack, these parameters are queried from the database using full object-path to particular geometrical object. The full object-path is built as a concatenation of all object-names in the hierarchy of PLMXML file:

<object-name of level1>/<object-name of level2>/<object-name of level3>/...

We always start out with the configuration of whole setup. We start with the name of the server room (level 1), followed by “/”. Next is the name of the rack (level 2), etc. This makes up the path to the important data stored for the CFD.

Inside this path there is the necessary data stored:

- Pressure p
- Temperature T
- Velocity U

Example: *HLRSServerroom/RackNECWC_01/inlet_1*

2.2.5.3 Orientation of velocity at inlet

The .STL-files used to define the geometry for CFD-simulation input need the following orientation convention.

The tessellation of .STL-file has to be done according to the right-hand-rule. The face normal vector, which results from this rule, has to point in direction of flow for the inlet.

2.2.6 COVISE and CFX based CFD Solver

For simulation of the heat-flow on room level, COVISE and Ansys CFX have been selected, as OpenFOAM based CFD calculation required too long time, making interactive usage of the CFD simulation impossible – a key-feature of the SVD-Toolkit.

As noted in D2.2 [D2.2], the Simulation Workflow COVISE (Collaborative Visualisation and Simulation Environment) is an extensible distributed software environment capable to integrate simulations, post-processing and visualization functionalities in a seamless manner. The CFD Solver performing CFD simulation is directly integrated into the COVISE workflow, including all necessary pre- and post-processing tasks. COVISE offers a networked SOAP based API and is accessible by all components that can make use of Web Service based components. In CoolEmAll, COVISE firstly retrieves simulation relevant data (step 4 presented in Figure 2-2) from the DEBB repository (containing geometry data and position of objects) and from the Database (containing results from DCworms, i.e., power usage and air throughput), passes over these data to the CFD Solver, receives results from the CFD Solver, post processes and visualizes simulation results allowing at the same time modification of certain parameters such as the arrangement of objects. Results of the simulation are written back into the Database (step 8), while modified geometrical parameters and arrangement of objects are used to update DEBBs (step 8), to be stored in the DEBB repository (step 12). Using COVISE, users can analyse their datasets intuitively and interactively in a fully immersive environment through state of the art visualization techniques, including volume and fast sphere rendering.

The Computational Fluid Dynamics (CFD) Solver is directly integrated into the COVISE workflow and enables to simulate and analyse complex heat flow and dissipation processes, and their consequences on flow guiding structures, such as compute-building blocks (DEBBs) in data centres. For this purpose a heat flow model defined by partial differential equations is defined. CFD solvers are using this model to calculate and simulate the interaction of liquids and gases with surfaces defined by boundary conditions of DEBB's geometry and other parameters. The results of a simulation, a heat-flow distribution map, are passed over to Simulation Workflow/COVISE and can be visualised using COVISE GUI integrated into CoolEmAll-Web-GUI. In addition, the temperature and airflow on inlet/outlets of the building blocks are extracted from the heat-flow distribution map and stored in the Database, ready to be used by metrics calculator to calculate assessment metrics.

The COVISE licence and its modules are described on the corresponding web-page <http://www.hlrs.de/organization/av/vis/covise/> and can be downloaded from <http://www.hlrs.de/covise/support/download>. COVISE is available for various platforms including: Linux (Fedora, SUSE; Red Hat, Ubuntu), Mac OS, and Windows (ia 32 and x64). Since version 4.5 COVISE will not run without a license anymore. To get a permanent license please contact your project partner

or Uwe Wössner (woessner@hlrs.de). COVISE with a demo license can be downloaded also from the CoolEmAll web-site [SVD-Toolkit].

ANSYS CFX software is a product of ANSYS and is a high-performance, general-purpose fluid dynamics program solver capable to solve CFD problems very fast. The ANSYS CFX solver can be obtained from the <http://www.ansys.com> product page.

2.2.7 Metric Calculator

As described in D2.2, the Metric Calculator is responsible for the assessment of the simulation results. Based on metrics identified and defined in D5.1, it assesses energy-efficiency and heat-efficiency of building blocks (DEBBs). The calculation itself is based on data/metrics that are retrieved from the Database. Results of the calculation are written back into the Database, to be retrieved and visualized by MOP GUI and the Report GUI.

The Metric Calculator is a Python command line application that can be called with many different parameters depending of the calculations performed. The calculation is based on metrics retrieved from the database.

The focus here is on hardware metrics available at node, node-group, rack and room level.

While the first prototype was able to calculate 6 metrics, the current implementation of the metric calculator for the final prototype allows us to calculate these 22 following metrics:

Hardware level metrics:

- CPU usage (minimum, maximum, average)
- Server usage (minimum, maximum, average)
- Memory usage (minimum, maximum, average)
- Power usage (minimum, maximum, average)
- Power consumption (minimum, maximum, average)
- Power IT consumption (minimum, maximum, average)
- Temperature (minimum, maximum, average)
- CPU Temperature (minimum, maximum, average)
- Energy consumption
- Productivity
- SWAP
- Cooling index low and high (minimum, maximum, average)
- Heat generation (minimum, maximum, average)
- DH-UR
- Imbalance of temperature (minimum, maximum, average)
- Imbalance of heat generation (minimum, maximum, average)
- Power Usage Effectiveness (3 and 4)
- Data Centre Infrastructure Efficiency

- Electricity cost
- Carbon emissions
- CAPEX
- Energy Wasted Ratio

Depending of the situation, the Metric Calculator will use one of the following databases:

- When calculating metrics for a real world experiment, data will be read from and stored into the TIMaCS database.
- When calculating metrics for a simulation, data will be read from and stored into the SVD database.
- When generating a report, previously calculated metrics will be read from the TIMaCS or SVD database depending of the type of experiment and transferred into the Web-GUI database.

Table 2-7: Software dependency list for Metric Calculator

Software name	License / Website	Description
Stand-alone RPC client	GNU Lesser General Public License www.timacs.de [MOP-package]	RPC based client allowing to insert and retrieve data from TIMaCS database
Python 2.6	Open Source, GPL compatible, Python Software Foundation License Version 2	All scripts are written in Python

2.3 Detailed description of the CoolEmAll-Web-GUI

The CoolEmAll-WEB-GUI provides integrated web based graphical user interface allowing interacting with the SVD-Toolkit and visualizing its results. It comprises several GUIs integrated into common web based GUI environment each of them capable of interacting with the corresponding SVD-Toolkit components through the tab-page.

After entering user-name and password via login page (as shown in Figure 2-4 below) user is redirected to the welcome-page Figure 2-5, which shortly describes the components of the SVD-Toolkit.

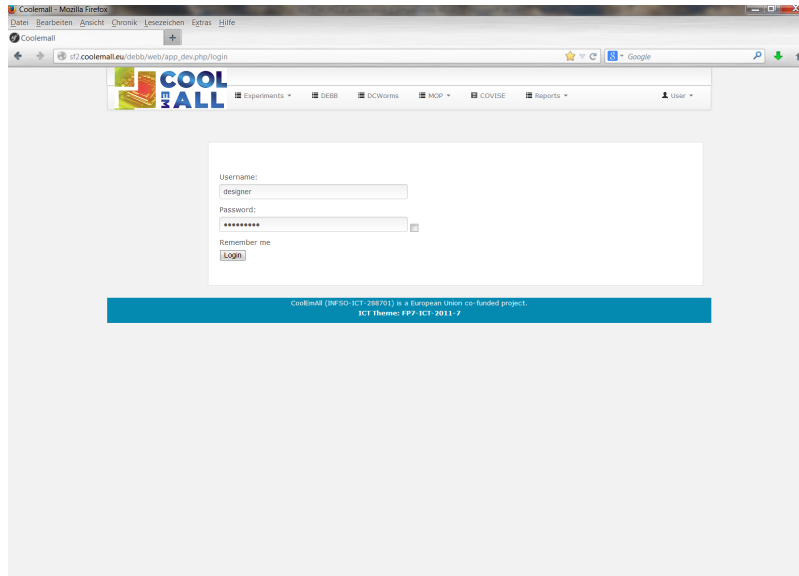


Figure 2-4: Login page

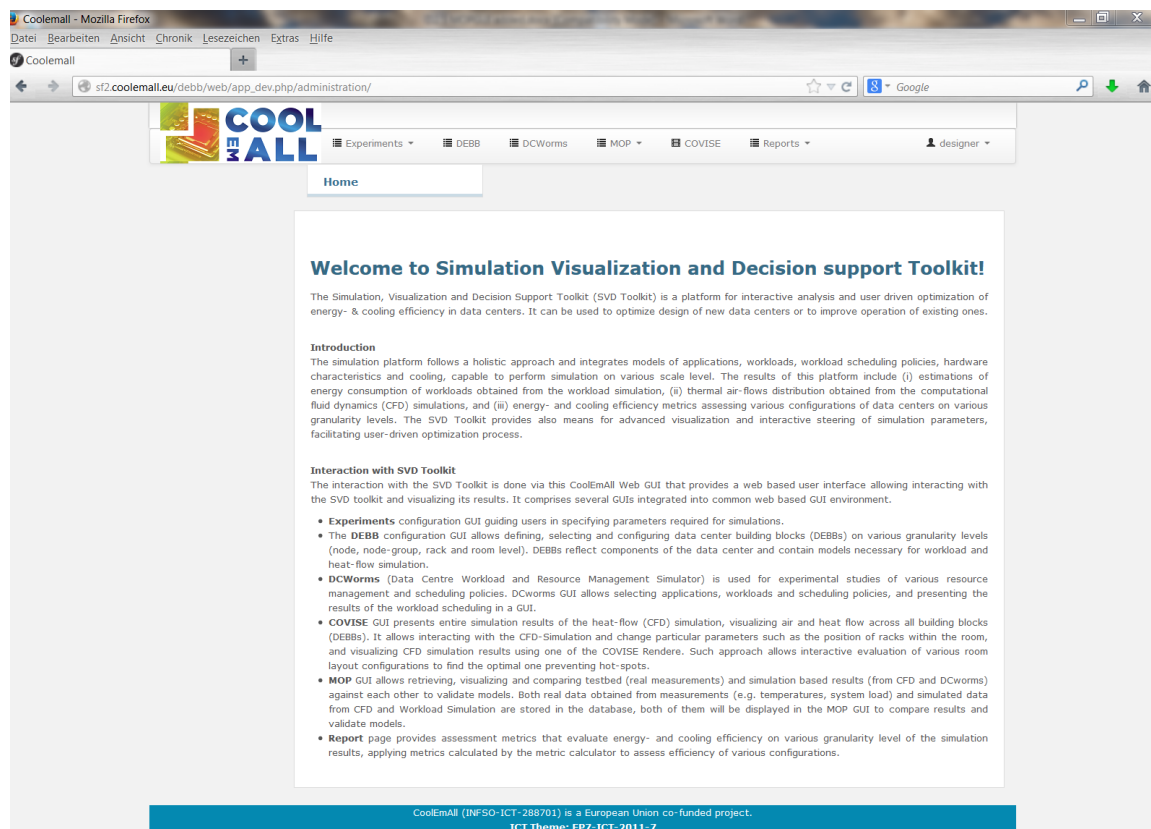


Figure 2-5: Welcome page

The interaction with the SVD-Toolkit is done via this CoolEmAll-Web-GUI that provides a web based user interface allowing interacting with the SVD-Toolkit

and visualizing its results. It comprises several GUIs integrated into the common web based GUI environment.

- **Experiment configuration** GUI guiding users in specifying parameters required for simulations.
- The **DEBB** configuration GUI allows defining, selecting and configuring data centre building blocks (DEBBs) on various granularity levels (node, node-group, rack and room level). DEBBs reflect components of the data centre and contain models necessary for workload and heat-flow simulation.
- **DCworms** (Data Center Workload and Resource Management Simulator) GUI is used for experimental studies of various resource management and scheduling policies. DCworms GUI allows selecting applications, workloads and scheduling policies, and presenting the results of the workload scheduling in a GUI.
- **COVISE** GUI presents entire simulation results of the heat-flow (CFD) simulation, visualizing air and heat flow across all building blocks (DEBBs). It allows interacting with the CFD-Simulation and change particular parameters such as the position of racks within the room, and visualizing CFD simulation results using one of the COVISE Renderers. Such an approach allows interactive evaluation of various room layout configurations to find the optimal one preventing hot spots.
- **MOP** GUI allows retrieving, visualising and comparing testbed (real measurements) and simulation-based results (from CFD and DCworms) against each other to validate models. Both real data obtained from measurements (e.g. temperatures, system load) and simulated data from CFD and Workload Simulation are stored in the database, both of them will be displayed in the MOP GUI to compare results and validate models.
- **Report** page provides assessment metrics that evaluate energy- and cooling efficiency on various granularity levels of the simulation results, applying metrics calculated by the metric calculator to assess efficiency of various configurations.
- **User**: additionally to the SVD-Toolkit GUIs there is also the menu User interface where there are options to register users on the CoolEmAll-Web-GUI, edit the profile of a particular user and the option to logout of the system when the user is logged in.

These are explained in the following sub-sections.

2.3.1 ExperimentConfigurator GUI

This is the main entry page for definition and selection of the experiment that will be executed on the SVD-Toolkit, controlled by the GUIs of the CoolEmAll-Web-GUI. On the CoolEmAll-Web-GUI the user has to select the option “Experiments” on the top menu where he/she can choose the list of all the experiments or create a new experiment, as we can see on Figure 2-6. The

ExperimentConfigurator allows the user to specify parameters required to execute the simulation through the definition of experiments and trials. The others GUIs can access the information saved on the ExperimentConfigurator via session variable.



Figure 2-6: ExperimentConfigurator–GUI Menu

Once the user chooses the option “List” the Experiment configuration GUI-page will be presented (as shown on Figure 2-7) in which the user can select different options related with the experiment and have an overview of all the experiments saved on the database.

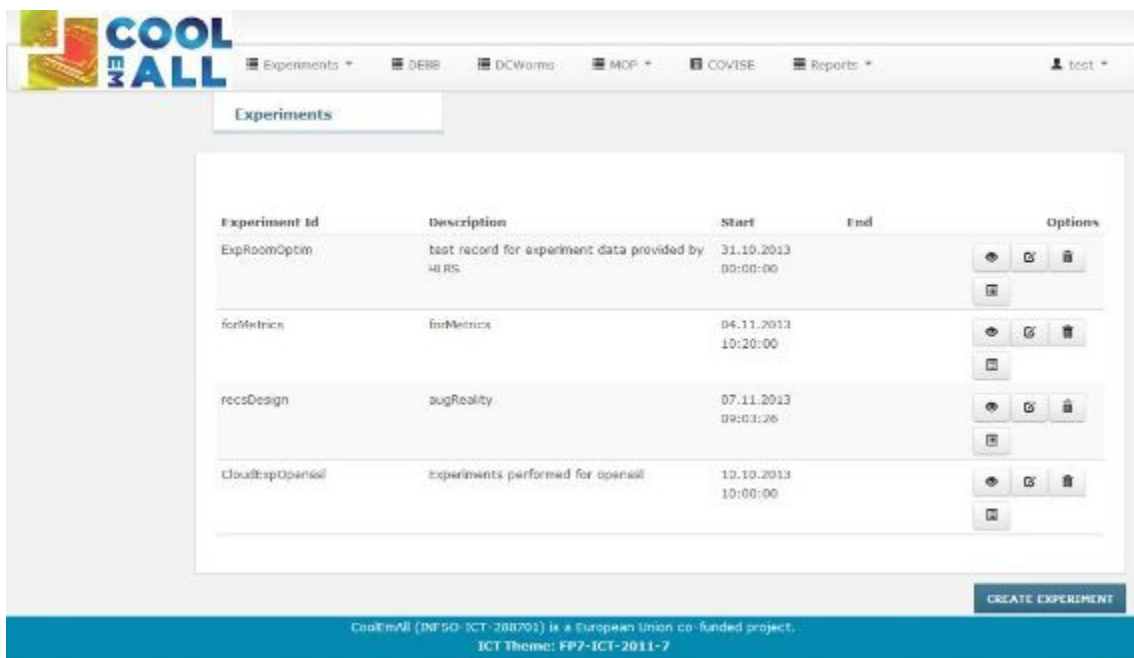


Figure 2-7: Experiment configuration GUI

When the user is creating a new experiment the information that he/she must provide on the interface presented on Figure 2-8, as required, is:

- Experiment ID: identifier of the experiment.
- Type: type of experiment, which can be DCworms, CFD, Testbed or All.
- Start: the timestamp when the experiment starts.

Additionally, as optional information, the user can provide:

- Description: short description for the experiment.
- End: the timestamp when the experiment finishes.

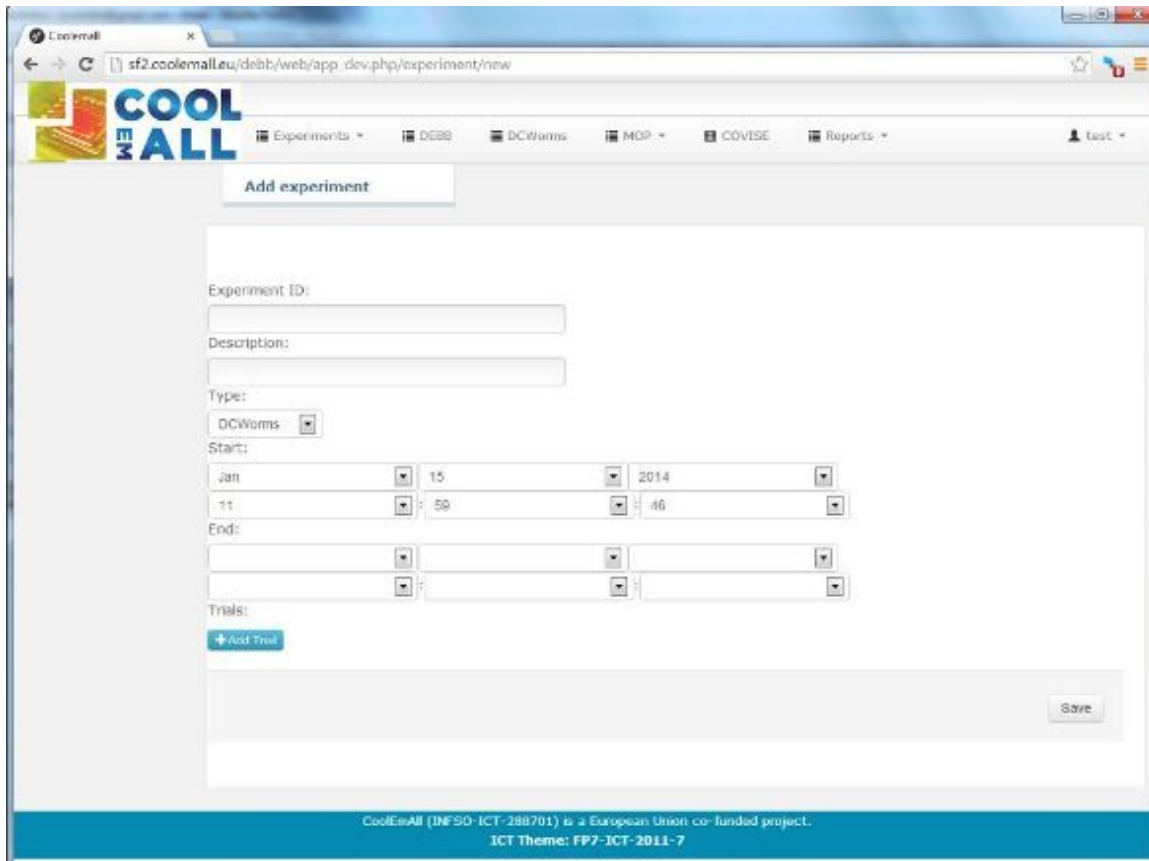


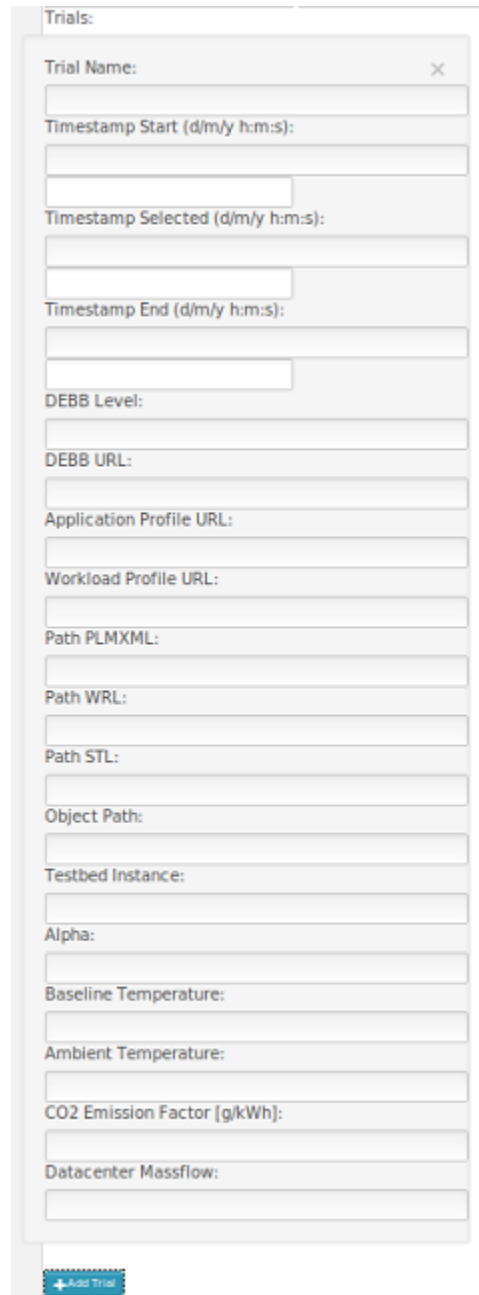
Figure 2-8: New experiment interface

In the same interface the user should define the trials to the experiment. The user has to select the option “Add Trial” whereby a new interface will be shown, see Figure 2-9, and fills the required fields. The required information for the trial is:

- Trial name: name of the trial.
- Start: the timestamp when the trial starts on the format (d/m/y h:m:s).
- DEBB level: level on the DEBB in which the experiment is performed.

Additional fields that can be filled are:

- Timestamp Selected: this is a timestamp selected for CFD simulation on the format (d/m/y h:m:s).



Trials:

Trial Name:

Timestamp Start (d/m/y h:m:s):

Timestamp Selected (d/m/y h:m:s):

Timestamp End (d/m/y h:m:s):

DEBB Level:

DEBB URL:

Application Profile URL:

Workload Profile URL:

Path PLXML:

Path WRL:

Path STL:

Object Path:

Testbed Instance:

Alpha:

Baseline Temperature:

Ambient Temperature:

CO2 Emission Factor [g/kWh]:

Datacenter Massflow:

Figure 2-9: Trial information

- Timestamp End: the timestamp when the trial finishes on the format (d/m/y h:m:s).
- DEBB URL: URL to the svn repository where the DEBB files are saved.
- Application Profile URL: URL to the svn repository where the Application profile files are saved.
- Workload Profile URL: URL to the svn repository where the Workload

- profile files are saved.
- Path PLMXML: Path on the svn repository where the PLMXML file is saved.
 - Path WRL: Path on the svn repository where the WRL file is saved.
 - Path STL: Path on the svn repository where the STL file is saved.
 - Object Path: Path of the metrics that are saved on the database, for the CFD simulation.
 - Testbed Instance: instance on the test bed in which the trial is performed.
 - Alpha: alpha defines the factor for calculations of power usage.
 - Baseline Temperature: operating temperature of the data centre / temperature with which air is supplied.
 - Ambient Temperature: temperature outside the data centre.
 - CO2 Emission Factor: CO2 emission factor measured in g/kWh.
 - Datacentre Massflow: datacentre massflow of the server room.

All the optional fields, both for experiment and trial, can be updated for the other GUIs or as result of the simulation process.

2.3.2 DEBBConfigurator GUI

The DEBBConfigurator GUI makes it possible for the hardware vendor and the end-user to easily create a valid DEBB without using complex XML Editors. The DEBB can then be exported to the SVN repository for other CoolEmAll-tools to be used, or downloaded as a ZIP File that contains different files:

- DEBB description file (.plmxml)
- DEBB component file (.xml)
- Pictures (.jpg, etc.)
- 3D models (.stl, .wrl)

In the CoolEmAll-Web-GUI the DEBBConfigurator is integrated as a Symphony2 “plugin”.

With the DEBBConfigurator the hardware vendor can define the hardware that is available for purchase. The process has a hierarchy-like structure. Meaning you first define the components, then build nodes with these components. And put these nodes in node groups, which are then put in racks, which are placed in a room in the end of the process. This can be done by adding the components (baseboard, processor, memory, heatsink, chassis, etc.) on the component level of the process. Infrastructure components such as cooling devices, power profiles, networks and flow pumps can also be defined on this level by the vendor. Several boundary conditions for the hardware can be set. This is important for later simulation processes or to determine the overall power usage. Furthermore sensors can be specified to give input during simulation, or afterwards. These Sensors can capture several data values like temperature, voltage, throughput, etc.

Therefore the vendor just clicks on the plus symbol next to the processors menu for example, fills the form with a product name, manufacture, clock speed, power usage, costs and other information and saves the component part. In addition it is possible to set several power usages for different workloads of the CPU for heat dissipation over the heatsink with a defined transfer rate (as set in the heatsink part) in a simulation.

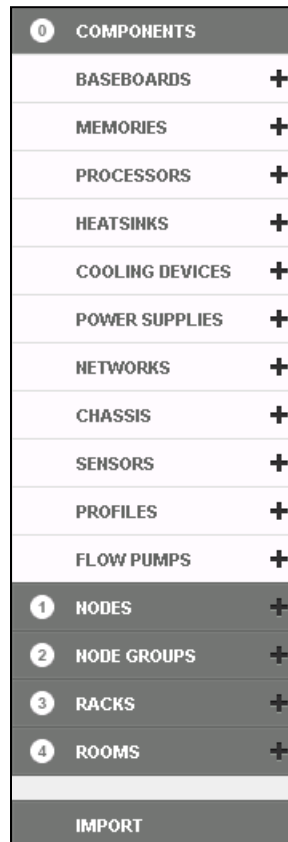


Figure 2-10: DEBB-menu

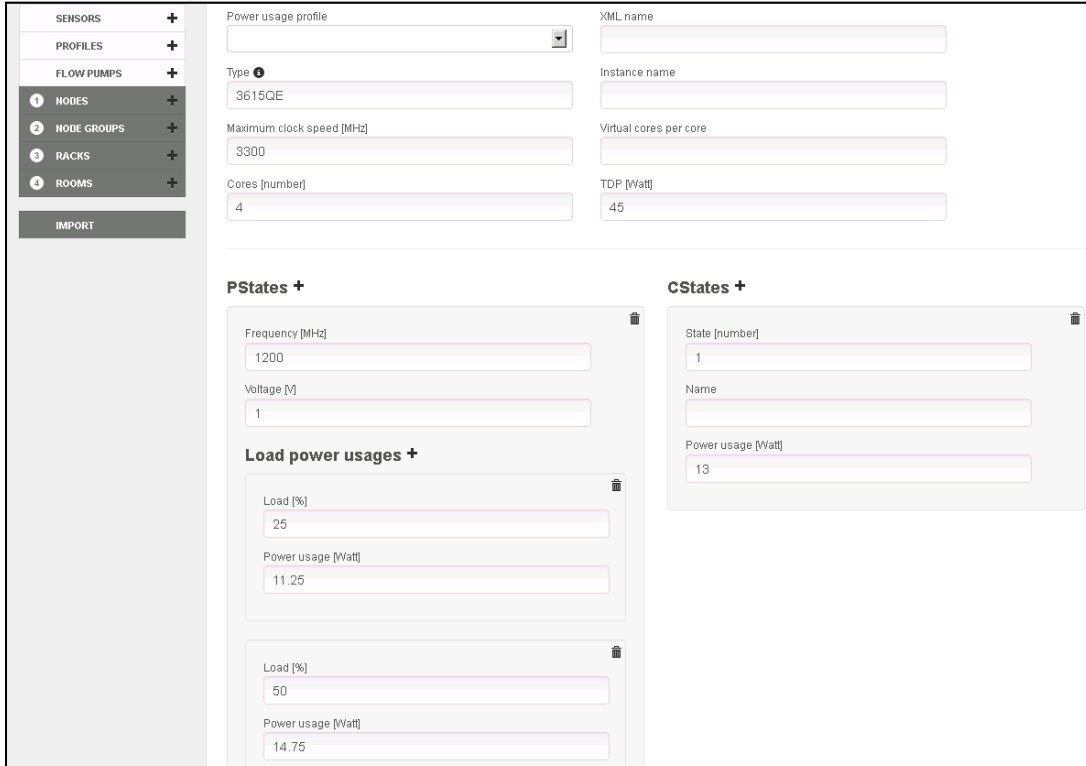


Figure 2-11: DEBBConfigurator GUI – Processor tab (components level)

General information in every components-creation-menu:

- Component id
- Manufacturer (mandatory)
- Product (mandatory)
- Label
- Costs [EUR]
- Costs (emission of CO₂ during the production and transportation of the part) [kg CO₂]
- Hostname
- Maximum power usage [W]
- Power usage profile (can be defined in the profiles tab)
- XML name
- Type
- Instance name

There are some special attributes that can be set for some parts:

Memory

- Capacity (mandatory) [MB]
- Interface

Processor

- Maximum clock speed (mandatory) [MHz]
- Cores [number]
- Virtual cores per core [number]
- TDP (Thermal Design Power) [W]
- PStates (diverse states can be created. e.g. 1000MHz, 800MHz)
 - o Frequency [Mhz]
 - o Voltage[V]
 - o Load power usages (diverse states can be created. e.g. 25%, 50%)
 - Load [%]
 - Power usage [W]
- CStates (diverse states can be created)
 - o State [number]
 - o Name
 - o Power usage [W]

Heatsink

- Transform (parameter for simulation)
- Transfer rate [NTU] (efficiency of energy transfer)
- Upload model files (for simulation an visual presentation)

Cooling device

- Class (fan, heatpipe, refrigeration, ILC, LCU, CRAH, HVAC)
- Maximum cooling capacity
- Cooling capacity rated
- Energy efficiency ratio
 - o LWT (water temperature entering the chiller)
 - o CWT (air temperature entering the condenser)
 - o Capacity
 - o Power usage [W]
 - o EER

Power supply

- Class (PSU, UPS, PDU, MVLVTransformer)
- Total output power (mandatory) [W]
- Typical efficiency (mandatory) [%]
- Power profile (can be defined in the profiles tab)

Network

- Interface (Physical Interface description like fibre, twisted pair, etc.)
- Technology
- Max bandwidth [bit/s]

Chassis

- Height [RU]
- View (top, front)
- Chassis image upload (shown as background-image when creating a node group with that chassis)

- Model file upload
- The option to add flow pumps
- The option to add nodes
- An area where nodes and flow pumps can be dragged into place, and where the size of the chassis is determined

Sensor

- Class (temperature, voltage, power, humidity, throughput, velocity)
- Unit (for the class like: °C, V, mW, kW, etc.)
- Min value
- Max value
- Factor (multiplier between the current unit and basic unit)
- Accuracy
- Input or not (Input is a flag describing that a sensors is a input value for the simulation or not. For example heat sources can be seen as sources without any output afterwards. Other sensors might be added for extracting results at the end of the simulation)

Flow pump

- Width [m]
- Height [m]
- Depth [m]
- Max rpm
- Typical efficiency [%]
- Mode (outlet, inlet)
- Transform (parameter for simulation)
- Upload model files (for simulation an visual presentation)

Profiles can be chosen in every components creation tab as power usage profile. They can have several states with different energy consumptions. So the profiles can be used for many hardware items, but must only be created once. They only have the following input boxes:

- Name (mandatory)
- Type
- Flow state (diverse states can be defined. e.g. off and 100% on)
 - o State (at least one is mandatory)
 - o Flow
 - o Power usage [W]
 - o Description
 - o Typical efficiency [%]

On the node level the vendor assigns the specific hardware to a node that will be choose able for the user. During the node creation the vendor can choose which CPU, baseboard, memory and other hardware should be part of a node. And for

a 3D visual presentation Model files (max 128MB per file) of the components are uploaded via the DEBB-GUI.

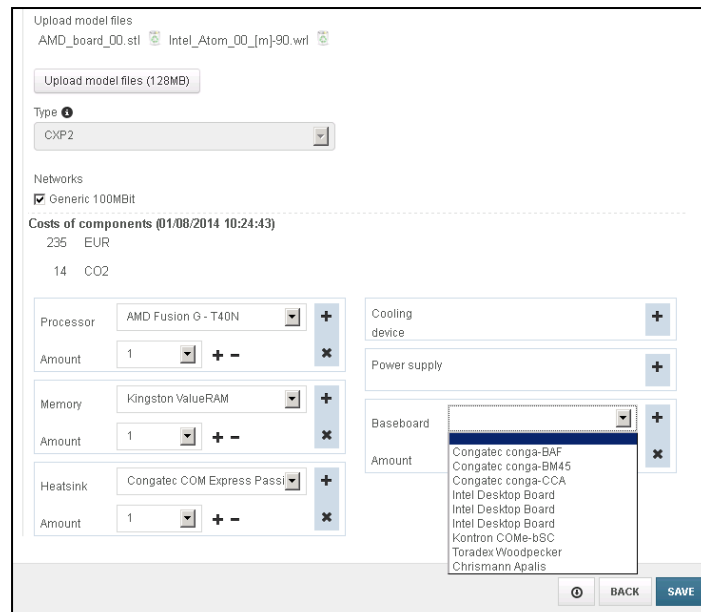


Figure 2-12: DEBBConfigurator GUI – Node creation

During the node generation these fields can be filled:

- Upload image
- Upload model files
- Component id
- Manufacturer (mandatory)
- Product (mandatory)
- Label
- Costs [EUR]
- Costs (emission of CO₂ during the production and transportation of the part) [kg CO₂]
- Hostname
- Maximum power usage [W]
- Power usage profile (components level profile)
- Instance name
- Width [m]
- Height [m]
- Depth [m]
- Mesh resolution (parameter for simulation)
- Location in mesh (parameter for simulation)
- Type (used to tell if that part can later be built into a chassis with a specific interface like: CXP2, PS)
- Networks (more than one can be chosen)

- Costs of components are displayed in [EUR] and [kg CO₂]
- Hardware component parts
 - o Processor
 - o Baseboard
 - o Memory
 - o Cooling device
 - o Heatsink
 - o Power supply

From node groups creation over racks to rooms, the end-user can then choose from a variety of hardware, chassis and racks he wants to have in his/her server room. At the end the costs are displayed, and the configuration can then be downloaded as a ZIP file or can be directly exported to SVN repository to be used for simulation or a visual impression of the server room.

First the user can select a chassis in the node groups tab (node group level) and set which and where the nodes should be located in the chassis. This is done by selecting a predefined node-slot and then adopting a proper node to that slot.

This information can be entered in the node creation tab:

- Component id
- Manufacturer (mandatory)
- Product (mandatory)
- Label
- Hostname
- Instance name
- Chassis (dropdown menu with the created chassis)
- Type
- Maximum power usage [W]
- Power usage profile (dropdown menu with the created power profiles)
- Mesh resolution (parameter for simulation)
- Location in mesh (parameter for simulation)
- XML name
- Costs [EUR]
- Costs (emission of CO₂ during the production and transportation of the part) [kg CO₂]
- Networks (more than one can be chosen)
- Select node

In the node selection only the proper type for the selected node can be chosen, e.g. only COM Express modules for a CXP2 type node. The unavailable options are disabled (see below).

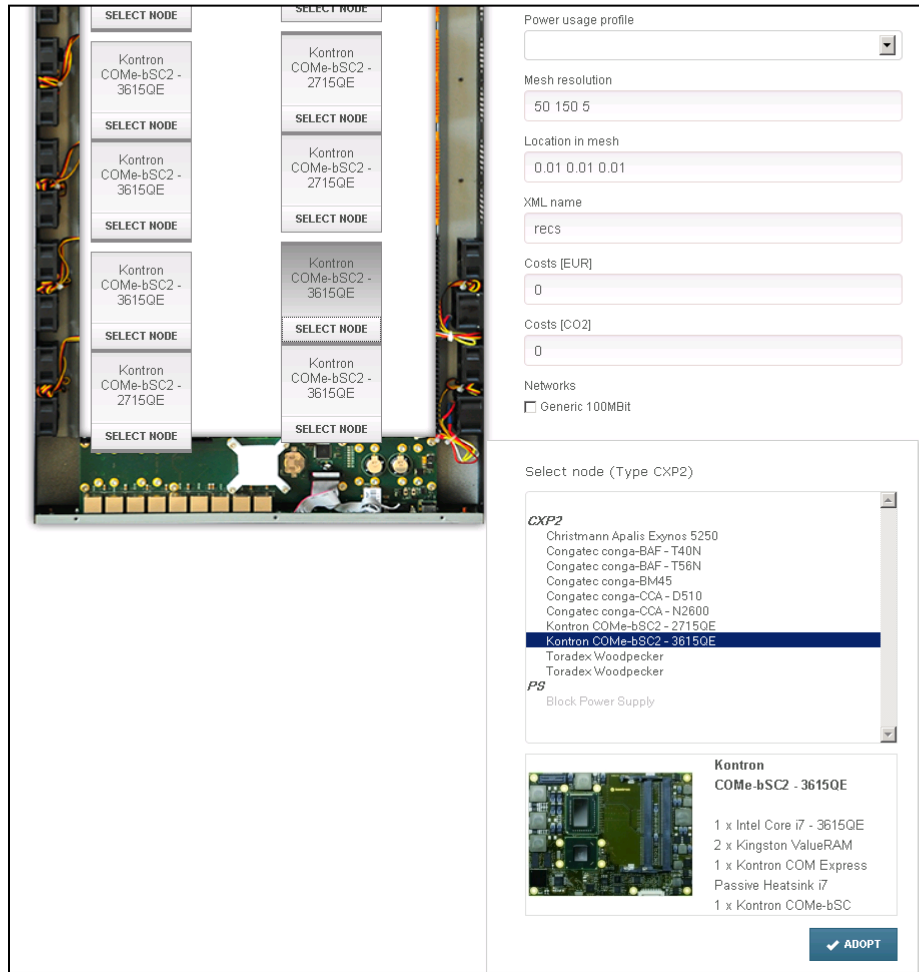


Figure 2-13: DEBB configuration GUI – Node group creation

After that the user can choose his/her preferred rack (rack level) and place the created node groups in the cabinet by clicking an appropriate rack-unit and picking a node group.

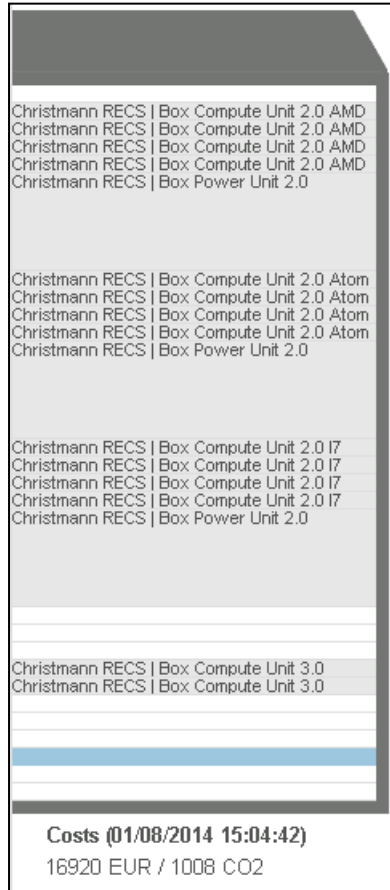


Figure 2-14: DEBBConfigurator GUI – Rack view with costs

Rack configuration fields:

- Component id
- Manufacturer (mandatory)
- Product (mandatory)
- Label
- Costs [EUR]
- Costs (emission of CO₂ during the production and transportation of the part) [kg CO₂]
- Hostname
- XML name
- Power usage profile (dropdown menu with the created power profiles)
- Type
- Instance name
- Width [m]
- Height [m]
- Depth [m]
- Gap bottom [m] (used for the XML file to position the chassis in the rack)
- Gap left (used for the XML file to position the chassis in the rack)
- Gap front (used for the XML file to position the chassis in the rack)

- Rack size [RU] (number of height-units the rack can hold)
- Mesh resolution (parameter for simulation)
- Location in mesh (parameter for simulation)
- Upload model files
- Node Group

The Node Group dropdown menu is used to select the node group that will be in the selected rack unit. It also shows the remaining free rack units.

With the DEBB-GUI it is possible to create a unique server room (room level) with a user-defined combination of nodes in a chassis, different chassis with different node configurations in a rack, and with different racks in a room. Racks or flow pumps can be added by clicking on the corresponding button. The user can determine how a rack is orientated and positioned in the room, just via drag'n'drop. Scaling the blueprint of the room with the mouse can change the room size. For customization any room layout can be uploaded as 3D model file.

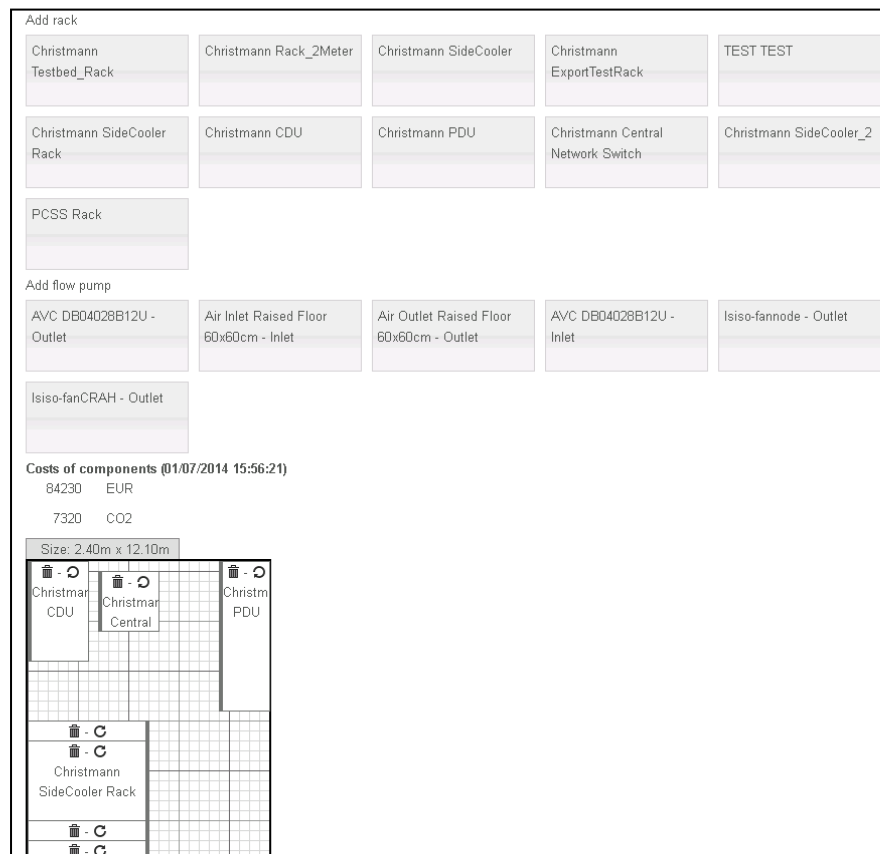


Figure 2-15: DEBB configuration GUI – server room

Room organization fields:

- Component id
- Name (mandatory)
- Building (mandatory)
- Height [m]
- XML name
- Mesh resolution (parameter for simulation)
- Location in mesh (parameter for simulation)
- Costs [EUR]
- Costs (emission of CO₂ during the production and transportation of the part) [kg CO₂]
- Upload model files
- Cooling devices (more than one can be chosen)
- Add rack / Add flow pump (all previously created racks and flow pumps can be used and positioned in the room)
- Costs of components (the cumulated costs of all components is shown in [EUR] and [kg CO₂])
- Blueprint of the server room (for easy positioning of components in the room)

With the import button in the bottom of the navigation menu it is possible to import a saved ZIP file. This file should have the same structure as the downloadable ones as shown below.

The downloadable ZIP holds the following files:

```

ZIP archive
| CoolingDevice_name_20
| Costs.xml
| FlowPump_name_8
| Heatsink_name_5
| Node_name_1.xml
| NodeGroup_name_1.xml
| PMLXML_roomname.xml
| rack_name_1.xml
| room_roomname_1.xml
|
+---objects
|   | X***_node1.stl
|   | X***_node1.wrl
|   \ X***_node2.stl
|
\---pics
   | node1_3.jpeg
   \ node2_8.jpeg
  
```

In the main folder the validated XML files are stored. The Costs.xml contains a list of the cost of every component in the room. In the PLMXML file in the main folder, the coordinates for the 3D models in the room are stored for assembling the whole room with all components. The component XML files holds information that was supplied by the hardware vendor and which STL and WRL models belong to the node.

The STL files stored in the “objects” folder are used for CFD simulation in the room. While the WRL files are used to display the objects in the room with colors in COVISE or any other capable program. Reference to these files is given in the PLMXML and the components XML file in the main folder.

When uploaded during the node-creation the pictures showing the nodes are saved in the “pic” folder by their node name.

The exported ZIP file and its content can also be used by other tools described in this deliverable. Hence it is possible for the user to optimize a room by just dragging the racks to another position in the room and simulate the new array to see what has improved.

A more detailed description of the features of the DEBBConfigurator can be found in the deliverable 3.5.

2.3.3 DCworms GUI

DCworms is provided as a part of SVD-Toolkit. For this reason it accepts and generates files in appropriate formats understood by other SVD-Toolkit components. In general, performing experiments with the DCworms requires establishing the simulation environment properties first. They include specification of input files, such as resource and workload/application models as well as a definition of workload and resource management policy. DCworms offers an intuitive GUI, which guides user through this stage. It provides means to navigate through workload and application profiles repositories and to choose the ones that will be used during the simulation phase. Moreover, DCworms GUI allows browsing a list of available workload and resource management policies that can easily be exchanged between different experiments. The following policies can be applied:

Table 2-8: Workload and resource management policies available within DCworms

Name	Level	Description
Rack_FCFS_LoadBalancing	Rack	Tasks are scheduled in FCFS order and assigned to Nodes in the way that balance the load
Rack_FCFS_ConsolidationHighPerf	Rack	Tasks are scheduled in FCFS order and assigned to Nodes, starting from high performance CPUs, to consolidate the load on the Nodes

Rack_FCFS_ConsolidationHighPerf_NodePowMan	Rack	Tasks are scheduled in FCFS order and assigned to Nodes, starting from high performance CPUs, to consolidate the load on the Nodes. Unused nodes are switched off and turn on if needed
Rack_FCFS_ConsolidationLowPower	Rack	Tasks are scheduled in FCFS order and assigned to Nodes, starting from low power CPUs, to consolidate the load on the Nodes.
Rack_FCFS_ConsolidationLowPower_NodePowMan	Rack	Tasks are scheduled in FCFS order and assigned to Nodes, starting from low power CPUs, to consolidate the load on the Nodes. Unused nodes are switched off and turned on if needed
Room_FCFS_LoadBalancing	Room	Tasks are scheduled in FCFS order and assigned to Rack and then to Nodes in the way that balances the load
Room_FCFS_ConsolidationHighPerf	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from high performance CPUs, to consolidate the load on Racks and then on Nodes
Room_FCFS_ConsolidationHighPerf_NodePowMan	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from high performance CPUs, to consolidate the load on Racks and then on Nodes. Unused nodes are switched off and turned on if needed
Room_ConsolidationLowPower	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from low power CPUs, to consolidate the load on the Racks and then on Nodes.
Room_ConsolidationLowPower_NodePowMan	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from low power CPUs, to consolidate the load on the Racks and then on Nodes. Unused nodes are switched off and turned on if needed

Room_FCFS_LoadBalancing_PowerCapping	Room	Tasks are scheduled in FCFS order and assigned to Rack and then to Nodes in the way that balances the load. For racks
--------------------------------------	------	---

		exceeding the power limit, a power capping strategy is applied.
Room_FCFS_ConsolidationHighPerf_PowerCapping	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from high performance CPUs, to consolidate the load on Racks and then on Nodes. For racks exceeding the power limit, a power capping strategy is applied.
Room_ConsolidationLowPower_PowerCapping	Room	Tasks are scheduled in FCFS order and assigned to Nodes, starting from low power CPUs, to consolidate the load on the Racks and then on Nodes. For racks exceeding the power limit, a power capping strategy is applied.

More details concerning available policies can be found in [D4.3] and [D4.6].

Each selection panel is supplemented by a short characteristic of the selected model. Figure 2-16 shows the main window of DCworms GUI.

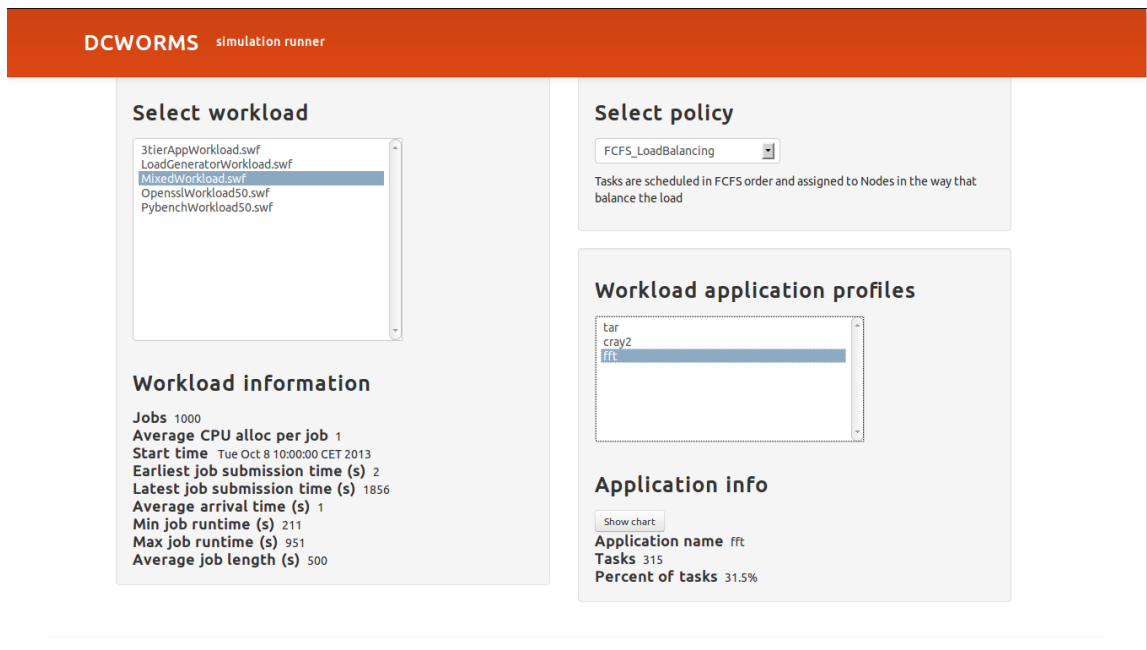


Figure 2-16: DCworms GUI

Additionally, DCworms GUI supports definition of load calendars (described also in Section 2.2.4) that determine the initial utilization of resources. For the given DEBB level, the user is able to specify the load distribution according to various statistical parameters.

The output of each simulation consists of a set of statistics that are written into the database and can be later viewed using the MOP GUI (where they can be seen in a graphical way), processed by the Metric Calculator or used for the purpose of CFD analysis. Moreover, in order to enable quick insight into an effectiveness of the selected policy, DCworms GUI also displays a set of energy efficiency metrics.

2.3.4 COVISE GUI

As noted, the COVISE GUI provides user web based access to COVISE, allowing interacting with the simulation by changing parameters such as rearranging positions of racks represented by shapes within the room and visualizing results of the simulation. Figure 2-17 provides overview of the web based GUI to COVISE.

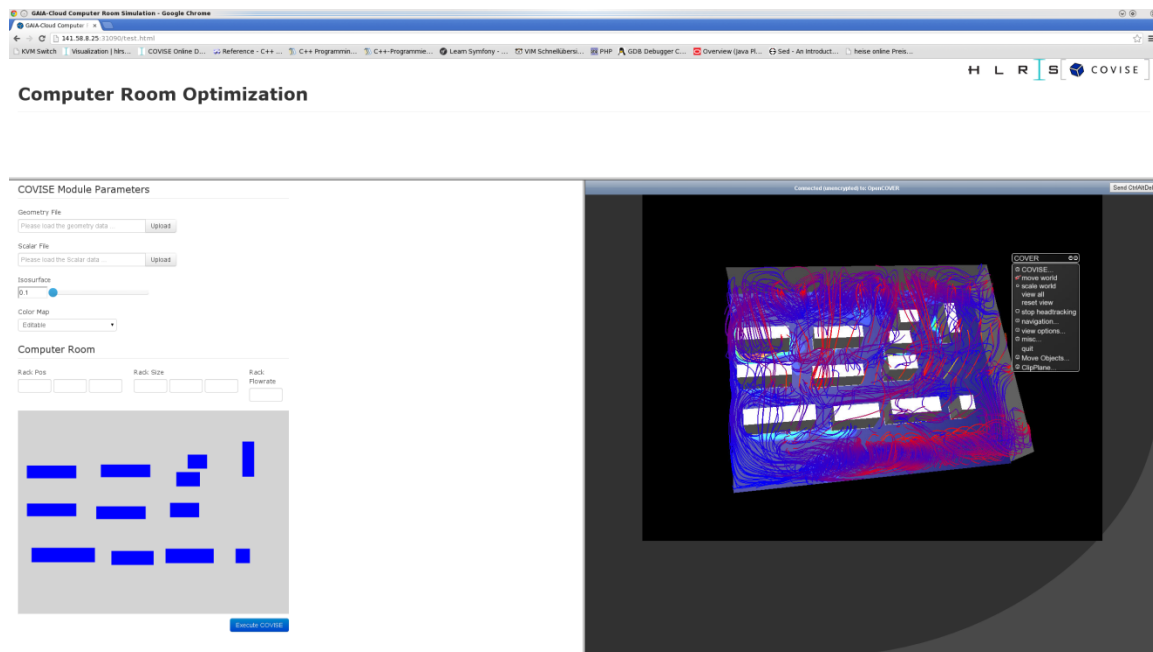


Figure 2-17: Covise-GUI parameters and visualisation

For the implementation of the web based GUI to COVISE, the following libraries and frameworks were used:

Table 2-9: Frameworks, libraries and licenses used for implementation of the COVISE-GUI

Software name	License / Website	Description
Apache CXF JavaScript Clients (v2.6.1)	Apache license. Version 2.0. http://cxf.apache.org/docs/javascript-clients.html	Apache CFX JavaScript Client
bootstrap (v2.3.2)	Apache license. Version 2.0. http://getbootstrap.com/	Front-end framework for faster and easier web development
Jquery(v2.0.3)	MIT license http://jquery.com/	JavaScript based query library
Log4Javascript (1.4.6)	Apache License. Version 2.0. http://log4javascript.org/	JavaScript based library for logging
fabric (v1.2.0)	Open source. MIT license. https://github.com/kangax/fabric.js/blob/master/LICENSE	JavaScript HTML5 canvas library
bootstrap-slider (v2.0)	Apache License. Version 2.0. http://www.eyecon.ro/bootstrap-slider	Slider for Bootstrap

2.3.5 MOP GUI

MOP GUI is used in the SVD-Toolkit to visualize 3D textured geometries from DEBB models along with data from the MOP Database. As both, real data obtained from measurements (e.g. temperatures, system load) and simulated data from CFD and Workload Simulation, are stored in the database, both of

them are displayed in the MOP GUI, which is integrated in the CoolEmAll Web GUI. The available data stored in the database is visualized in MOP GUI in two forms: colour maps on top of 3D models and 2D charts.

MOP GUI involves following parameters.

- Selection of the DEBB to be visualized

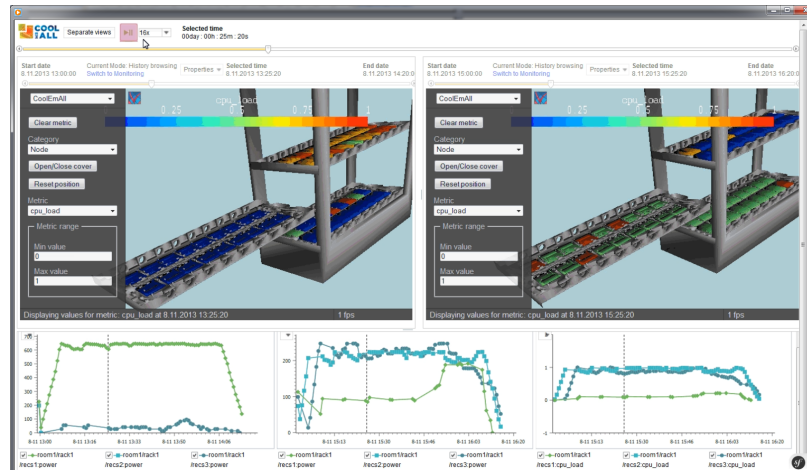


Figure 2-18: MOP GUI Visualization

2.3.6 Metric Calculator and report-GUI

The Metric calculator GUI is in charge of assessing experiments by calculating a selection of efficiency metrics. As an outcome, it generates an assessment report of the experiment.

In order to obtain such a report, the Metric Calculator requires an access to the monitoring infrastructure database (when calculating metrics for an experiment made on a real data-centre) or the SVD database (for a simulated experiment) to retrieve the measurements data necessary to calculate metrics, which will then be stored back into their respective databases. Finally these data will be transferred to the web-GUI database in order for the metric calculator GUI to be able to access them.

The Report-GUI will then display a summary table of metrics calculated for each components at different level.

Each of the metric named in the following lists are described in the deliverable D5.1 ([D5.1]).

For a simulation experiment the report will contain the following metrics:

- Key Metrics
 - Electricity cost
 - Carbon emissions
 - Power usage efficiency 3

- Power usage efficiency 4
- Total energy consumption
- Energy wasted ratio
- Productivity
- CAPEX
- Room level
 - Deployed hardware utilisation ratio
 - Load (minimum, average, maximum)
 - Power usage efficiency 3
 - Power usage efficiency 4
 - Energy wasted ratio
 - Data-centre infrastructure efficiency
 - Power used (minimum, average, maximum)
 - Energy
 - Productivity
- Rack level
 - Deployed hardware utilisation ratio
 - Load (minimum, average, maximum)
 - Power used (minimum, average, maximum)
 - Energy
 - Productivity
- Node-group level
 - Deployed hardware utilisation ratio
 - Load (minimum, average, maximum)
 - Power used (minimum, average, maximum)
 - Space, Watts And Performance (SWAP)
 - Energy
 - Productivity
- Node level
 - Load (minimum, average, maximum)
 - Power used (minimum, average, maximum)
 - Energy
 - Productivity

For a real world experiment the report will contain the following metrics:

- Data-centre level
 - Due to missing measurements at this level no metrics are calculated
- Room level
 - Due to missing measurements at this level no metrics are calculated
- Rack level
 - Energy
 - Productivity
 - Cooling index low (minimum, average, maximum)
 - Cooling index high (minimum, average, maximum)

- Imbalance of CPU temperatures (minimum, average, maximum)
- Node-group level
 - Deployed hardware utilisation ratio
 - Space, Watts And Performance (SWAP)
 - Energy
 - Cooling index low (minimum, average, maximum)
 - Cooling index high (minimum, average, maximum)
 - Imbalance of CPU temperatures (minimum, average, maximum)
- Node level
 - CPU temperature (minimum, average, maximum)
 - Inlet temperature (minimum, average, maximum)
 - Outlet temperature (minimum, average, maximum)
 - CPU usage (minimum, average, maximum)
 - Server usage (minimum, average, maximum)
 - Network usage (minimum, average, maximum)
 - Memory usage (minimum, average, maximum)
 - Power used (minimum, average, maximum)
 - Power usage (minimum, average, maximum)
 - Productivity
 - Cooling index low (minimum, average, maximum)
 - Cooling index high (minimum, average, maximum)

The Metric-Calculator itself is able to generate the same metrics for simulated and real world experiments, but missing measurements, sensor and simulated data make it necessary to have different reports while inputs for the calculator are not similar for both environments.

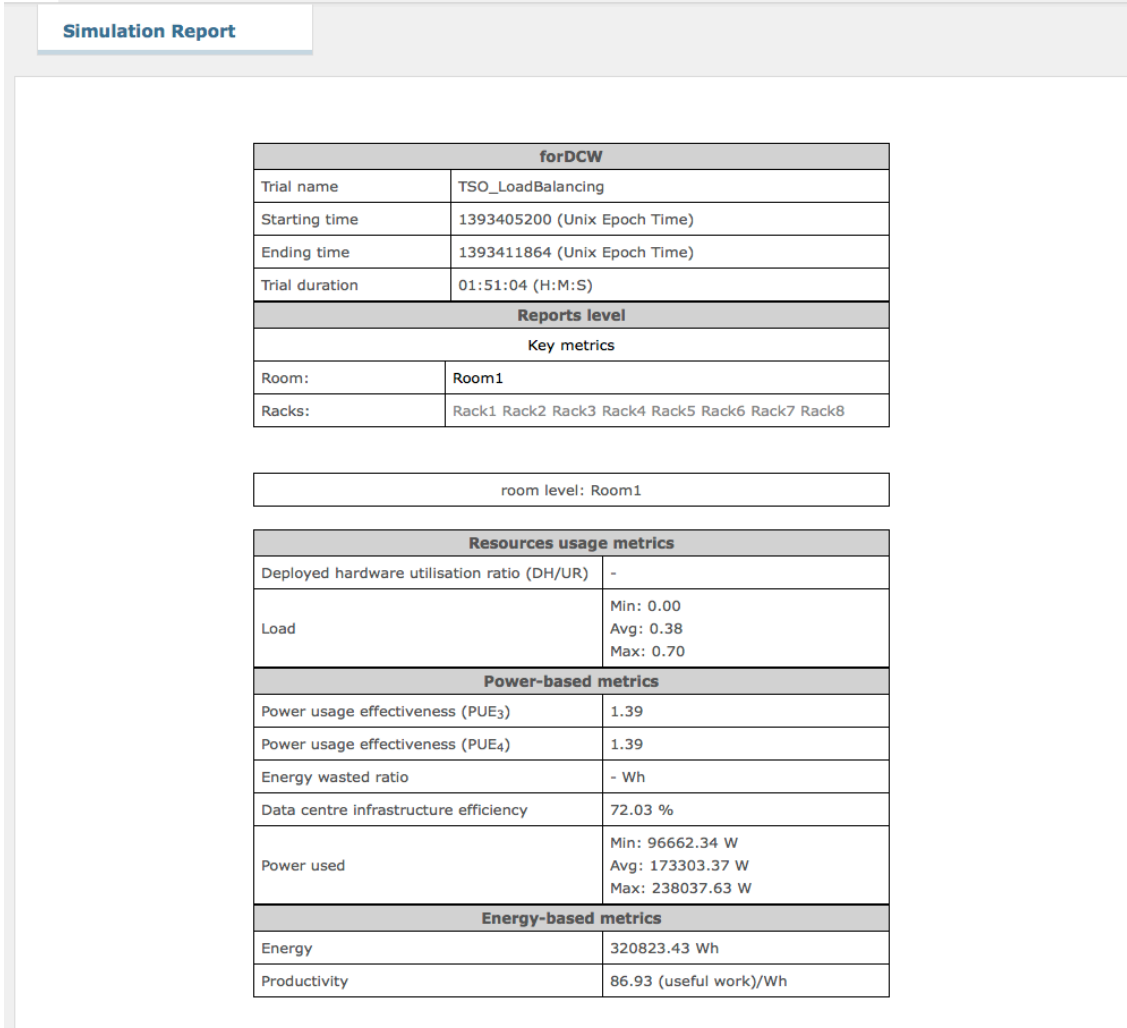


Figure 2-19: Simulation Report from Metric Calculator report GUI

2.3.7 User interface

This is the interface where the user account can be created. The user can also edit his/her profile, as we can see on Figure 2-20. On the CoolEmAll-Web-GUI the user has to select the option “User” on the top menu where he/she can choose from the options “Register”, “Edit Profile” or “Logout”.

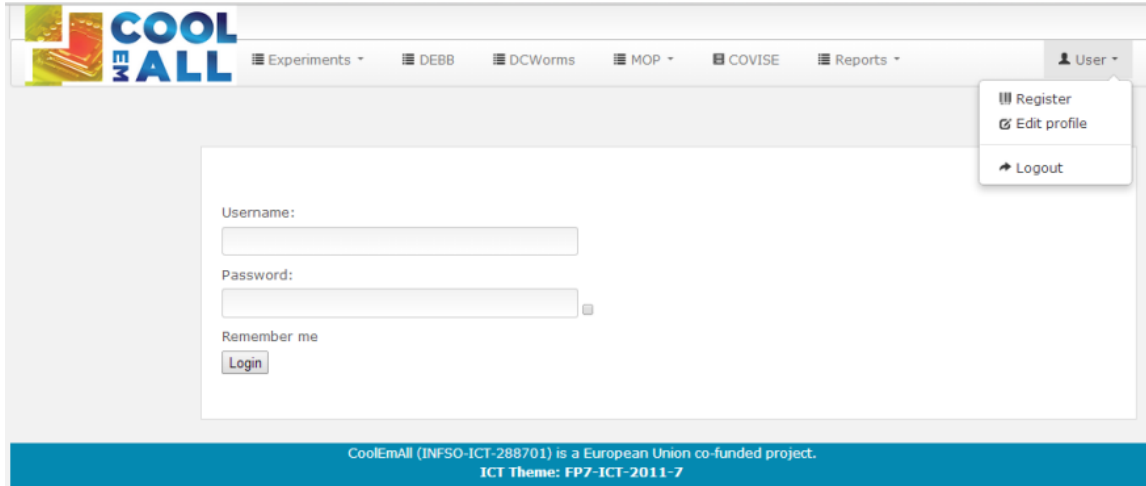

 The screenshot shows the CoolEmail web interface. At the top left is the CoolEmail logo. A navigation bar contains links for Experiments, DEBB, DCWorms, MOP, COVISE, and Reports. On the right, a 'User' dropdown menu is open, showing options for Register, Edit profile, and Logout. Below the navigation bar is a login form with fields for Username, Password, and a Remember me checkbox, followed by a Login button. At the bottom, a blue footer contains the text: 'CoolEmail (INFOS-ICT-288701) is a European Union co-funded project. ICT Theme: FP7-ICT-2011-7'.

Figure 2-20: User menu

Once the user chooses the option “Register” a new interface will be presented (as shown on Figure 2-21) in which the user has to provide the necessary information to create the account.

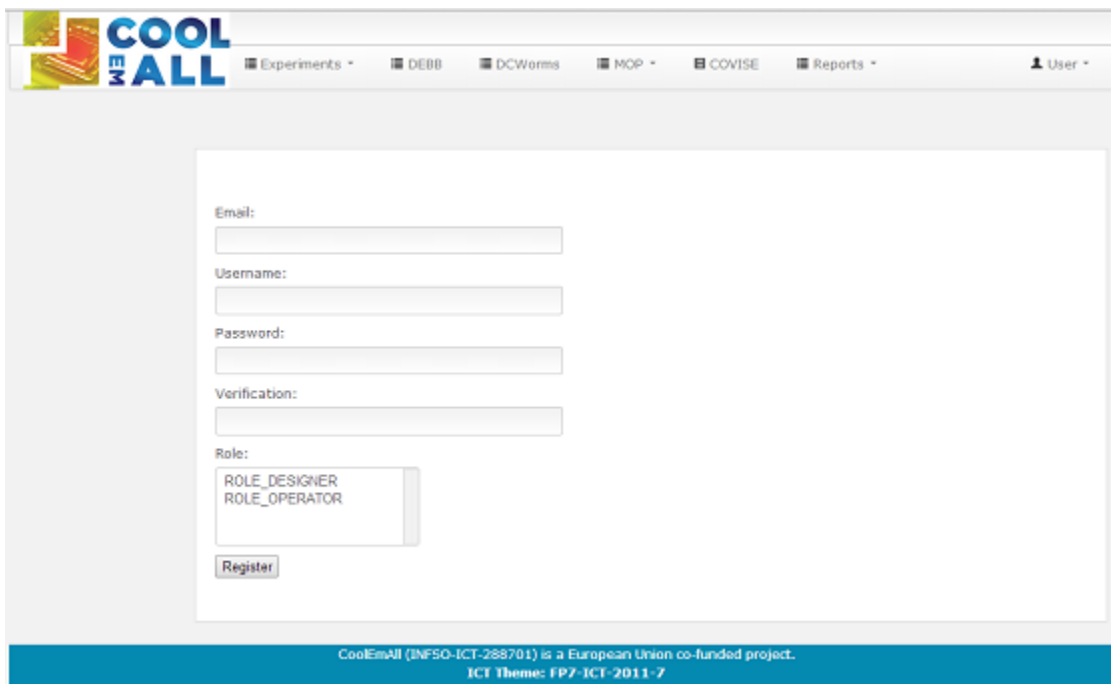

 The screenshot shows the CoolEmail registration form. It features the same navigation bar as Figure 2-20. The main form area contains fields for Email, Username, Password, and Verification. Below these is a Role selection dropdown menu with options for ROLE_DESIGNER and ROLE_OPERATOR, and a Register button. The footer at the bottom is identical to Figure 2-20, stating: 'CoolEmail (INFOS-ICT-288701) is a European Union co-funded project. ICT Theme: FP7-ICT-2011-7'.

Figure 2-21: Register interface

When the user is creating a new account the information that he/she must provide on the interface presented on Figure 2-21, as required, is:

- Email: user email.

- User name: desired name of user.
- Password: password to access to the CoolEmAll-Web-GUI.

Additionally, as optional information, the user can provide:

Role: there are two roles on the system: (i) Designer who has access to the GUIs ExperimentConfigurator, DEBBConfigurator, COVISE and ReportGUI; and (ii) Operator who has access to all the components GUIs of the SVD-Toolkit. The user can have none, one or both roles.

The second option is “Edit Profile” where the user can change the information saved during the registration process and additional information needed to execute the COVISE-GUI. These fields, as presented on Figure 2-22, are:

- Covise Server: server name in which Covise is running.
- Covise Port: port that is using Covise on the server name.

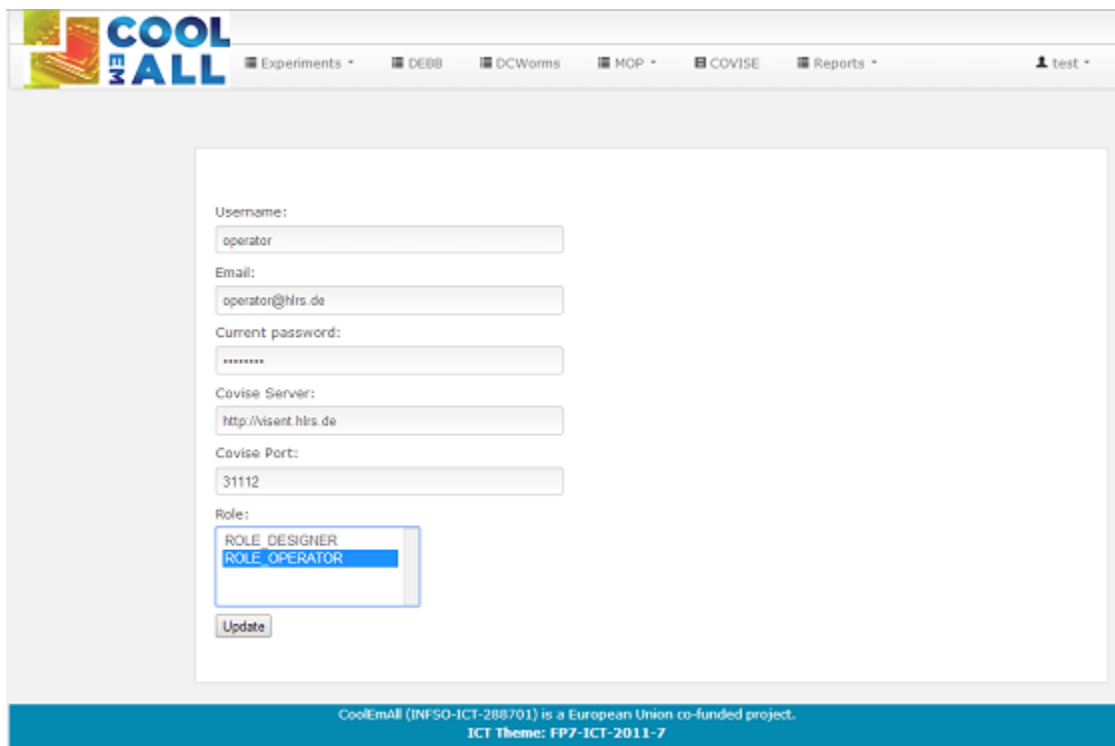


Figure 2-22: Register interface

Finally, the option “Logout” closes the session and returns the user to the login page presented on Figure 2-4.

2.4 Addressing the EcoDesign Directive

As described in [D3.6] in the year 2009 the European Union has released a directive regarding the energy efficient production and operation of energy consumption relevant products [EurPar]. The main goal is to improve the

efficiency of those products and to form a fair platform for trading these products within the EU without any disadvantages in competing countries. The following section gives an insight of how the concept of the EcoDesign Directive is reflected in the SVD-Toolkit.

When designing a product or a facility with the SVD-Toolkit, there are several metrics that can be processed, calculated and finally analysed. The most important metric in the scope of the EcoDesign directive is surely CO₂, thus the costs in CO₂ can be entered into the DEBBconfigurator, from a single component up to the CO₂ costs to provide a modelled facility. These CO₂ costs contain an estimation of the produced CO₂ in tons that will be emitted into the air, water and ground in case of the production of the specific component. The following factors could be included in this calculation:

- CO₂ equivalent to the energy needed to produce the component taking into account the respective environmental impact of various used materials
- CO₂ emitted through transportation
- CO₂ equivalent to the used electric power over the estimated lifetime
- CO₂ equivalent to the expected environmental stress through noise, electromagnetic radiation, etc. over the estimated lifetime
- CO₂ equivalent to the amount of created waste after the lifetime
- CO₂ savings when recycling or reusing the component

At higher levels of the DEBBconfigurator, these CO₂ costs get automatically summed up and displayed to the user so that he is easy able to compare the “greenness” of his specific configuration.

Analogue the CO₂ costs, the maximum power usage of each component can be entered, summed up and compared.

With these indicators it is easy for a user to see which components and configurations fit best into its company philosophy – e.g. to have a very green configuration with a low CO₂ footprint which on the other side might cost a little more.

When using the SVD-Toolkit to generate a report and calculate metrics related to an experiment, these metrics, defined by the DEBBconfigurator, will be extracted and integrated into the metric report.

In this way the end-user is able to visualize all the metrics in one place and can easily take decisions by taking all these costs data into consideration.

In the trial interface of the ExperimentConfigurator the CO₂ Emission Factor was also included. So, this factor can be updated for the other components of the SVDtoolkit during the execution of the experiments.

3 Usage of the CoolEmAll-Web-GUI and SVD-Toolkit

In this section we describe usage of the CoolEmAll-Web-GUI, new in version 2 of the Toolkit, presented in section 3.1, and of the SVD-Toolkit components, using command-line options by advanced users, in section 3.2.

3.1 Usage of the CoolEmAll-Web-GUI

In this section we present usage of the CoolEmAll-Web-GUI by explaining interaction of user with each of the GUIs of the CoolEmAll-Web-GUI, along with the interaction of the GUIs with other SVD-Toolkit components.

3.1.1 General Flow

The purpose of the CoolEmAll-Web-GUI is to provide a graphical user interface to SVD-Toolkit, which simplifies for users the interaction with the toolkit and between the components. In this section we briefly describe how to work with the CoolEmAll-Web-GUI. The general flow is as follows:

There are three options when you access the GUI:

1. Create and execute a new Experiment:
 - a. The user selects the ExperimentConfigurator GUI, inserts the information related to the experiment and trial. If the user doesn't provide the DEBB-URL he/she will be redirected to the DEBBConfigurator GUI in order to configure a DEBB specification.
 - b. Depending on the type of experiment that the user provides he/she will be redirected to the respective GUI:
 - i. If the experiment type is "DCworms" then the DCworms GUI is active.
 - ii. If the experiment type is "CFD", then the next GUI will be COVISE-GUI.
 - iii. If the experiment type is "Testbed", then the next GUI will be MOP-GUI.
 - iv. In the case that experiment type is "All", the user will be redirect first to the DCworms GUI, then to the COVISE-GUI and finally to the MOP-GUI in order to supply the necessary information for the execution of the component.
2. Go direct to a particular GUI component, in order to get the current status and/or continue with the execution of a given experiment. In this case the user has to select an experiment and a started trial.
3. View reports. In this option the user is able to select either the MOP-GUI or the Metric-Calculator GUI.

3.1.2 ExperimentConfigurator selection dialogue

To start an experiment the user must select the TAB of the ExperimentConfigurator GUI and execute the following steps, see Figure 3-1:

1. Create a new experiment. The user must select the type of experiment to be executed (DCworms, CFD, Testbed or All).
2. The user provides the Experiment Id.
3. The user has to select the option “create a new trial” and provide both the timestamp and the DEBB level in which the experiment is performed. The trial status is updated.
4. The user receives a message that the operation was successful.
5. The user provides additional information (optional), such as SVN version, DEBB URL, application profile URL and workload Profile URL. This information is saved on the database.
6. It shows to the user a message about the result of the trial configuration.
7. When the user selects the option “Next”, a query is done on the database in order to know what is the next component that the user can access is. The GUI of the next component is activated.

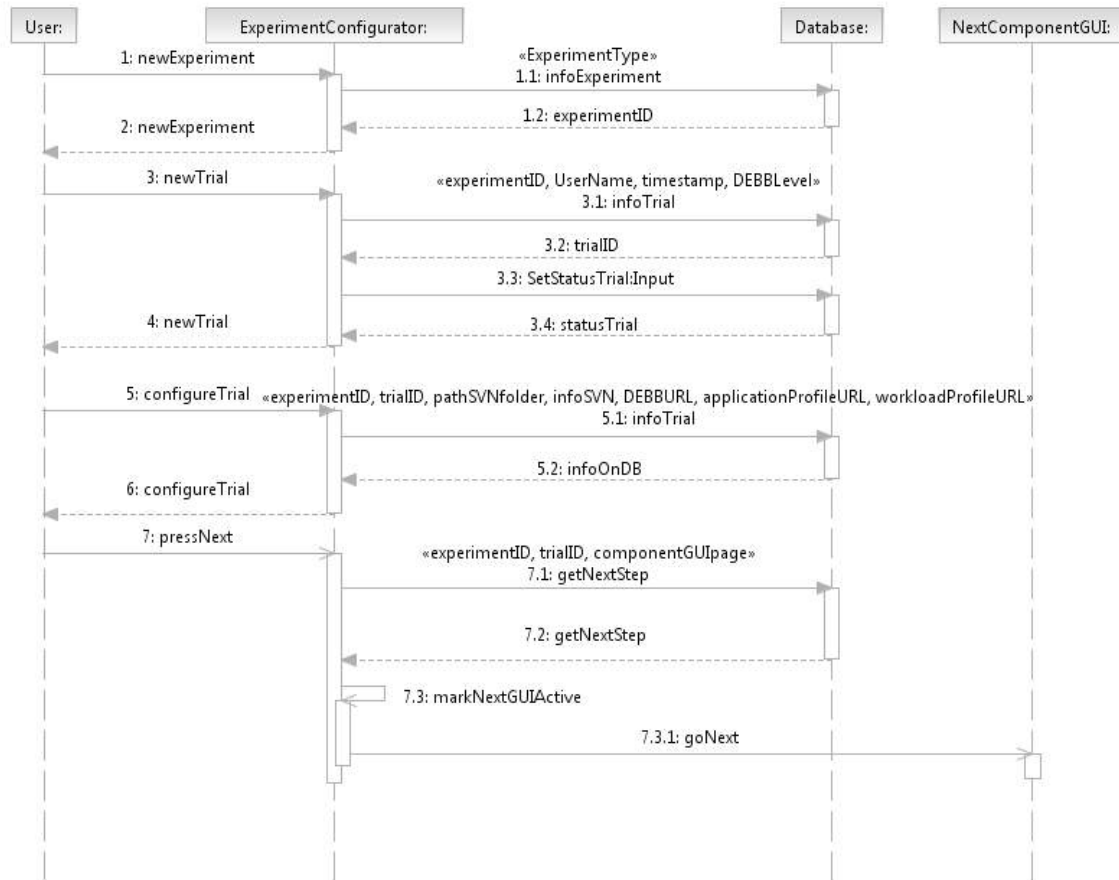


Figure 3-1: Sequence diagram of ExperimentConfigurator GUI.

In case the user has already created an experiment and trial in the past, then after logging into the Symphony2, all trials, which are not finished, will be shown. The user selects then the trial that he/she wants to check / modify / execute:

- User log ins into the Symphony2
- A request is send to the database DB to get for particular User-Name
 - All the trials that are “active” or last modified
 - User selects the appropriate trial.
 - Context belonging to this trial is loaded from the DB and passed to the server (and its variables)
 - User is now able to check the status of the experiment (when simulation is finished, a corresponding flag is stored in DB for particular trial)
 - Based on the status, the corresponding tab is automatically marked as active – i.e. view results of the simulation.

3.1.3 DEBBConfigurator GUI

To start a DEBB specification the user must select the TAB of the DEBB configuration GUI and execute the following steps, see Figure 3-2:

1. Create a DEBB specification. The DEBBConfigurator GUI must get the value of the necessary variables to execute the DEBB configuration GUI.
2. The user has to provide the information to create the DEBB model (as described in section 2.3.2 of this deliverable). The output of the process is saved on the SVN repository when the corresponding button in the node groups, racks or rooms TAB is clicked. The necessary variables, such as the SVN path to the DEBB files, are saved in the database. A default SVN path is suggested and can be changed to store the same DEBB under different names for different experiments.
3. When the exportation to the SVN repository is finished the user will be prompted a “Next” button. When the user selects the option “Next”, a query is done on the session variable “context” in order to know which the next component that the user can access is. This depends on the type of experiment (DCworms, CFD, Testbed or all) the user has chosen while creating the experiment. The GUI of the next component is called. If no experiment is selected the “Next” button will not show up.

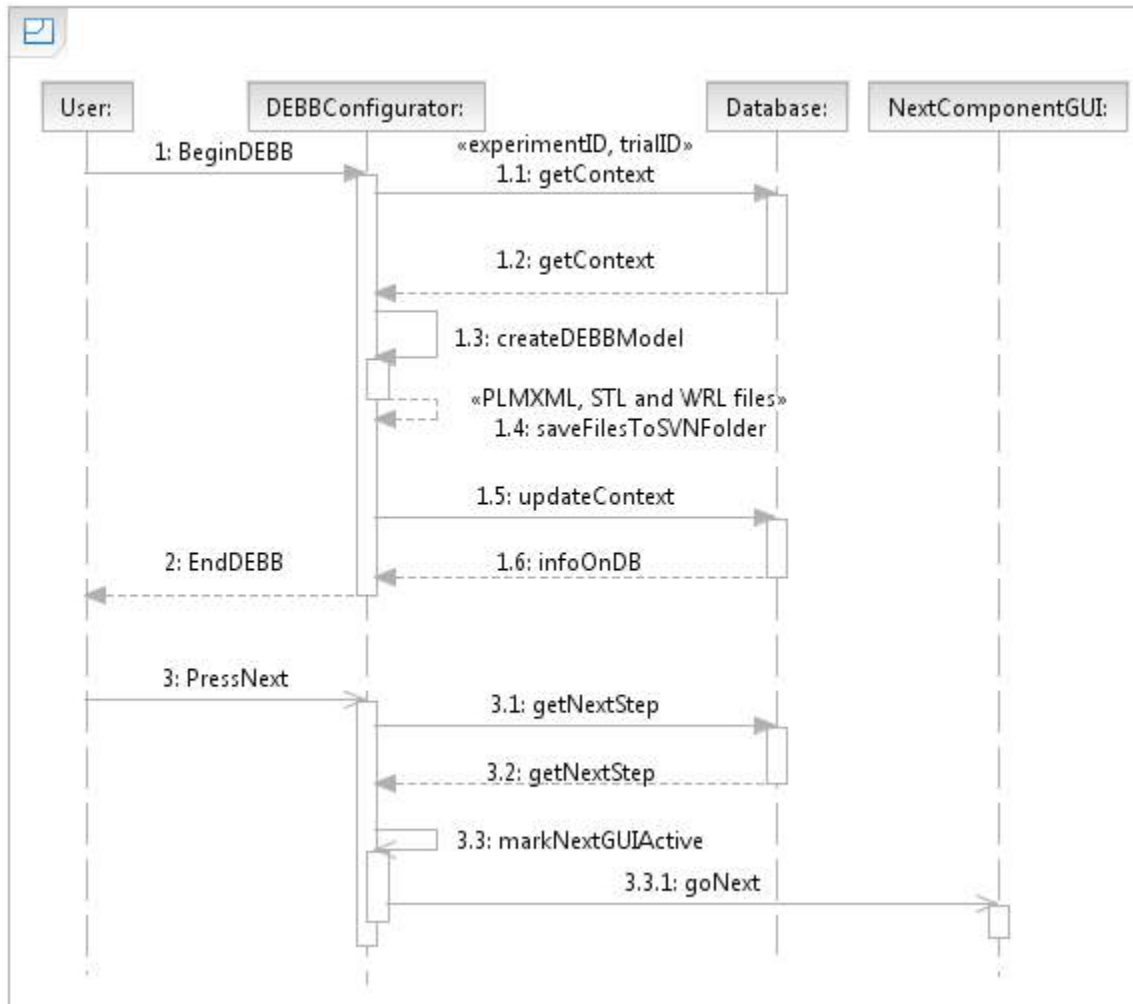


Figure 3-2: Sequence diagram of DEBBConfigurator GUI.

3.1.4 DCworms GUI

To start a DCworms simulation the user must select the TAB of the DCworms GUI and execute the following steps, see Figure 3-3:

1. Create DCworms simulation. The DCworms GUI must get the value of the necessary variables to execute it.
2. The user has to provide additional information to execute the DCworms component: 1) workload with workload and resource management policy; or 2) utilization levels of resources; or 3) workload with workload and resource management policy and utilization levels of resources.
3. The user starts the execution of the simulation, when all the requirements are reached.

4. The output of the process is saved on the SVN repository. The necessary variables are saved on the database.
5. When the user selects the option “Next”, a query is done on the database in order to know which the next component that the user can access is. The GUI of the next component is activated.

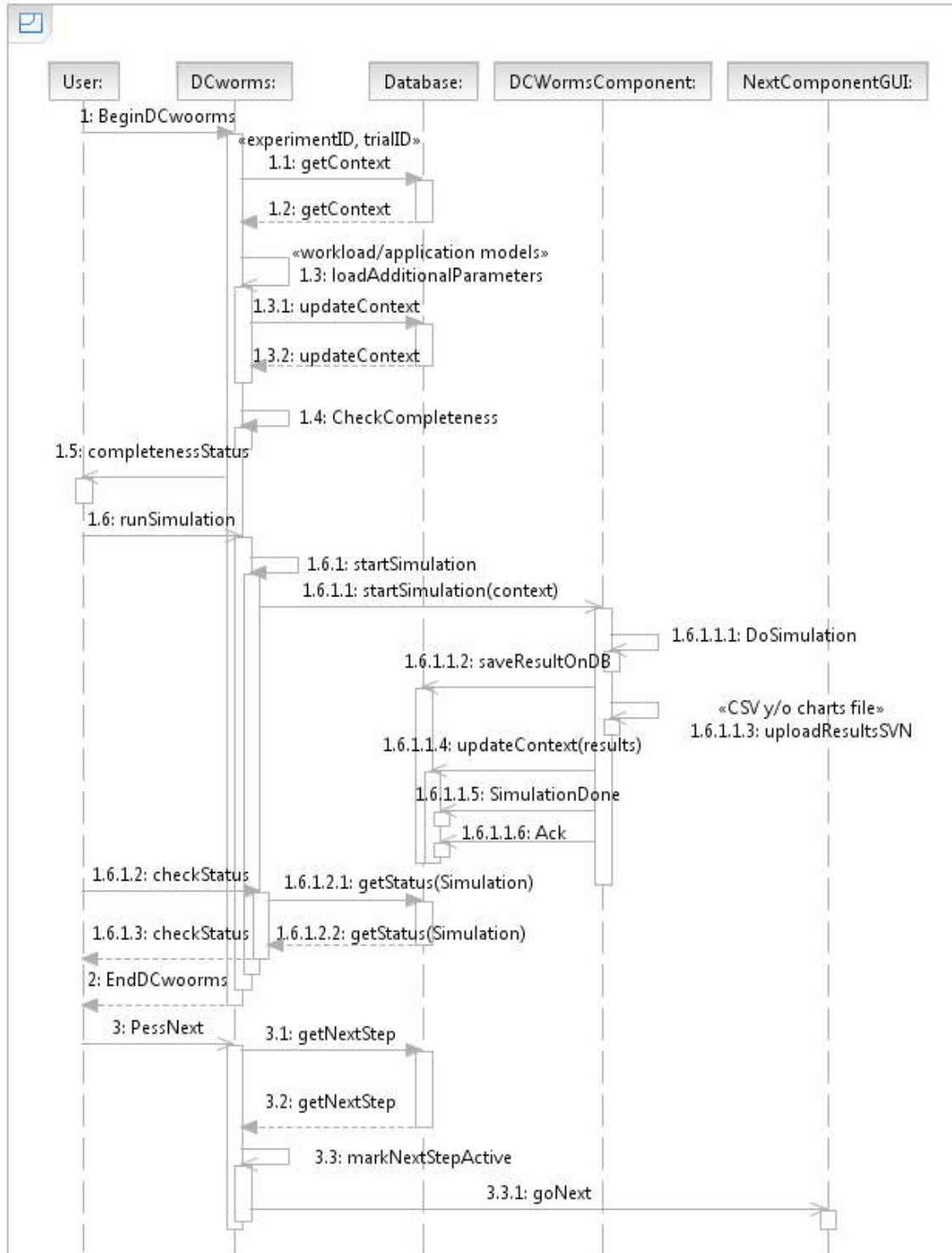


Figure 3-3: Sequence diagram of DCworms GUI

3.1.5 COVISE GUI

COVISE GUI presents the entire simulation results of the CFD-simulation, visualizing airflow across all building blocks (DEBBs) and enabling interaction with the simulation, allowing the user to interactively change the simulation parameters that affects the position (arrangement) of objects.

To start a COVISE simulation the user must select the TAB of the COVISE GUI and execute the following steps, see Figure 3-4:

1. Create COVISE simulation. The COVISE GUI must get the value of the necessary variables to execute it.
2. The user could provide additional information to execute the COVISE component – including arrangement of racks within the room.
3. The user starts the execution of the simulation, when all the requirements are reached.
4. The output of the process is visualised in the GUI, while essential metrics as extracted from the simulation results and are saved to the database.
5. When the user selects the option “Next”, a query is done on the database in order to know which the next component that the user can access is. The GUI of the next component is activated.

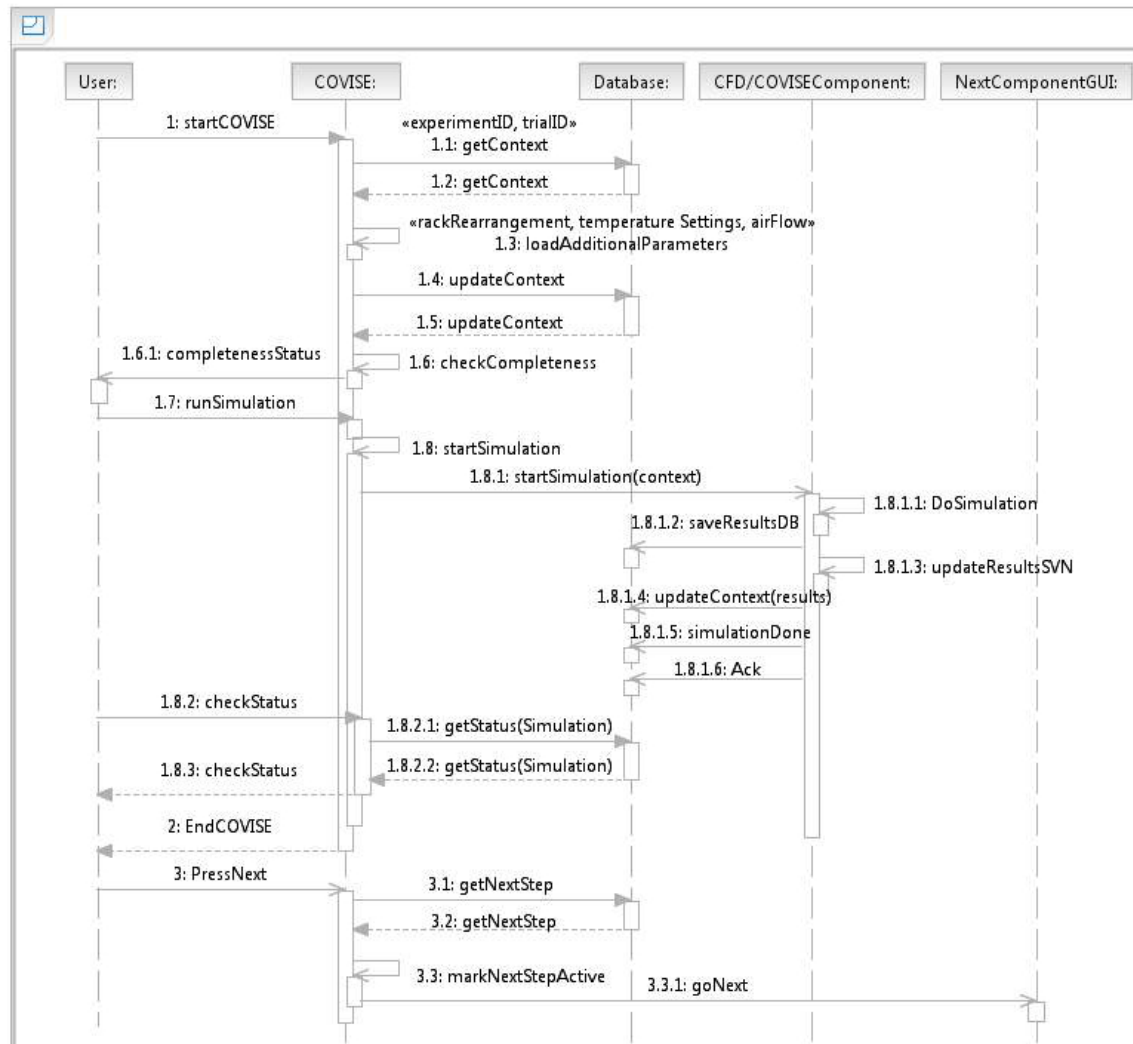


Figure 3-4: Sequence diagram of COVISE GUI.

3.1.6 MOP GUI

To start MOP GUI the user must select the TAB of the MOP-GUI and execute the following steps, see Figure 3-4:

1. View metrics in 3D view and 2D line charts. The MOP-GUI must get the value of the necessary variables to present the appropriate model and metrics.
2. When the user selects the option “Next”, a query is done on the database in order to know which the next component that the user can access is. The GUI of the next component is activated.

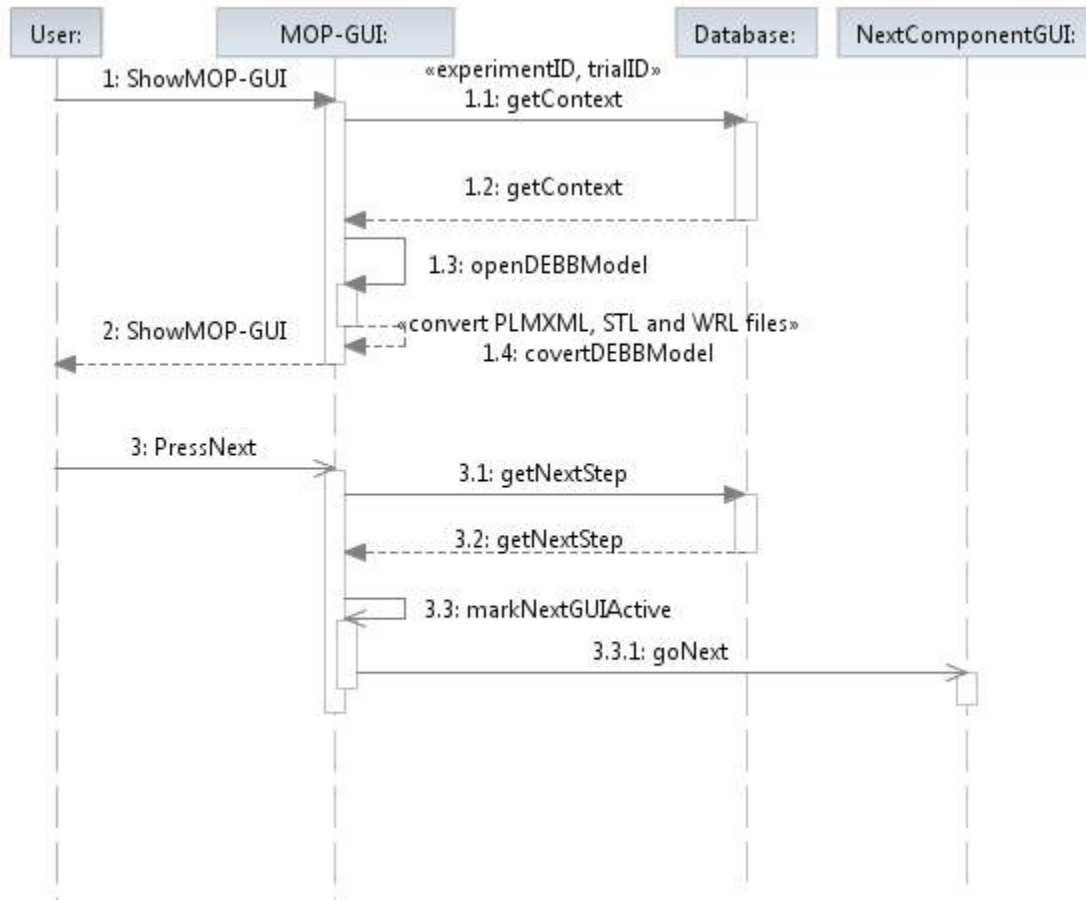


Figure 3-5: Sequence diagram of MOP GUI.

3.1.7 Metric Calculator and Report-GUI

The report-GUI provides access to report created by the metrics calculator.

In order to use the metric calculator report-GUI, the user must execute the following steps:

1. Select an already existing experiment and trial using the “List” sub menu from the “Experiments” TAB.
2. Select the “Reports” TAB and the “Metric Calculator” sub-menu.
3. If the report is not yet generated the user has to follow on-screen instructions to start the calculation.
4. The output of the process is stored in the web-GUI database.
5. If the report has already been generated the user will be able to navigate through all the data-centre levels (from the data-centre itself to the node) and see resulting metrics.

Once the user is on a report page she/he is able to navigate through the metrics using the links included in the Reports level table.

Lists of the following elements will progressively appear when clicking elements containing them:

- Key metrics (default selected level)
 - Rooms
 - Racks
 - Nodes-groups
 - Nodes

Each of these elements is a link to the corresponding report.

3.2 Expert usage of SVD-Toolkit components

In this section we describe how SVD-Toolkit components are used to enable execution of experiments (simulation) by advanced users, applying command line options. The description in this section is mostly originated from [D2.4].

3.2.1 Application Profiler

As stated in D2.4, the Application Profiler is quite simple to use. Each time an application is run on the test-bed, the application is run afterwards to produce its profile using monitored information available in TIMaCS. The resulting XML file is stored in the SVN hierarchy following the official CoolEmAll architecture. DCworms can then read these files for simulation purposes.

3.2.2 SVN Repository

As stated in D2.4, the SVN repository provides access to: application profiles, workload-profiles, DEBBs, and experiment configurations, used by SVD-Toolkit components for execution of experiments/simulations. In order to interact with the repository, on the client side, the user runs an Apache Subversion client application - typically a command line client, but possibly a GUI client as well. There exist a number of SVN clients, capable to access SVN server (repository). The most used command line options by SVN clients are:

- svn checkout - to checkout a working directory from the svn server
- svn add - to add a new file or directory to repository
- svn update/up – to update local copy with files from SVN server.
- svn commit/ci – to recursively sends local changes to the SVN server
- svn list – to display files in a directory for any given revision
- svn update – r <revision-number> - to check out specific revision

The usage of repository is done according to structure and conventions described in section 2.2.2.

3.2.3 Database

As stated in D2.4, the database includes several methods via RPC that can be called to insert and to retrieve data. To simplify query of the database, we implemented a script based API:

- **coolemall_getExperimentsList**
 - Return a list of all the experiments saved on the database.
- **coolemall_getLastMetricByMetricName object_path metric_name [experiment_ID trial_ID]**
 - Return the last metric specified by metric_name, object_path, experiment_ID and trial_ID. Experiment_ID and trial_ID are optional. The metric contains the last time and value recorded.
- **coolemall_getLastMetricsByHostName object_path [experiment_ID trial_ID]**
 - Return the last metrics of a specified object_path, for a given experiment_ID and trial_ID. Experiment_ID and trial_ID are optional.
- **coolemall_getMetricNames object_path [experiment_ID trial_ID]**
 - Return all the metrics saved for a particular object_path on a specified experiment_ID and trial_ID. Experiment_ID and trial_ID are optional.
- **coolemall_getHostNames [experiment_ID trial_ID]**
 - Return all object_path for which metrics are saved from a given experiment_ID and trial_ID. Experiment_ID and trial_ID are optional.
- **coolemall_getRecordsByMetricName object_path metric_name [experiment_ID trial_ID start_time, end_time]**
 - Return a list of metrics that contains record objects. Each record has three attributes: time, value and output. The arguments experiment_ID, trial_ID, start_time and end_time are optional. The argument start_s in seconds specifies the earliest record to be returned. No records newer than end_s (in seconds) are returned.
- **coolemall_getSelectedMetric object_path metric_name t_start t_selected**
 - Return the value of the metric_name given the timestamp start and the timestamp selected.
- **coolemall_getTrialsList experiment_ID**
 - Retrieve all trial_ids for a specified experiment_ID
- **coolemall_putMetricDB "metric_attribute" [,metric attribute ...]**
 - Insert into the database the parameters specified in the string by command line. Each attribute is a set of tuples *key:value* separated by comma (,) that represent the metric. For example:
"experiment_id:id_2,time:139893248,..."

3.2.4 DCworms

As stated in [D2.2] and presented in 2.1 the input to the workload simulation phase consists of workload and application profiles as well as DEBB model. In general, in DCworms all of this information is included within the single configuration file that is passed as an input parameter. This file has a typical, Java resource bundle format. List of all available parameters is presented below.

workload=workload.swf

resdesc=serverRoom.xml

workload parameter specifies the path and name of the file containing the workload profile. For the purposes of the workload description within the SVD-Toolkit we adopted Standard Workload Format (SWF) [SWF]. In addition to the predefined labels in the header comments, we introduce support of a new one that is used to provide information about types of applications used within the given set of tasks. In this way, a workload profile contains the references to the corresponding application profiles that will be loaded and linked during the simulation. More details of the workload and application profiles and the format of the particular descriptions can be found in [D2.3].

resdesc parameter points to the path and name of file with the resource characteristics. DCworms is also able to read DEBB files (the format of such files was described in [D3.2]) by translating them to its own resource description file. This file also contains reference to workload and resource management policy used for particular experiment (within the SVD-Toolkit this information is provided by DCworms GUI).

There are a number of management policies provided with DCworms and available from the DCworms GUI perspective. They include various approaches starting from load balancing strategy, through consolidation methodology up to energy usage optimization one, that take into account both application and resource profiles. Moreover, most of them come in a version with node power management approach for both ComputeBox levels or with power capping strategy on the room level. More details concerning the policies can be found in [D4.3] and [D4.6].

To perform experiments using DCworms, the user needs to execute the following command, passing the path to the configuration file as a program argument:

```
sbatch --partition=aux runDCworms.sh
experiment1/RECSexperiment1.properties
```

The simulation is controlled by the testbed queuing system (SLURM) and it is requested to be started on aux partition. The partition is designed for running computations outside the main monitored part of the testbed to not influence the measurements. It consists of two worker nodes that have their own dedicated power lines and are physically separated from rest of the testbed.

3.2.5 CFD using OpenFOAM

As stated in D2.4, the interaction with the CFD-solution within 1st prototype was done via a command line based script.

Before the invoking of the script can be done a simulation environment needs to be set up. This consists foremost of a working installation of OpenFOAM and a setup of OpenMPI [OpMPI]. This is supposed to be done before you start with the setup of the bespoke CFD-solution.

For setting up an automated simulation based on this first prototype it is most convenient to do so in a dedicated directory. In our case this directory is called “auto_OpenFOAM”. In this working directory it is supposed to have the following subdirectories set up: One directory should be called “DEBB”. In this directory the governing .PLMXML-file and the geometry representing .STL-files are stored.

Then there is another important directory, which is called “control_files”. This directory is particularly important when it is of interest to run several different simulations. In this directory simulation independent solver parameter are stored. These variables are editable but it is not necessary and not recommended to do so. These variables are reused for each independent simulation and therefore these variables are copied to the corresponding files of the simulation case. Another important directory is called “TIMaCS”. In this directory are the scripts for invoking the database for automatic creation of boundary and initial conditions stored, as well as the scripts for putting back the simulation results to the database. Then there is directory called “SRC”. In this directory the source code and the executables for the automated setup of the simulation case is stored. So there should be no necessity for the user to interact with this directory.

Then there are some more .TXT and .XML-files. These files are described in the order the setup script invokes them. First there is a file called Definition_Quader.xml. In this file the definitions are made for invoking blockMesh. blockMesh sets up the surrounding region in which the geometry to simulate is meshed. Here it is necessary to make sure that the geometry fits into the surrounding region created by blockMesh and therefore the size of this region can be adjusted accordingly in this file called “Definition_Quader.xml. Next the user finds a file called locationInMesh.txt. This file is important for snappyHexMesh. To enable snappyHexMesh to automatically create a mesh based on the region created by blockMesh and the .STL-files transformed by the PLMXML-parser it is necessary to specify a point inside the mesh, in a region where the mesh should be kept. The point inside the mesh region to keep is stated in the file called “locationInMesh.txt”

In the file called temperature.txt temperature values for boundaries, which are not referenced in the database, are stored. This is necessary as there are no temperature sensors available on these boundaries, which are for example sidewalls. These values should be submitted in degrees Celsius in the format found in the file and are editable by the user.

After a simulation was done a new directory called “case1” appears. In this directory the data for the whole simulation can be found. And this directory is deleted and remade every time a simulation is started automatically. So if the user desires to keep the whole simulation case it is best to move this directory to a different location before another automatic simulation is started.

When all this setup is complete, the simulation can be started automatically just by calling the script called Autorun with submitting the path to the location on the .PLMXML-file and the name of the host for which the metrics are stored. The retrieving of all relevant data and the calculation is then performed automatically.

3.2.6 CFD using COVISE with Ansys CFX

In version 2 of the Toolkit, simulation of the server-room is done using COVISE with Ansys CFX. As noted, the Simulation Workflow COVISE (Collaborative Visualisation and Simulation Environment) is an extensible distributed software environment capable to integrate simulations, post-processing and visualization functionalities in a seamless manner. The CFD Solver performing CFD simulation is directly integrated into the COVISE workflow, including all necessary pre- and post-processing tasks. Figure 3-6 visualizes workflow (presented in the figure on the left) within the COVISE used (i) to integrate input parameters coming from various sources necessary to setup simulation, (ii) execute simulation remotely using corresponding (CFX) solver, (iii) post-process and visualize results using renderer (presented in the figure on the right).

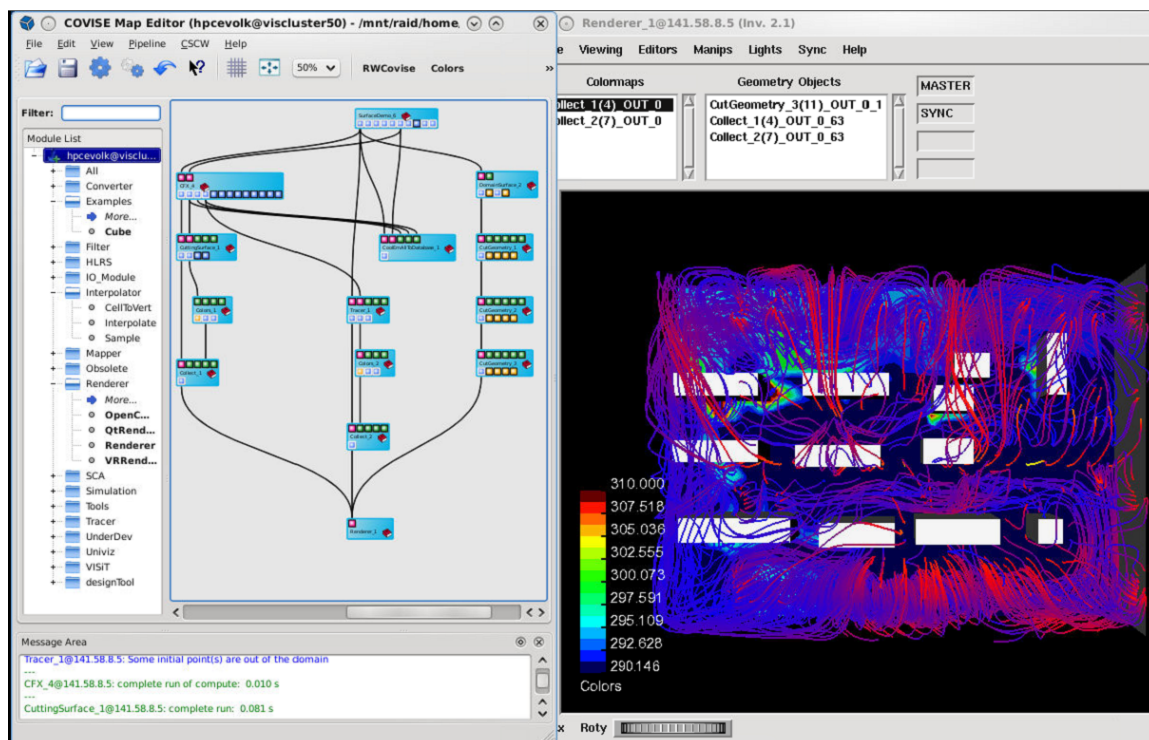


Figure 3-6: COVISE visualisation environment and outcome of the heat-flow simulation

3.2.7 Metric Calculator

The Metric Calculator is a Python command line application that can be called with many different parameters depending of the calculation.

Measurements and calculations can be done at each level of the data-centre. Starting at the data-centre level itself, the room level, the rack level, the node-group level and finally the node level.

In order to be able to compare simulation and real world experiments the Metric Calculator can be executed on both environments.

Each metric can be calculated individually (as shown in the following listing) at the desired level. Depending of the provided parameters the Metric Calculator is able to determine if the calculation has to be done on real or simulated data and adapt the calculation.

Listing of available commands:

```
./metricCalc.py cpu_usage[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum cpu usage of the hardware element identified by its path between two timestamps.

```
./metricCalc.py server_usage[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum server usage of the hardware element identified by its path between two timestamps.

```
./metricCalc.py network_usage[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum network usage of the hardware element identified by its path between two timestamps.

```
./metricCalc.py memory_usage[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum memory usage of the hardware element identified by its path between two timestamps.

```
./metricCalc.py power_usage[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum power usage of the hardware element identified by its path between two timestamps.

```
./metricCalc.py power[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum power consumption of the hardware element identified by its path between two timestamps.

```
./metricCalc.py temperature[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum temperature of the hardware element identified by its path between two timestamps.

```
./metricCalc.py cpu_temperature[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum cpu temperature of the hardware element identified by its path between two timestamps.

```
./metricCalc.py energy <path> <start_s> <end_s>
```

Calculate the energy consumed of the hardware element identified by its path between two timestamps.

```
./metricCalc.py productivity <path> <start_s> <end_s> <useful work>
```

Calculate the productivity of the hardware element identified by its path between two timestamps.

```
./metricCalc.py swap <path> <start_s> <end_s> <useful work>
```

Calculate the space, watts and performance metric (SWaP) of the hardware element identified by its path between two timestamps.

```
./metricCalc.py cooling_index_low <path> <start_s> <end_s>
```

Calculate the cooling index low of the hardware element identified by its path between two timestamps.

```
./metricCalc.py cooling_index_low_ts[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum cooling index low of the hardware element identified by its path between two timestamps.

```
./metricCalc.py cooling_index_high <path> <start_s> <end_s>
```

Calculate the cooling index high of the hardware element identified by its path between two timestamps.

```
./metricCalc.py cooling_index_high_ts[_min, _avg, _max, _mma] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum cooling index high of the hardware element identified by its path between two timestamps.

```
./metricCalc.py heat_generation[_min, _avg, _max, _mma, _tot] <path> <start_s> <end_s>
```

Calculate the minimum, average and maximum heat generation of the hardware element identified by its path between two timestamps.

```
./metricCalc.py dh-ur <path> <start_s> <end_s>
```

Calculate the deployed hardware utilisation ratio (DH-UR) of the hardware element identified by its path between two timestamps.

```
./metricCalc.py dh-ur(cpu) <path> <start_s> <end_s>
```

Calculate the deployment hardware utilisation ratio considering CPU (DH-UR_{CPU}) of the hardware element identified by its path between two timestamps.

```
./metricCalc.py imbalance_of_temperature <path> <start_s> <end_s> [usedNodes]
```

Calculate the imbalance of temperature of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py imbalance_of_temperature_ts[_min, _avg, _max, _mma] <path> <start_s> <end_s> [usedNodes]
```

Calculate the minimum, average and maximum imbalance of temperature of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py imbalance_of_temperature_of_cpu <path> <start_s> <end_s> [usedNodes]
```

Calculate the imbalance of temperature of CPU of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py imbalance_of_temperature_of_cpu_ts[_min, _avg, _max, _mma] <path> <start_s> <end_s> [usedNodes]
```

Calculate the minimum, average and maximum imbalance of temperature of CPU of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py imbalance_of_heat_generation <path> <start_s> <end_s> [usedNodes]
```

Calculate the imbalance of heat generation of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py imbalance_of_heat_generation_ts[_min, _avg, _max, _mma] <path> <start_s> <end_s> [usedNodes]
```

Calculate the minimum, average and maximum imbalance of heat generation of the hardware element identified by its path between two timestamps (the calculation can optionally be done only considering a set of used nodes).

```
./metricCalc.py pue <path> <start_s> <end_s>
```

Calculate the power usage effectiveness of the hardware element identified by its path between two timestamps.

```
./metricCalc.py dcie <path> <start_s> <end_s>
```

Calculate the data centre infrastructure efficiency (dcie) of the hardware element identified by its path between two timestamps.

```
./metricCalc.py electricity_cost <path> <start_s> <end_s>
```

Calculate the electricity cost of the hardware element identified by its path between two timestamps.

```
./metricCalc.py carbon_emissions <path> <start_s> <end_s>
```

Calculate the carbon emission of the hardware element identified by its path between two timestamps.

Some of the metrics can be calculated in two different ways. They are differentiated in the previous listing by the `_ts` suffix meaning that the metric can be calculated for each timestamp of the defined interval (with the `_ts` suffix) or over the whole interval (without the suffix). In order to simplify the usage of this tool, the Metric Calculator also includes some scripts that automatically calculate a predefined set of metrics. These scripts are used by the Report-GUI to start the calculation for a selected experiment and trial and store the results in the simulation or real world experiments database and finally transfer them into the web-GUI database in order to be able to access these data. In addition to the previously seen parameters these scripts accept an additional one, a link to the DEBB's files describing the specifications of the hardware used for the simulation or the real world experiment.

Computing and storing metrics for a simulation experiment is done using the following command:

```
./simulationMetrics.py <exp> <trial> <start> <end> [debb url]
```

4 Test of the CoolEmAll-Web-GUI and SVD-Toolkit

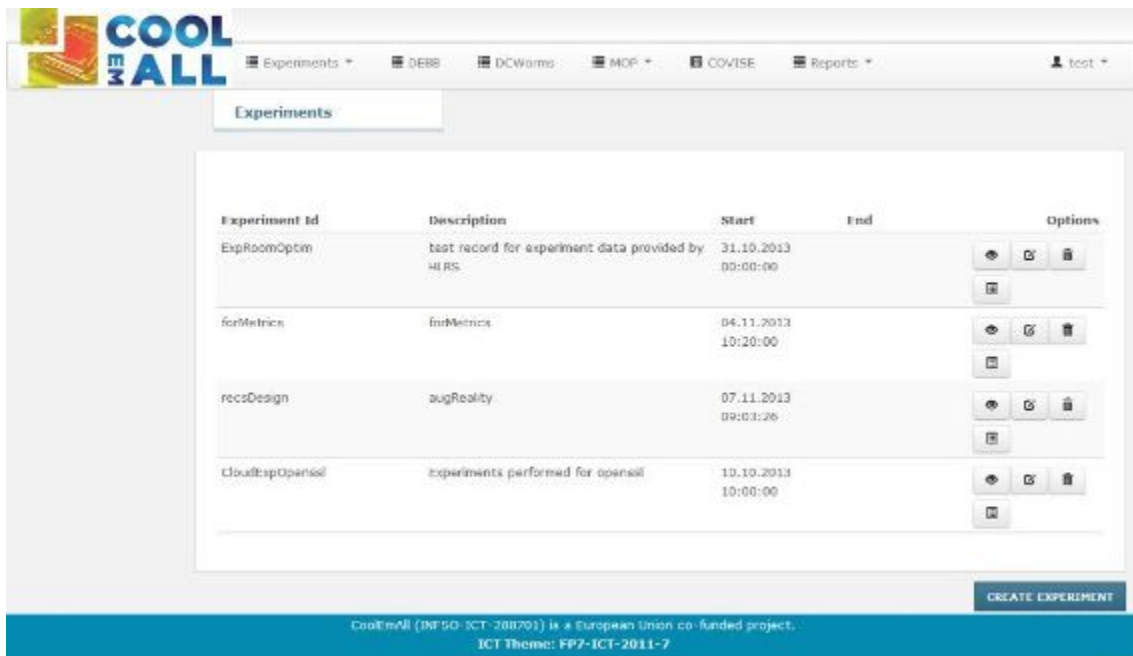
In this section we describe tests of the CoolEmAll-Web-GUI and its interaction with the SVD-Toolkit components in section 4.1 (new in final release of SVD-Toolkit. In Annex A we present the execution of the complete flow), and command line based tests of the SVD-Toolkit components described in section 4.2.

















4.1 Test of the CoolEmAll-Web-GUI

In this section we describe test of the CoolEmAll-Web-GUIs.

4.1.1 ExperimentConfigurator GUI

In section 2.3.1 we described the experiment configurator GUI, capable of reading parameters, entered by the users, necessary to configure and setup the SVD-Toolkit and its components to perform experiments. The overview of the experiments is shown in the figure below and allows definition and selection of experiments. In the Experiment Configurator section the user can list all the experiments saved, choosing the option “Experiments” and then “List” on the top menu. Then the listing of the experiments will be shown, as we can see on Figure 4-1.




Experiment Id	Description	Start	End	Options
ExpRoomOptim	test record for experiment data provided by HI RS	31.10.2013 00:00:00		   
forMetrics	forMetrics	04.11.2013 10:20:00		   
recoDesign	augReality	07.11.2013 09:03:26		   
CloudExpOpses	experiments performed for opses	10.10.2013 10:00:00		   




[CREATE EXPERIMENT](#)

CoolEmAll (INF50-ICT-200701) is a European Union co-funded project.
ICT Theme: FP7-ICT-2011-7

Figure 4-1: Overview of Experiments

In this interface the user can select the options:

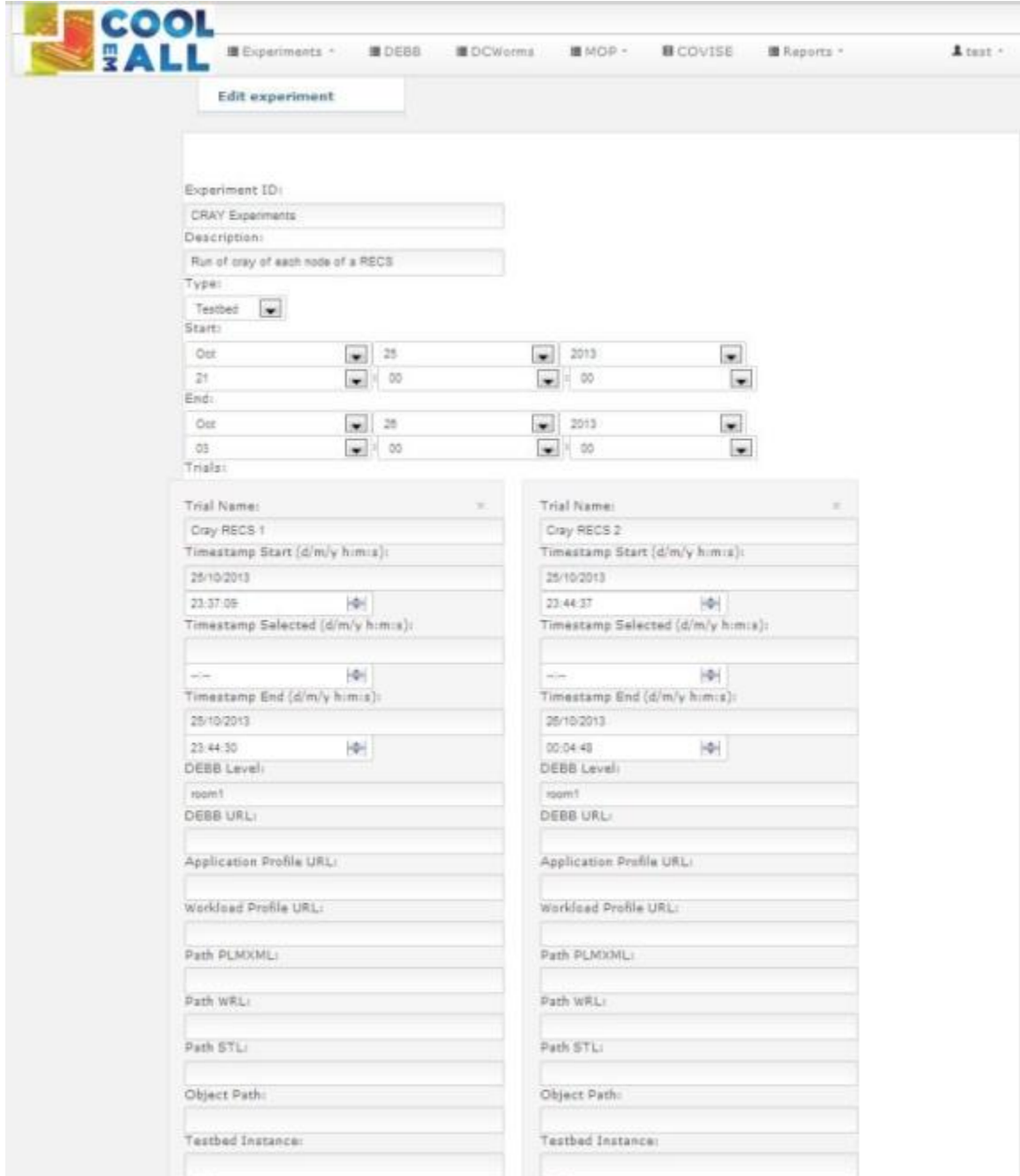
- “Show” () to see more information of the experiment;

- “Edit” () to change parameters of the experiment;
- “Delete” () the experiment or
- “Trials” () to list all the trials of a particular experiment.

For creating new Experiments the user has two options:

1. Select the option “New” on the top menu section “Experiments”.
2. Click on the “CREATE EXPERIMENT” button at the bottom of the page.

Both options will present the interface to configure a new experiment, as shown on Figure 2-8, whereby the user has to input all the parameters required defined on Section 2.3.1, both for experiment and trials. At the end of the process we have the view as presented on Figure 4-2.



The screenshot shows the 'Edit experiment' interface in the COOL M3 ALL system. At the top, there is a navigation bar with 'Experiments', 'DEBB', 'DCWorms', 'MOP', 'COVISE', and 'Reports' menus, and a user profile 'test'. Below the navigation bar is an 'Edit experiment' button. The main form contains the following fields:

- Experiment ID:** CRAY Experiments
- Description:** Run of tray of each node of a RECS
- Type:** Testbed
- Start:** Oct 25, 2013 21:00
- End:** Oct 26, 2013 03:00
- Trials:** Two trial configuration panels are shown side-by-side.
 - Trial 1 (Cray RECS 1):** Timestamp Start: 25/10/2013 23:37:05; Timestamp Selected: --; Timestamp End: 25/10/2013 23:44:30; DEBB Level: room1; DEBB URL: ; Application Profile URL: ; Workload Profile URL: ; Path PLMXML: ; Path WRL: ; Path STL: ; Object Path: ; Testbed Instance:
 - Trial 2 (Cray RECS 2):** Timestamp Start: 25/10/2013 23:44:37; Timestamp Selected: --; Timestamp End: 25/10/2013 00:04:48; DEBB Level: room1; DEBB URL: ; Application Profile URL: ; Workload Profile URL: ; Path PLMXML: ; Path WRL: ; Path STL: ; Object Path: ; Testbed Instance:

Figure 4-2: Experiment with two trials.

In order to have the information available of a particular trial for the other GUIs, the user has to set the trial information on the session. In this way all components can read/update the information of a particular trial. The user chooses an experiment (on the options Experiments -> List) and then he/she selects the option trials. A list with the trials of the experiment will be shown, see Figure 4-3. In this interface the user has to select the last option "Set Context" (

0). In this interface we have also the options to “Show” (👁️), “Edit” (✎) and “Delete” (🗑️) a trial of the list.

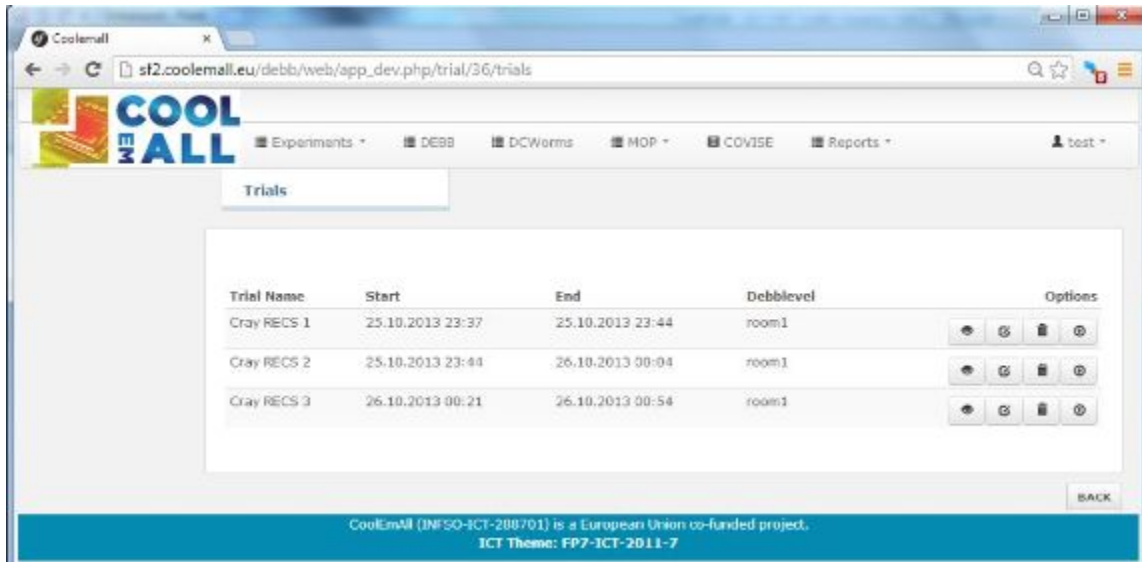


Figure 4-3: List of trials of a particular experiment.

Each GUI-Component can use/change the session variable during execution of the trial/experiment. The ExperimentConfigurator provides the option to update the value of this variable. Once the variable is set on the context two new buttons are shown on the interface, as presented on Figure 4-4.

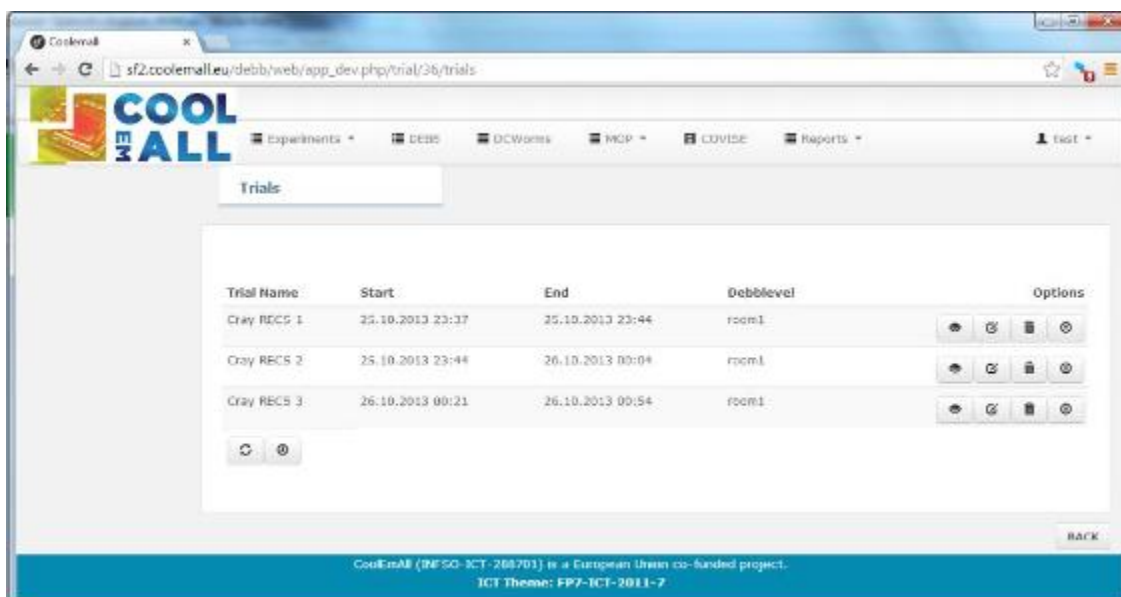




Figure 4-4: Trail on the context.

The First one, “Update Context” (), allows the user to change the value of the variable on the session and the other, “Save Context” (), saves the value of the session variable on the database.

4.1.2 DEBBConfigurator GUI

In this test a predefined node (Kontron COMe-bSC2 - 3615QE + Kontron COM Express Passive Heatsink i7) was put into a node group (test_D2.5), and into a rack (test_D2.5), which was placed in a room. The resulting room was exported as a ZIP file, which was named with the current time stamp.

This ZIP must then contain the following files:

- Node_KontronCOMebSC23615Q_*.xml
- Heatsink_KontronCOMExpressPassiveHeatsinki7_*.xml
- NodeGroup_test_D25_*.xml
- FlowPump_AVADB04028B12U_*.xml (chassis fan)
- Rack_test_D25_*.xml
- Room_test_D25_*.xml
- PLMXML_test_D25_*.xml
- Costs.xml
- pics - > the a picture of the board
- objects -> the board, heatsink, chassis, fan and rack STL and WRL files

The transform coordinates for the heatsink included the following gaps:

- The rack stands at x: 1,5m; y: 1,5m; z: 0m
- It has a gap of x: 0,037m; y: 0m; z: 0,2m
- The chassis is at RU 10 in the rack. 1RU = 0,04445m
- The node in the chassis is at this position: x: 0.25634m; y: 0.29995m; z: 0.01739m
- The node is 0,01m high

These offsets are listed in the exported PMLXML file.

This results in the following position for the heatsink:

$$X = 1,5m + 0,037m + 0.25634m = 1,79334m$$

$$Y = 1,5m + 0m + 0,29995m = 1,79995m$$

$$Z = 0m + 0,2m + (10 * 0,04445m) + 0,01739m + 0,01m = 0.67189m$$

The following steps have been done to create the ZIP archive:

1. Create a new node group with the “Christmann RECS | Box Compute Unit”

2. The product name is “test_D2.5”
3. Adopt a the “Kontron COMe-bSC2 – 3615QE” node in column 2, row 7 counted beginning from left and at the top of the page (back of the chassis)

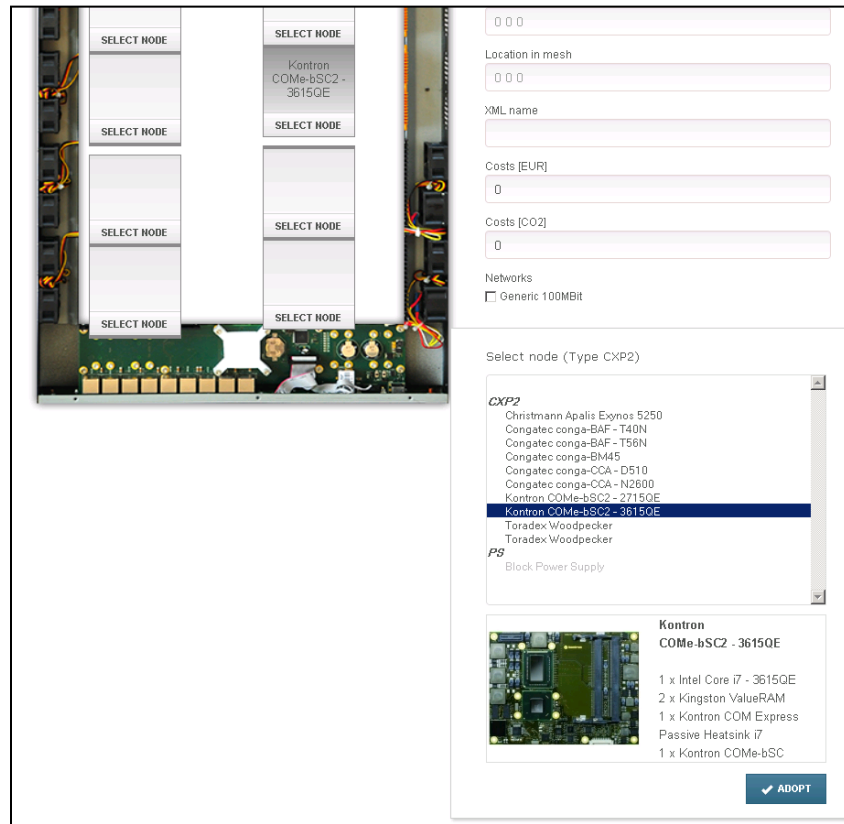


Figure 4-5: node group creation tab with adopted node

4. Save the node group
5. Create a new rack
6. The gaps, size and the STL and WRL model files were taken from the “Christmann Testbed_Rack”
7. Save the rack
8. Create a new room named “test_D2.5”
9. Place the “test_D2.5” rack into the room
10. Drag the rack to the position at x=1,5m; y=1,5m
11. Save the room
12. Download the archive file

Result:

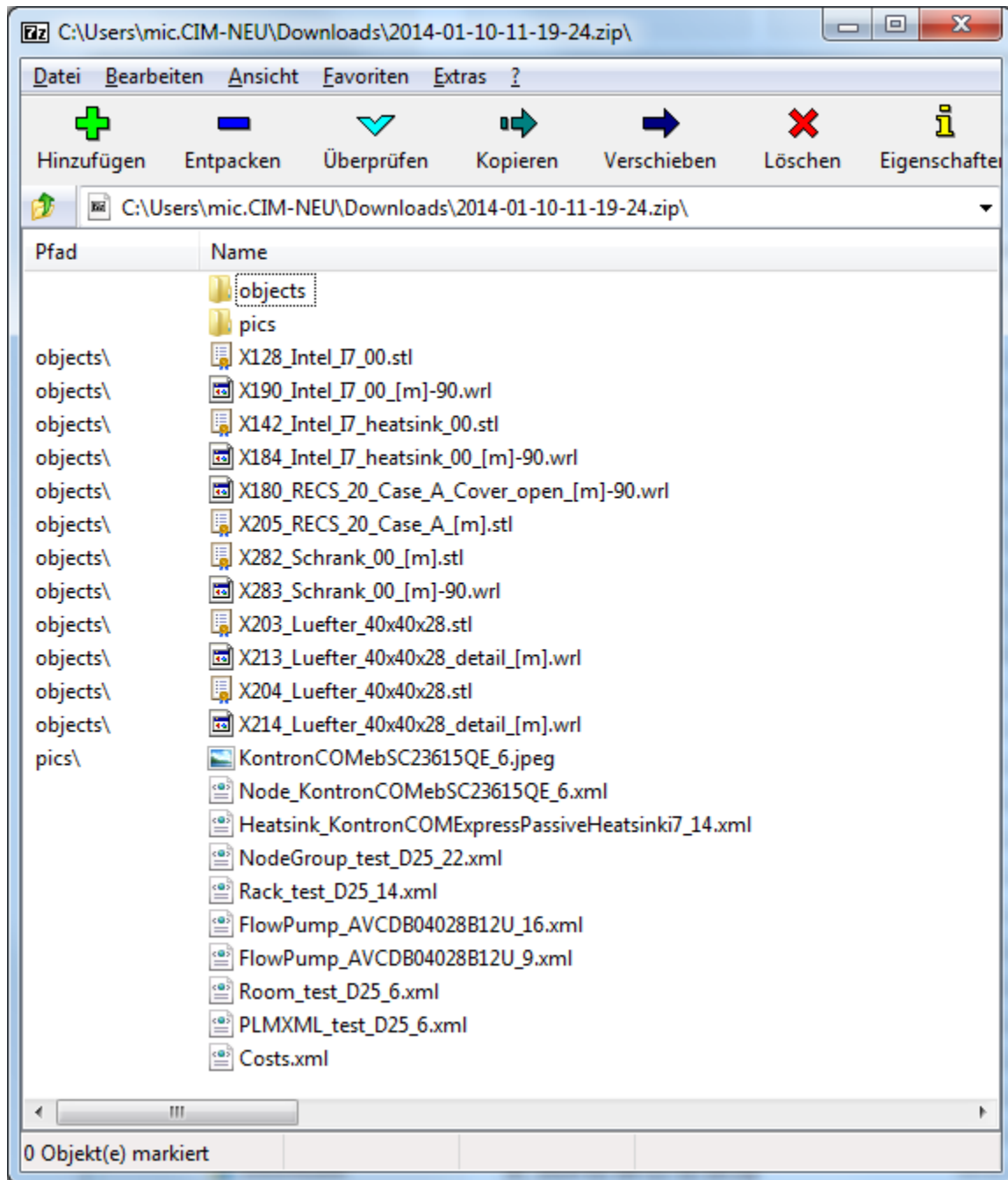


Figure 4-6: Exported room as ZIP file

All expected files are in the archive. The numbers at the end of the names were assigned by the DEBBConfigurator. The X*** numbers in front of the model names were assigned during the upload process.

As seen below the “Transform id” in line 30 shows the correct distances (X; Y; Z) in [m] for the Intel node in relation to the chassis. The last four numbers in a “Transform id” line give the X, Y, Z, coordinates followed by a 1 at the end of the

transformation, which is a scale factor for the whole object. These transformations are given for every part in the PMLXML and are added as shown at the beginning of the test by the software that uses them. The heatsink for example is listed with a 1cm Z-axis offset (line 15 below) in relation to the node it belongs to. This 1cm gap is listed as 0.01, because the standard scale unit for the DEBBconfigurator is in Meter.

```

1 <ProductRevisionView id="iview14_1" name="Heatsink_1">
2   <UserData id="iview14_1_1">
3     <UserValue value="Heatsink" title="DEBBLevel"></UserValue>
4     <UserValue value="KontronCOMExpressPassiveHeatsinki7_14"
5       title="DEBBComponentId"></UserValue>
6     <UserValue value="Heatsink_KontronCOMExpressPassiveHeatsinki7_14.xml"
7       title="DEBBComponentFile"></UserValue>
8   </UserData>
9   <Representation id="iview14_1_objects_X142_Intel_I7_heatsink_00_stl" format="STL"
10     location="./objects/X142_Intel_I7_heatsink_00.stl"></Representation>
11   <Representation id="iview14_1_objects_X184_Intel_I7_heatsink_00_m_90_wrl"
12     format="VRML" location="./objects/X184_Intel_I7_heatsink_00_[m]-
13     90.wrl"></Representation>
14 </ProductRevisionView>
15 <ProductInstance id="inst14_1" name="Heatsink_1" partRef="iview14_1">
16   <UserData id="inst14_1_1">
17     <UserValue title="power-sensor" value="power"></UserValue>
18     <UserValue value="Heatsink_inst14_1" title="label"></UserValue>
19   </UserData>
20   <Transform id="id14_01">1 0 0 0 1 0 0 0 0 1 0 0 0 0.01 1</Transform>
21 </ProductInstance>
22 <ProductRevisionView id="iview06_2" name="i7_0_01" instanceRefs="inst14_1">
23   <UserData id="iview06_2_1">
24     <UserValue value="Node" title="DEBBLevel"></UserValue>
25     <UserValue value="KontronCOMebSC23615QE_6" title="DEBBComponentId"></UserValue>
26     <UserValue value="Node_KontronCOMebSC23615QE_6.xml"
27       title="DEBBComponentFile"></UserValue>
28   </UserData>
29   <Representation id="iview06_2_objects_X128_Intel_I7_00_stl" format="STL"
30     location="./objects/X128_Intel_I7_00.stl"></Representation>
31   <Representation id="iview06_2_objects_X190_Intel_I7_00_m_90_wrl" format="VRML"
32     location="./objects/X190_Intel_I7_00_[m]-90.wrl"></Representation>
33 </ProductRevisionView>
34 <ProductInstance id="inst06_2" name="i7_0_01" partRef="iview06_2">
35   <UserData id="inst06_2_1">
36     <UserValue value="i7_0_inst06_1" title="label"></UserValue>
37   </UserData>
38   <Transform id="id06_02">-1 0 0 0 -0 -1 0 0 0 0 1 0 0.25634 0.29995 0.01739 1
39     </Transform>
40 </ProductInstance>

```

In Figure 4-7 the summed up results can be seen in COVISE as a correct positioned node, heatsink and chassis in the rack.

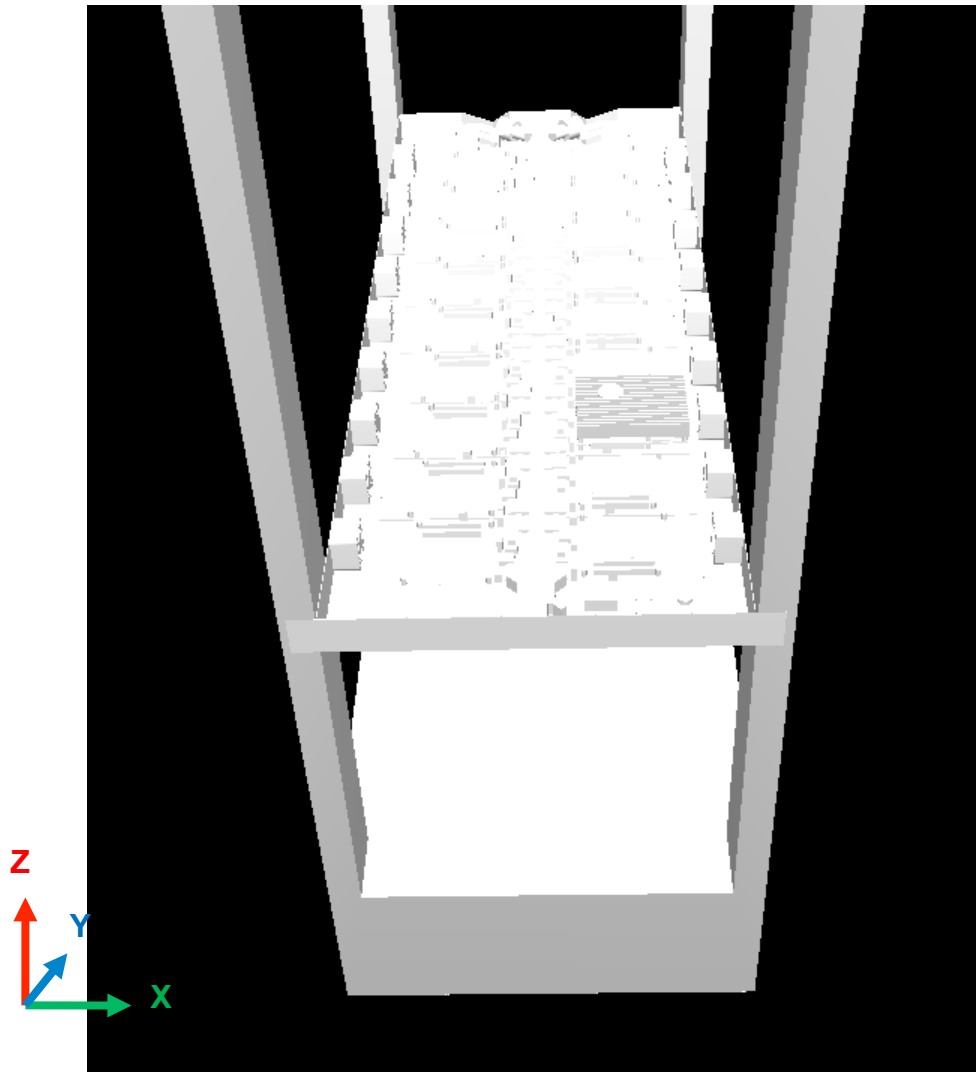


Figure 4-7: Positioned heatsink in the chassis

4.1.3 DCworms GUI

After setting up the simulation parameters with the experiment configurator, the user can proceed to the DCworms GUI window that provides the means to complete the workload simulation process. As mentioned in Section 2.3.3, DCworms GUI allows users to update DCworms with workload specification and information about management policy. Left panel allows users to browse the list of all the workloads uploaded to the repository, with respect to the hierarchy of created directories. Each selected workload is complemented with brief description of its main characteristics. They include number of jobs, their average runtime, arrival rate, etc. Furthermore, selection of workload results in displaying a list of application that constitutes the workload. Browsing that list allows users to view the details of their profiles (as presented in Figure 4-8).

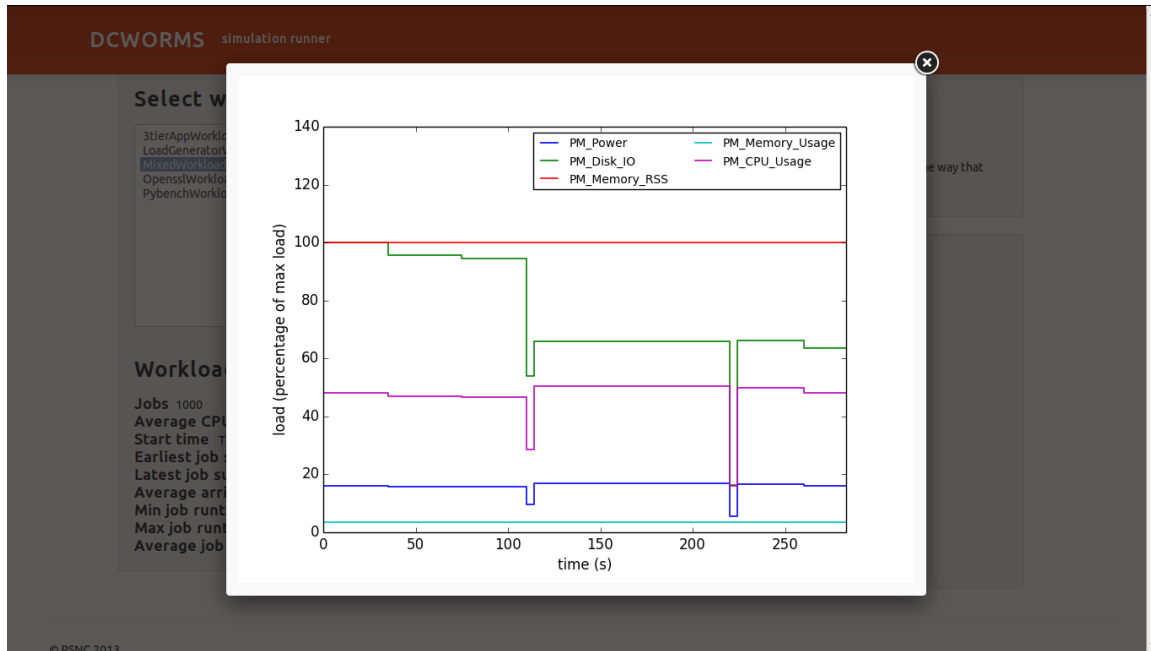
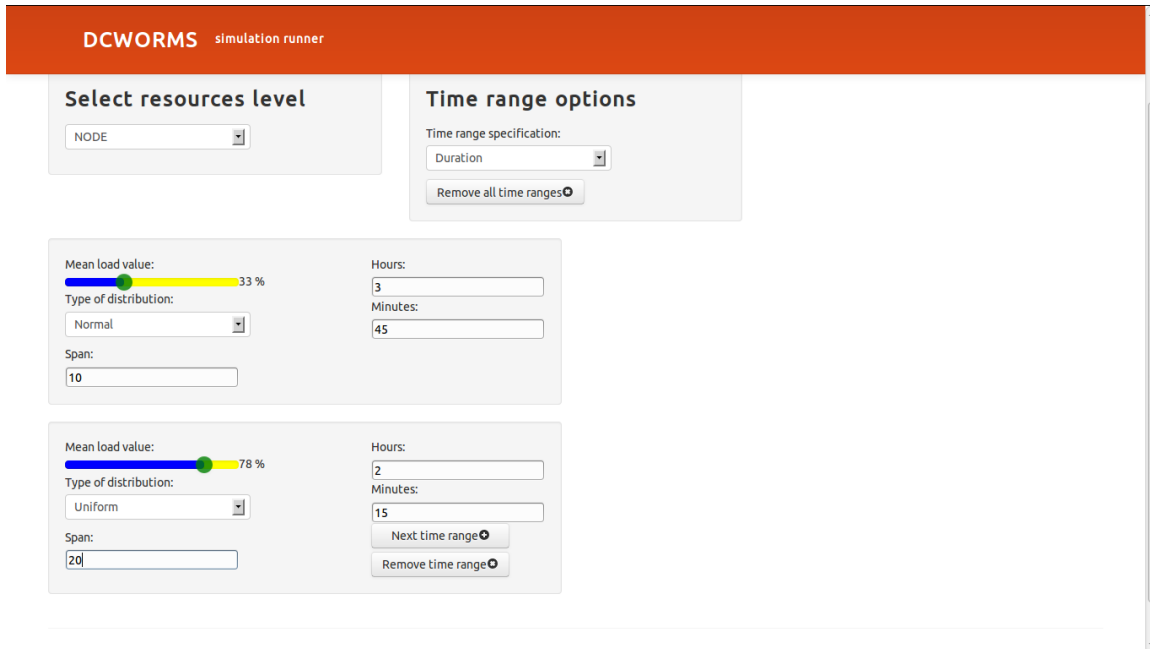


Figure 4-8: Preview of application profile within DCworms GUI

Finally, the upper right panel delivers information about the workload and resource management policy to be applied during the simulation. Sample configuration of the experiment was presented in Figure 2-16 in Section 2.3.3.

The second tab of DCworms GUI allows the specification of load calendars. For the given resource level, chosen from the list of defined levels for the analysed DEBB, the user can provide load distribution according to mean and standard deviation values. View of this panel can be seen in the following figure (Figure 4-9).



The screenshot shows the DCworms simulation runner interface. At the top, there is a red header with the text "DCWORMS simulation runner". Below this, the interface is divided into several sections:

- Select resources level:** A dropdown menu labeled "NODE".
- Time range options:** A section for "Time range specification" with a "Duration" dropdown and a "Remove all time ranges" button.
- Load specification (Top):** A panel with a "Mean load value" slider set to 33%, "Type of distribution" set to "Normal", and a "Span" input field with the value "10". To the right, there are input fields for "Hours" (3) and "Minutes" (45).
- Load specification (Bottom):** A panel with a "Mean load value" slider set to 78%, "Type of distribution" set to "Uniform", and a "Span" input field with the value "20". To the right, there are input fields for "Hours" (2) and "Minutes" (15), along with "Next time range" and "Remove time range" buttons.

Figure 4-9: DCworms GUI – load specification window

The “Start Simulation” button placed on the “Run Simulation” tab in DCworms GUI invokes simulation process. It also triggers the update of database with the information coming from DCworms GUI. Available console allows the user to follow the progress of the simulation. After the simulation is completed, all the gathered and calculated data are written into the database. Later on they can be analysed by Metric calculator or viewed in MOP GUI. Apart from that, an overview of energy efficiency metrics is also presented by DCworms GUI, in additional dialog window (as shown in Figure 4-10).

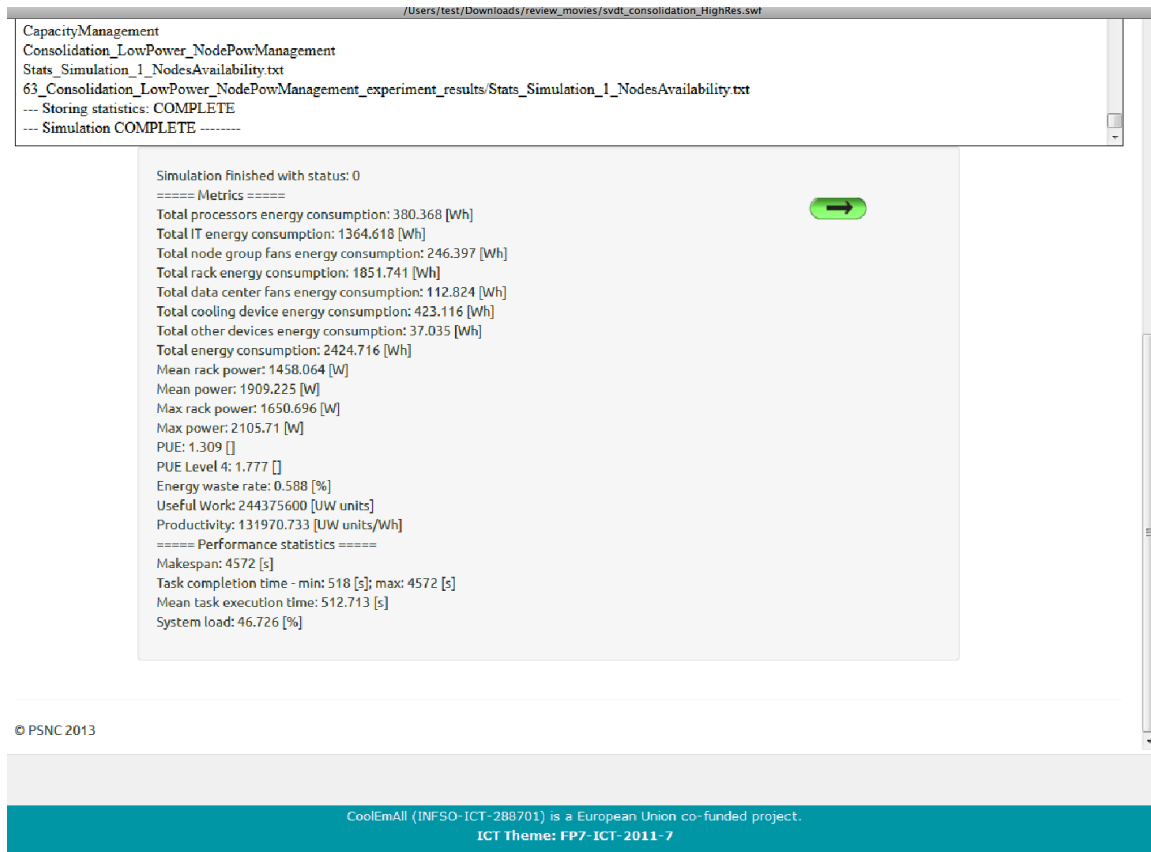


Figure 4-10: DCworms GUI - energy efficiency metrics

4.1.4 MOP-GUI

There are two available modes of displaying MOP-GUI:

- Standard mode, which presents one 3D view and three charts. In this mode the user can view results of one experiment. The standard mode is presented in Figure 4-11
- Comparison mode, which presents two 3D views and three charts, each chart can be assigned to show metrics for first or second 3D window. 3D windows can present different experiments/trials/models. In this mode the user can compare results of two experiments regardless of whether it is a simulation result or real testbed measurements. The comparison mode is presented in Figure 4-12

In this document there are described only basic MOP-GUI test; for further information please check the D4.7 - Second prototype of Module Operation Platform GUI

Both modes are accessible from the main portal menu as subtabs of the MOP tab. In each case MOP-GUI presents the model and results that are assigned to those chosen by the user experiment and trial. Experiment and trial can be also

set inside MOP-GUI. In this case the user can easily choose experiments to compare and quickly change the presented case.

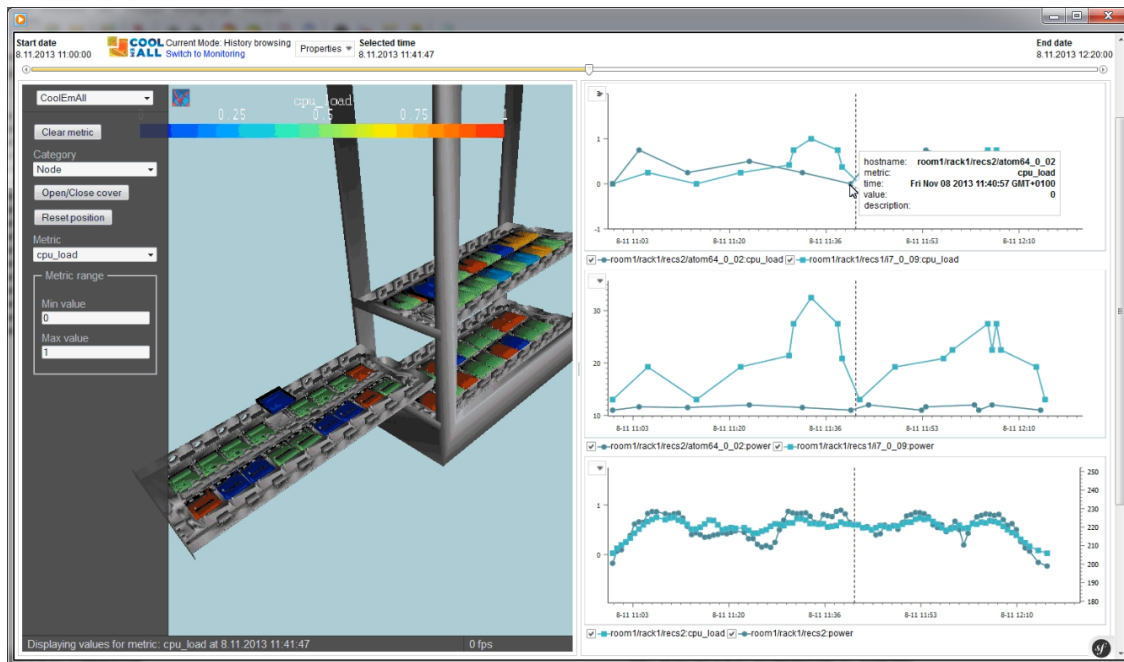


Figure 4-11: MOP-GUI - Standard mode view

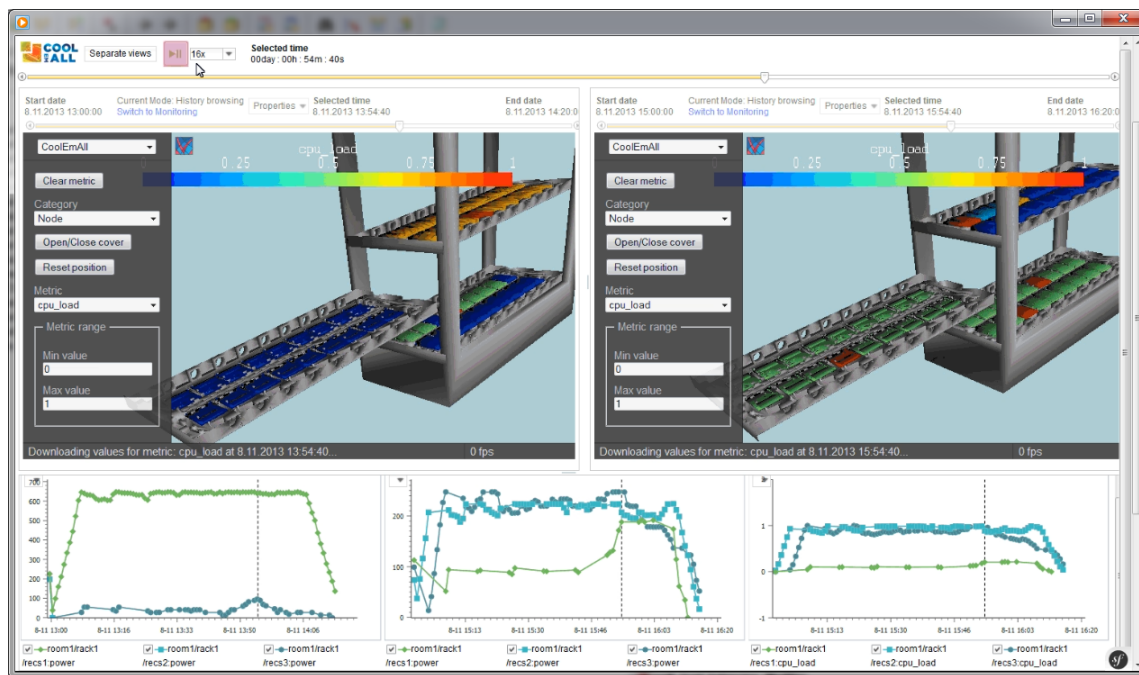


Figure 4-12: MOP-GUI - Comparison mode view

4.1.5 COVISE-GUI

As noted, COVISE GUI offers web-based graphical interfaces presenting entire simulation results of the CFD-simulation, visualizing air flow across all building blocks (DEBBs) and enabling interaction with the simulation, allowing the user to change interactively the simulation parameters that affect position (arrangement) of objects. The figure below visualizes COVISE-GUI capable of rearranging positions of racks (left) represented by shapes and visualizes results of CFD simulation (right). After rearranging the positions of racks in the room and pressing simulation button, the visualisation of the simulation is updated according the new positions, as presented in Figure 4-13.

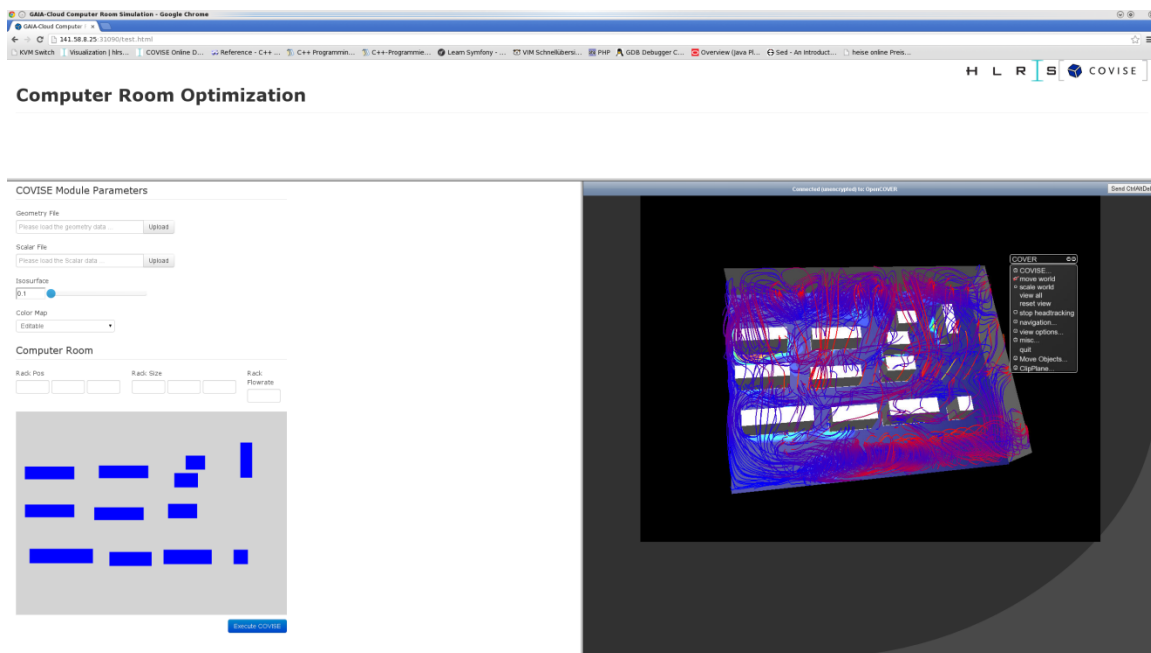


Figure 4-13: COVISE GUI

4.1.6 Metric Calculator and Report-GUI

Once an experiment and trial have been selected, we can access the Reports TAB and go to the Metrical Calculator sub-menu allowing us to interact with the Metric Calculator component.

In order to test the Report-GUI and its interaction with the Metric Calculator we tried many different usage scenarios.

When a trial has not been correctly selected the following error will be displayed:

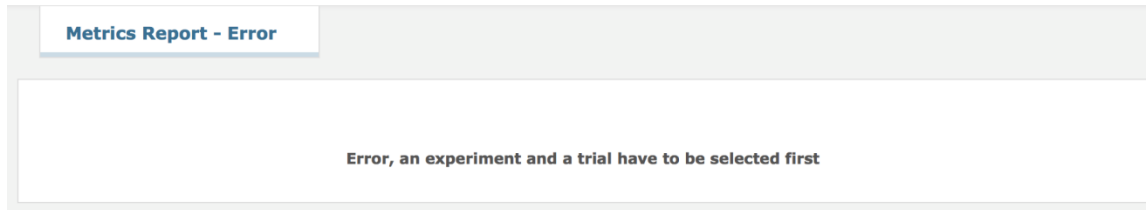


Figure 4-14: Report-GUI experiment selection error

When a report has not yet been calculated the following error will be displayed:

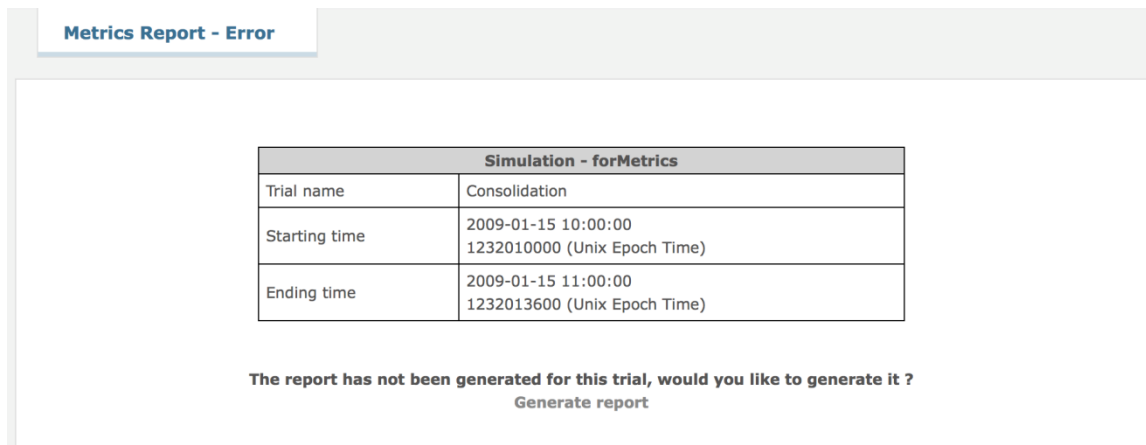


Figure 4-15: Report-GUI missing report error

When a report is currently being generated the following error will be displayed:

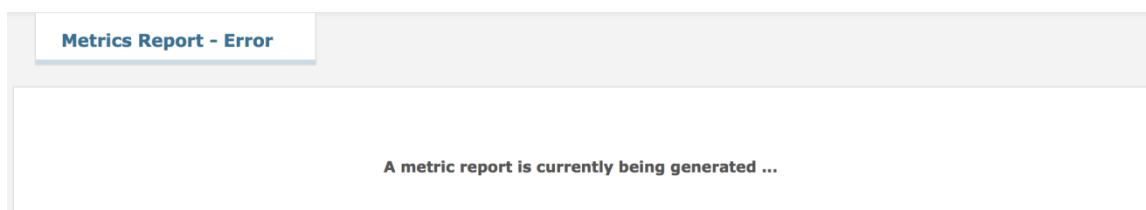


Figure 4-16: Report-GUI currently computing error

When everything is set and computed, the Report-GUI component will display the resulting metrics:

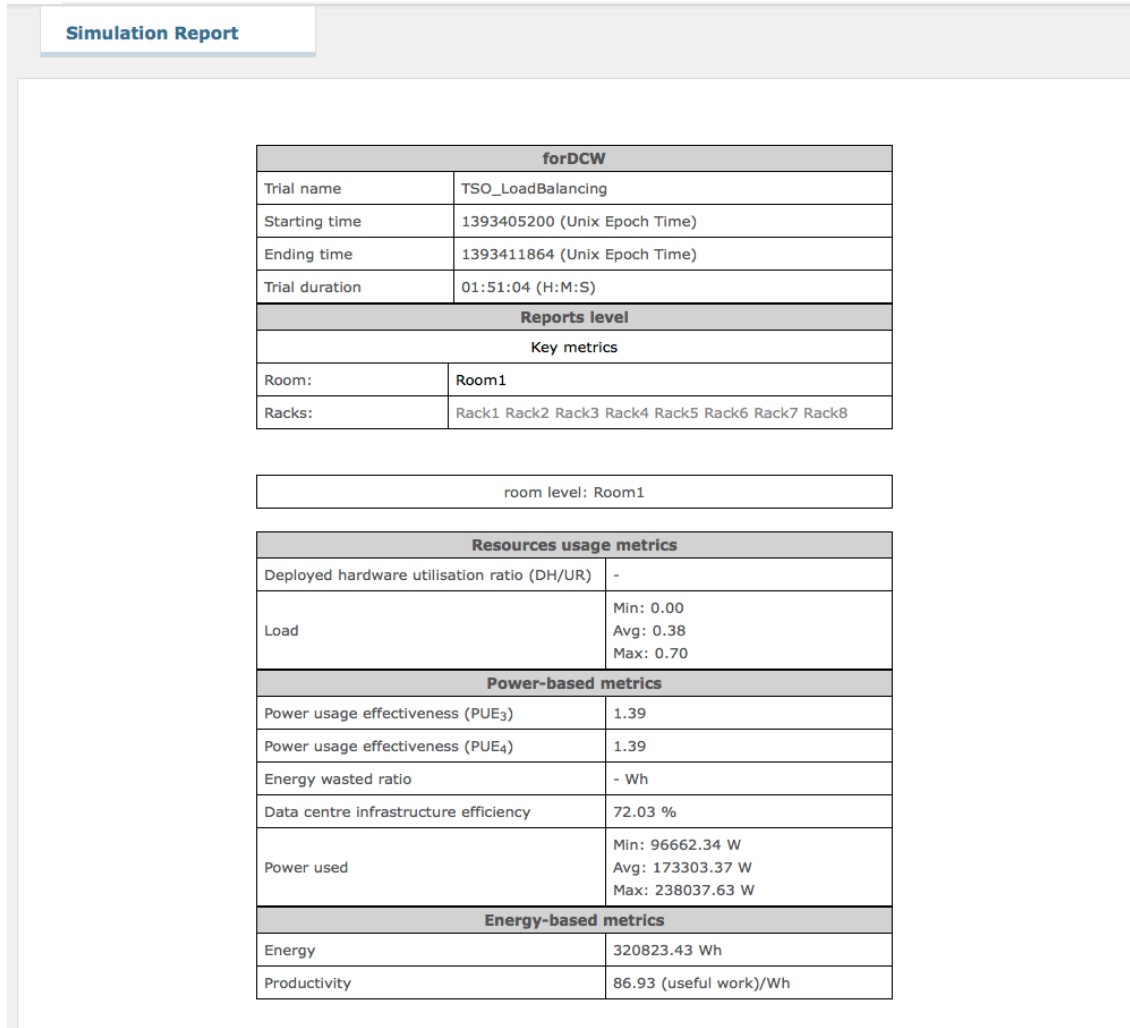


Figure 4-17: Report-GUI normal report

In addition to these basic scenarios we tried the Report-GUI with every different scenario stored in the experiment database of the Web-GUI.

4.2 Test of the SVD-Toolkit components

In this section we describe tests of SVD-Toolkit components that are accessed by the command line and are similar to v1 of the Toolkit, including Application profiler, Repository, Database, Data Centre workload simulator, CFD solver, COVISE and metric calculator. The content in this section is originated mostly from D2.4, extended by section 4.2.6.

4.2.1 Application Profiler

The application profiler was tested by creating the profile of one the HPC benchmark EP with different frequencies. The faster the processor, the lower the number of phases that are detected because some slight behaviour changes do not have enough impact at faster speed to be detected as new phases.

Profile at fastest speed

```

<resourceConsumptionProfile>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT4S</duration>
    <behaviour name="cpu">
      <value>99</value>
    </behaviour>
    <behaviour name="network">
      <value>29</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT67S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>88</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT8S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>29</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT63S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>85</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT69S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>85</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT61S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>80</value>
    </behaviour>
  </resourceConsumption>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT72S</duration>
    <behaviour name="cpu">
      <value>98</value>
    </behaviour>
    <behaviour name="network">
      <value>76</value>
    </behaviour>
  </resourceConsumption>
</resourceConsumptionProfile>

```

Profile at slowest speed

```

<resourceConsumptionProfile>
  <resourceConsumption>
    <referenceHardware>Intel_i7</referenceHardware>
    <duration>PT4S</duration>
    <behaviour name="cpu">
      <value>99</value>
    </behaviour>
    <behaviour name="network">
      <value>28</value>
    </behaviour>
  </resourceConsumption>

```

```

</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT46S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>92</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT18S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>38</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT72S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>90</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT12S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>40</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT72S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>90</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT4S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>44</value>
  </behaviour>
</resourceConsumption>
<resourceConsumption>
  <referenceHardware>Intel_i7</referenceHardware>
  <duration>PT93S</duration>
  <behaviour name="cpu">
    <value>98</value>
  </behaviour>
  <behaviour name="network">
    <value>89</value>
  </behaviour>
</resourceConsumption>
</resourceConsumptionProfile>

```

4.2.2 SVN Repository

As noted, the interaction with SVN repository is done via svn-client (svn command). The following example shows interaction with the svn:

```
volk@timacs:~/coolemall-repository$ svn co https://svn.coolemall.eu/svn/repository
```

```
A repository/users
```

```

A repository/common
A repository/common/debbs
A repository/common/workloads
A repository/common/applications
Checked out revision 51.
volk@timacs:~/coolemall-repository$ cd common/
volk@timacs:~/coolemall-repository/repository/common$ ls
applications debbs workloads
volk@timacs:~/coolemall-repository/repository/common$ mkdir experiments
volk@timacs:~/coolemall-repository/repository/common$ ls
applications debbs experiments workloads
volk@timacs:~/coolemall-repository/repository/common$ svn ci
Adding      common/experiments
Committed revision 52.
volk@timacs:~/coolemall-repository/repository/common$

```

4.2.3 Database

To execute the Python wrapper it is necessary to start the coolemalldb demon:

```

[timacs@recs1 coolemall] ./bin/coolemall
usage: coolemall <option>
    -h|--help          This blurb.
    -k|--kill|--stop  Stop coolemalldb.
    -s|--start        Start coolemalldb.
    -i|--status       Show status of coolemalldb.

```

Example invocation:

```
coolemall --start
```

The test of the collemall DB using script based API (described in Section 3.3 in D2.4) are show below:

```

[timacs@recs1 coolemall]$ ./coolemall_getRecordsByMetricName
testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05 mem_usage
[(1361574545L, 42356.230000000003, 'ok-low')]

```



```
[timacs@recs1 coolemall]$ ./coolemall_getMetricNames
```

```
testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05
```

```
[cpu_usage, mem_usage, usr_load]
```

```
[timacs@recs1 coolemall]$ ./coolemall_getHostNames
```

```
[testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05,      testbed/hlrs/hpc/hw/rack1/recs1,
testbed/hlrs/hpc/hw/rack1]
```

```
[timacs@recs1 coolemall]$ ./coolemall_getLastMetricsByHostName
```

```
testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05
```

```
[Metric(name='cpu_usage',   output='ok-low',   value=6.21,   source='nagios',
host='testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05',   time=1361574550L,
performance='ok-2.5'),   Metric(name='mem_usage',   output='ok-low',
value=42356.230000000003,   source='nagios',
host='testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05',   time=1361574545L,
performance='ok-1.5'),   Metric(name='usr_load',   output='ok-low',
value=12.210000000000001,   source='nagios',
host='testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05',   time=1361574525L,
performance='ok-2.5')]
```

```
[timacs@recs1 coolemall]$ ./coolemall_getLastMetricByMetricName
```

```
testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05 cpu_usage
```

```
host = testbed/hlrs/hpc/hw/rack1/recs1/i7_0_05
```

```
name = cpu_usage
```

```
output = ok-low
```

```
performance = ok-2.5
```

```
source = nagios
```

```
time = 1361574550
```

```
value = 6.21
```

```
[timacs@recs1 coolemall] ./coolemall_getExperimentsList
```

```
[exp_1, exp_2, exp_3]
```

```
[timacs@recs1 coolemall] ./coolemall_getTrialsList exp_1
```

```
[trial_1, trial_2, trial_3]
```

#Including data onto database

```
[timacs@recs1 coolemall] ./coolemall_putMetricDB "experimentID: exp_1,
trialID: trial_1, name: cpu_usage, time: 1361574539, value: 12356.21, object_path: tes
tbed/hlrs/hpc/hw/rack1/recs1/i7_0_05, source: nagios, performance: ok-
2.5, output: ok-low"
```

```
[timacs@recs1~]$ coolemall_getSelectedMetric
sim/expSelected/tSelected/cfd/hw/hlrs/rack1/recs1/inlet_01 test_selected
1393842120 1393843981
('499.683',)
```

4.2.4 DCworms

Apart from simulation of complex distributed computing systems, DCworms has been also used to simulate execution of workloads on resources defined by DEBBs for RECS. Results of this work have been presented in [DCworms2012] and in [CoolEmAll_RECS].

In [CoolEmAll_RECS] the impact of resource allocation policies on power draw and outlet temperatures of RECS system was studied. Based on DEBB a description of RECS unit containing 18 Intel i7 nodes was built.

The evaluated workload had the given characteristics (Table 4-1):

Table 4-1: Workload characteristics

load intensity	70%
number of tasks	1000
tasks interval [s]	560
Application classes	scalable CPU-intensive – 34% single threaded – 33% IO-intensive – 33%

As a workload management policy simple FCFS with Relaxed Backfilling approach strategy assigning tasks to nodes in Random manner was used.

Power estimations were based on power measurements made for each application type.

The following figure (Figure 4-18) shows the power consumption chart generated based on the data gathered during the DCworms simulation.

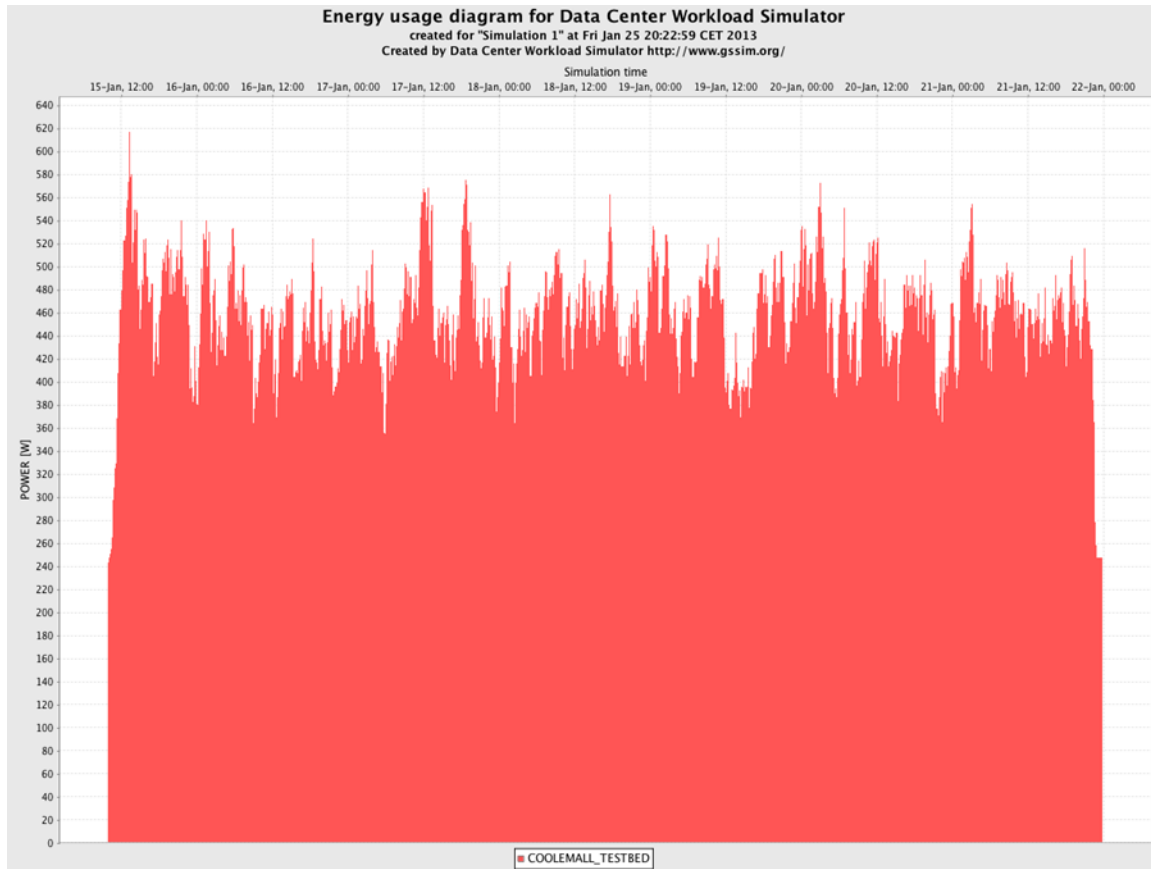


Figure 4-18: Power usage chart generated for the DCworms simulation

4.2.5 CFD simulation using OpenFOAM

For test purposes to main test cases on RECS level are considered and in this stage executed. First there is the flow through a compute node, in this case the RECS-design of project partner Christmann is used for a reference case. Second the test was done on a random compute room. These test cases were considered most viable because these cases are most likely to be used by end users.

4.2.5.1 Flow through RECS

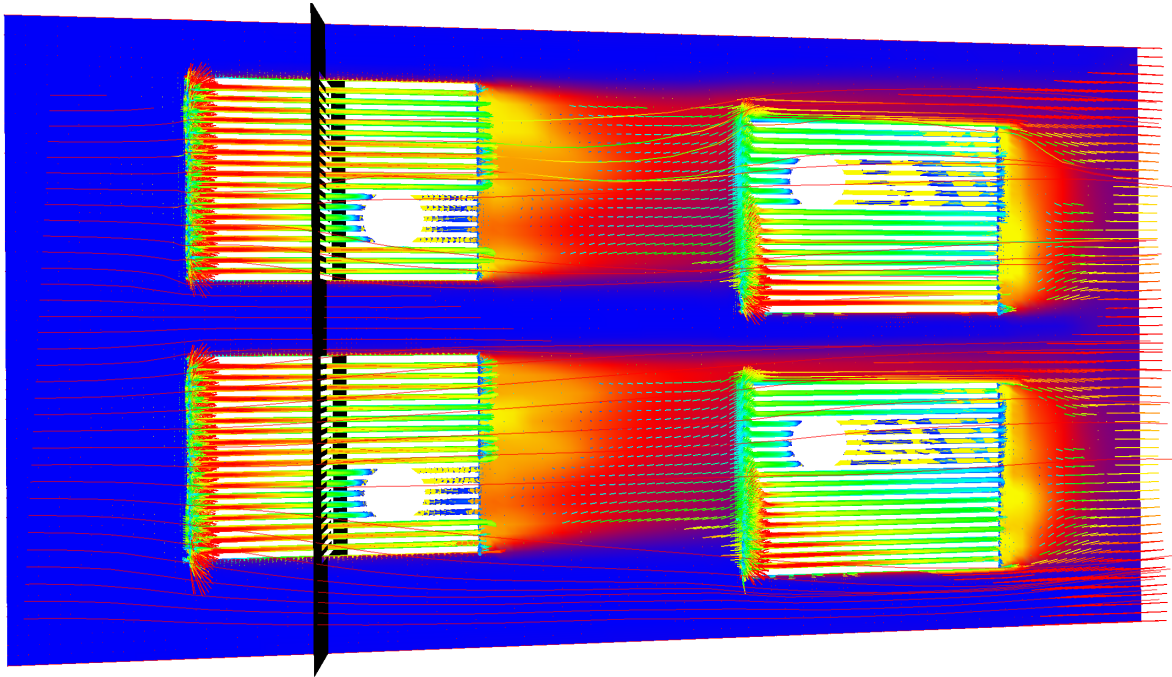


Figure 4-19: velocity and temperature distribution inside RECS

This figure shows the velocity and temperature distribution inside a compute node of RECS type. The colour along the plane represents the temperature distribution in conjunction with the heatsinks of the CPUs. The streamlines and the velocity vectors in conjunction with their colour represent the velocity distribution. Hot temperature is coloured in red and colder temperature goes over green to blue. Velocity is coloured in a similar way and red means high velocities.

4.2.5.2 Flow through Compute Room

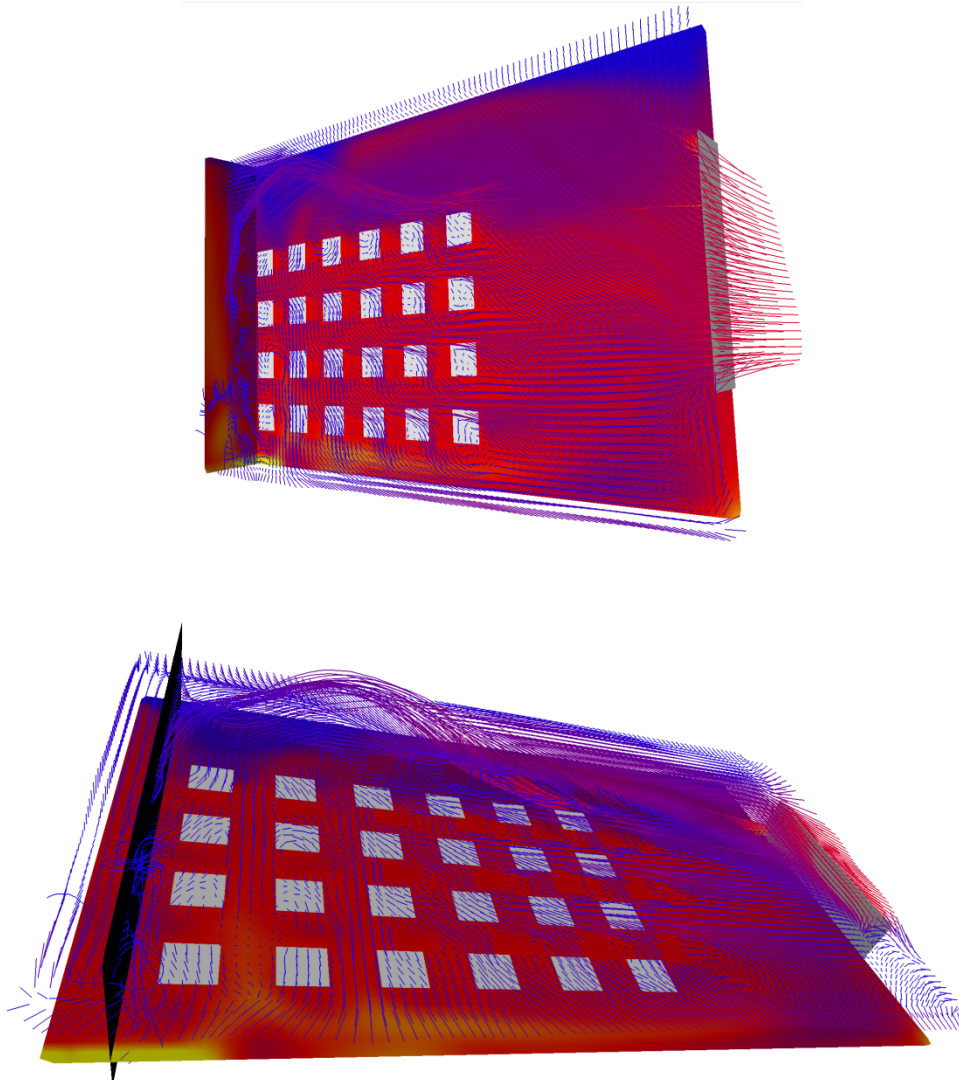


Figure 4-20: velocity and temperature distribution inside a compute room

Figure 4-20 shows temperature and velocity distribution inside a randomly chosen compute room. The 24 squares represent the inlets for the air inflow, which are also the top of the server racks inside this compute room. Temperature is again represented by the colour of the cutting plane. Velocity magnitude and velocity direction is shown by the colour and direction of the streamlines and the velocity vectors.

4.2.6 CFD simulation using COVISE with Ansys CFX

As described in scope of section 3.2.6, the simulation of heat-flow in server-room was done using COVISE with Ansys CFX solver. Figure 3-6 presented in section 3.2.6 visualized configuration of the workflow within the COVISE environment used (i) to integrate input parameters coming from various sources necessary to setup simulation, (ii) execute simulation remotely using corresponding (CFX) solver, (iii) post-process and visualize results using renderer (presented in the Figure 3-6 on the right).

Another configuration of the server-room and visualisation of the simulation results using COVISE and its render is presented in Figure 4-21.

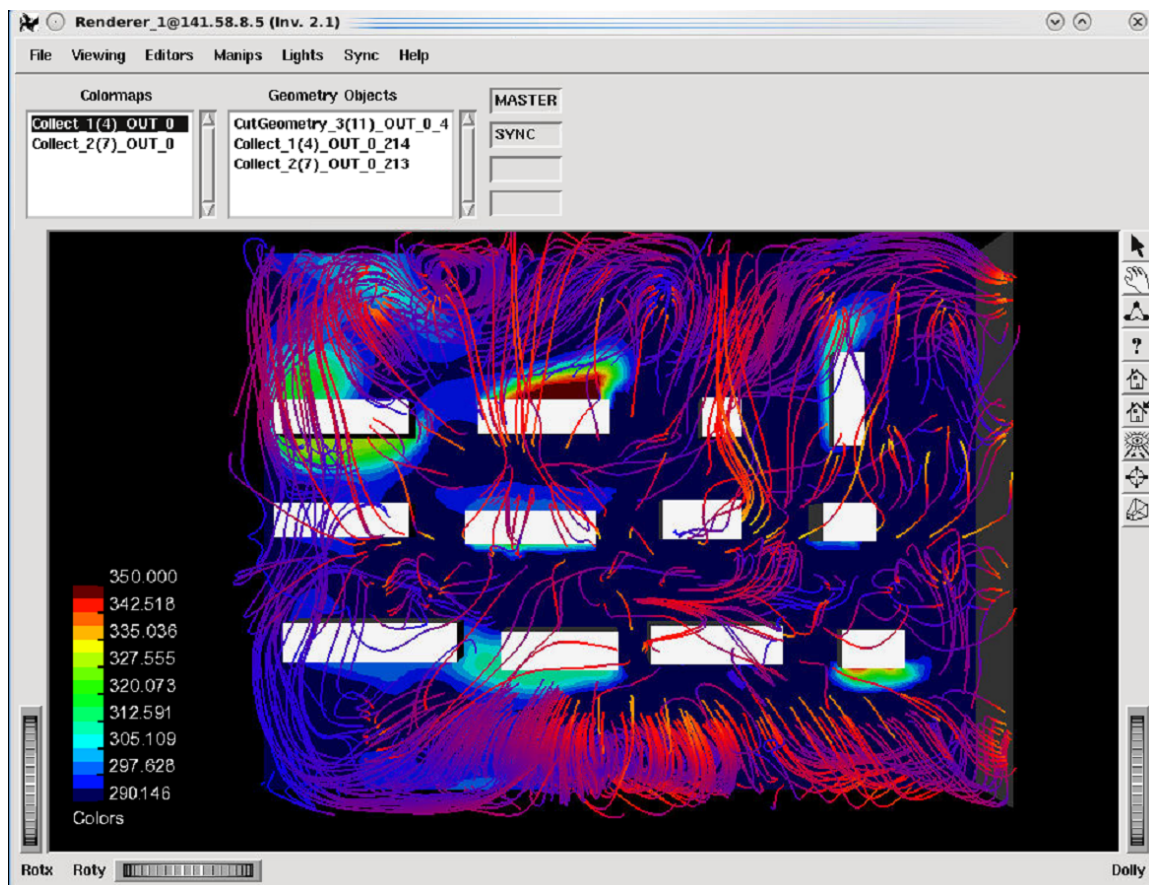


Figure 4-21: Visualisation of the heat-flow distribution within a room using COVISE renderer

4.2.7 Metric Calculator

In this section we present tests of the Metric Calculator. The Metric Calculator has been used to calculate metrics on a wide variety of experiments made both in a real and simulated environment. The output has been compared to results

obtain by calculating metrics using some other tools (including Excel sheets) in order to check their validity.

Some examples of calculations are shown in the following lines. Timestamps are based on seconds since standard epoch of 1/1/1970.

Imbalance of temperature calculation (testbed):

Calculation of the imbalance of temperature between the nodes of the node-group recs1

```
[user@recs1 MetricCalc]$ ./metricCalc.py imbalance_of_temperature
testbed/psnc/hpc/hw/rack1/recs1 1362752604 1362755664

[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName
testbed/psnc/hpc/hw/rack1/recs1/ ImbalanceOfTemperature -From:1362752604-To:1362755664

    host = testbed/psnc/hpc/hw/rack1/recs1
    name = ImbalanceOfTemperature -From:1362752604-To:1362755664
    output = OK
    performance =
    source = metricCalc
    time = 1362755664
    value = 19.2183893608
```

Power calculation (testbed):

Calculation of the power consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py power_usage testbed/psnc/hpc/hw/rack1/recs1
1362752604 1362755664

[user@recs1 MetricCalc]$ coolemall_getRecordsByMetricName testbed/psnc/hpc/hw/rack1/recs1
Power 1362752604 1362755664

    [NumRecord(1362752609000000000L, 252.0, 'OK'),
    NumRecord(1362752621000000000L, 229.0, 'OK'),
    NumRecord(1362752633000000000L, 250.0, 'OK'),
    NumRecord(1362752645000000000L, 239.0, 'OK'),
    NumRecord(1362752657000000000L, 224.0, 'OK'),
    NumRecord(1362752669000000000L, 238.0, 'OK'),
    NumRecord(1362752681000000000L, 224.0, 'OK'), ...]
```

Minimum power calculation (testbed):

Calculation of the minimum power consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py minimum_power testbed/psnc/hpc/hw/rack1/recs1
1362752604 1362755664
```

```
[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName
testbed/psnc/hpc/hw/rack1/recs1/ MinPower-From:1362752604-To:1362755664
    host = testbed/psnc/hpc/hw/rack1/recs1
    name = MinPower-From:1362752604-To:1362755664
    output = OK
    performance =
    source = metricCalc
    time = 1362755664
    value = 215.0
```

Maximum power calculation (testbed):

Calculation of the maximum power consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py maximum_power testbed/psnc/hpc/hw/rack1/recs1
1362752604 1362755664
```

```
[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName
testbed/psnc/hpc/hw/rack1/recs1/ MaxPower-From:1362752604-To:1362755664
    host = testbed/psnc/hpc/hw/rack1/recs1
    name = MaxPower-From:1362752604-To:1362755664
    output = OK
    performance =
    source = metricCalc
    time = 1362755664
    value = 275.0
```

Average power calculation (testbed):

Calculation of the average power consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py average_power testbed/psnc/hpc/hw/rack1/recs1
1362752604 1362755664
```

```
[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName
testbed/psnc/hpc/hw/rack1/recs1/ AvgPower-From:1362752604-To:1362755664
    host = testbed/psnc/hpc/hw/rack1/recs1
    name = AvgPower-From:1362752604-To:1362755664
    output = OK
    performance =
    source = metricCalc
    time = 1362755664
    value = 236.233910486
```


Energy consumption calculation (testbed):

Calculation of the energy consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py energy testbed/psnc/hpc/hw/rack1/recs1 1362752604 1362755664
```

```
[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName testbed/psnc/hpc/hw/rack1/recs1/ Energy-From:1362752604-To:1362755664
```

```
host = testbed/psnc/hpc/hw/rack1/recs1
name = Energy-From:1362752604-To:1362755664
output = OK
performance =
source = metricCalc
time = 1362755664
value = 200.864444444
```

Power Usage Effectiveness (simulation):

Calculation of the average power consumption of the node-group recs1.

```
[user@recs1 MetricCalc]$ ./metricCalc.py pue sim/CapacityManagement/LoadBalancing/DCWORMS/metrics/room1
```

```
[user@recs1 MetricCalc]$ coolemall_getLastMetricByMetricName sim/CapacityManagement/LoadBalancing/DCWORMS/metrics/room1 MC_PUE-From_1383904800000-To_1383919634000
```

```
host = sim/CapacityManagement/LoadBalancing/DCWORMS/metrics/room1
name = MC_PUE-From_1383904800000-To_1383919634000
output = ok
performance = ok
source = metricCalc
time = 1383919634000
value = 1.28280601679
```

5 Summary

In scope of D2.4 we described realization of the 1st prototype of the SVD-Toolkit, demonstrating its functionality and capability to simulate and assess efficiency of various configurations of servers and data centres. However, the usage of the 1st prototype required expert knowledge to apply its command line based interfaces for data centre optimisation. To overcome the high expertise required and to simplify usage of the SVD-Toolkit, we developed in scope of the final prototype web based Graphical User Interfaces to SVD-Toolkit, summarized as CoolEmAll-Web-GUI, allowing interacting with the SVD-Toolkit and visualizing its results. The CoolEmAll-Web-GUI comprises several GUIs integrated into the common web based GUI environment each capable of interacting with the corresponding SVD-Toolkit component through a web page in a guided manner. Such a web based CoolEmAll-Web-GUI simplifies usage of the SVD-Toolkit and makes installation of the SVD-Toolkit components at user-side unnecessary, as it is sufficient to install components at the provider sides and access them remotely via web-interfaces.

In this deliverable we described realization, usage and test of the CoolEmAll-Web-GUI (and its components), along with the components of the final prototype of SVD-Toolkit. The test and usage described in this deliverable demonstrated that SVD-Toolkit and its web based interfaces can be easily used even by a non-skilled users: (i) to define parameters necessary for execution of simulation, (ii) to design data centre building blocks (DEBBs) on various level of granularity, (iii) to select application- and workload-profiles, execute workload simulation and find the best scheduling strategy for energy-savings, (iv) to visualize results of the workload simulation while comparing outcomes with real measurements, (v) to setup and simulate heat-flow distribution within a server or a data centre and visualize results, and finally (vi) to calculate power-consumption and assess energy- and cooling-efficiency of various DEBB configurations, applications, workload and environmental conditions.

Extending the SVD-Toolkit by CoolEmAll-Web-GUI completes CoolEmAll's vision and its holistic approach in the design and operation of data centres, enabling full life-cycle optimisation of cooling- and energy-efficiency of data centres on various scale level. The full life-cycle optimisation offered by SVD-Toolkit to the users is achieved by enabling: (a) modular design, configuration and arrangement of data centre components on various scale level (DEBB design) taking hardware and facility characteristics into account, (b) capturing application profiles in real environment and generating workload-profiles reflecting real demand of various applications and use-groups, (c) evaluating various workload scheduling policies for selected hardware configuration, applications profiles and workloads, to optimize operation and increase energy-savings, (d) evaluating interactively heat-flow distribution in various data centre configurations to optimize data centre layout and improve cooling-efficiency, (e) assessing all above mentioned aspects in a report, providing metrics on productivity, energy-efficiency, cooling-efficiency, monetary- and CO₂ costs of data centres configuration and operation. Such an

approach and capabilities offered by SVD-Toolkit achieves the objective of the CoolEmAll project to enable designers and operators of a data centre to reduce its energy impact by combining the optimization of IT, cooling and workload management.

6 Annex A. Description of test implemented to assess CoolEmAll methodology

Model a Data Centre using CoolEmAll Data Efficiency Building Blocks

Description:

Model a room with DEBB configurator using some pre-modeled nodes to simplify the configuration process.

Test steps:

- 1) Access **CoolEmAll WEB-GUI**:
<http://sf2.coolemall.eu>
- 2) Login with credentials (username/password): test/test
- 3) Go to **DEBB tab** > From DEBB configurator tab we are able to setup and configure different components for each of the levels (component, node, node group, rack, room)
- 4) Navigate through the DEBB configurator menu (left side) to get an overview of the different components that we are able to model (baseboards, processors, memories, heatsinks, power supplies, nodes, etc.).



Figure 22: DEBB configurator menu

- 5) Go to **NODE GROUPS** sub-menu to start the creation of a node group, in this step we are going to create a Resource Efficient Computing & Storage (RECS) server hardware using pre-configured nodes. Click "**CREATE NODE GROUP**" at the bottom and fill the form using the following sample parameters:

Table 6-1: Node Groups parameters

Computer ID	Manufacturer	Product
MyRECS	Chistmann	RECS Box Compute MyRECS

Label (not mandatory)	Hostname (not mandatory)	Instance name (not mandatory)
Chassis (After selection the nodes position have been displayed) Chirstman RECS Box Compute Unit	Min allowed temperature (not mandatory)	Max allowed temperature (not mandatory)
Type Intel i7	Power usage profile (not mandatory)	Mesh resolution 0 0 0
Location in mesh 0 0 0	XML name myrecs	Costs [EUR] (not mandatory)
Costs [CO2] (not mandatory)	Networks Generic 100Mbit	

Populate the compute unit selecting “**Kontron COMe-bSC2 - 3615QE**” or “**Kontron COMe-bSC2 - 2715QE**” (or any other supported node) at your choice as nodes within the RECS, click **ADOPT** to add your unit. When all the nodes are selected press **SAVE** button, afterwards you will be able to use your new RECS in a RACK. At this stage you should be able to download the archive that represents the compute node.

You can upload the new RECS to the common repository from the **NODE GROUPS** menu, clicking **Add to svn** button.

SVN Upload ×

Change SVN upload path if necessary (the content of the directory will be overwritten):

/repository/common/debbs/trunk/PSNC_Little_Server_Room/

The directory would be deleted completely.

Close
Upload

Figure 23: SVN upload path

- 6) Afterwards, go to **RACKS sub-menu**, you will be able to insert RECS servers into the RU, clicking into the RACK figure displayed at top-left in the main window.

Table 6-2: RACK parameters

Computer ID	Manufacturer	Product
MyRACK	Chistmann	Testbed_Rack
Label	Costs [EUR]	Costs [CO2]
(not mandatory)	(not mandatory)	(not mandatory)
Hostname	XML Name	Power usage profile
Chirstman RECS Box Compute Unit	myrack	(not mandatory)
Type	Instance name	Width [m]
Intel i7	(not mandatory)	(0.5)
Height [m]	Depth [m]	Gap bottom [m]
(1.7)	(1.15)	0.2
Gap left [m]	Rack size [RU]	Flow direction
0.037	(choose the rack size. i.e:20)	(not mandatory)
Mesh resolution	Location in mesh	Current power usage
0 0 0	0 0 0	(not mandatory)

Populate the rack unit selecting the node group created in the previous step.

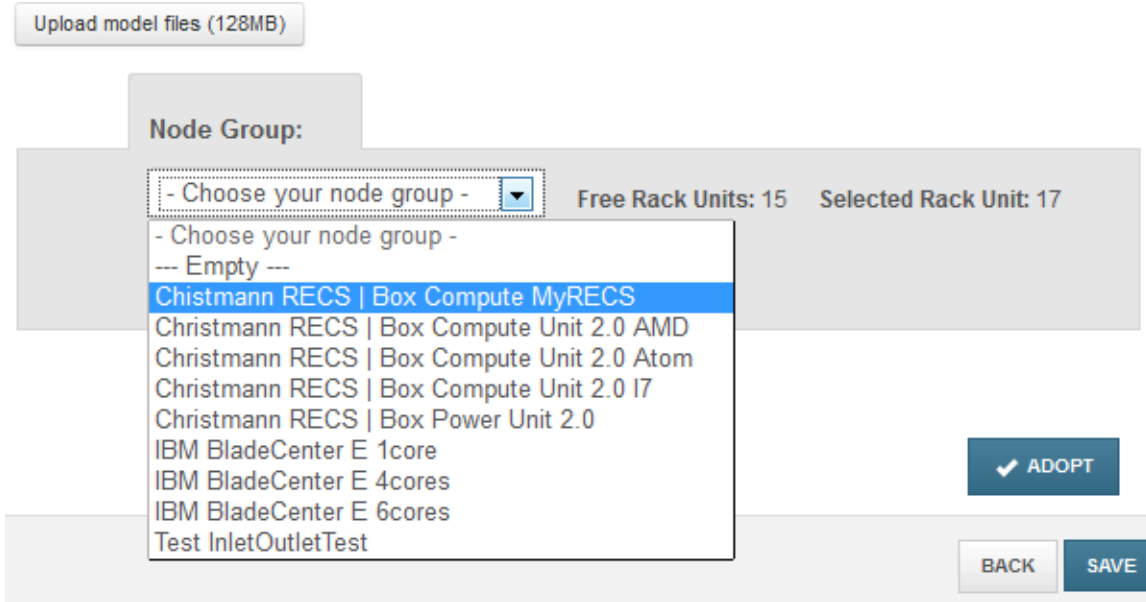


Figure 24: Node Group break-down menu

When the rack units are populated press **SAVE** button, afterwards you will be able to use your new RACK in a room. At this stage you should be able to download the archive that represents the rack.

You can upload the new RACK to the common repository from the **RACK** menu, clicking **Add to svn** button.

- 7) After the creation of the RACK, you should be able to populate a room with your already created RACK composed by RECS. Go to **Room sub-menu** and use your configured racks to populate the room:

Table 6-3: Room parameters

Computer ID	Name	Building
MyRoom	MyLittleRoom	PSNC
Height[m]	XML name	Mesh resolution
2.5	myroom	0 0 0
Location in mesh	Costs [EUR]	Costs [CO2]
0 0 0	(not mandatory)	(not mandatory)

Select the components of your room by clicking into them, the element selected will appear into the graph window automatically, the graph window can be resized dragging the right-bottom corner.

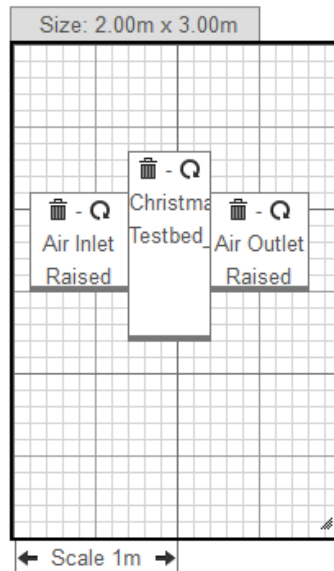


Figure 25: Room graph window

When all the components are selected press **SAVE** button, afterwards you will be able to use your new room. At this stage you should be able to download the archive that represents the **DEBB**.

Results:

PLXML file describing the DEBB

Simulate a Data Centre Workload using DCworms to compare different resource scheduling policies

Description:

DCworms is a simulation tool that can be used to verify power usage and thermodynamic models based on selected scheduling algorithm, resource allocation algorithm and resource management policy. The steps described below will compare the simulation of OpenSSL workload using a load balancing strategy versus a consolidation strategy in low-power nodes. The DEBB input for this test is pre-charged in the system.

Test steps:

- 1) Access CoolEmAll-WEB-GUI:
<http://sf2.coolmall.eu>
- 2) Login with credentials (username/password): test/test
- 3) First all we need to decide which room will be used in our new experiment:
Go to the DEBB configurator tab > ROOMS , and duplicate PSNC – MyTestRoom. If it is needed, you can edit the room parameters before

upload the model to the common repository or use the room already created in the previous experiment.

- 4) Once we have the room, we can setup the configuration for a new experiment. Go to the **Experiments** tab and click **New** from the up-down menu.
Fill the form with the setup parameters for your new experiment and save the experiment afterwards.

Table 6-4: Experiment parameters

Experiment ID	Description	Type
MyExperiments	(not mandatory)	DCworms
CO2 Emission Factor [g/kWh]	Start	End
(not mandatory)	(mm/dd/yy) (hh/mm/ss)	(mm/dd/yy) (hh/mm/ss)

- 5) When the experiment is created, we can add trials associated with the experiment definition. Edit the experiment and click **Add Trial** button in the experiment view.

Table 6-5: Room parameters

Trial Name	Timestamp Start (d/m/y h:m:s)	Timestamp Start (d/m/y h:m:s)
MyNewTrial		DCworms
Timestamp Start (d/m/y h:m:s)	DEBB Level	DEBB URL
	room	https://svn.coolmall.eu/svn/repository/common/debbs/trunk/MyTestRoom
Application Profile URL	Workload Profile URL	Path PLXML
(not mandatory)	(not mandatory)	https://svn.coolmall.eu/svn/repository/common/debbs/trunk/MyTestRoom/PLMXML_PSNCMMyTestRoom_12.xml
Path WRL	Path STL	Object Path
(not mandatory)	(not mandatory)	(not mandatory)
Testbed Instance	Alpha	Baseline Temperature
(not mandatory)	(not mandatory)	(not mandatory)
Ambient Temperature	DataCentre MassFlow	
(not mandatory)	(not mandatory)	

- 6) Once the experiment is saved and the associated trials created, next step is set the trial configuration as a context variable in order to be used by the simulation and visualization tools. Go to **Experiments** tab > **List** > Click Trials button in **MyExperiments** > Click **Set Context** button for the trial you want to simulate. “The Trial information has been saved in the session on the variable: context.” message should be displayed.
- 7) Go to **DCWorms** tab to create the **Simulation arrangement**. Select the **OpenSSL** workload and automatically the profile associate will be charged.
- 8) Select the policy used in the simulation (i.e.: Room_FCFS_LoadBalancing or Room_FCFS_ConsolidationLowPower_NodePowMan)
- 9) Go to **Run Simulation** within DCworms sub-menu and click **Start Simulation**. The simulation results will be displayed in the simulation window.
- 10) Next step is the visualization of results, it can be done in the **Reports** tab > **Metric Calculator** or using MOP-GUI, go to **MOP** tab > **Standard Mode**
- 11) In case of MOP visualization you have to add the properties of the data series that you want to visualize. Navigate through the DEBB hierarchy to select in example power measurements for each of the compute nodes and add the series to display the power plot associated to these measurements.

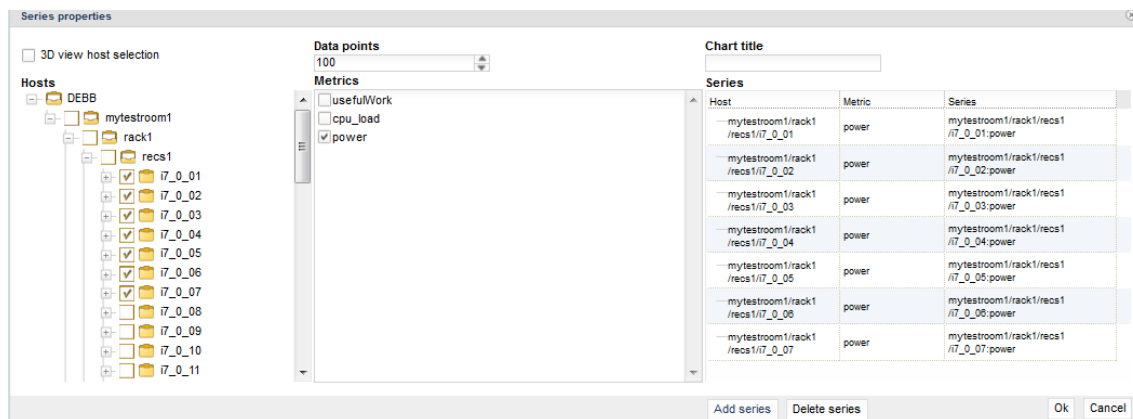


Figure 26: Adding data series to MOP

- 12) Analyze the results displayed as well as the measurements captured by the MetricCalculator and repeat the steps from 9 to 13 with a different scheduling policy.

Results:

Table 6-6: DCWorms result comparison

Room_FCFS_LoadBalancing	ROOM_FCFS_ConsolidationLowPower_NodePowMan
Simulation finished with status: 0 ===== Metrics ===== Total processors energy consumption: 727.36 [Wh] Total IT energy consumption: 727.36 [Wh] Total node group fans energy consumption: 452.826 [Wh] Total rack energy consumption: 1356.536 [Wh] Total data centre fans energy consumption: 240.135 [Wh] Total cooling device energy consumption: 6.57 [Wh] Total other devices energy consumption: 271.307 [Wh] Total energy consumption: 1874.548 [Wh] Mean rack power: 1067.671 [W] Mean power: 1475.377 [W] Max rack power: 1172.092 [W] Max power: 1601.245 [W] PUE: 1.382 [] PUE Level 4: 2.577 [] Energy waste rate: 59.436 [%] Useful Work: 512475783 [UW units] Productivity: 377782.79 [UW units/Wh] ===== Performance statistics ===== Makespan: 4574 [s] Task completion time - min: 518 [s]; max: 4574 [s] Mean task execution time: 513.108 [s] System occupancy: 37.393 [%] System load: 33.394 [%]	Simulation finished with status: 0 ===== Metrics ===== Total processors energy consumption: 206.175 [Wh] Total IT energy consumption: 206.175 [Wh] Total node group fans energy consumption: 211.118 [Wh] Total rack energy consumption: 479.647 [Wh] Total data centre fans energy consumption: 110.631 [Wh] Total cooling device energy consumption: 2.211 [Wh] Total other devices energy consumption: 95.929 [Wh] Total energy consumption: 688.418 [Wh] Mean rack power: 377.592 [W] Mean power: 541.943 [W] Max rack power: 1124.598 [W] Max power: 1543.995 [W] PUE: 1.435 [] PUE Level 4: 3.339 [] Energy waste rate: 3.699 [%] Useful Work: 146348530 [UW units] Productivity: 305117.244 [UW units/Wh] ===== Performance statistics ===== Makespan: 4573 [s] Task completion time - min: 517 [s]; max: 4573 [s] Mean task execution time: 512.558 [s] System occupancy: 37.361 [%] System load: 44.458 [%]

Simulation of Computational Fluid Dynamics (CFD) with re-arrangement of racks

Description:

In this test we describe server-room optimization using CFD simulations from the COVISE-GUI included within the CoolEmAll-WEB-GUI. The heat and air flow simulations will help data centre operators to detect and avoid hot-spots; afterwards they can try to resolve them re-arranging the racks within the room. The end-users will be able to identify problems and find a better placement of the racks in the rooms avoiding these problems.

Test steps:

- 1) Access CoolEmAll-WEB-GUI:
<http://sf2.coolmall.eu>
- 2) Login with credentials (username/password): test/test

- 3) First all we need to decide which room will be used in our CFD simulation. To set a trial on the session variable (context) you have to go to **Experiments > List > trials > Set Context**
- 4) After the selection of the trial to simulate there is another configuration step that need to be set before run the simulation. Go to **MOP > Standard Mode** and select a time point in the time-line displayed on top of the MOP window, afterwards you have to select **Set time for CFD** from the up-down menu.

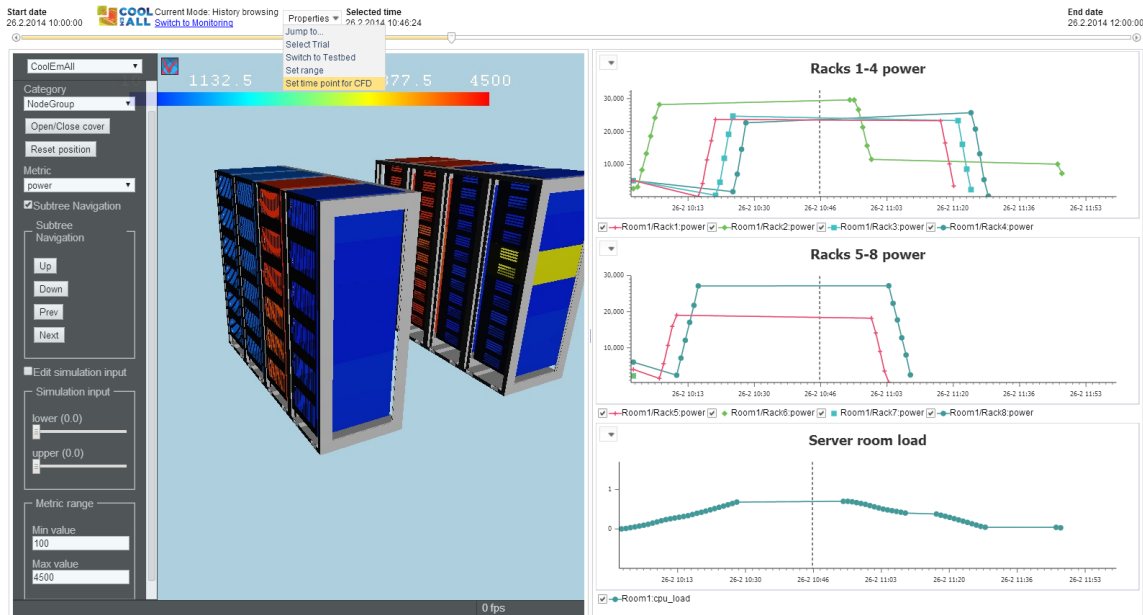


Figure 27: MOP - setting time for CFD simulation

- 5) Once the time point is selected you can go to **COVISE** tab to execute the air and heat flow simulations after the re-arrangement of racks within the room. The COVISE GUI has been displayed in a dual view mode, on one hand at the left you can select the parameters that configure the simulations, and on the other hand at the right window you can visualize the results of the simulation.
- 6) Next step is to re-arrange some of the servers displayed at bottom-left in the room view. After the re-arrangement you should click **Execute COVISE** and automatically you will see how the heat and airflow distribution change with the new position of the racks.
- 7) Before and after the simulation change the **Isosurface** parameter on top-left to visualize the heat distribution at different surface levels.

Results:

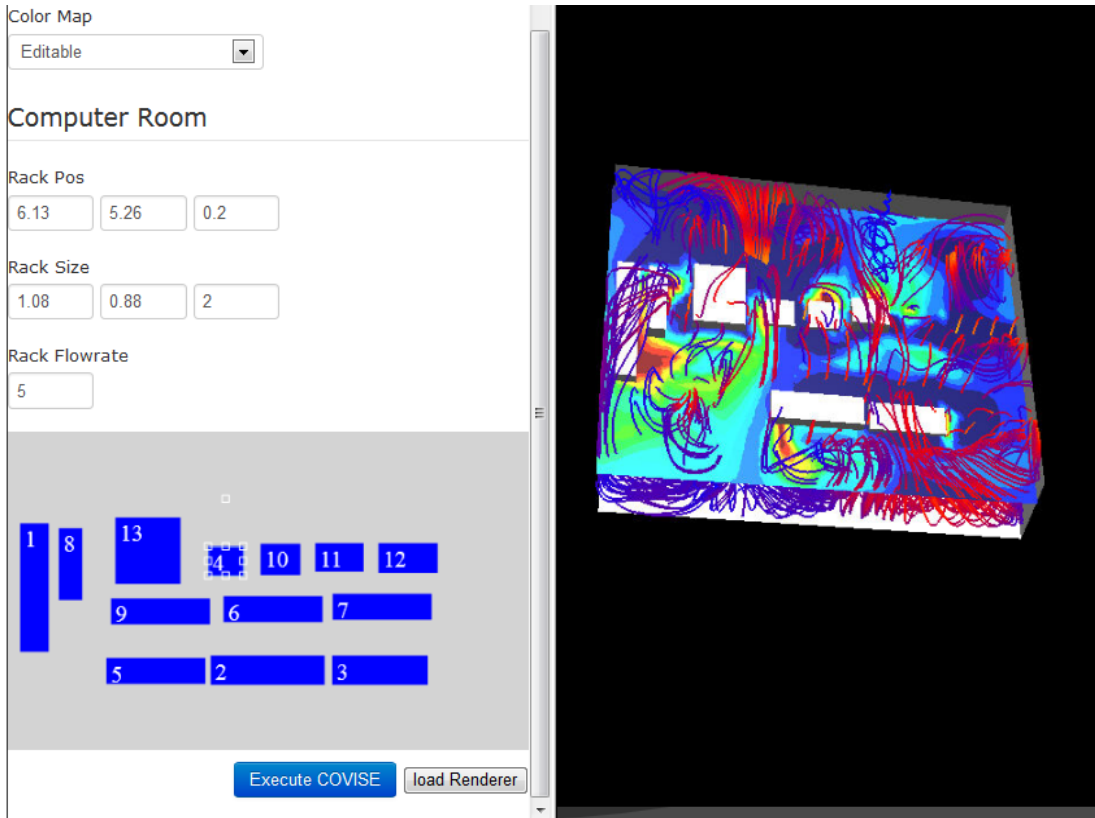


Figure 28: CFD heat and air flow simulations

7 References

- [SVD-Toolkit] download link for SVD Toolkit:
http://www.coolmall.eu/web/guest/download/-/document_library_display/g2Kj/view/57456
- [SF2] High Performance PHP Framework for Web Development – Symfony. <http://symfony.com/>
- [ApSu] Apache Subversion. <http://subversion.apache.org/>
- [PaVi] ParaView - Open Source Scientific Visualization.
<http://www.paraview.org/>
- [OpMPI] Open MPI: Open Source High Performance Computing.
<http://www.open-mpi.org/>
- [D2.2] CoolEmAll Deliverable D2.2 Design of the CoolEmAll simulation and visualisation environment, 2012
- [D2.3] CoolEmAll Deliverable D2.3 First definition of the hardware and software models, 2013
- [D2.3.1] CoolEmAll Deliverable D2.3.1 Update on definition of the hardware and software models, 2013
- [D2.4] First release of the simulation and visualisation toolkit, 2013
- [D2.5] Second release of the simulation and visualisation toolkit, 2014
- [D3.2] CoolEmAll Deliverable D3.2 First definition of the modular compute box with integrated cooling, 2013
- [D3.6] CoolEmAll Deliverable D3.6 *Final release of the rack-level and the modular compute boxes*, 2014
- [D4.3] CoolEmAll Deliverable D4.3 First set of resource management and scheduling policies, 2013
- [D4.7] CoolEmAll Deliverable D4.7 - Second prototype of Module Operation Platform GUI, 2014
- [D4.6] CoolEmAll Deliverable D4.6 Second set of resource management and scheduling policies, 2014

- [D6.1] CoolEmAll Deliverable D6.1 - Validation Scenarios, Methodology and Metrics, 2012
- [DCworms2012] Kurowski, K., Oleksiak, A., Piatek, W., Piontek, T., Przybyszewski, A., Weglarz, J. (2013) DCworms - a tool for simulation of energy efficiency in distributed computing infrastructures, Simulation Modelling Practice and Theory, in revision.
- [CoolEmAll_RECS] Da Costa G., Jarus, M., Oleksiak, A., Piatek, W., Volk, E., vor dem Berge, M., Modeling Data Centre Building Blocks for Energy-efficiency and Thermal Simulations, 2012
- [CoolEmAll-SVN] CoolEmAll project subversion repository, <https://svn.coolmall.eu/svn/repository>
- [SWF] Parallel Workload Archive, <http://www.cs.huji.ac.il/labs/parallel/workload/>
- [SLURM] <https://computing.llnl.gov/linux/slurm/>
- [SVN] Apache Subversion software versioning and revision control system, <http://subversion.apache.org>
- [TORQUE] TORQUE Resource Manager, <http://www.adaptivecomputing.com/products/open-source/torque>
- [OpenFOAM] OpenFOAM User Guide Version 2.1.1, <http://www.openfoam.org/docs/user/>
- [GPL] GNU General Public License <http://www.gnu.org>
- [LGPL] GNU Lesser General Public License <http://www.gnu.org>
- [MPL2] Mozilla Public License Version 2.0 <http://www.mozilla.org/MPL/2.0>
- [MIT] The MIT License <http://www.opensource.org/licenses/MIT>

[EurPar] *Directive 2009/125/EC*. Retrieved from
[http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?
uri=OJ:L:2009:285:0010:01:EN:HTML](http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:285:0010:01:EN:HTML). 2009