Context-Aware Cloud Topology
Optimisation and Simulation

Model Integration Method and Supporting Tooling

Project Deliverable D5.1

Henning Groenda, Christian Stier (FZI),

P-O Östberg, Jakub Krzywda (UmU),

James Byrne, Sergej Svorobej (DCU),

Zafeirios Papazachos (QUB),

Craig Sheridan, Darren Whigham (FLEXIANT)

Due date: 30/11/2014
Delivery date: 26/11/2014

This project is funded by the European Union under grant agreement no. 610711

Dissemination Level

X	PU	Public
	PP	Restricted to other programme participants (including the Commission Services)
	RE	Restricted to a group specified by the consortium (including the Commission Services)
	CO	Confidential, only for members of the consortium (including the Commission Services)

Version History

Version	Date	Change	Author
0.1	04/07/2014	Added Document Structure	Christian Stier (FZI)
0.2	24/07/2014	Initial Version of the Infrastructure Model Chapter	Christian Stier (FZI)
0.3	24/07/2014	Detailed Review	Henning Groenda (FZI)
0.4	30/07/2014	Incorporated Feedback from the Review, First Part of Section on Integration Tooling	Christian Stier (FZI)
0.5	08/08/2014	Integrated initial versions of sections on individual toolkits written by DCU and QUB	Christian Stier (FZI)
0.6	10/08/2014	Updated descriptions of the logical data centre models and optimisation triggers	P-O Östberg (UmU)
0.7	14/08/2014	Updated logical model figure and open points	Christian Stier (FZI)
0.8	14/08/2014	Updated CactoOpt optimisation deliverables descriptions	P-O Östberg (UmU)
0.9	14/08/2014	Added CactoOpt optimisation triggering description + partial API description	P-O Östberg (UmU)
0.91	14/08/2014	Updated descriptions of the Logical Data Centre Model	P-O Östberg (UmU)
0.92	18/08/2014	Further updates of the model descriptions and minor changes to the text. A few comments added.	P-O Östberg (UmU)
0.93	19/08/2014	Added section on interactions between CactoScale and CactoOpt	Christian Stier (FZI)
0.94	20/08/2014	Incorporated initial version of section on redeployment and reconfiguration	Craig Sheridan (FLEX)
0.95	21/08/2014	Revised and expanded CactoSim Release Planning	Sergej Svorobej (DCU)
0.96	21/08/2014	CactoScale-related parts	Zafeirios Papazachos (QUB)
0.97	22/08/2014	Review	Henning Groenda (FZI)
0.98	25/08/2014	Consistency checking, expanded upon a few sections	Christian Stier (FZI)
0.99	26/08/2014	Added management summary and introduction chapter	Christian Stier (FZI)
0.100.1	26/08/2014	Updated simulation section (Section 3: Release Planning) and Licensing for CactoSim	James Byrne (DCU) Sergej Svorobej (DCU)
0.100.2	01/08/2014	Reviewed, consolidated and expanded the sections	Christian Stier (FZI)
0.100.3	03/09/2014	Incorporated additional information on the realisation of optimisations in Flexiant Cloud Orchestrator	Darren Whigham (FLEX) Christian Stier (FZI)
0.100.4	05/09/2014	Fixed a few inconsistencies, updated figures and description of Infrastructure Models	Christian Stier (FZI)
0.100.5	15/09/2014	Updated release planning section for CactoSim	Sergej Svorobej (DCU)
0.100.6	19/09/2014	Updated release planning section for CactoScale	Zafeirios Papazachos (QUB)
0.100.7	14/10/2014	Added section for Component Deployment Middleware	Henning Groenda (FZI)
0.100.8	17/10/2014	Added description for updated Optimisation Plan Model	Henning Groenda (FZI)
0.100.9	28/10/2014	Preparation for internal Review	Henning Groenda (FZI)
0.100.10	11/11/2014	Addressed reviewer's comments	Henning Groenda (FZI)
0.100.11	14/11/2014	Addressed reviewer's comments	Jakub Krzywda (UmU)
0.100.12	21/11/2014	Addressed comments from the internal review	Christian Stier (FZI)
0.100.13	25/11/2014	Addressed comments from the internal review	Henning Groenda (FZI)

EXECUTIVE SUMMARY

The CACTOS project aims to improve the operational efficiency of cloud data centres by supporting data centre operators in the planning and operation of heterogeneous data centres. One major goal of CACTOS is to enable automated capacity and resource management for virtualised infrastructure environments built upon the Infrastructure as a Service (IaaS) paradigm. This document outlines the model-driven methodology developed for the integration of runtime monitoring of cloud-based data centres with runtime optimisation techniques.

The CACTOS project develops an integrated solution for runtime monitoring, optimisation and predictive analysis of data centres. The solution supports data centre providers in managing and planning data centres. CACTOS consists of two toolkits:

- The CACTOS *Runtime Toolkit* enables automated resource planning and optimisation for IaaS data centres.
- The *CACTOS Prediction Toolkit* supports what-if analyses for existing or planned data centre topologies that account for effects caused by automated resource optimisation.

While the focus of this document is to describe the integration methodology developed to couple runtime monitoring and optimisation for cloud data centres in the CACTOS Runtime Toolkit, the outlined methodology was developed to facilitate the integration across all toolkits developed in CACTOS. Hence, the integration of optimisation and monitoring with the simulative predictions in the CACTOS Prediction Toolkit is also discussed.

The main contributions of this deliverable are the *CACTOS Cloud Infrastructure Models* that define the common language through which the runtime analytics tool, CactoScale, and the optimisation tool, CactoOpt, exchange information on the data centre's structure and operational state. The models allow for the capturing of the deployment of Virtual Machines (VMs) on the middleware used in cloud data centres. Additionally, they track measurements and metrics that reflect the operational efficiency of the data centre. Instances of the CACTOS Cloud Infrastructure Models are constructed and maintained by CactoScale. CactoOpt uses the captured models as input for its optimisations.

This document gives an overview on the developed models and how they are utilised in the context of a holistic integration process. It relates to other deliverables by integrating the information on CactoScale's runtime monitoring (D4.2 Preliminary offline trace analysis), CactoOpt's topology optimisation algorithms (D3.1 Prototype Optimization Model) and the simulative what-if analyses of CactoSim (D6.1 CactoSim Simulation Framework Initial Prototype) for data centres. Furthermore, recent (D5.2.1 CACTOS Toolkit Version 1) and planned releases of the CACTOS toolkit (D5.2.2 CACTOS Toolkit Version 2) and the licensing models proposed for the individual CACTOS tools are outlined. The feature scope and integration of these features has served as the foundation for the requirements analysis of the developed integration methodology. The current iteration of the CACTOS Cloud Infrastructure Models capture all essential characteristics required to support an integration of current and planned features in all toolkits. Future iterations will improve the usability of the developed models and extend them to address newly identified requirements.

In addition, this document describes the development process of the toolkits and the infrastructure used throughout the CACTOS project. The document discusses the setup of CACTOS' development and build infrastructure and sketches the chosen architecture for the infrastructure. A holistic development process for both CACTOS Runtime Toolkit and the CACTOS Prediction Toolkit was chosen in order to facilitate early as well as Continuous Integration throughout and beyond the project's life cycle. The build infrastructure was set up

following the principle of Continuous Integration and allows for continued development and integration of all tools developed in the CACTOS projects, as well as the tools that they build upon.

Finally, the document discusses different licensing models for the release of both toolkits. In line with the effort to keep the results of the CACTOS project open for further development and use by the Open Source community, this document proposes to release all major project contributions under the Eclipse Public License Version 1.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Description
CACTOS	Context-Aware Cloud Topology Optimisation and Simulation
CDO	Connected Data Objects
CI	Continuous Integration
EMF	Eclipse Modeling Framework
FPGA	Field Programmable Gate Array
GPGPU	General Purpose Computation on Graphics Processing Unit
GPU	Graphics Processing Unit
HDFS	Hadoop File System
IaaS	Infrastructure as a Service
JEE	Java Enterprise Edition
OS	Operating System
OSGi	Open Service Gateway initiative
PDU	Power Distribution Unit
PSU	Power Supply Unit
PU	Processing Unit
QoS	Quality of Service
VM	Virtual Machine
VMI	Virtualisation Middleware Integration
XMI	XML Metadata Interchange
NAT	Network Adress Translation
VEPA	Virtual Ethernet Port Aggregator
VN-Link	Cisco Virtual Network Link

I. INTRODUCTION

Data centre operators are facing an increase of complexity involved in the efficient management of data centre resources. A surge in heterogeneity of computational resources and storage has made the manual management of applications and physical resources in data centres extremely effort-intensive. Alternative computational paradigms such as GPGPU and specialised hardware solutions such as ARM-based processing nodes have seen a broad adoption in data centres. Virtualisation technologies and cloud middleware frameworks have been introduced and extended to cope with the increasing heterogeneity. Virtualisation technologies facilitate the efficient use of hardware resources by co-locating multiple customer applications on the same physical machine. In the Infrastructure as a Service (IaaS) context, customer applications are encapsulated as Virtual Machines (VMs). These VMs each are automatically deployed by cloud middleware frameworks such as OpenStack without the involvement of the data centre operator. Currently the mapping of VMs to physical nodes is carried out using simple heuristics that abstract from key properties of VMs. Additionally, they do not automatically deconsolidate VMs that are running on the same physical node when the node is overutilised. Consequently, IaaS cloud data centres still heavily rely on the manual intervention of data centre operators to establish sufficient Quality of Service (QoS) for both data centre customers and operator.

In the CACTOS project, a holistic approach for the monitoring, analysis, optimisation and predictive analysis of IaaS cloud data centres is being developed. For this, tools for runtime monitoring and offline analysis (CactoScale), runtime optimisation (CactoOpt) and what-if-analyses using simulation (CactoSim) are brought together and integrated into two toolkits. These toolkits support the data centre operator in the automated operation of a data centre (CACTOS Runtime Toolkit) and resource planning and data centre design (CACTOS Prediction Toolkit). Both toolkits use the same optimisation algorithm framework (CactoOpt). Optimisations are used in the CACTOS Runtime Toolkit to identify adaptation actions that increase the QoS for data centre customers and operators. The CACTOS Runtime toolkit employs the optimisation to evaluate the QoS impact of changes to the infrastructure and optimisation policies using simulative analyses.

This document sketches the methodology by which the runtime measurement and monitoring of CactoScale is integrated with CactoOpt's optimisation algorithms. The methodology is applied in the implementation of the CACTOS Runtime Toolkit. A first iteration of this toolkit has been presented in (D5.2.1 CACTOS Toolkit Version 1). CactoOpt continuously evaluates the current deployment of Virtual Machines (VMs) on the infrastructure of a cloud data centre in order to identify optimisation actions. These actions aim to improve QoS for both the cloud customers as well as the data centre operators. The document also sketches how both the CACTOS Prediction Toolkit can be coupled with the CACTOS Runtime Toolkit to perform QoS predictions on the topology of a real data centre.

This document is structured as follows. Chapter II provides an overview on the tooling developed in the CACTOS project and the interactions between the individual tools. Chapter III outlines the release plans and feature scope for the tools. Chapter IV discusses the CACTOS Cloud Infrastructure Models that form the basis of CACTOS' model-driven integration methodology. Chapter V addresses the integration methodology in detail with a description of control flows and interactions between the tools. Chapter VI delineates the architecture of CACTOS' development and build infrastructure and provides the rationale for the chosen solution. Chapter VII describes how the storage and interpretation of models are managed by the individual tools. Chapter VIII shows licensing requirements and models of the tool releases.

II. CACTOS TOOLING OVERVIEW

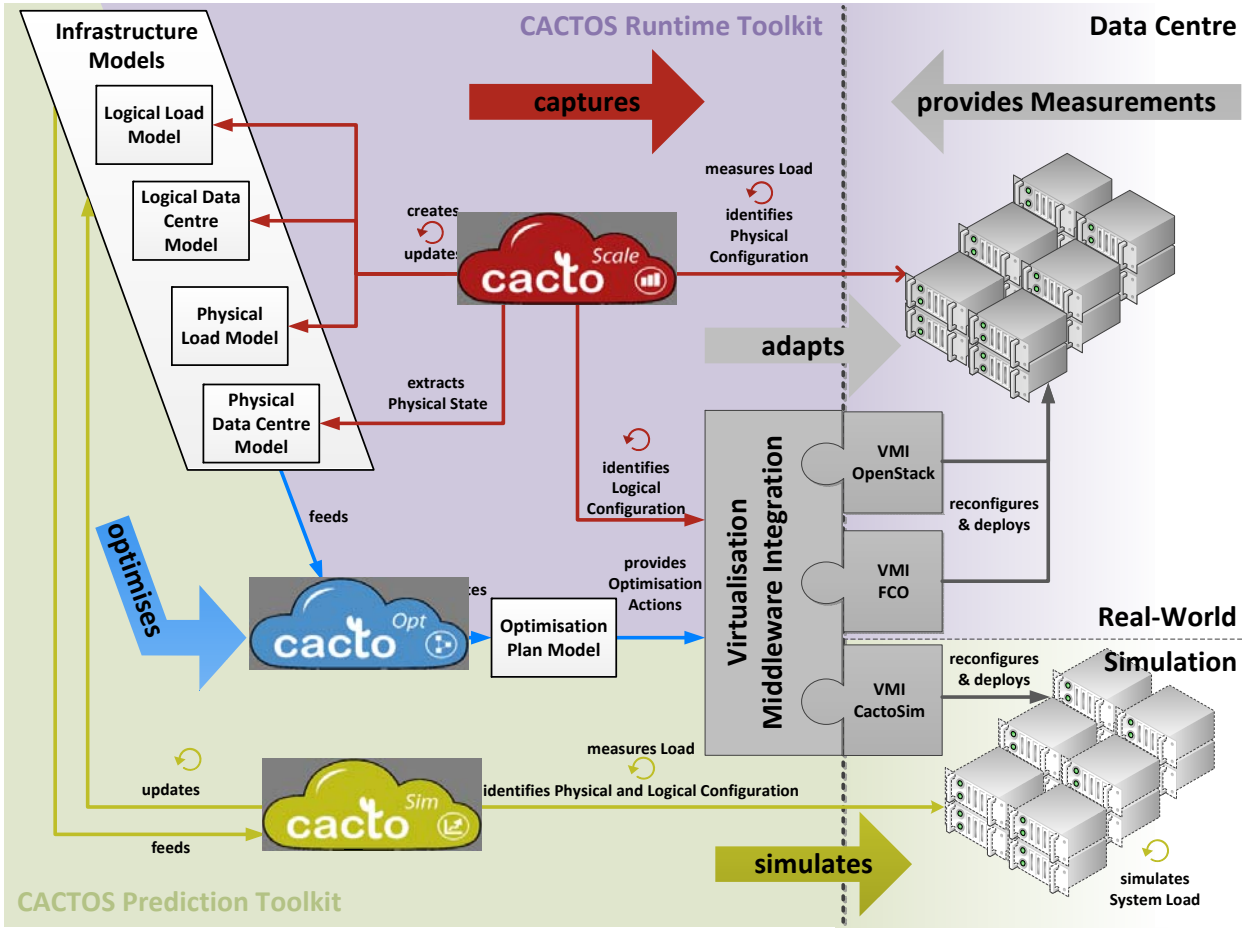

Figure 1 Overview on the CACTOS Tooling Landscape

This section provides an overview of the CACTOS tooling. CACTOS brings together analysis, optimisation and simulation of large-scale data centres, enabling data centre operators to utilise their infrastructure in a more efficient manner. Figure 1 illustrates the architecture and relation of the tools in the CACTOS toolkits. CACTOS consists of the CACTOS Runtime Toolkit for managing real-world data centres (purple area) and the CACTOS Prediction Toolkit for the simulation and analysis of data centres (green area). The toolkits combine the functionality of CactoScale for capturing information in real-world data centres, CactoOpt for optimising the topology of data centres, and CactoSim for simulating data centres. A Virtualisation Middleware Integration (VMI) layer is also included which decouples technology-specific data centre cloud computing platforms, e.g. OpenStack[1] or FCO[2] from the generic optimisation and simulation core of the toolkits. The CACTOS Runtime Toolkit brings together CactoScale and CactoOpt to enable data centre operators to automate deployment and configuration management of the data centre. The CACTOS Prediction Toolkit integrates the predictive simulations of CactoSim with the optimisation algorithms of CactoOpt. Using the CACTOS Prediction Toolkit, data centre operators can evaluate the QoS of different data centre topologies optimised through different policies under different user loads. The three main tools and their respective functionality are described in more detail as follows.

[1] http://www.openstack.org/
[2] http://www.flexiant.com/flexiant-cloud-orchestrator/

CactoOpt uses the Infrastructure Model instances for creating optimisation plans as part of the CACTOS Runtime Toolkit. Examples for optimisations on the infrastructure are VM migrations or an increase of resources assigned to a Virtual Machine. The optimisations are stated in the Optimisation Plan Model. It describes the order of a set of atomic optimisation actions proposed by CactoOpt. The Virtualisation Middleware Integration carries out the individual optimisation actions on the specific middleware used in the cloud data centre, e.g. OpenStack or FCO. The Virtualisation Middleware Integration follows the order of actions as suggested by the Optimisation Plan Model.

CactoSim allows for performing simulations based on Infrastructure Models. In future iterations the effect of the employed infrastructure optimisation algorithms will be accounted for by CactoSim's QoS predictions. CACTOS Cloud Infrastructure Models extracted from real-world data centres by CactoScale can be used directly as input for simulation. Alternatively, the data centre operator can modify these models according to potential change scenarios. She can also create purely synthetic models to estimate the QoS benefit of data centre expansions or reorganisations. CactoSim simulates the data centre described by Infrastructure Models and is planned to provide load measurements on the simulated infrastructure in the same way as CactoScale. It is further planned that CactoSim allows for predicting the impact real world data centre optimisation mechanisms have on the QoS of data centres. Specifically, CactoSim will be coupled with the optimisation algorithms of CactoOpt. The integration of CactoOpt's optimisation algorithms is achieved through adaptation rule templates that realise the actions in an optimisation plan proposed by CactoOpt.

The following sections provide a more detailed overview of the individual tools CactoScale, CactoOpt and CactoSim. The description is an excerpt from the tooling overview in (D5.2.1 CACTOS Toolkit Version 1).

1. CACTOSCALE

CactoScale serves multiple data capturing roles in the CACTOS project. It provides a scalable data collection framework for monitoring cloud computing data centres. Furthermore, it is designed to perform online data analytics of events and conditions within a data centre. These events will be utilised as triggers for data centre optimisations in future iterations of the CACTOS Runtime Toolkit.

CactoScale also provide offline analysis functions. In future iterations, behaviour models for Virtual Machines or running systems based on measurements and log data will be generated as part of the offline analysis. This section provides a quick overview on CactoScale, however, more detailed information is available in (D4.1 Data Collection Framework) and (D4.2 Preliminary offline trace analysis).

a) PURPOSE AND FEATURES

CactoScale features a data collection framework capable of monitoring large scale distributed systems and cloud computing data centres. The data collection framework is coupled with data analytics tools, which allow for the parallel processing of the data using the MapReduce framework. Fundamental capabilities of the toolkit include:

- Scalable data collection – The data collection framework relies on an agent-based monitoring facility and historic data is stored on a Hadoop Distributed File System which provides scalability and robustness.
- Data analytics for "Big Data" – CactoScale utilises Hadoop's MapReduce framework and therefore enables parallel processing of the collected data.
- Monitoring of cloud systems and alerting – Different metrics, log files and error logs are monitored and collected. Performance analysis and anomaly detection is feasible based on historical analysis of these metrics and logs. An alerting mechanism will allow for efficient operation and quick reaction on the data

centre operator side and can serve as a trigger for optimisations in future versions of the CACTOS Runtime Toolkit.

- Filtering, clustering and correlation algorithms - Focusing on effective algorithms to achieve correlation and co-analysis of traces from different sources, e.g. to create behaviour models in the offline case.

The release provided as part of (D5.2.1 CACTOS Toolkit Version 1) includes:

- Monitoring of metrics including network, CPU, memory, storage and virtualised metrics
- Customised import modules for real-world traces of the MolPro scientific application provided by UULM and the logging on the black box level experienced at Flexiant with no insight into VM internals.
- Offline statistical analysis tool for imported traces

b) UTILISED BASE TECHNOLOGIES

CactoScale is utilising a range of existing Apache software tools - Chukwa, Hadoop, HBase and Pig. Chukwa is a data collection tool for monitoring of distributed systems, which is built on top of Hadoop Distributed File System and MapReduce framework. Chukwa is extended in order to collect data for the VMs. The design of Chukwa allows us to use the agent modules to invoke a variety of low-level monitoring tools such as sar, iostat, df and top. The virt-Top tool is utilised in order to retrieve information regarding the VMs. The virt-Top tool is part of the popular libvirt toolkit. libvirt supports a great number of hypervisors including the

- KVM/QEMU Linux hypervisor
- Xen hypervisor on Linux and Solaris hosts
- LXC Linux container system
- OpenVZ Linux container system
- User Mode Linux paravirtualised kernels
- VirtualBox hypervisor
- VMware ESX and GSX, Workstation and Player hypervisors
- Microsoft Hyper-V hypervisor
- IBM PowerVM hypervisor
- Parallels hypervisor
- Bhyve hypervisor

libvirt is also able to identify and monitor virtual networks built upon bridging, NAT, VEPA and VN-LINK techniques. In addition, it can be used to extract storage information regarding the utilised IDE/SCSI/USB disks, FibreChannel-, LVM-, iSCSI-, NFS-based storage networks and file systems. Libvirt allows covering the existing variety of Cloud middleware solutions.

HBase is a NoSQL database modelled after Google's Bigtable, which provides random, real-time read/write access to Big Data. HBase comprises a set of tables. Pig, which is the last of the aforementioned tools, is a high-level platform for creating MapReduce programs used with Hadoop. CactoScale interfaces with the other CACTOS tools through the use of a Java-based infrastructure and the Infrastructure Models.

c) INTERFACES FOR INTEGRATION

The Infrastructure Models created or updated by CactoScale are stored in a Runtime Model Storage using EMF CDO technology.

The logical configuration of data centres, e.g. the deployment of VMs, is elicited via the Virtualisation Middleware Integration and via additional management tools, e.g. libvirt.

2. CACTOOPT

CactoOpt is composed of a set of optimisation algorithms and tools designed to allow for data centre operator control, optimisation, and – together with CactoSim – evaluation of data centre infrastructure optimisation plans. Optimisation mechanisms evaluate data centre properties, e.g., Virtual Machine deployments and configurations, and attempt to (by ways of evaluation heuristics and optimisation computations) find ways to more efficiently (according to a given objective function) configure and execute workloads in the data centre.

This section provides a quick overview of CactoOpt. More detailed information is available in (D3.1 Prototype Optimization Model).

a) PURPOSE AND FEATURES

The purpose of CactoOpt is to facilitate the optimisation of data centre infrastructures with respect to the efficient provisioning of computational and storage resources to Virtual Machines in the context of IaaS-based cloud data centres.

CactoOpt follows a sensor-actuator model where infrastructure topology and load models capture a sensor view of the surrounding world (i.e. the data centre state) and CactoOpt optimisation recommendations are viewed as actuators for the optimisation core. Heuristic functions and optimisation algorithms are developed based on this model and Optimisation Plan Models are created accordingly.

Planned and supported features of CactoOpt include the ability to

- Plan for optimised deployment of Virtual Machines
- Perform heuristics-based evaluation of current deployments of Virtual Machines
- Continuously identify opportunities for improvements in Virtual Machine deployment layouts
- Plan for migrations of Virtual Machine within and between clusters in data centres
- Plan for optimised dynamic reconfiguration for both virtual and physical resources within the data centre, e.g. by scaling or dynamic frequency regulation
- Plan for optimised admission control and horizontal scaling of Virtual Machines for cloud applications.

In the CACTOS Toolkit Version 1 release, CactoOpt supports algorithms for initial placement and scaling of Virtual Machines. There are built-in heuristic functions for evaluating the placement efficiency and comparison of optimisation actions. This version focuses on Virtual Machine state and does not consider holistic effects of individual optimisation actions on the entire data centre. Optimisation algorithms are developed in prototypical (simulated) environments and (in this version) implement simple, greedy functions (e.g., finding the node that best matches a certain resource capacity description) and will be further refined in later versions. Further optimisation and heuristics functions will support more advanced modelling of application behaviour, virtual and physical machine load, and the prediction of application behaviour.

The following subsections outline the aforementioned capabilities and features of CactoOpt in greater detail.

PLANNING THE OPTIMISED DEPLOYMENT OF VIRTUAL MACHINES

CactoOpt inspects the infrastructure information in order to determine the optimal placement of arriving Virtual Machines. The infrastructure information includes the current state of the physical and virtualised infrastructure

and load measurements on both virtual and physical resources. Optimisations are performed on the basis of a high-level objective function such as the maximisation of efficiency with which a resource (e.g., memory) is used, the minimisation of overall power consumption, or the maximisation of server consolidation (e.g., the number of Virtual Machines per physical machine).

PERFORMING HEURISTICS-BASED EVALUATIONS OF CURRENT VM DEPLOYMENTS

CactoOpt defines heuristic functions that express the utility of arbitrary Virtual Machine deployments. One potential instantiation of the function is a summarised cost function for the load Virtual Machines place on local network segments or a predicted energy consumption cost on Virtual Machine, node or rack level. Deployment evaluation cost functions are then used to evaluate and compare current deployment plans against alternative plans from simulation or optimisation plans.

OPTIMISING VM DEPLOYMENT AND MIGRATION PLANNING

CactoOpt identifies opportunities for improvements in Virtual Machine deployment layouts. Based on the identified optimisation operations, a plan for migrations of Virtual Machine within and between clusters in data centres is set up. Key to this ability is the definition of cost functions that accurately model the cost of migration of Virtual Machines (not only in direct transfer costs or performance delays, but also in resulting network load and potential impact on co-located / neighbouring machines and workloads). Important use cases for this ability include not only the use of Virtual Machine migration as an actuator in optimisation, but also in interactive use cases where data centre operators may want to weigh the potential gains of migrating a workload against the risks and costs associated with the action.

ENABLING THE DYNAMIC RECONFIGURATION OF LOGICAL AND PHYSICAL RESOURCES WITHIN THE DATA CENTRE

In future releases of CactoOpt, the virtual resources of VMs as well as the state of physical resources may be changed to dynamically adapt to changes in incoming request rates to servers. CactoOpt will use information on planned or predicted peaks in workload to determine when such a reassignment makes sense. An example for such a logical reconfiguration is the increase of RAM and CPU cores assigned to VMs in anticipation of a load peak. A reconfiguration of physical hardware may consist of an adjustment of CPU frequencies or power state in a similar manner.

OPTIMISED ADMISSION CONTROL AND SCALING OF VIRTUAL MACHINES FOR CLOUD APPLICATIONS

This ability can be used differently in the usage context of the CACTOS Runtime Toolkit as well as the CACTOS Prediction Toolkit. For the runtime data centre environment it supports automatic adjustment of the number of Virtual Machines spawned for a particular cloud application. In predictions, it additionally serves to evaluate alternative deployment plans for Virtual Machines and to predict the impact changes in the infrastructure and usage profile have on the QoS of applications deployed in the data centre. The key to this ability is the formulation of application workload models and load prediction techniques that can capture workload fluctuations with a sufficiently high accuracy. Thereby, the costs of Virtual Machine instantiations can be weighed against the costs of potential service-level agreement violations that would result from delays.

b) UTILISED BASE TECHNOLOGIES

CactoOpt is realised in Java and stores application and workload data in a SQL database for analysis and prediction purposes. The database is decoupled from CactoScale as the stored information is specific to the optimisation algorithms. The database performance is sufficient for the current algorithms and testbed. It will be analysed for the next releases if the performance is still appropriate. Future optimisation algorithms can be developed in

proprietary environments, e.g., Matlab and customised C-based solver environments. In such a case, they are wrapped in Java using Java-native bridging technologies such as the Java Native Interface (JNI). The optimisation functions will be provided as OSGI / Equinox bundles in order to facilitate the use within both CACTOS toolkits.

c) Interfaces for Integration

CactoOpt uses the Infrastructure Models in the Runtime Model Storage (if used within the CACTOS Runtime Toolkit) or the ones provided by CactoSim (if used within the CACTOS Prediction Toolkit) in order to create Optimisation Plan Models. The recommendations contained in such a plan can then be executed using the Virtualisation Middleware Integration. The Optimisation Plan Model and Infrastructure Models are described in detail in (D3.1 Prototype Optimization Model) and presented in context with the integration methodology in section V.3.

3. CactoSim

The aim of the model-driven discrete event simulation tool CactoSim is to produce accurate system behaviour forecasts, which can aid in the planning and management of a data centre. CactoSim does exactly that in context-aware fashion for cloud computing data centres. By using CactoSim, experimenters are able to validate and evaluate system configuration scenarios and obtain valuable decision support information.

This section provides a quick overview on CactoSim; more detailed information is available in (D6.1 CactoSim Simulation Framework Initial Prototype).

a) Purpose and Features

CactoSim is a context-aware cloud topology simulation framework. It is implemented as an Eclipse feature, which is a set of plugins. In an IaaS scenario from a data centre operator point of view, fine-grained knowledge of application internals is usually not available. The data centre operators do not have access to detailed information of applications that are deployed in VMs. Thus, CactoSim abstracts from these internals and utilises a higher-level black box behaviour model. CactoSim addresses both physical and virtual resources: the mapping of VMs to physical resources and the hierarchy of racks and nodes are essential characteristics of IaaS systems that are captured by the CACTOS Cloud Infrastructure models.

The version of CactoSim released as part of (D5.2.1 CACTOS Toolkit Version 1) provides prediction capabilities in the cloud context including processor, storage and memory to the simulated resources. In order to represent system workload based on variety of applications running in the virtual landscape we also introduce black box and grey box behaviour modelling practices allowing resource utilisation prediction without detailed software component model knowledge.

CactoSim plays a major role in the evaluation and validation for the optimisation algorithms used in CactoOpt. CactoSim uses CactoOpt and takes the recommended Optimisation Plans into account, modifying the topology.

Features available in this release are:

- **Grey box and black box behaviour models** – alternative high-level behaviour and performance prediction models for VMs where little or no knowledge on the internal behaviour of hosted applications is available.
- **Memory Model** – representation of physical node memory and its throughput.

The additional features of CactoSim in its final version will be:

- **Deployment and behaviour modelling** – allows modelling the structure of IaaS cloud environments, including the hardware bound deployment and behaviour models from which predictions on the resource demands of VMs can be derived.
- **Support for self-adapting systems** – extends considering optimisation and even allows to take into account modifications on individually deployed systems within the data centre to represent reactions to system conditions i.e. QoS violations or workload change.
- **Resource utilisation reports** – the log of the simulated system resource utilisation can be accessed and analysed right after the simulation.

b) UTILISED BASE TECHNOLOGIES

CactoSim is built on the two existing tools Palladio (The Palladio component model for model-driven performance prediction) and SimuLizar (Performance analysis of self-adaptive systems for requirements validation at design-time) .

Palladio is a component-based architecture simulator and allows performing model-driven QoS predictions for component-based software systems defined in the Palladio Component Model (PCM). The focus of the PCM lies on reasoning on non-functional properties of systems in the design stage and selecting the best architecture alternative. Consequently, software architects and component developers designing the architecture of the system are assumed to have insight into the assembly of and dependencies between individual software components. They model individual services and their behaviour at a white box level. Palladio models physical resources and the virtualisation layer in a very abstract manner

SimuLizar is based on PCM but extends it for self-adaptive systems. Its ability to create self-adaptation rules will be utilised and extended by CactoSim. This enables the realisation of the Virtualisation Middleware Integration with CactoSim and therefor the execution of optimisation plans.

c) INTERFACES FOR INTEGRATION

The significant strength of CactoSim in comparison to other cloud simulation tools comes from the integration with the CACTOS Runtime toolkit. The CACTOS project enforces a common data exchange format and API between CactoScale, CactoOpt and CactoSim components by utilising meta-models specifically tailored for the operation of IaaS cloud data centres. Having these common standards in place across toolkits it is ensured that all toolkits can share information using a common abstraction.

In this release CactoSim is able to fetch the Infrastructure Models from real-word data centres from the Runtime Model Storage. These models are then stored in the Prediction Model Storage for further modification or analyses.

The Prediction Model Storage enables storage and access to different versions of individual Infrastructure Model instances. This allows storing multiple versions based on the same Infrastructure Model, e.g. one that is initially populated from a real data centre and then extended by additional hardware nodes. This supports the lightweight comparison of different data centre or optimisation algorithm alternatives.

The results of a simulation are stored using the persistency framework EDP2 that is developed as part of the Palladio tooling. The files containing the results are persisted on the machine executing the simulation. Recorded results can be analysed using distribution and time-series plots as part of the Eclipse-based Palladio tools. Furthermore, the results can be extracted as comma-separated tables for an analysis in statistical tools, storage in databases or an analysis in other graphical user interfaces.

III. RELEASE PLANNING

Table 1 Project-Wide Tool Releases

Tool	Date	Content	Deliverables
CACTOS Toolkits	30/09/2014	First Version	D5.2.1, MS5
	31/12/2014	Integration Tooling Operational for Small-Scale Testbed	D5.3, MS7
	31/03/2016	Second Version Suited for the Operation of a Small Cloud Testbed	D5.2.2, MS11, MS12
CactoScale	31/03/2014	Conceptual Design of the Data Collection framework	D4.1
	30/09/2014	Data Collection Framework supporting the Extraction of the System State as Infrastructure Models Including Preliminary Parallel Trace Analysis	D5.2.1, D4.2, MS5
	31/12/2014	Operational Prototype Ready for Use in Small Testbeds	D5.3, MS7
	30/06/2015	Parallel Trace Analysis	D4.3
	31/03/2016	Integrated Data Collection and Analysis Frameworks	D5.5, D4.4, MS11, MS12
CactoOpt	30/09/2014	Prototype Optimisation Model	D5.2.1, D3.1, MS5
	31/12/2014	Predictive Cloud Application Model	D5.3, D3.2, MS7
	30/09/2015	Extended Optimisation Model	D3.3, MS9
	30/09/2016	Final Optimisation Model	D3.4
CactoSim	30/09/2014	Simulation Framework Initial Prototype	D5.2.1, D6.1, MS5
	30/09/2015	Simulation Framework Intermediate Prototype	D6.3
	31/05/2016	Simulation Framework Final Prototype	D6.4, MS12

The CACTOS toolkits and the individual tools are released in multiple iterations. Table 1 shows the individual and overarching CACTOS release plans and main target deliverables. Only this first dependency to tool deliverables and milestones are listed for brevity. The following sections describe the specific features of each release and the refinement of the overall plan.

The CACTOS toolkits will be delivered following the Continuous Integration (CI) methodology, meaning that all changes in the CACTOS tooling will be integrated via a common build server infrastructure.

1. CACTOSCALE

CactoScale will be released in 4 separate versions. The release plan in Table 1 is designed to provide prototypes, which include unique features in each version and also enhance features existing in previous versions. The Tables 2-5 describe the features of the released version of CactoScale.

Table 2 Features for the CactoScale Release Version 1

	Date: March 31th 2014
Feature Name	**Scope and Description**
Conceptual Design of the Data Collection Framework	This release defines the requirements of the data collection framework to be deployed for the project. The requirements include the performance and energy indicators that the data collection framework is expected to log and analyse, the agreed data formats and the architecture of the data collection tool. This release also describes the existing datasets from Flexiant and the University of ULM.

Table 3 Features for the CactoScale Release Version 2

	Date: September 30th 2014
Feature Name	**Scope and Description**
Data Collection Framework supporting the Extraction of the System State as Infrastructure Models Including Preliminary Parallel Trace Analysis	A data collection framework supporting the extraction of the system state as Infrastructure Models. The framework provides measurements of the cloud platform on CPU, memory, network, storage and VM resource consumption. This initial version provides preliminary offline trace analysis for existing traces.

Table 4 Features for the CactoScale Release Version 3

	Date: December 31st 2014
Feature Name	**Scope and Description**
Operational Prototype Ready for Use in Small Testbeds	This release supports the automatic generation of physical and logical model instances based on the extraction of the system state as infrastructure models. This prototype version will be validated for use in small testbeds.

Table 5 Features for the CactoScale Release Version 4

	Date: June 30th 2015
Feature Name	**Scope and Description**
Parallel Trace Analysis	A data collection framework which processes data online and in-situ to enable real-time decision-making, such as proactive VM migration to cope with anticipated failures, or proactive server activation to cope with anticipated load spikes. We will develop a framework for parallel analysis of workload traces and system logs, coupled with HDFS for in-memory processing of the data. To this end, this task will implement pre-processing of raw data logs for storage in HDFS. To avoid interference of the trace data processing infrastructure, scheduling and buffer allocation methods will be developed that isolate the resources (cores, memory, interconnect) used by the logging infrastructure from the resources used by actual workloads. Furthermore, the outcome from the validation in M18 will be considered in order to refine any existing methods to improve performance.

Table 6 Features for the CactoScale Release Version 5

	Date: March 31st 2016
Feature Name	**Scope and Description**
Integrated Data Collection and Analysis Frameworks	This version presents the integration of the algorithmic data analysis framework and the data collection tool prototypes in production-mode setups.

2. CACTOOPT

CactoOpt will be released in 4 separate versions. The release plan in Table 1 is designed to provide prototypes, which include unique features in each version and also enhance features existing in previous versions. The Tables Table 8-Table 10 describe the features of each version of CactoOpt.

Table 7 Features for the CactoOpt Release Version 1

	Date: September 30th 2014
Feature Name	**Scope and Description**
Prototype Optimisation model	An initial optimisation model to demonstrate the interfacing of the model with preliminary characterisation templates describing workloads and infrastructures. All performance predictions are made on Virtual Machine level and are based on historical data from Virtual Machine monitoring and logs. The initial version of the optimisation model is intended to lay the foundation work on infrastructure and workload models, as well as preliminary versions of the optimisation actuators (i.e. what the optimisation engine is able to affect in the target infrastructure), and will not contain any advanced optimisation algorithms for infrastructure optimisation.

Table 8 Features for the CactoOpt Release Version 2

	Date: December 31st 2014
Feature Name	**Scope and Description**
Predictive Cloud Application Model	A prediction-capable model of cloud services, execution environment, user behaviour, and quality characteristics like service times, workloads, scalability, and burstiness. Application behaviour quantified in terms of hardware resource capacity requirements (e.g., CPU, RAM, etc.) as expressed in encapsulating Virtual Machines. Preliminary prediction capabilities based on application resource utilisation pattern models. An initial workload approximation and prediction model that describes the resource capacity requirements of cloud applications and outlines foundational work in workload characterisation and application classification. From the initial models defined in this release, more advanced resource capacity predictions will be developed and deterministic models that characterise and predict application behaviour will be developed.

Table 9 Features of the CactoOpt Release Version 3

	Date: September 30th 2015
Feature Name	**Scope and Description**
Extended Optimisation Model	An enhanced version of the optimisation model that will feature predictive capabilities based on further developed application resource utilisation models. Building and extending on the prototype optimisation and predictive cloud application models, the extended optimisation model incorporates more advanced optimisation algorithms, e.g., integer and linear programming optimisation as well as heuristics-based optimisation models that can be applied to resource level optimisation scenarios such as initial Virtual Machine placement, Virtual Machine migration, vertical scaling of Virtual Machines (and their resource assignments). The purpose of the extended optimisation model is to finalise the initial versions of optimisation interfaces and models defined in the prototype models, and demonstrate the potential of optimisation tools based on these models. Furthermore, the outcome from the validation in M18 will be considered in order to refine any existing methods to improve performance of optimisation.

Table 10 Features of the CactoOpt Release Version 4

	Date: September 30th 2016
Feature Name	**Scope and Description**
Final Optimisation Model	The final optimisation model to demonstrate joint optimisation of network usage and system resources. The model should also be capable of considering consolidation effects together with workload scheduling and migration. The final optimisation model will naturally build (and extend) on the extended optimisation model and will feature more advanced and fine-tuned optimisation mechanisms that operate on both resource and (holistic) data centre level. Higher-level optimisations are intended to both interact with and use resource-level optimisations to achieve holistic data-centre level infrastructure optimisation objectives.

3. CACTOSIM

This section is condensed from the supporting documentation (D6.1 CactoSim Simulation Framework Initial Prototype). Further information on the release planning of CactoSim can be found in the aforementioned document.

Simulation is an essential tool for data centre planning and optimisation. Depending on the design and purpose of an experiment, simulations can be carried out with models that capture a certain hardware configuration, network topology and workload. The models allow changing each aspect independently from the others to evaluate their effect on the predicted QoS of the whole composition. Besides structural information, these models can be populated with behaviour descriptions based on actual events occurring within a data centre, e.g. HDD failure, VM admission or CPU utilisation changes. Using the simulation information on the resource and system utilisation under controlled and reproducible conditions can be obtained. This eliminates the complexity, hardware usage,

and costs associated with running a benchmark on actual hardware. The use of simulation gives the ability to predict system behaviour and produce information for large-scale data centres without even starting a single VM.

Predicting the effect of changes in the data centre, e.g. other storage systems with higher I/O performance might appear to be trivial. However, assuming linear dependencies between properties such as I/O performance and application performance are over-simplifications. Applications from different domains such as scientific computing and enterprise services require a more in-depth simulation that differentiates computation-bound and I/O-bound periods as well as access times and contention during resource access. Application behaviour models allow hiding some of this complexity from the users. Instead of reasoning on system traces, users specify a set of application characteristics, e.g. different storage access types. The applications are then simulated for a user-defined data centre environment specification. Besides a specification of the physical infrastructure topology, this also includes the selection of data centre topology optimisation algorithms whose effect on QoS is to be simulated.

CactoSim aims to deliver a simulation framework that relies on the data traces and their analysis provided by CactoScale and serve as means of validation for CactoOpt at a large scale. CactoSim plays the role of a context-aware advanced decision-support tool for data centre management. Operators are able to model heterogeneous landscape components in order to validate and evaluate optimisation algorithms via what-if-analyses basing results on such metrics as cost and risk. Produced results, in form of simulated forecasts, are calculated based on collected historical system data such as resource utilisation, usage patterns, failure rates and existing optimisation strategies.

CactoSim is planned to be delivered in three prototype releases: an initial prototype release (CactoSim 1.0) in Month 12, an intermediate release (CactoSim 2.0) in Month 24, and a final release (CactoSim 3.0) in Month 32. The following tables describe proposed features, scope and description of each of these releases. Note that the features included in these tables are planned features and are subject to change over the duration of the project as research is carried out into the requirements and design of CactoSim taking related component/tool development into account.

Table 11 Features for the CactoSim Release Version 1

		Date: September 30th 2014
CactoSim Release 1.0 (Initial Prototype)	**Feature Name**	**Scope and Description**
	Base Discrete Event Simulation Engine	Base engine providing capability to model the operation of a system as a discrete sequence of events in time using SimuCom within the Palladio environment.
	Initial Component Model	This release utilises the Palladio component model (PCM) which captures the software architecture with respect to static structure, behaviour, deployment/allocation and resource environment. Using this as a base, this will be extended in future releases.
	Initial Self-Adaptive System Analyser	This release utilises the SimuLizar plug-in for Palladio in order to provide support for modelling self-adaptive cloud computing systems at design time.
	Initial Black and Grey Box Behaviour Model	Ability to represent the application behaviour without full knowledge of its inner workings, performance prediction being based on collected historical data.

	Initial Memory Model	Extension to the utilisation of the PCM model providing an addition to the existing CPU and HDD models representing hardware memory.
	Initial Graphical User Interface	For this version, a modified version of the Eclipse-based graphical user interface (GUI) of Palladio is used. The GUI supports graphical drag-and-drop based modelling of data centre resources and workloads.

Referring to Table 11, the following is a description of the planned features for the CactoSim initial release (CactoSim 1.0):

- *Base Discrete Event Simulation Engine.* This is required in order to provide the user with the capability to model the operation of the system as a discrete sequence of events. For this release, SimuCom is used to provide this capability within the Palladio environment.
- *Initial Component Model.* This release utilises the Palladio component model (PCM) as the basis for simulations. Instances of the CACTOS Cloud Infrastructure Models are translated into PCM model instances using model transformations. Future releases will expand the PCM towards a native simulation of the CACTOS Cloud Infrastructure Models.
- *Initial Self-Adaptive System Analyser.* This release utilises the SimuLizar plug-in for Palladio in order to provide support for modelling self-adaptive cloud computing systems at design time later on. Future releases will expand SimuLizar as described in the respective overview on the releases.
- *Initial Black and Grey Box Behaviour Model.* In order to forecast data centre resource utilisation we need to model the behaviour of each VM. However, cloud data centre providers usually only have access to the VM attributes and coarse workload history information. These few parameters tell very little about the internal software design and are insufficient for creating detailed application behaviour models. Therefore, the first release of CactoSim utilises black box behaviour models for IaaS applications. As the name implies, these models provide the ability to simulate IaaS cloud applications on a coarse-grained level. In these models, users are described solely in terms of their resource consumption levels. Temporal changes in the load are extracted through manual analysis of historical workload traces of the node.
- *Initial Memory Model.* As the internally employed Palladio Component Model is focused on high-level architectural performance predictions, it does not include a specific model for the *Memory* resource. In CACTOS we have identified possible requirement for such a model within the Enterprise Application use case. The proposed initial memory model will have support for the throughput prediction and help to identify possible further extension needs.
- *Initial Graphical User Interface.* This version uses the Eclipse-based drag and drop graphical user interface provided by Palladio. It offers use of extended visual modelling tools of the Eclipse Modelling Framework (EMF) and Graphical Modeling Framework (GMF).

Table 12 CactoSim Release 2.0: Features, Scope and Description (Subject to Change)

		Date: September 30th 2015
CactoSim Release 2.0 (Intermediate Prototype)	**Planned Feature Name**	**Planned Scope and Description**
	Self-Adaptive System Analyser	This release utilises the SimuLizar plug-in for Palladio in order to provide support for modelling self-adaptive cloud computing systems at design time. Planned modifications for year 2 include the integration of real world optimisation models with the simulation environment. Features will be developed towards the automated retrieval of Logical and Physical Data Centre Models from CactoScale's CDO Model Repository and automated transformation between CactoOpt's OptimisationPlan models and adaptation operations that are performed in the simulated data centre environment.
	Intermediate Black and Grey Box Behaviour Model	Ability to represent the application behaviour without full knowledge of its inner workings, performance prediction being based on obtained historical data and other relevant assumptions.
	Intermediate Energy Model	Gives capability to simulate energy consumption of the cloud data centre.
	Layered Component Model	Introduction to resource containers for Virtual Machines, which includes a representation of Hypervisor resource access abstraction layer.

In the following, a description of the planned features for the CactoSim intermediate release (CactoSim 2.0) is outlined on the basis of Table 12:

- *Self-Adaptive System Analyser.* This release utilises the SimuLizar plug-in for Palladio in order to provide support for modelling self-adaptive cloud computing systems at design time. Planned modifications for year 2 include the use of real world optimisation models within the simulated environment. Features will be developed towards CactoScale model retrieval (logical and physical data centre models) and automated transformation of CactoOpt optimisation models for use in simulation. The adoption of optimisation models will allow for optimisation rules to be executed directly within the simulation.
- *Intermediate Black and Grey Box Behaviour Model.* In order to forecast data centre resource utilisation we need to model the behaviour of each VM. However, cloud data centre providers usually only have access to the VM attributes and coarse workload history information. These few parameters tell very little about the internal software design and are insufficient for creating detailed application behaviour prediction models. Therefore, the first release of CactoSim concentrates on solving this problem by

introducing black and grey box behaviour models for IaaS applications. As the names imply, the behaviour models provide the ability to simulate IaaS cloud applications through coarse granular behaviour models. The black box behaviour models are based and described solely in terms of their resource consumption levels. Their temporal change is determined by matching them to the historical workload data of the node.

- *Intermediate energy model.* Energy consumption prediction is a very important topic for CACTOS and cloud computing in general. It is planned that the initial energy model release will concentrate on static energy consumption data available from the infrastructure specification together with the behaviour models. The energy models will be calibrated using representative benchmarks experiments carried out at node level. This forms the first step towards more sophisticated energy consumption predictions that also account for the migration of VMs and the different power states of hardware resources.
- *Layered component model.* It is planned that this version will give the capability to model and experiment with simulations of resource containers for Virtual Machines, which includes a representation of the hypervisor resource access abstraction layer.

Table 13 CactoSim Release 3.0: Features, Scope and Description (Subject to Change)

		Date: May 31st 2016
CactoSim Release 3.0 (Final Prototype)	**Planned Feature Name**	**Planned Scope and Description**
	Final Self-Adaptive System Analyser	This final release provides support for modelling self-adaptive cloud computing systems at design time, including the use of real world optimisation models within the simulated environment. Features include CactoScale model retrieval (logical and physical data centre models) and automated transformation of CactoOpt optimisation models.
	Final Black and Grey box Behaviour model	Ability to represent the application behaviour without full knowledge of its inner workings, performance prediction being based on obtained historical data and other relevant assumptions.
	I/O Model	Gives ability to adjust the simulation model to include disk I/O consumption measurements. This model will address the recent advancement in the fast and network storage development.
	Final Energy Model	Gives capability to simulate energy consumption of the cloud data centre including transient cost of migrating VMs and switching the power states of nodes.
	Final Layered Component Model	Introduction to resource containers for Virtual Machines, which includes a representation of Hypervisor resource access abstraction layer.

Referring to Table 13, the following are a description of the planned features for the CactoSim final release (CactoSim 3.0):

- *Final self-adaptive system analyser.* This final release provides support for modelling self-adaptive cloud computing systems at design time, including the use of real world optimisation models within the simulated environment. Features include CactoScale model retrieval (logical and physical data centre models) and automated transformation of CactoOpt optimisation models.
- *Final black and grey box behaviour model.* In order to forecast data centre resource utilisation we need to model the behaviour of each VM. However, cloud data centre providers usually only have access to the VM attributes and coarse workload history information. These few parameters tell very little about the internal software design and are insufficient for creating detailed application behaviour prediction models. Therefore, the first release of CactoSim concentrates on solving this problem by introducing black and grey box behaviour models for IaaS applications. As the names imply, the behaviour models provide the ability to simulate IaaS cloud applications through coarse granular behaviour models. The black box behaviour models are based and described solely in terms of their resource consumption levels. Their temporal change is determined by matching them to the historical workload data of the node.
- *I/O model.* The I/O model gives ability to adjust the simulation model to include disk I/O consumption measurements. This model will address the recent advancement in the fast and network storage development. It will be designed to account for differences between read/write speeds and architectural differences between standard Hard Drive Devices, Solid State Disks, RAID versions and Network Attached Storage.
- *Final energy model.* The final version of CactoSim accounts for occurring operational delays and energy consumption that results from VM launches, migrations and switching between the hardware power states offered by power management APIs such as ACPI.
- *Final layered component model.* It is planned that this version will give the capability to model and experiment with simulations of resource containers for Virtual Machines, which includes a representation of the hypervisor resource access abstraction layer. By introducing a layered component model, separate virtual resources assigned to VMs by the hypervisor can be modelled.

IV. THE CACTOS CLOUD INFRASTRUCTURE MODELS

The *CACTOS Cloud Infrastructure Models*, or in short Infrastructure Models, represent the infrastructure of the data centre managed and optimised by the CACTOS Runtime Toolkit. The models define the general structure of a data centre's hardware environment, the virtualisation layer and the load measurements for both physical and virtual layers of the data centre.

The CACTOS Cloud Infrastructure Models were created and are being evolved using a Model-Driven Software Development (MDSD) process. This has drastically reduced the work necessary to adapt the models to iteratively gathered requirements and reduced the effort for ensuring consistency between the abstractions of the physical and virtual layer of the CACTOS tools.

In Model-Driven Software Development, the domains of the developed software are abstracted in common *meta-models*. A meta-model is a formal representation of the knowledge of entities and their connections and dependencies in a certain domain. Changes to the domain model aren't performed manually in implementation code but rather are made directly in the model. This goes beyond the scope of model-based development where the models are mainly used as documentation artefacts. For model-driven development the knowledge from the models is automatically transformed into source code via generative techniques based on model-to-text transformations. MDSD helps avoid a drift between the abstraction and implementation.

The CACTOS tools, namely CactoOpt, CactoSim and CactoScale use the implementation generated from the meta-model to communicate information regarding the data centre's hardware and software environment, as well as the measured system load. All information exchange between the tools is based on the common meta-model. The major advantage lies in avoiding inconsistencies in the information exchange between the tools. The CACTOS Cloud Infrastructure Model is part of the interface that captures the structure and abstraction of the managed data centre.

The CACTOS Cloud Infrastructure Model consists of four models, namely the *Physical Data Centre Model*, the *Logical Data Centre Model*, the *Logical Load Model* and the *Physical Load Model*.

The Physical Data Centre Model represents the physical infrastructure of the data centre. This includes, but is not limited to, the hierarchical rack-node structure, a network model and a description of the hardware installed in each node.

The Logical Data Centre Model models the virtualisation layer, e.g. the Virtual Machines deployed on each of the compute nodes and their virtual communication infrastructure.

The Physical Load Model contains load measurements taken on the physical infrastructure that are caused by the applications running on the data centre.

The Logical Load Model has the same purpose as the Physical Load Model but for the virtualised resource environment: Instead of measurements on physical resources, e.g. the CPU utilisation, it is used to represent measurements performed on virtualised resources like virtual CPUs

Section 1 outlines the structure and entities modelled in the Physical Data Centre Model. Section 2 presents the Logical Data Centre Model in greater detail. The Physical Load Model is explained in the following section 3. Section 4 delineates the Logical Load Model. Section 5 presents utility models.

1. Physical Data Centre Model

The Physical Data Centre Model defines the structure of a data centre's physical infrastructure. It is composed of a Core and an Architecture Type model. The Core model structures the data centre into a hierarchy of racks and nodes, which are interconnected by a physical network. The Architecture Type model allows for the defining of a set of available architecture types. Thereby the deployment of Virtual Machines instances specifically built for certain processing architectures such as X86- and RISC-based processors can be modelled.

Section a) describes the core model in detail. Section b) briefly explains this Architecture Type Repository Model.

a) Core Model

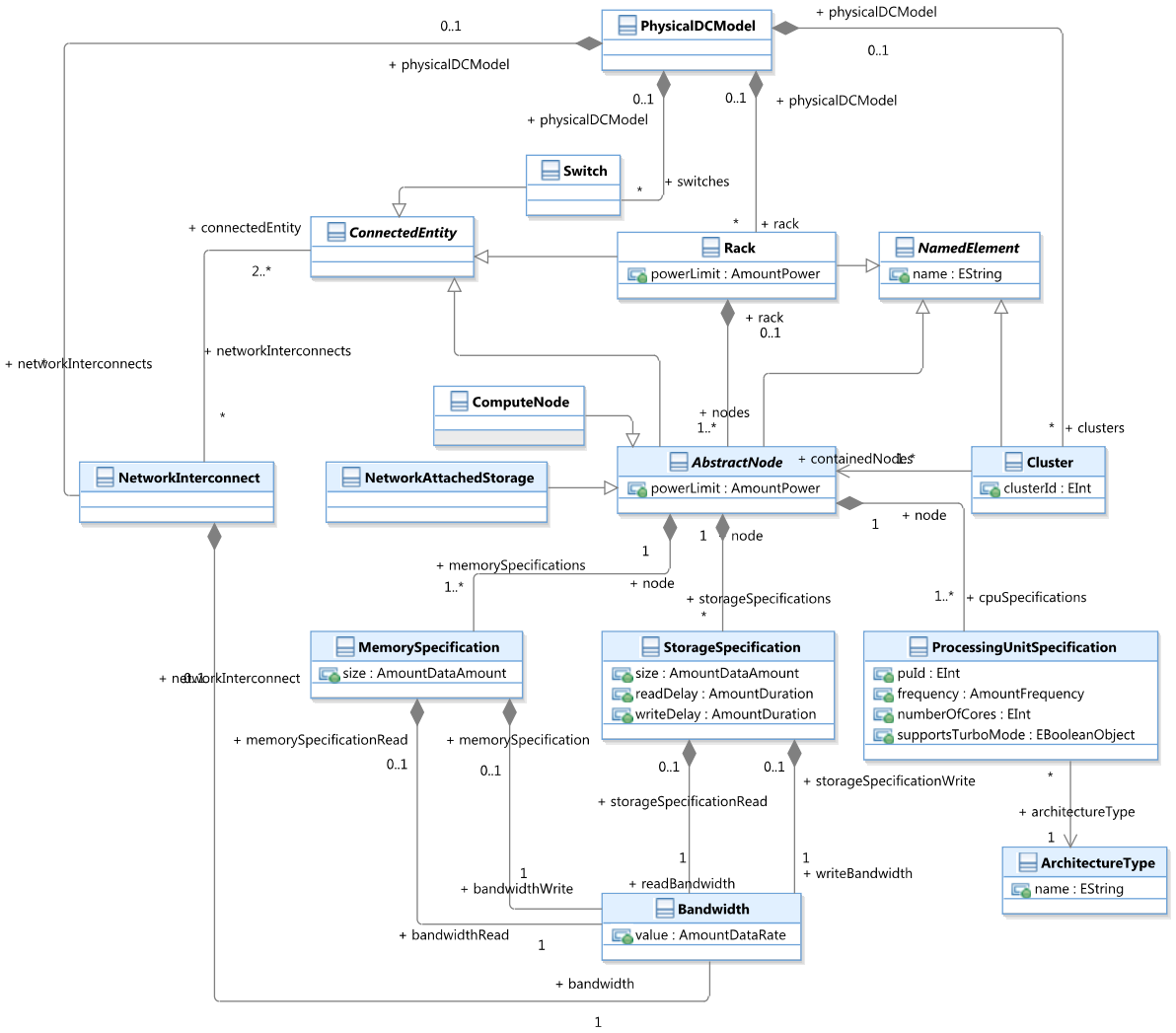

Figure 2 Core Model Sub-Model of the Physical Data Centre Model

The Core sub-model of the Physical Data Centre Model defines a data centre's physical infrastructure and its hierarchical structure. It is depicted in Figure 2. In the following the model elements and their relationships will be discussed.

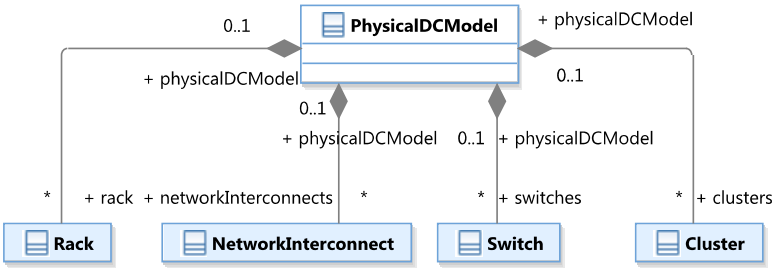

Figure 3 View on the PhysicalDCModel and the Nested Model Elements

Each data centre, represented by the *PhysicalDCModel,* contains multiple *Racks*, *NetworkInterconnects, Switches* and *Clusters*, as can be seen in the sub-view of the overall model that is depicted in Figure 3.

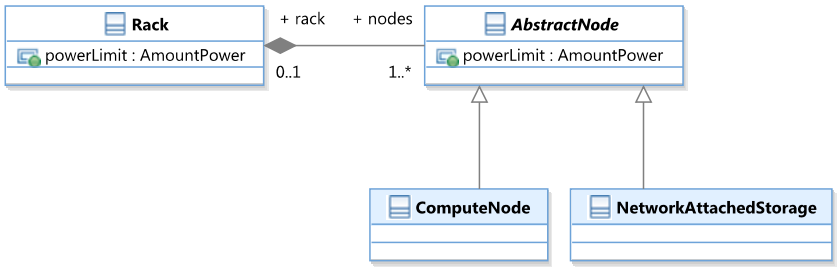

Figure 4 Relationship between Racks and Nested Nodes

Figure 4 sketches the rack-node hierarchy. Each Rack represents a rack unit in a data centre that holds a set of *AbstractNodes*. Both racks and nodes have a *powerLimit* attribute. The attribute defines the peak power that can be supplied by the Power Supply Units (PSU) or Power Distribution Units (PDU) in a node or rack. An AbstractNode can be either a *ComputeNode* or a *NetworkAttachedStorage node*. A ComputeNode corresponds to a regular data centre node on which OS or hypervisor instances can be launched. NetworkAttachedStorage is a specialised node type that is used for storage and cannot run a Virtual Machine.

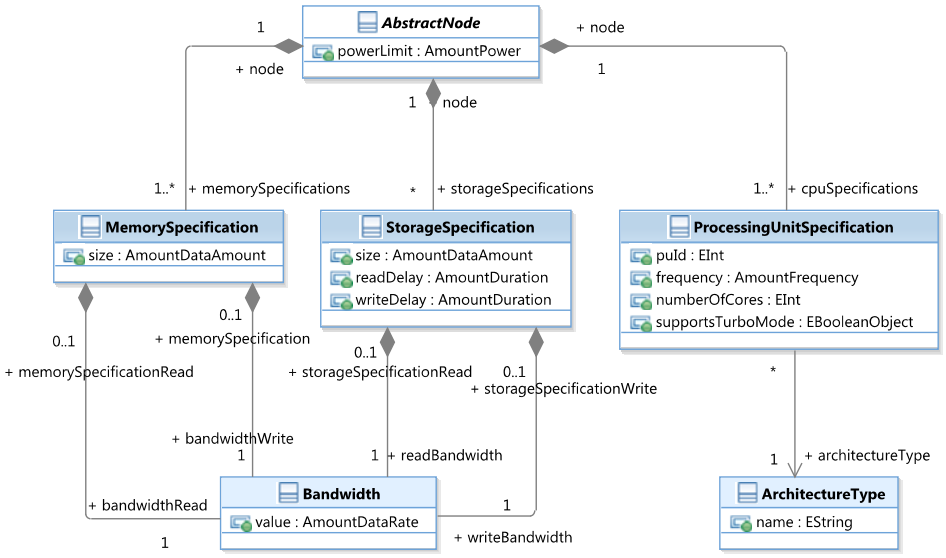

Figure 5 Components and Structure of an AbstractNode

All AbstractNodes have a common structure, as is depicted by Figure 5. They contain memory represented by *MemorySpecification*, storage defined by *StorageSpecifications* and processing units characterised by *ProcessingUnitSpecifications*.

Each *MemorySpecification* comes with a definition of its *size* as well as its read and write *Bandwidth*. An example use of the model elements would entail the specification of 256 GB DDR3 RAM with a read and write bandwidth of 12800 MB/s.

The storage that is specified as *StorageSpecifications* is characterised by its absolute *size*, average *read* and average *write delay*, along with the absolute *read* and *write bandwidth*. Using the StorageSpecification, a Samsung 840 PRO SSD would be specified to have a readBandwidth of 540 MB/s, a writeBandwidth of 520 MB/s and a size of 512 GB. Its writeDelay and readDelay could be specified as 0.05 ms based on measurement results from the AIDA64 random access benchmark.

ProcessingUnitSpecification defines the properties of a Processing Unit (PU). Typical representatives of PUs are CPUs, Graphical Processing Units (GPUs) and specialised processing resources such as Field Programmable Gate Arrays (FPGAs). Every PU in a node is represented by a separate *ProcessingUnitSpecification*. PU´s properties are the ID of the processing unit, its frequency and the number of cores in the processing unit. The Boolean flag *supportsTurboMode* defines whether the processing unit supports a turbo mode for dynamically overclocking the processing unit. An example technology for this is the *Intel Turbo Boost*. In order to support the modelling of today's heterogeneous computing infrastructure, the architecture of a processing unit can be specified via its *ArchitectureType*. Thereby RISC-based CPU architectures can be differentiated from X86-based CPUs, and CPUs can be distinguished from GPUs. These architecture types are defined with the Physical Data Centre's *Architecture Type Repository* model. This model is shown in section b).

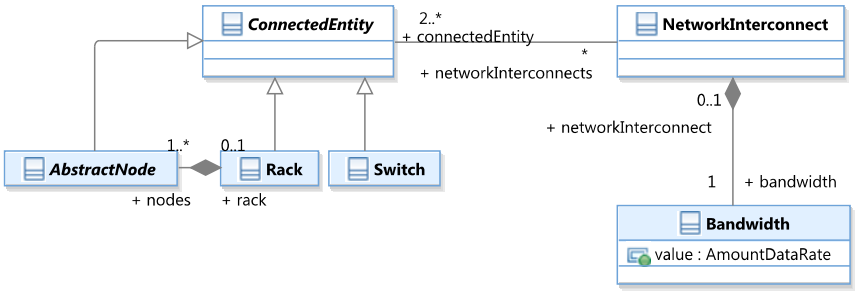

Figure 6 Relationship between NetworkInterconnect and ConnectedEntity

There are different ways the nodes in a data centre can be connected with each other. Figure 6 depicts the model elements and relationships of the Physical Data Centre Model that are used for modelling the network connections.

Every *NetworkInterconnect* connects a group of *ConnectedEntities* among themselves. A NetworkInterconnect defines a multidirectional network connection through which multiple entities are connected. For the NetworkInterconnect the achievable network transfer rate is defined as the *bandwidth* that can be reached on that network segment. ConnectedEntity subsumes AbstractNodes, Racks and *Switches*. A NetworkInterconnect can establish a connection between an arbitrary set of ConnectedEntities. Hence the model has the expressiveness to capture network connections of the nodes among themselves as well as hierarchically structured connections that centrally route the network connections of nodes through their rack.

Switches are used to link up multiple ConnectedEntities by connecting them to the same Switch.

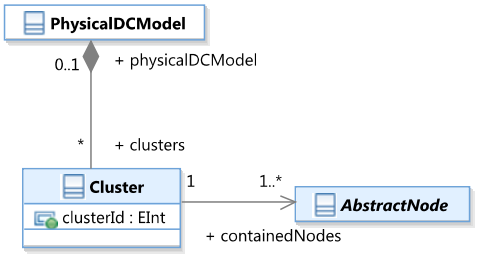

Figure 7 Depiction of the Clusters and the Contained Nodes

A *Cluster* groups together a set of nodes. Clusters aren't part of the rack-node hierarchy as can be seen in Figure 7. Rather, their definition is orthogonal to the grouping of nodes into racks: A cluster can group nodes that are placed in multiple racks. Clusters are used to distinguish compartments of the data centre that are independently managed. It isn't mandatory to define these clusters in an instance of the model. They can, however, be used when a data centre operator partitions the data centre's physical infrastructure, e.g. for different customers or applications.

b) Architecture Type Repository Model

The CACTOS Physical Data Centre Models accommodate the need to differentiate between processing architectures in the *Architecture Type Repository* model. Traditionally, data centres provided homogeneous

computing infrastructure landscapes that typically consisted of x86-based processors. Modern data centres have evolved beyond this, offering heterogeneous and multi-purpose infrastructure landscapes. Regular CPUs are complemented by accelerators that are specialised on certain computing tasks, such as GPGPU and FPGAs. Besides x86 architectures, RISC-based architectures are increasingly used. Since processing unit architectures offer drastically different instruction sets, operating systems and applications need to be implemented and compiled to execute on processors of the respective architecture type.

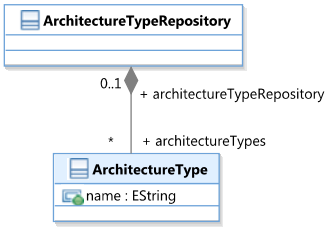

Figure 8 Architecture Type Repository Model

Figure 8 depicts the Architecture Type Repository Model. The model's root element is the *ArchitectureTypeRepository*. The ArchitectureTypeRepository is used to host a set architecture of architecture type description, e.g. for x86 and RISC. The ProcessingUnits described in section a) can then refer to these predefined architecture types. Multiple Physical Data Centre Models all can reuse the same architecture type repository. This helps avoid inconsistencies between different instances of the Physical Data Centre Models.

Every ArchitectureTypeRepository contains a set of architecture types. Each of the architecture types is characterised by its name. Examples for architecture types are x86 32-bit, x86 64-bit, Power PC, ARMv7, ARMv13, Alpha for CPUs and CUDA for GPGPU. Architecture types cover the support of instruction feature set of processors, e.g. SSE2, as well.

2. Logical Data Centre Model

The Logical Data Centre Model describes the layout, composition, and mapping of the virtual to the physical infrastructure in the data centre. It is composed of two parts, the logical data centre core model and the logical data centre hypervisor model.

a) Core Model

Similar to the Physical Data Centre Model's Core sub-model, the Logical Data Centre Core sub-model encompasses description of a number of Virtual Machine level features such as amount of provisioned memory, virtual processing unit (virtual CPU) type and settings; a set of virtual-physical machine mappings and related properties, such as the CPU affinity settings, specification of storage types and access qualities for Virtual Machines; as well as a set of bootstrapping and migration data such as the size and location of Virtual Machine disk images and virtual network configurations and link qualities.

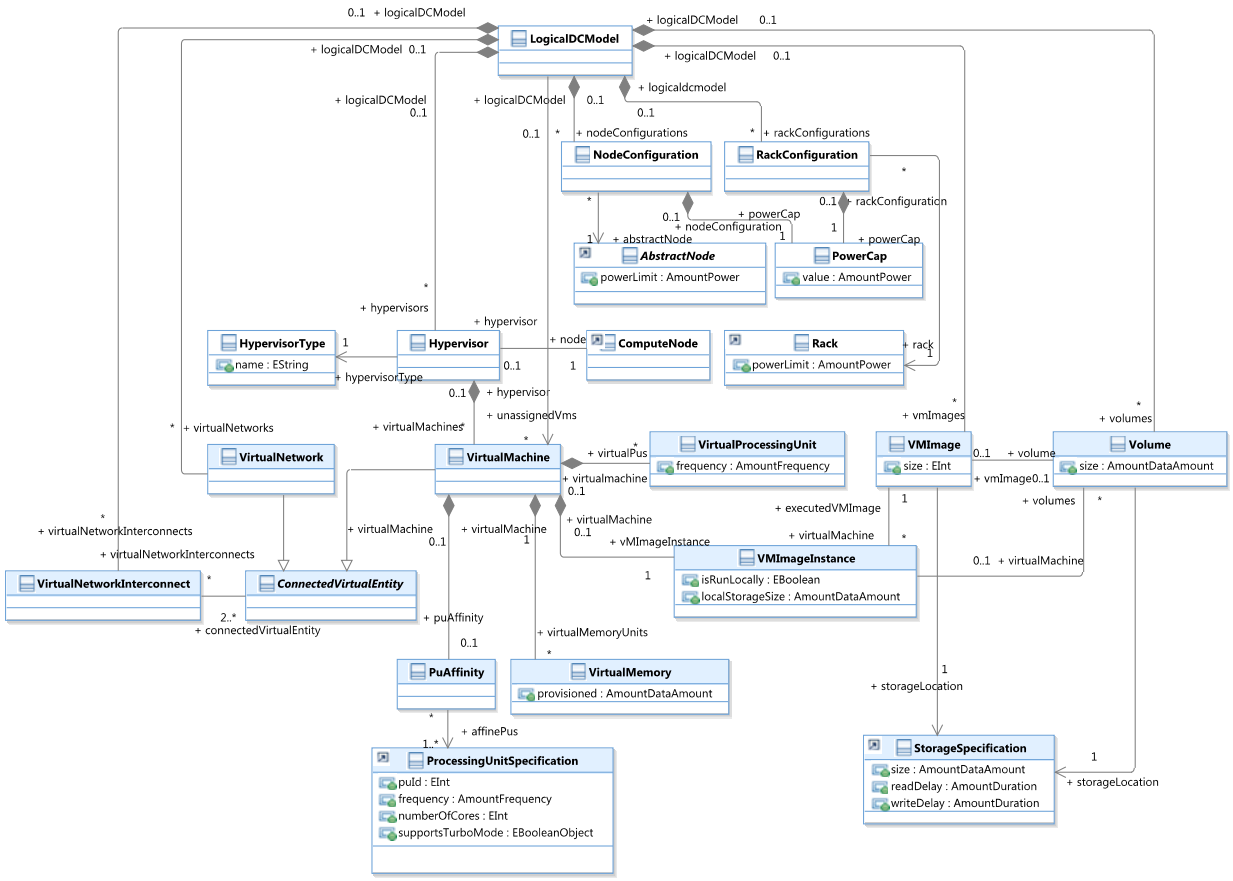

Figure 9 The Logical Data Centre Model representing the Virtualisation Layer within the Data Centre

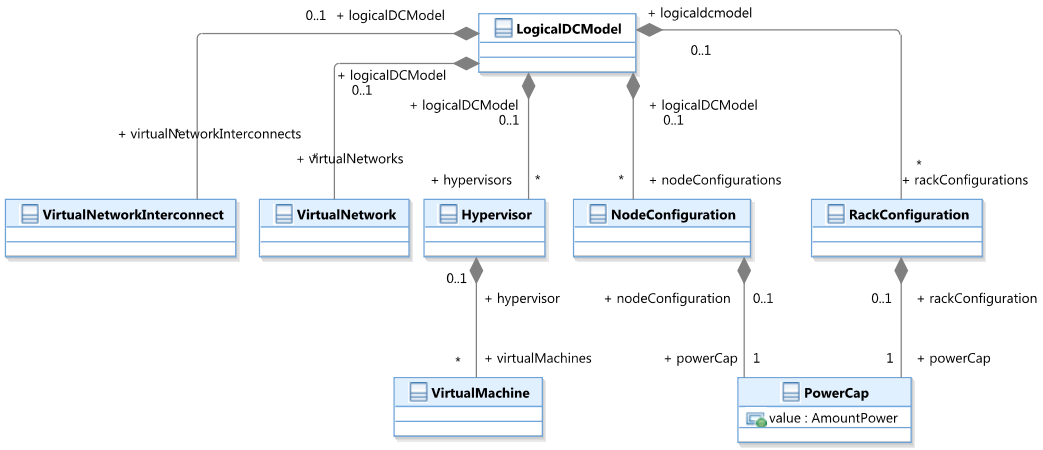

Figure 10 View on the Core Sub-model of the Logical Data Centre Model Containing the Central Elements

Figure 9 gives an overview on the full Core model. As illustrated by the view shown in Figure 10, the Logical Data Centre Model consists of three major parts described in the following: network description, Virtual Machine description, and power infrastructure description.

The network description that is part of the logical data centre core model describes Virtual Machine network interconnectivity. VirtualNetwork and VirtualNetworkInterconnect elements describe the links between VirtualMachine instances. The data centre may define multiple virtual networks composed of links between the virtual network interconnections of individual Virtual Machines. Each Virtual Machine may define multiple virtual network interconnects and participate in multiple virtual networks. Virtual networks are model abstractions for approaches as SDN or VLAN spanning logical network channels within the physical environment.

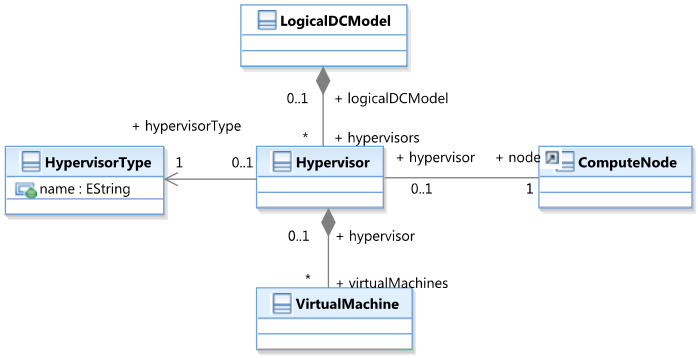

Figure 11 View on the Hypervisor Model Entity and Mapping of Virtual Machines to Physical Machines

Figure 11 shows how the Logical Data Centre Model (LogicalDCModel) links to the Physical Data Centre Model (PhysicalDCModel, described in the previous section) through the Hypervisor element, which specifies the hypervisor type (HypervisorType) for a physical machine (ComputeNode) as well as links between Virtual Machine instances (VirtualMachine) and hosting physical machines. This link essentially describes the mapping between the virtual and physical nodes, as well as the type of hypervisor used on the physical nodes described in the Physical Data Centre Model. Multiple Virtual Machines may be mapped to the same physical machine, but each Virtual Machine may only be mapped to a single physical machine at a time.

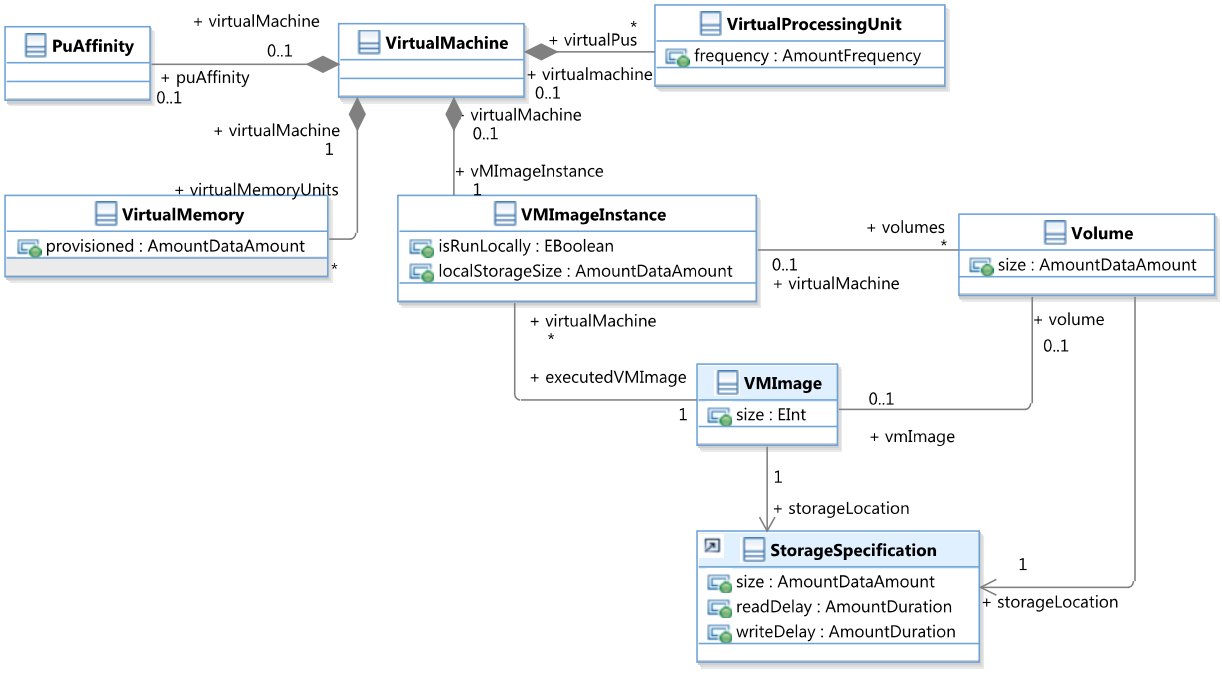

Figure 12 Virtual Machines and Instantiation

A Virtual Machine is hosted on physical machines (ComputeNode) and managed by hypervisors (Hypervisor). Virtual machine disk images and their locations are described using VMImage, Volume, and StorageSpecification entities (see Figure 12). Connections to instantiations of Virtual Machines are modelled using VMImageInstance entities. Individual properties of Virtual Machine hardware resource assignments are described in PuAffinity (CPU) and VirtualMemory (RAM) entities.

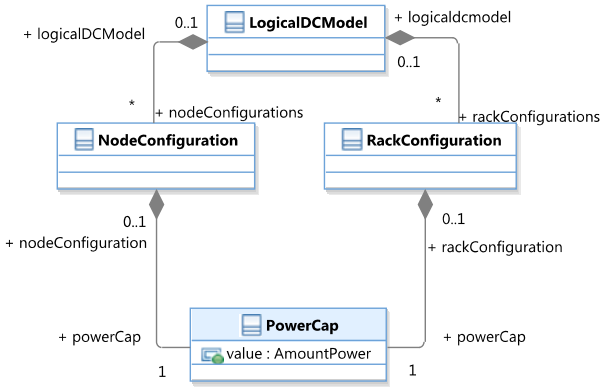

Figure 10 Node and Rack Power Distribution Configurations Contained in the Logical Data Centre Model

The Logical Data Centre Model also encompasses information regarding limitations in power distribution at both node and rack level using NodeConfiguration, RackConfiguration, and PowerCap entities, as shown in Figure 10. Relocation restrictions for power distribution optimisations are included in the model as follows: Individual properties of Virtual Machine hardware resource assignments are described in PuAffinity (CPU) and VirtualMemory (RAM) entities, and hardware power limitations are described at both node and rack level in NodeConfiguration, RackConfiguration, and PowerCap entities. These properties allow optimisation routines to monitor and plan for balanced power consumption within and between virtual data centre nodes and clusters.

b) HYPERVISOR MODEL

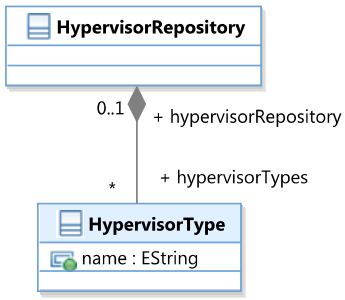

Figure 13 Hypervisor Repository Model for Managing Hypervisor Types

All available types of hypervisors are stored in a central HypervisorRepository, as shown in Figure 11. Each hypervisor is characterised by its name. A HypervisorType represents a specific hypervisor implementation, e.g. Hyper-V, ESXi or Xen. The name can include version information, if required. Since Virtual Machine images are often specific to certain hypervisor types the type information needs to be maintained. By having the Hypervisor

model separate from the Core model, one Hypervisor model instance can be reused for multiple instances of the Core model.

3. PHYSICAL LOAD MODEL

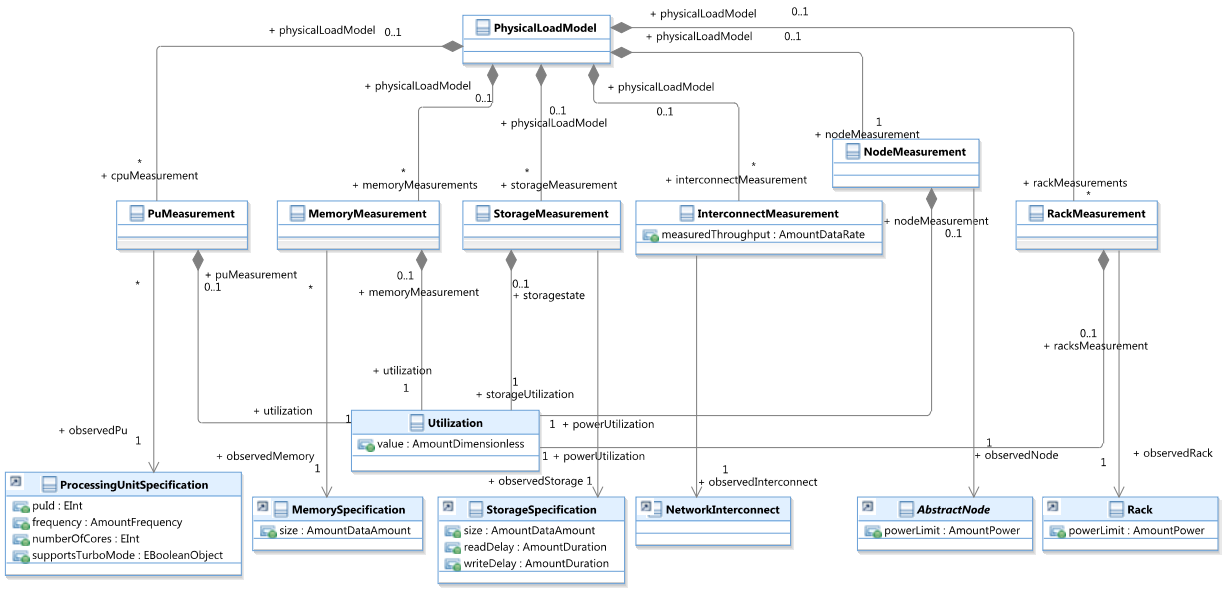

Figure 14 CACTOS' Physical Load Model. It is used to link Measurements performed on the Physical Resources of the Data Center with the Physical Data Center Model.

The *Physical Load Model* is used to link measurements performed on the physical resources of the data centre with the Physical Data Centre Model. It associates all measurements with the following physical resources: processing units, memory, storage, network connections, as well as node and rack-specific power measurements. For all resources, their respective *Utilization* is measured and stored in the *PhysicalLoadModel*. In the cases of *NodeMeasurement* and *RackMeasurement* the *powerUtilization* contains the ratio between measured current power consumption and the maximum peak power that can be supplied by the power distribution unit in the respective component. The *PuMeasurement* contains measurements taken on a processing unit specified by a *ProcessingUnitSpecification*. The *MemoryMeasurement* encompasses the utilisation measured for a specific MemorySpecification. *StorageMeasurement* contains the proportion of space taken up on a storage unit such as an HDD. It is measured for exactly one storage unit that is represented by a *StorageSpecification* in the Physical Data Centre Model. The *InterconnectMeasurement* represents a measurement taken on a network interconnect. For the network interconnects utilisation metrics are difficult to come up with as the maximum throughput heavily depends on the configuration and the setup of both virtual and physical network interconnections. Consequently, *measuredThroughput* stores an absolute value in order to allow the optimisation to reason on bottlenecks. The physical load model also encompasses information regarding measurements on the limitations in power distribution at both node (NodeMeasurements) and rack (RackMeasurement) level.

4. LOGICAL LOAD MODEL

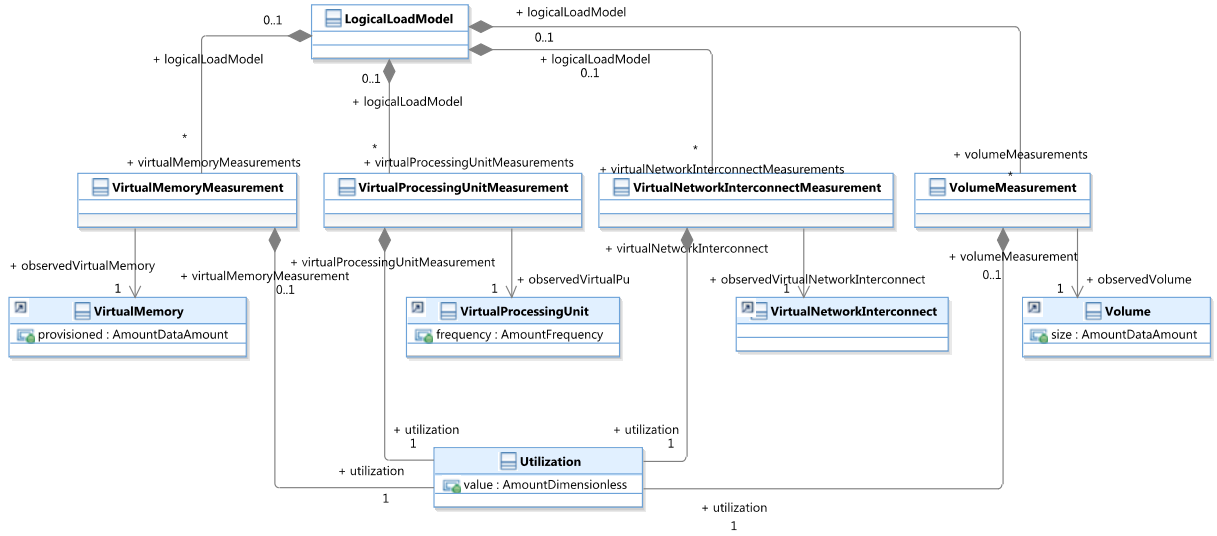

Figure 15 Depiction of the Logical Load Model. The Logical Load Model contains Load Measurements taken on the Virtualised Resources.

The logical load model contains load measurements taken on the virtualised resources. Virtual resources for which these measurements are taken are the virtual memory, processing units and volumes, which are assigned to individual VMs, and the virtual network connections between the VMs. The measurements on the individual virtual resource types are represented by *VirtualMemoryMeasurement*, *VirtualProcessingUnitMeasurement*, *VolumeMeasurement* and *VirtualNetworkInterconnectMeasurement* respectively. All the resources (*VirtualMemoryMeasurement*, *VirtualProcessingUnitMeasurement*, *VirtualNetworkInterconnectMeasurement* and *VolumeMeasurement*) are associated with *Utilization* through multiple attributes in order to express the degree by which they exhaust the proportion of resources provisioned to them.

The virtual memory measurement represents the ratio of memory currently in use by a VM and the amount of memory provisioned to the VM. The *VirtualProcessingUnitMeasurement* links to a VirtualProcessingUnit and contains the *utilization* of the virtual processing unit. The *VirtualNetworkInterconnectMeasurement* annotates a *VirtualNetworkInterconnect* with an utilisation measurement. The *VolumeMeasurement* links to the *observedVolume* size which has a value of AmountDataAmount to describe the size of the VM's volume as a measurement. The volume's current size is kept as a measurement rather than a ratio since there are no natural boundaries on the size of a volume that need to be accounted for when performing runtime optimisations.

5. UTILITY MEASUREMENT MODEL

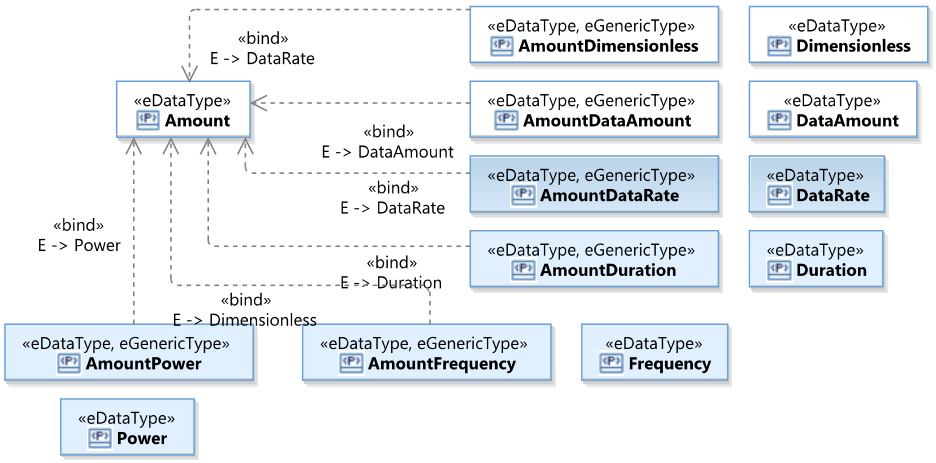

Figure 16 The Utility Measurement Model

Figure 16 depicts the *Measurement Model*. The Measurement Model was defined so that measurements and hardware specifications can be specified together with their international standardised units. The model helps avoiding ambiguities and consistency problems that occur when it isn't clear e.g. if the network throughput unit is KBit/s, MBit/s or KByte/s. It defines the template-based mapping of model elements to Java classes representing the different types and units of measurements.

On the side of the implementation the Open Source library JScience (JScience) is used for linking measured values with their units of measurement. The Utility Measurement Model replicates the structure of JScience's *Amount* (JScience, Amount Interface Documentation, 2006) and *Quantity* (JScience, Quantity Interface Documentation, 2006) class/interface hierarchy.

Classes that implement JScience's Quantity interface represent "any type of quantitative properties of thing. Mass, time, distance, heat and angular separation are among the familiar examples of quantitative properties" (JScience, Quantity Interface Documentation, 2006). The classes *Power*, *DataAmount*, *DataRate*, *Duration*, *Frequency* and *Power* all implement the Quantity interface. Due to technical limitations of UML2, in which the CACTOS meta-models were created, the implements-relationship between *Power*, *DataAmount*, *DataRate*, *Duration* and *Frequency* to the interface Quantity could not explicitly be modelled.

Amount corresponds with JScience's generic *Amount<Q extends Quantity>* class. It represents a quantifiable amount of a quantitative property defined by the Quantity of type Q. As Power, DataAmount, DataRate, Duration, Frequency and Power all extend Quantity in JScience they parameterise the Utility Measurement Model's Amount using UML2's template binding mechanism. For each parameterisation of the Amount class, a utility class is defined, e.g. *AmountDataRate*. AmountDataRate is equivalent to the parameterised Java class Amount<DataRate>.

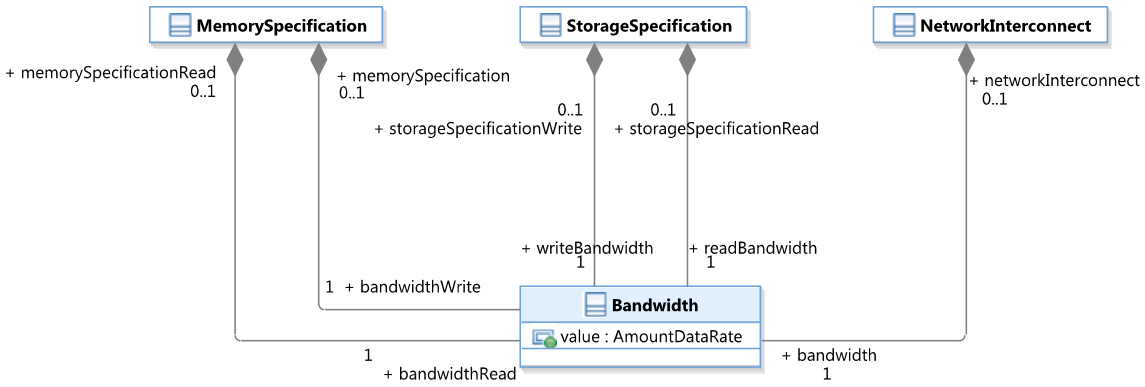

Figure 17 Example for the Use of the Utility Measurement Model's Elements

Figure 17 shows an exemplary use of the Utility Measurement Model in the Physical and Logical Data Centre Core models. The storage (StorageSpecification), memory (MemorySpecification) as well as network connections (NetworkInterconnect) all are attributed with a description of their bandwidth. While storage and memory accesses are typically specified to the basis Byte, the network bandwidth is conventionally specified in relation to the basis Bit. Bandwidth doesn't just store the *value* as a double value. In order to make it clear which unit was used for the specification *AmountDataRate* not only stores the measured value but also its unit.

V. INTEGRATION METHODOLOGY

In this chapter, the main workflows between CACTOS' tooling is described. For each pair of tools, it is described how the tools interact and how this interaction is enabled through CACTOS' model integration approach.

1. PROCESSING CHANGES IN THE ENVIRONMENT

CactoScale is utilising an agent-based monitoring architecture to retrieve load and infrastructure information from the data centre at runtime. The agents and collectors monitoring mechanisms are provided by apache Chukwa module that is extended to support monitoring of a virtualised environment. An agent is running on every node, which collects the output of different monitoring utilities such as iostat, sar, Virt-Top, cigar, top, ps and Df. The output is then collected by one of the available collectors. The collector processes the data and registers the input to HBase.

CactoScale is also capable of gathering and extracting information from application and error logs. An agent installed in each Virtual Machine is able to collect log files and transmit these data to the collectors for processing. The agents can also be paired with in-situ analytics modules to cover the cases where high sampling rates of numerical indicators (e.g. utilisation) are needed, but also to filter the data that flows to the database for post-processing.

2. CREATING AND UPDATING INFRASTRUCTURE MODELS

CactoScale collects system measurements and log data in HBase distributed database where they can be analysed in parallel. The information exchange between CactoScale, CactoOpt and CactoSim components is done by using instances of specifically designed meta-models. The designed meta-models aim at achieving better integration and data exchange amongst the different components of the CACTOS Runtime Toolkit and CACTOS Prediction Toolkit. CactoScale provides measurements of the system load and status of the infrastructure by creating and sharing instances of the Physical Data Centre Model, the Logical Data Centre model, and the Physical and Logical Load models. These instances are stored and accessed in the Runtime Model Repository.

3. TRIGGERING OPTIMISATIONS AND OPTIMISATION PLANS

This section describes how optimisations are triggered through the CactoOpt Infrastructure Optimiser. Part of this section is an excerpt from (D5.2.1 CACTOS Toolkit Version 1). The presented optimisation plan model is an update to the one described in (D3.1 Prototype Optimization Model).

CACTOS infrastructure optimisations are actions performed by the CACTOS toolkit to improve the efficiency of the target infrastructure. From a high-level perspective they entail a multi-step process. In the first step the current state of the infrastructure is inspected, and a set of models representing the system state are built. In the second step optimisation reasoning is performed based on these model instances and an optimisation plan is built for the infrastructure represented in the model. Finally, the individual actions are enacted within the optimisation plans to improve the efficiency of the infrastructure. The interface between the optimisation core of CactoOpt and the rest of the CACTOS toolkit architecture can be described in terms of a sensor-actuator model. The sensors (i.e. what the optimiser sees of the surrounding world) consists of the physical and logical data centre and load models described in Section V, and contains all state the optimiser builds prediction and optimisation models on. The

actuators (i.e. what the optimiser can affect in the surrounding world) consists of a set of optimisation recommendations generated by the optimiser, e.g., actions such as “place Virtual Machine x on physical machine y” or “migrate Virtual Machine x to physical machine z”. The optimisation actions (including their internal order and relationships between them) are described in optimisation plans using a set of predefined actions defined in the XML-based CACTOS infrastructure optimisation language. Optimisation plans are parsed, interpreted, and translated to cloud platform operations by the Virtual Middleware Integration components developed in the CACTOS project. The Virtual Middleware Integration components are specific to middleware solutions such as FCO and OpenStack. For simplicity, the interface between the CactoOpt optimisation core and the rest of the CACTOS toolkit architecture consists only of a single method:

OptimisationPlan generateOptimisationPlan (PhysicalDCModel pdcm, LogicalDCModel ldcm,
PhysicalLoadModel plm, LogicalLoadModel llm,
Deadline deadline)

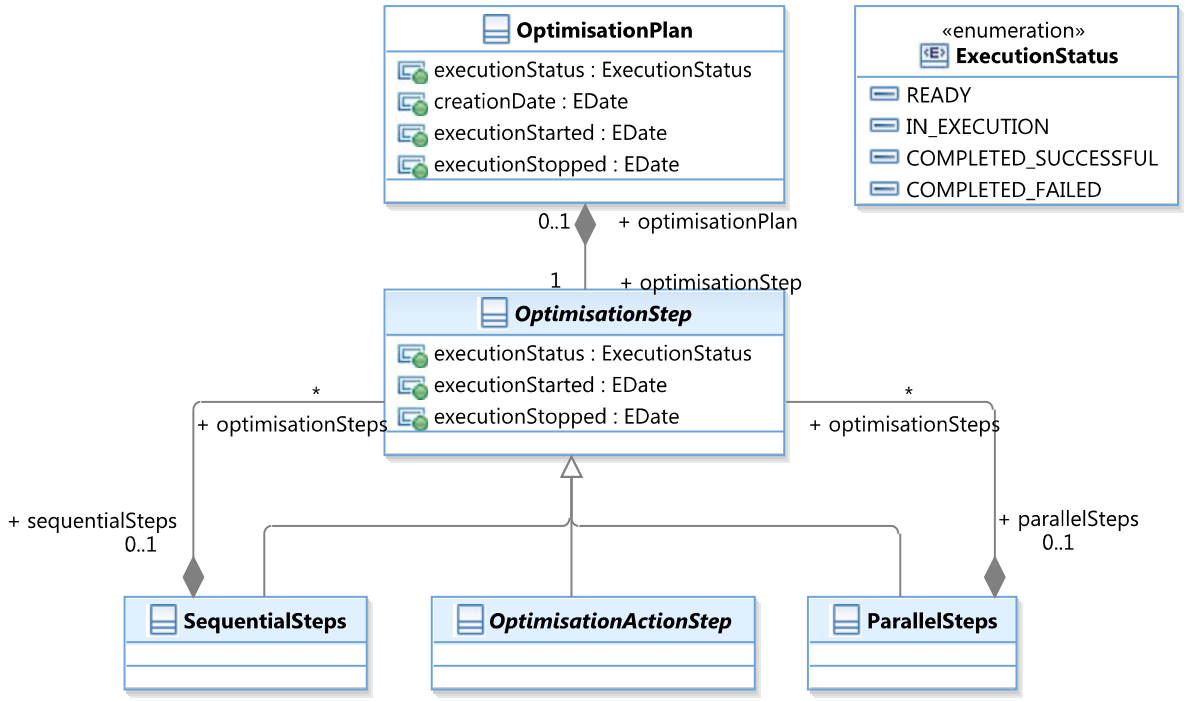

Figure 18 Overview of the OptimisationPlan Model

The result of *generateOptimisationPlan* method is a list of recommended changes to the Logical Data Centre Model and the Physical Data Centre Model in form of an *OptimisationPlan* model instance. The changes can for example affect the deployment and configuration of components on the data centre.

Figure 18 provides an overview of the optimisation plan model. The presented model is an update to the one presented in (D3.1 Prototype Optimization Model) and provides additional management information. It is described in the following.

An *OptimisationPlan* describes all recommended changes. It has an *executionStatus*, which is *READY* as default. Management information describes the time of its creation by CactoOpt (*creationDate*) as well as the points in time for starting (*executionStarted*) and completing (*executionStopped*) its execution, if it is or was executed. An optimisation plan contains the necessary implementation steps via the *optimisationStep* reference. The

executionStatus of an executed plan is *IN_EXECUTION* as long as there is at least on step pending execution. It is *COMPLETED_SUCCESSFUL* if all steps are successful and *COMPLETED_FAILED* if at least one step was not successful. Only one plan can be IN_EXECUTION at the same time preventing interference. The optimizer can decide which plan is executed next ensuring the best outcome.

The *ExecutionStatus* provides information on the execution of a plan. It can be *READY* for execution, *IN_EXECUTION* or completed. The completion could be successful (*COMPLETED_SUCCESSFUL*) or failed (*COMPLETED_FAILED*).

An *OptimisationStep* describes a single implementation step of the plan. It provides information on the execution status and execution times comparable to *OptimisationPlan*. It is abstract and subclasses allow describing the order and basic implementation actions (*OptimisationActionStep*).

Change steps can be executed sequentially or in parallel. A set of steps that can be executed in parallel is nested in *ParallelSteps*, a *SequentialSteps* contains steps that need to be executed sequentially. Possible changes include actions such as the initial placement, VM migration and vertical scaling of resources assigned to VMs. Please refer to (D3.1 Prototype Optimization Model) for an in-depth explanation of the action types and their characteristics.

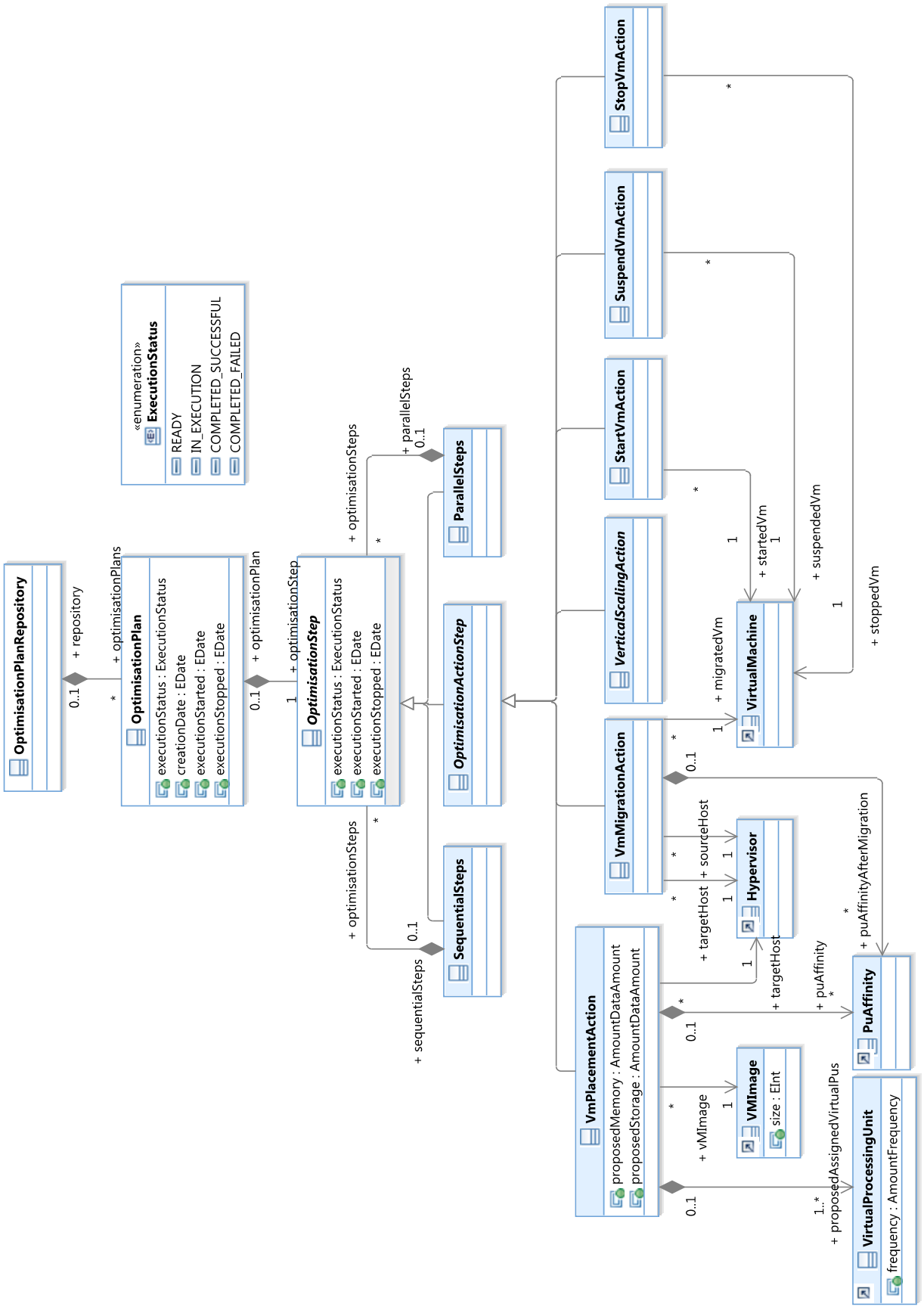

Figure 19 All Elements of the Updated Optimisation Plan Model

Figure 19 shows all elements of the optimisation plan model. It contains the additional element *OptimisationPlanRepository*, which allows storing several plans in one location. This can be used to store several recommended plans for the same data centre.

The optimisation process of CactoOpt can be triggered in two ways: automatically and manually. Automatic invocation can be triggered by the toolkit itself, e.g. periodically on some configured schedule or in response to some internal or external event such as a hardware failure, a utilisation limit being reached or simply the arrival of a new Virtual Machine. Currently, the optimisation processed is triggered by the Cyclic Optimiser component as explained in section VII.1. The simulation triggers optimisations analogous to the runtime scenario through the Cyclic Optimiser Simulation component. Manual triggering of CactoOpt is also fundamentally supported by CactoOpt's optimisation method. This allows for administrators to schedule optimisation following certain events, e.g. following infrastructure maintenance.

4. ISSUING REDEPLOYMENT AND RECONFIGURATION

This section outlines how redeployment and reconfiguration actions in the form of OptimisationActionSteps within an Optimisation Plan can be executed on the middleware of real and simulated data centres. As was explained in sections VII.1 and VII.2, these actions are executed on the APIs of the actual cloud middleware by Virtual Middleware Integration components. This section focuses on the interfaces of the cloud middleware of both real data centre middleware and the simulated middleware of CactoSim.

a) REAL-WORLD ENVIRONMENT

In a real world Cloud environment, the Flexiant Cloud Orchestrator's (FCO) middleware supports the whole range from admission control to redeployments. Optimisation actions are based on a set of key metric data exposed by the FCO platform and processed via CactoScale. This enables CactoOpt to make deployment and optimisation decisions by analysing the data, building infrastructure representations and writing out deployment and optimisation plans to the Runtime Model Storage.

To complete these actions the Virtual Middleware Integration component for FCO will execute the optimisation plans by mapping them to calls on three APIs: The Admin API, the User API and System API. These APIs provide SOAP methods that encapsulate reconfiguration actions available to both administrators and users in an IaaS data centre. An overview of supported reconfigurations is available on (Flexiant Cloud Orchestrator Documentation - SOAP Documentation: Calling the SOAP Admin API and User API). The system API is further subdivided into Cluster API, Network API, Server API, Storage API and Virtual Data Centre API. Each of the APIs encapsulates management and configuration concerns for a group of entities in the data centre controlled by the FCO middleware. A more detailed presentation of Flexiant Cloud Orchestrator's (FCO) middleware APIs is available in (D5.2.1 CACTOS Toolkit Version 1).

In the following we provide a few example mappings between optimisation actions suggested by an Optimisation Plan and the FCO API. Processing the actions and their order within a plan determines the required actions executed by the FCO API's. The current release of VMI FCO supports the migration of a VM, starting a VM and stopping a VM.

Migration actions to move a VM from one physical node to another can be carried out using the Server API method `migrateServer`. The FCO platform then migrates the VM represented by a Server object from one

node to another. Reading the VM UUID and target host IP address from the action in the optimisation plan provides the necessary information.

Stopping and starting an existing VM is realized by the `changeServerStatus()` Admin API. This uses the UUID of a VM stored in the model.

Further mappings to be implemented are if the action requires that a new VM is created. This will employ the User API with the `createServer` method. The method takes a time of deployment and a set of parameters that describes the VM as well as the resources assigned to the VM. The method signature is as follows:

```
Job createServer(Server skeletonServer, List<String> sshKeyUUIDList, Time
when)
```

The properties of the VM are bundled into the `skeletonServer` object. Properties include placement information that maps the VM to a cluster and node. As mentioned previously, images can also be imported via uploads by the user.

A full list of the available configuration parameters of a VM can be found on (SOAP Server).

All vertical scaling actions require the VM to be shut down. To configure an existing VM the FDL `Server` class can be used. Through this the server resources available to the VM can be modified, e.g. the number of virtual CPU cores and available memory. A full list of properties of the `Server` class can be found in the documentation on the Flexiant website (Flexiant Cloud Orchestrator Documentation - FDL Server).

One example for a vertical scaling action is an increase of RAM assigned to a VM. In order to execute the action, the server object associated with the VM representation that is the target of the vertical scaling action is looked up using the FCO API. This is done using the VM's identifier String that is taken from the plan. Then, FCO API will shut down the VM and change the RAM to the set amount as stated in the optimisation plan and finally restart the VM

b) VIRTUAL ENVIRONMENT

CactoSim incorporates the execution of CactoOpt's OptimisationPlan via templates written in model transformation languages such as QVT-Operational. CactoSim defines a separate model transformation for each AdaptationActionStep in the OptimisationPlan. The Virtual Middleware Integration Component of CactoSim realises the adaptation action by transforming the current state of the simulated models.

5. CACTOSCALE AND CACTOOPT

This section sketches the information flow between CactoScale and CactoOpt that is necessary to enable the automatic optimisation of initial deployment and redeployment of Virtual Machines in the data centre. The CACTOS Cloud Infrastructure Models outlined in chapter IV constitute the common exchange format in which all information on the current state and structure of the data centre are persisted. A model representation of the current data centre state is persisted in the Runtime Model Storage, as is discussed in section VII.

This section builds upon the description of the integration-driven mechanisms from the previous section. The succeeding sections explain the interactions between CactoScale and CactoOpt on the Runtime Model Storage and discuss the control flow that is involved in creating and receiving the current system state.

a) INTERACTIONS

CactoScale collects information on the structure and runtime state of the data centre infrastructure using distributed Chukwa-based agents. CactoScale's Collector component puts together and aggregates the collected information into a representation of the infrastructure. It then stores this information for offline analysis in an HBase database.

The CDO Model Generator component of CactoScale is responsible for persisting the current system state in CACTOS Cloud Infrastructure Model instances. For this purpose the CDO Model Generator periodically queries the database for current data centre information. Once it has collected this information, it updates the Logical Data Centre Model with changes in the virtual topology. Additionally, both Physical and Logical Load Models are updated with recent utilisation metrics. Changes to the models are persisted in the Runtime Model Storage.

CactoOpt interacts with CactoScale via the Runtime Model Storage. It fetches the current state of the data centre and load measurements from the repository. In order to ensure that CactoOpt always accesses the models in a consistent state, both CactoOpt and CactoScale only access the models inside of transactions.

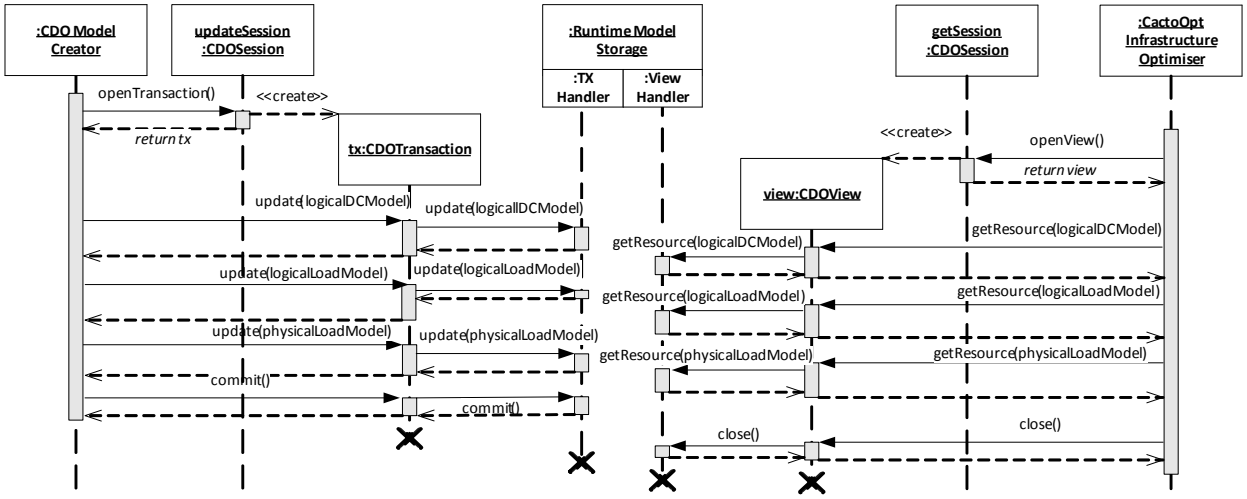

Figure 20 Sequence Diagram Depicting a Simultaneous Model Update by the CDO Model Generator and the CactoOpt Infrastructure Optimiser

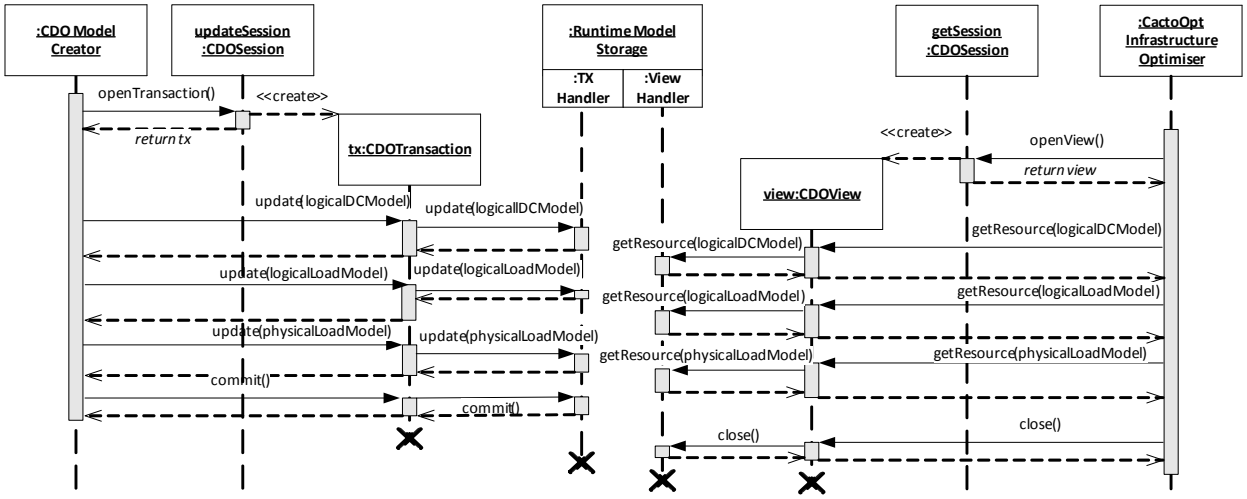

Figure 20 describes how the updates submitted by CactoScale interact with the queries of CactoOpt on the models stored in the Runtime Model Storage. In the CACTOS Runtime Toolkit implementation, the Runtime Model Storage

is realised as a CDO Model Repository component instance. In order to update the CACTOS Cloud Infrastructure Models, CactoScale's CDO Model Generator creates a transaction on this Runtime Model Storage. Within this transaction it then submits a set of update operations on the instances of the CACTOS Cloud Infrastructure Models. Simultaneous to CactoScale's update transaction CactoOpt opens up a read-only transaction *view* on the repository. Within the view the Infrastructure Optimiser of CactoOpt fetches the current state of the CACTOS Cloud Infrastructure Models. Since the changes by CactoScale on the Model Repository haven't been made visible yet through a commit, the view represents the state of the model instances prior to CactoScale's update operations. Once the CDO Model Generator of CactoScale has completed its updates to the Runtime Model Storage, all consecutive transactions and views access the updated model instances.

b) CONTROL FLOW

As there is no direct interaction between CactoOpt and CactoScale, there is no immediate control flow between them. CactoOpt and CactoScale only communicate via the Runtime Model Repository. The CDO Model Repository used in the Runtime Model Storage utilises the transactional functionality of a connected DBMS to ensure that the modifications submitted by CactoScale always leave the models in a consistent state.

CactoOpt queries the current state of the data centre and measured load information from the Runtime Model Repository once an optimisation operation has been triggered. Since updates performed by CactoScale's CDO Model Generator are only made visible once it has committed the corresponding transaction, CactoOpt always gains access to the most recent consistent model version.

6. CACTOSCALE AND CACTOSIM

This section describes planned interactions and control flow between CactoScale and CactoSim components. The reasoning and use cases are also covered earlier in VII.3.c).

a) INTERACTIONS

Similar to the interactions with CactoOpt mentioned earlier, CactoSim interacts with CactoScale via the Runtime Model Storage. The Runtime Model Storage is realised via the CDO Model Repository, offering full support for transactional operations on the CACTOS Cloud Infrastructure Models.

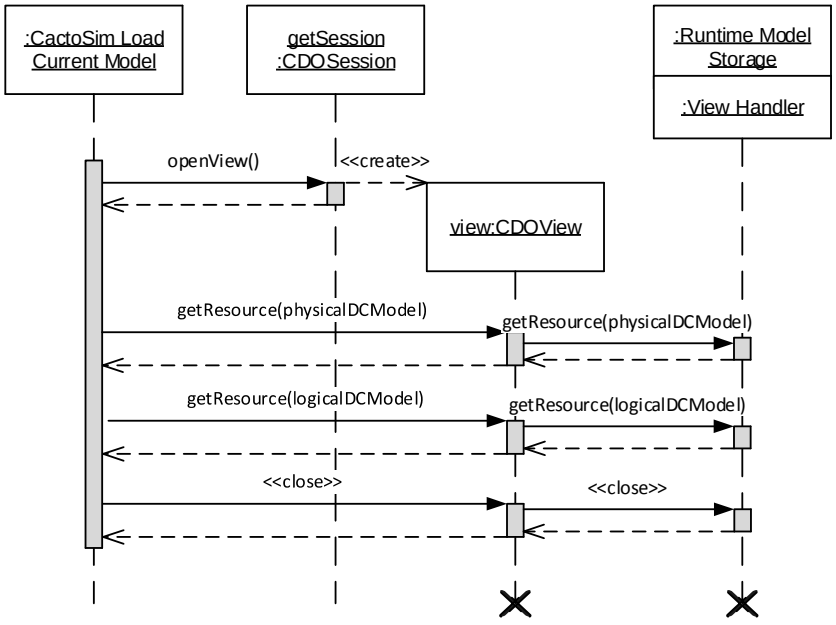

Figure 21 Sequence Diagram of interactions between CactoSim and Runtime Model Storage

Part of CactoSim's functionality is the feature allowing data centre operators to load current datacentre models from the Runtime Model Storage and store them locally in the Prediction Model Storage. As shown in Figure 21, CactoSim interacts only with the view exposed by the Runtime Model Storage therefore it has no control over the data update frequency or other data collection options.

b) CONTROL FLOW

Simulations through the CactoSim Engine are triggered manually by the user using CactoSim's IDE. Before simulations are started, the user can use CactoSim's tooling to retrieve up-to-date sets of models that represent the current state of a data centre. The current model representations are fetched from the Runtime Model Storage. The retrieved models include the Logical Data Centre Model and the Physical Data Centre Model, both of which are then stored locally in Prediction Model Storage for further use within simulations. Instances of the Load Models are not taken under consideration by the simulation, as the simulation itself puts load on the simulated system based on behaviour models. Instead, separate behaviour models need to be manually modelled to simulate the load that is put onto the simulated system, during the first release. In future iterations, CactoScale will support the generation of such behaviour models. CactoSim will be developed to enable the use of the extracted behaviour models as part of the simulation.

As mentioned previously, CactoScale ensures that the CACTOS Cloud Infrastructure models stored in the Runtime Model Storage are consistent and represent the latest available live datacentre version. When CactoSim retrieves the CACTOS Cloud Infrastructure Models from the repository, they are saved to the local Prediction Model Storage which is realised using an EMF Store. These models then can be used for immediate simulation or further manipulation by the cloud operator.

The translation between the fine-grained CACTOS Cloud Infrastructure model and the PCM representation of the entities employed by CactoSim is handled using model transformations.

7. CACTOOPT AND CACTOSIM

This section describes the planned interactions and control flow between CactoOpt and CactoSim components.

a) INTERACTIONS

Simulations carried out using CactoSim will be used to validate cloud topology optimisation models utilised by CactoOpt and provide feedback to the datacentre operator. The produced results then can be analysed, ensuring that the employed topology optimisation strategies work efficiently for the datacentre model that is being evaluated.

It is planned that the CactoSim Engine utilises the Cyclic Optimisation Simulation component to periodically trigger optimisations for the current state of the simulated data centre. For this, the Cyclic Optimisation Simulation component passes the current simulation state to the CactoOpt Infrastructure Optimiser. Once the optimisation has been carried out, CactoOpt triggers the execution of the Optimisation Plan on the CactoSim-specific Virtual Middleware Integration component. The integration component then performs the optimisation actions one the CACTOS Cloud Infrastructure Models simulated by the CactoSim engine.

b) CONTROL FLOW

The Cyclic Optimiser Simulation component periodically triggers optimisations on the CactoOpt Infrastructure Optimiser. Once the CactoOpt Infrastructure Optimiser has compiled an Optimisation Plan with Optimisation Operations, CactoOpt passes this plan to the Virtual Middleware Integration Component of CactoSim. The integration component then carries out the optimisation actions on the current simulation state. Finally, the CactoSim Engine proceeds with the simulation of the updated optimised infrastructure topology.

Optimisations will be periodically triggered. The Logical Datacentre Models and the Physical Datacentre Models, for which the optimisations are performed, are taken from the Prediction Model Storage and consist of models extracted from real datacentres by CactoScale, modified versions of the datacentre or synthetic models used for planning purposes.

The feature of having the same topology optimisation models executed in the real cloud orchestration and the simulation environment allows for a data centre operator to gain valuable information into the system performance forecast. This decision supporting approach brings simulation models and real world cloud environment closer together reducing complexity of simulation adaptation through integration.

VI. DEVELOPMENT AND BUILD INFRASTRUCTURE

In order to allow for an efficient development and build process, a holistic development and build infrastructure was set up for CACTOS. The following presents the architecture and main architectural decisions taken in order to verify and scrutinize the decision at later stages. The build process is supported by a build infrastructure based on the principle of Continuous Integration (CI). CI is a software development practice where the code developed by different teams or members of a team is integrated regularly in intervals no longer than a day. Nowadays CI is commonly supported by build servers that automatically fetch, compile and test developed software components. If successful, the assembled software can then be provisioned to users via update sites or repositories.

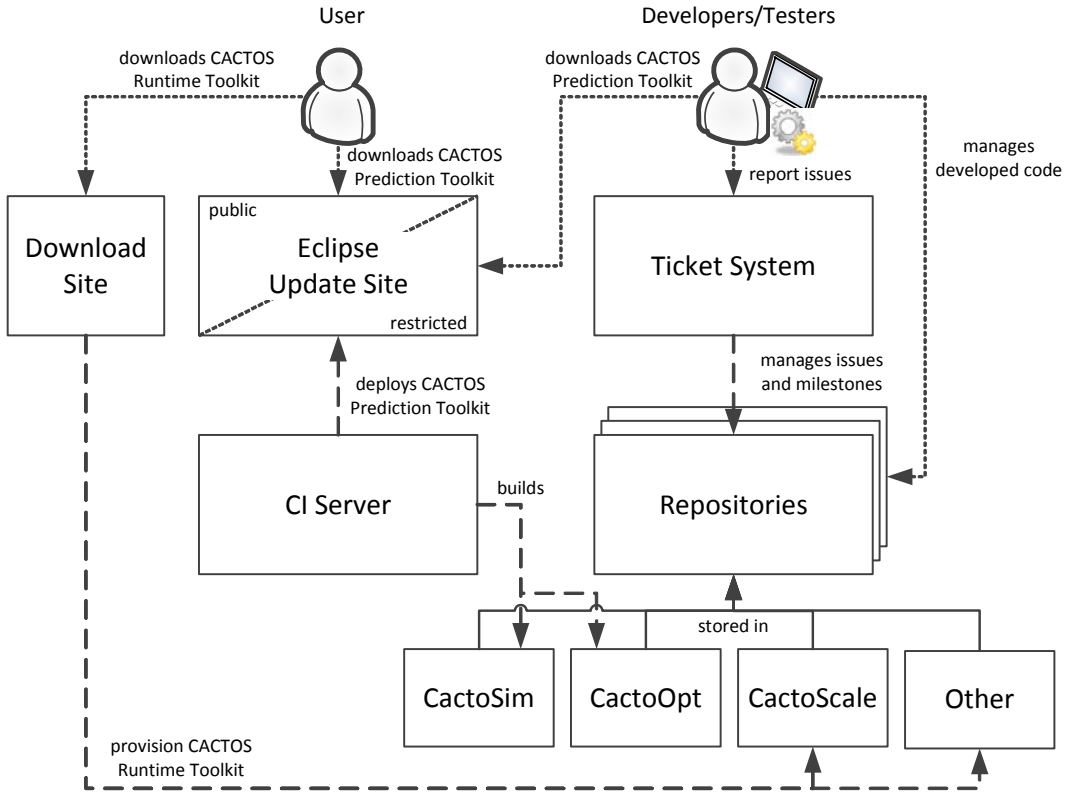

Figure 22 Overview on the Development and Build Infrastructure of CACTOS

Figure 22 depicts the general architecture of CACTOS' development and build system. The code developed for the CACTOS tools and the COTS tools that are utilised by them are stored in repositories. Versioning of the code is handled through versioning systems such as SVN or Git. Versioning take care about different versions of files but does not address release management. A ticket system is used to track development progress and handle reported issues for the individual tools, as well as the integrated CACTOS toolkit. The ticket system is additionally utilised for milestone and release planning. The CI server regularly collects the toolkit code from the repositories and creates a holistic build from it. If the build is successful and a set of predefined automated tests is passed, the CI server creates individual tools as well as integrated products and deploys them on an Eclipse update site for users and developers to download. Besides the publicly available update site with official releases there is an update site for internal testing.

The CACTOS Runtime toolkit is not provided via the Eclipse update site. It is distributed via a separate download site as a COTS software distribution. Unlike the Eclipse-based CACTOS Prediction toolkit the CACTOS Runtime toolkit is set up manually.

Both the CACTOS Runtime Toolkit as well as the CACTOS Prediction Toolkit use CactoOpt to optimise the infrastructure they manage or simulate. In order to minimise the effort of deploying CactoOpt in a real data centre as well as the context of simulation, CactoOpt is provided as an OSGi bundle that can be deployed inside containers capable of running OSGi bundles. One example for such a container is Equinox (The Eclipse Foundation, 2014).

In the following sections, the chosen build system and the decision rationale is discussed based on a set of alternatives that were considered. Section 1 discusses the requirements of the individual tools developed in CACTOS towards the build process for the creation of the holistic CACTOS Prediction toolkit. It also delineates prerequisites that need to be met by the tooling to make them compatible with an automated CI process. Section 2 outlines the software solutions chosen for the build system components and the decision rationale behind choosing said components. Sections 3 through 5 sketch different options for the previously discussed development infrastructure components. Section 6 discusses the selection of middleware for deploying components in a distributed manner.

1. Requirements from the CACTOS Tooling

The tools developed in CACTOS each pose different requirements towards the build infrastructure that need to be considered in its design and setup. These requirements will be discussed in this section.

a) CactoScale

CactoScale utilises COTS such as Chukwa, HBase and Hadoop to collect, persist and analyse measurement data in the data centre. The software tools used by CactoScale are based on different framework solutions. Most frameworks are within the Apache Hbase and Hadoop domain. The connection to the Runtime Model Storage is realized using OSGi and EMF technology. Setting up HBase and Hadoop requires manual deployment and configuration effort. The connection to the Runtime Model Storage requires a configuration file and the deployment within an OSGi container. The only step that could be automated for CactoScale is the build of the Java code used by CactoScale to leverage the COTS tooling. The CI server cannot actually perform integration tests for CactoScale without connecting it to the COTS tooling. As this instrumentation cannot be automated without investing serious effort, the CACTOS runtime toolkit has been excluded from the CI process.

b) CactoOpt

CactoOpt is utilised to optimise the configuration of virtual and physical resources and their deployment. As it is planned for CactoOpt to offer the option to optimise the infrastructure using specialised background algorithm technology that isn't part of the CACTOS project, it is crucial that the licensing terms of the background technology aren't violated by the development infrastructure, e.g. by having their code be publicly available.

c) CactoSim

CactoSim builds upon the Open Source projects Palladio and SimuLizar. The Palladio project already supports CI through its own build server and process. It is desirable to integrate CactoSim with the CI platform of Palladio so that features added to Palladio can also seamlessly be made available in CactoSim if useful. Validated extension of the CACTOS Infrastructure Model can be moved later on to the Palladio Component Model. The Open Source

community benefits from this co-development of CactoSim and the Palladio code base. A timely integration of stable code in the overall code base is of utmost importance to ensure that incompatible or conflicting code is not created.

2. CACTOS Build Infrastructure Selection

This section outlines the build infrastructure of CACTOS and provides the rationale for the chosen setup.

a) Ticket System

It was decided to use the JIRA issue tracker for handling tickets and milestone planning in CACTOS. Deciding factors for going with JIRA were its maturity and usability. A dedicated JIRA server was set up so that all development progress and outstanding issues can be tracked and coordinated project-wide.

b) CI Server

The CACTOS toolkit is built using Palladio's Jenkins instance. The main advantage of using a common build server with Palladio is the increased integrability of the CactoSim simulator extensions with the underlying Palladio simulation framework. It also avoided a significant effort in the setup of a Jenkins build server for Eclipse plugin-based applications.

c) Repository

For managing the code base of the tooling a dual solution was chosen. The extensions CactoSim makes to the underlying Palladio and SimuLizar core are stored and managed in the central Palladio SVN repository. Thereby a feature drift between CactoSim and the actively developed Palladio-based projects it builds upon is avoided and CI is enabled.

3. Ticket System Selection

An effective use of a ticket or issue tracking system is a necessary step towards developing high-quality software. This is especially true when multiple software components need to be integrated, as is the case for CactoSim. Ticket systems allow developers to keep track of and manage development milestones and tasks, as well as issues reported by the users.

Table 14 Overview of Ticketing Systems and Deployment Evaluated in CACTOS

Legend	Positive	Neutral	Negative	Uncertain

Tool Name	Long-Term Availability	Licensing Requirements	Maturity	Effort (Setup and Maintenance)	Visibility
GitHub Ticketing	+	+	+	-	+ (paid) o (free)
CACTOS JIRA	depends on hosting	+	+	-	+
Prose Ticket	EU project, depends on	Fully Open Source	-	Little public documentation,	+

System	funding			requires registration	
Palladio JIRA	+	Fully Open Source, Palladio-centric	+	+	o
CACTOS Redmine	+	o	o	+	+

Table 14 gives an overview of the alternative ticket systems and deployment options considered for CACTOS' development infrastructure. The depicted options will be outlined in more detail in the following sections.

a) GitHub Issue Tracker

The GitHub issue tracker fully integrates with GitHub's Git repository hosting services. Thus it can only be used in conjunction with Git repositories hosted by GitHub. The GitHub issue tracker is proprietarily hosted by GitHub and the number of projects that can be used to track the development process is tied to and limited by the chosen payment plan. Repositories outside of the ones hosted at GitHub cannot be integrated into the issue tracker. This is problematic for CactoSim since it builds upon the Palladio simulator that is actively developed on a separate SVN server. In order to utilise a GitHub issue tracker for the whole CACTOS project a copy of the Palladio codebase would have to be migrated to GitHub, endangering the consistency of the code base and timely integration of stable code by the Open Source Community around Palladio.

b) JIRA

JIRA is a commercial ticket server that supports the full range of managing issues and development tasks. JIRA can connect to a multitude of different repository solutions including Git and SVN. The repositories can be separately hosted from JIRA. The advantage of this is that CactoScale, CactoOpt, CactoSim and the tooling utilised by them can be hosted on separate repositories.

The advantages and disadvantages of available hosting options are discussed in the following.

PALLADIO JIRA

CactoSim builds upon the Open Source simulator framework Palladio. There already is a JIRA issue tracker in place for Palladio-related projects that could also be used for the development of CactoSim. This issue tracker can, however, not be used for projects outside of the Palladio context due to the requirements of the Open Source project license to which it is bound. Other components developed in CACTOS, namely CactoOpt, CactoScale and the CACTOS toolkit would have to be managed in a separate issue tracker. Issues and milestones concerning multiple components would have to be manually synchronised across the ticket systems.

CACTOS JIRA

The hosting of a dedicated CACTOS JIRA allows for the greatest possible control over access and administration rights in the ticket system. Unlike with Palladio's JIRA the setup of a dedicated JIRA instance using a commercial license doesn't pose any requirements on the licensing of the developed software. Although all CACTOS components are made available under Open Source licenses, some of the tools such as CactoOpt might provide the option to utilise specialised closed source background algorithms and technology as part of future releases. Utilising a commercial JIRA setup allows to also track issues related to these background technologies.

The disadvantages of having a dedicated CACTOS JIRA are the administration effort and the uncertainty related to its long-term availability. Both GitHub's and Palladio's issue tracker are and have been used in a multitude of long-term development processes and are expected to be available in the foreseeable future.

We considered two hosting services for the project-specific JIRA that are outlined hereinafter.

Hosted by Atlassian

Atlassian offers a JIRA hosting service for its JIRA product line, including the JIRA issue tracker. Their hosting service is, however, significantly more expensive than self-deploying JIRA on a dedicated server even in the middle term. Another issue connected to the monthly cost is the uncertainty regarding the long-term availability of the toolkit when it is hosted under these conditions.

Self-hosted

While the maintenance and setup effort is higher for a self-hosted JIRA, this option is also cheaper in the middle and long term. The long-term availability of the issue tracker can also be ensured at a reduced cost since it can be deployed on a less powerful node depending on the development velocity.

c) CACTOS REDMINE

A Redmine ticket system is already available for CACTOS as part of the project's internal wiki page. Hence the setup effort is limited to the setup of issue tracking categories and the configuration and instrumentation of subversions. Although the required core functionality such as the integration with repositories, support for tracking issues and milestones is also available in Redmine, its feature scope is more limited compared to JIRA. The usability of Redmine also falls short of JIRA's.

d) OSP ISSUE TRACKER

The OpenSourceProjects.eu (OSP) issue tracker is an issue tracker developed and provided by the Promoting Open Source in European Projects (PROSE) project. Its goal is to promote and support European projects through a common development platform. OSP requires the projects using the platform to fully be made available under an Open Source licensing agreement. This is problematic as it doesn't allow tracking issues that are associated with the use of proprietary background technology as part of future extensions to the CACTOS tooling. The two major disadvantages of the OSP platform as a whole are its immaturity and unclear long term availability.

4. CONTINUOUS INTEGRATION INFRASTRUCTURE SELECTION

At the core of every CI infrastructure stands a build server. The build server automatically collects the current versions of specified software components from a set of repositories and creates a single piece of deployable software from it. If successful, the build server runs a set of tests on the integrated software. These tests are defined by the software developers. A new software build is only made available if the build and all tests were successful. Otherwise the developers are notified of any issues that arose in building or testing the integrated software.

In the CACTOS project we opted to use a Jenkins build server. Jenkins supports the integration of a wide range of build mechanisms, such as Maven Tycho and Buckminster. Jenkins is able to instrument all popular repository types out of the box. Additional repositories and other functionality are supported by the wide array of plugins that are available for Jenkins.

Two hosting options were considered for CACTOS' Jenkins Build server. Both options will be discussed in the succeeding sections.

a) Palladio Build Server

The Palladio Jenkins build server is used for building projects from the Palladio context. The server is fully configured and set up with all required plugins to build Palladio-based products as well as other Java-based applications. Its long-term availability is ensured as it is used in many long-term community and research projects.

b) CACTOS Build Server

As CACTOS has no project-specific requirements beyond the features available for the Palladio build server, there is little benefit in setting up an own build server. On the contrary, it requires additional effort and resources for setting up and running the build server. The only benefits over using the Palladio build server are as follows: An own build server gives full control over the whole build server and processes hosted on it, and not just administrative capabilities in the CACTOS projects. Consequently, build resources do not have to be shared with other projects utilising the same build server.

5. File Repository System Selection

This section gives an overview of the repository technologies and hosting options that were evaluated for the use in the CACTOS project.

a) SVN

Apache Subversion (SVN) is an Open Source versioning and repository software solution released under the Apache license. SVN is widely used by the Open Source Community as well as the commercial development space.

CACTOS SVN

Using the CACTOS SVN for the development of the CACTOS toolkits has multiple advantages. First, it gives the project partners full control over the rights management. Certain subdirectories, e.g. for managing background algorithms can have their access rights restricted. Another benefit compared to the other options is the project focus and reduced integration effort by using a common repository. Exceptions to this are the extensions made to Palladio in the context of CactoSim: Maintaining and developing these extensions separate from the Palladio repository severely increases the necessary development and integration effort.

PALLADIO SVN

The Palladio SVN repository hosts projects related to Palladio. In the context of CACTOS it can be used to host CactoSim and its extensions to Palladio and SimuLizar. It isn't suited to host CactoOpt and CactoScale code and binaries as they aren't related to Palladio. Read access to the Palladio SVN and all code hosted in it are made publicly available which makes it unsuited to host background code and tooling.

b) Git

Git is an Open Source versioning and repository software solution released under the GNU General Public License v2. Unlike with SVN it isn't possible to only check out subdirectories of repositories. Instead, users always create full copies of the utilised repository. The main advantage of Git's over SVN's approach lies in its support of distributed development. The local copies aren't just copies that necessarily need to be synchronised with a central repository once a commit is issued. Instead, every copy is a repository on which commits can be issued.

In order to synchronise changes in a local repository with a central development repository, a merge is performed on both repositories.

GITHUB GIT REPOSITORY

GitHub is a provider that hosts Git repositories. Public repositories can be hosted free of charge while hosting private repositories costs between $25 and $200 per month for 10 to 125 private repositories. The benefit of GitHub's Git hosting services over other providers is the full integration with the GitHub issue tracker.

Using Git for CactoSim would require migrating all the Palladio-based plugins it builds upon from SVN to Git. Otherwise their development would be very difficult as managing CactoSim's extensions separate from the Palladio repository would result in a significant overhead and inconsistencies between CactoSim and Palladio due to the ongoing development of Palladio. The migration would incur a significant effort. For the other tools – CactoOpt, CactoScale and other tooling – the usage of Git poses less problematic. If Git was utilised only for some of the different repositories this would significantly increase the required integration effort as different repository implementations would have to be instrumented by the build infrastructure.

One major advantage of GitHub's hosting services is that they supply an issue tracker for all the hosted repositories as was described in section 3.a).

CACTOS GIT REPOSITORY

Hosting a CACTOS-specific Git server provides no benefit over the use of the GitHub Git repository beyond having full administrative control. This comes at a high price when compared to the use of GitHub's services. If an independent repository were to be hosted for CACTOS, an issue tracker would have to be set up manually on top of the required work for setting up the Git main repository itself.

c) OSP Repository

OSP offers both SVN and GIT repositories for hosting projects. For these repositories the same advantages and disadvantages as for the OSP ticketing system apply: The long-term availability of the repositories is unclear and the integration with the platform-specific ticketing system is immature.

d) Partner-Specific Repositories

Another option for developing code in Palladio is the maintenance of separate, tool-specific repositories for each of the tools. The advantage of this is that it allows all partners to maintain control on the access to their code base. Yet the separation of development over multiple repositories makes the integration of CactoOpt, CactoScale and CactoSim into one common toolkit much more challenging.

6. Component Deployment Middleware Selection

Different parts of the CACTOS tools can be distributed and deployed on different machines within a data centre, e.g. CactoOpt and the Virtualisation Middleware Integration. This requires a middleware for handling the execution and communication of components in a distributed fashion and allowing a coherent access across CACTOS. We evaluate the three technology decisions to use OSGi, JEE or the Custom as solutions.

Table 15 Evaluated Component Deployment Middleware Solutions

Component Deployment Middleware	License Availability	Development and Maintenance Effort	Security	Integration with External Tools	Re-use within CACTOS and Homogeneity
OSGi	+	+	+	+	+
JEE	+	+	+	+	-
Custom	+	-	-	-	+

Table 15 summarizes the evaluation, which is described in the following sections. The following categories were evaluated: License Availability checked if at least one implementation has a license, which can be directly used in CACTOS and with the EPL license. Development and Maintenance Effort covers the effort required within the CACTOS consortium. Security addresses if authentication, authorization and protected communication adhere to current standards. Integration with External Tools estimates the ease to integrate CACTOS tools with non-CACTOS tools, e.g. in a dashboard or control panel. Re-use within CACTOS and Homogeneity addresses the ease of code re-use between the different CACTOS tools and the homogeneity of the development environment.

The decision was made to use OSGi because it ranks highest.

a) OSGi

A combination of Apache CXF, Jersey, Felix and Karaf covers all required OSGi features with an Apache License Version 2.0, which is compatible with EPL. Development and Maintenance is provided by the Apache community and all Apache projects are actively maintained. State-of-the-Art security concepts are implemented and can be used. Regarding the high standards of Apache projects they should be well-tested. The projects provide access via remote OSGi and Web Service (REST and SOAP) communication standards. The integration with external tools is therefore well supported. The development environment Eclipse and CactoSim are based on OSGi. Code and plug-ins can be shared between the different tools easily. A homogeneous development environment with the same tools and technologies can be provided.

b) JEE

JEE containers, e.g. Apache Axis2, are available with an Apache License Version 2.0, which is compatible with EPL. Development and Maintenance is provided by the Apache community and the container is actively maintained. State-of-the-Art security concepts are implemented and can be used. Regarding the high standards of Apache projects they should be well-tested. The projects provide access via Web Service (REST and SOAP) communication standards. The integration with external tools is therefore well supported. The development environment Eclipse and CactoSim are based on OSGi. Code cannot be shared easily, as JEE has a different concept than OSGi plug-ins. The development environment would not be homogeneous as CactoSim and any other Eclipse-based tooling uses OSGi technologies.

c) Custom

A custom solution could be released under any license. Development and Maintenance would have to be provided by the CACTOS consortium. Security concepts need to be implemented and tested. There is a high chance that this

error-prone activity requires high effort for ensuring proper behaviour. The integration with other tools would require the additional implementation of communication standards, e.g. to provide proper Web Service handling. It is not desirable to re-use existing frameworks. The development environment could be selected as fit.

VII. INTEGRATION TOOLING

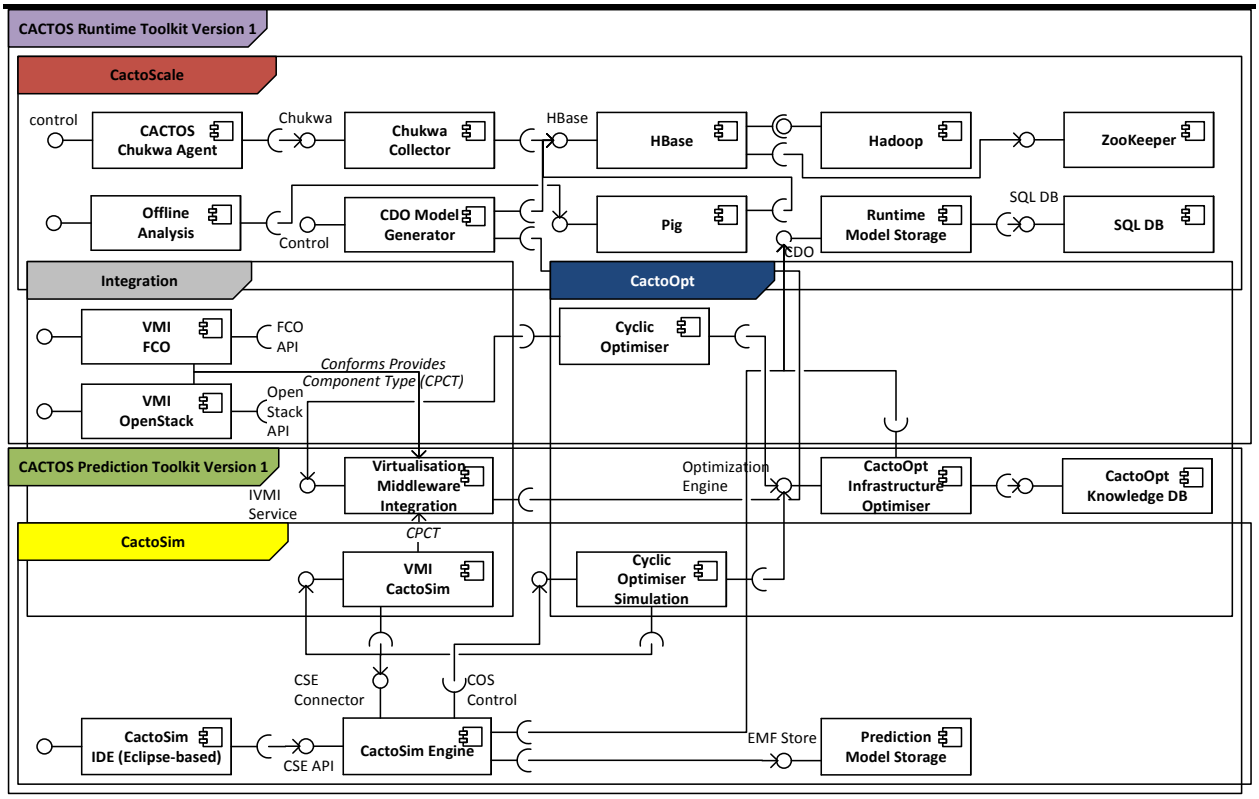

Figure 23 Structural Assembly of Components in the CACTOS Architecture

This section provides an overview on the tooling used in both the CACTOS Runtime Toolkit and the CACTOS Prediction Toolkit. Figure 23 depicts the architecture of the toolkits, showing interconnections and dependencies between both toolkits. The *Integration* package crosses boundaries of the toolkits and tools and denotes the components responsible for the integration with Cloud Middleware.

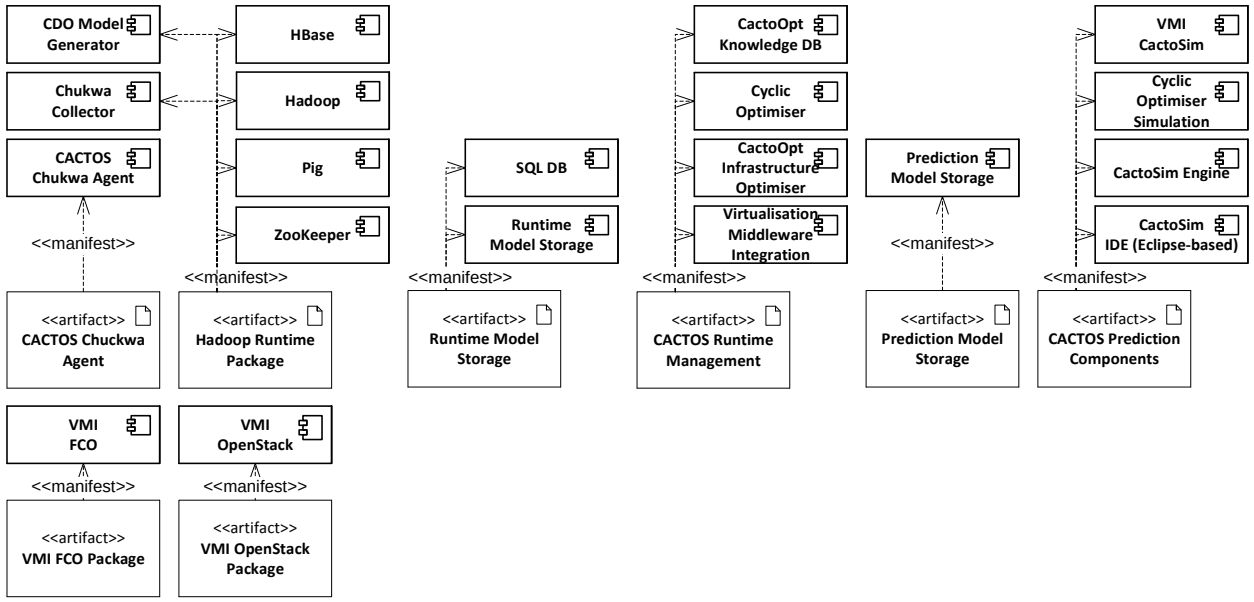

Figure 24 Assembled Artefacts

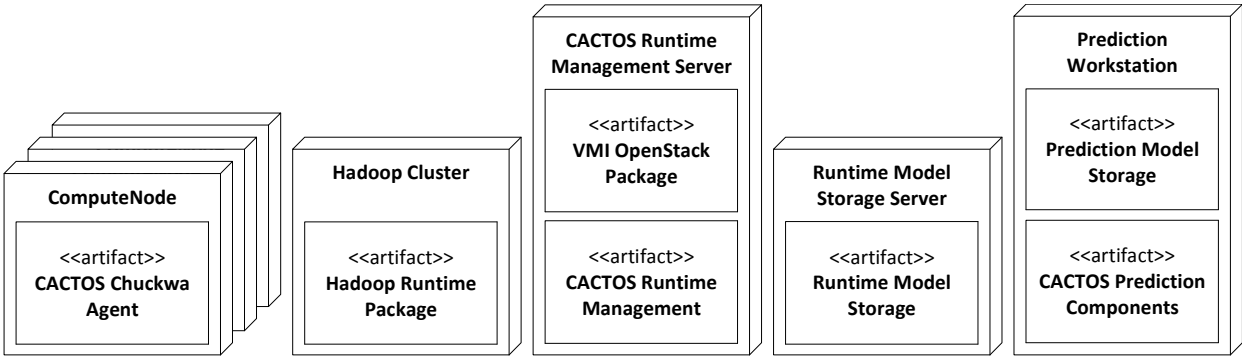

Figure 25 Artefact Deployment Example

Figure 24 shows the assembly of artefacts from the CACTOS Architecture for deployment. Figure 25 provides an example how these artefacts can be deployed within a data centre's infrastructure. Both Figures use Unified Modeling Language (UML) notations.

In the following, an overview of the technical dependencies between the tools will be given. For CactoScale, CactoSim and CactoOpt only the components with a major role in the integration of the CACTOS Runtime Toolkit and the CACTOS Prediction Toolkit are displayed. For a high-level overview with a focus on functionality provided by the CACTOS toolkits, please refer to chapter II.

Section 1 focuses on the relations between the components that are part of the CACTOS Runtime Toolkit. Section 2 sketches the relationship of components for the CACTOS Prediction Toolkit. Section 3 outlines how the current system state is persisted and communicated between the components of the toolkits. Section 4 sketches how instances of the CACTOS Cloud Infrastructure Model instance are transformed into instances of CactoSim's simulation models.

1. CACTOS RUNTIME TOOLKIT

CactoScale and CactoOpt are part of the CACTOS Runtime Toolkit. The *Runtime Integration Control* component provides an interface to the data centre operator. Through this interface the operator can control and monitor the data centre infrastructure and trigger optimisations on it. In order to allow managers to get an insight into the operational state and QoS of the data centre, extensions of the interfaces towards a high-level graphical monitoring in the form of a Runtime Integration Monitoring component might be desirable but are not planned for the prototypes.

CactoScale collects its measurements through *Chukwa Agent* components that are distributed over the data centre nodes. The Chukwa agents funnel the collected information towards Chukwa collectors that then persists the information in the Hadoop-based *HBase* database. HBase is an open-source, distributed, versioned, non-relational database, which provides Google Bigtable-like capabilities. HBase builds upon the functionality provided by Hadoop and the Hadoop File System (HDFS). HBase is a key-value store that works as sparse, persistent, distributed, multidimensional, sorted map. It works as an ordered map by associating keys to values allowing the efficient handling of vast amounts of data.

The primary task of the Chukwa Collector component is to parse the collected data from the agent and store the extracted information on the connected HBase database. The Chukwa Collector component can optionally be deployed in distributed manner. When deployed in such a manner, it provides a fail-safe solution in case one of the component instances stops responding. Furthermore, the distributed deployment of the Chukwa Collector

component allows for the monitoring tool to be scaled by adding more collector modules. In this way, a large number of nodes can be monitored. A scaling is not required for the testbeds.

CactoScale stores information on the current system state and topology in the connected Runtime Model Storage component. The Runtime Model Storage is responsible for persisting the current system state in CACTOS Cloud Infrastructure Model instances. The CDO Model Generator module periodically queries the database for current data centre information. Once it has collected this information, it updates the CACTOS Cloud Infrastructure Models. The Logical Data Centre Model is updated with changes in the virtual topology. Additionally, both Physical and Logical Load Models are updated with recent utilisation metrics. Changes to the models are persisted in the Runtime Model Storage, as is explained further in section 3. The Physical Data Centre Model isn't updated at the same frequency since updates to the physical infrastructure occur very rarely. As of now, the creation and update of the Physical Data Centre Models requires some manual input. Some of the properties, such as the hierarchical nesting of nodes into racks need to be set manually.

The *Cyclic Optimiser* periodically triggers optimisations on the *CactoOpt Infrastructure Optimiser* component. Different *Infrastructure Optimisation* component implementations can be utilised to identify a set of infrastructure optimisations for the managed data centre. The *CactoOpt Infrastructure* Optimiser component accesses the model representation of the current system state from the Runtime Model Storage. It then launches an infrastructure optimisation through a call on an *Infrastructure Optimisation Algorithm* for this model instance. The optimiser algorithms use the *CactoOpt Knowledge DB* to make decisions based on historic measurement data. The optimisation operations proposed by CactoOpt are then realised in the data centre by the *Virtualisation Middleware Integration (VMI)* component. There are different implementations of the VMI component that each apply CactoOpt's proposed optimisation plan for specific cloud middleware APIs, e.g. of OpenStack.

2. CACTOS PREDICTION TOOLKIT

The CACTOS Prediction Toolkit provides services through which the QoS of data centres can be predicted using simulations. It can be used a stand-alone tool or in combination with the CACTOS Runtime Toolkit. The CACTOS Prediction Toolkit provides services with which the impact of infrastructure optimisations can be predicted. Besides this it can also be used to evaluate how changes in the workload affect the QoS.

The integration of CactoOpt's optimisation algorithms replicates the assembly of runtime management and optimisation components in the CACTOS Runtime Toolkit. The *Cyclic Optimiser Simulation* component periodically triggers optimisations on the CactoOpt Infrastructure Optimiser component. Instead of fetching the model instance representing the current system state, the CactoOpt Infrastructure accesses the Infrastructure stored within the CactoSim Engine. The Optimisation plan proposed by CactoOpt is then passed back to the optimiser and realised in the simulated data centre by the *Virtualisation Middleware Integration (VMI)* CactoSim component.

3. STORING AND SYNCHRONISING MODEL INSTANCES

This section outlines the CACTOS Cloud Infrastructure Models are stored and synchronised between the components of the CACTOS Runtime Toolkit and the CACTOS Prediction Toolkit. Section a) provides a detailed overview on the deployment and assembly of CactoScale components that are used for creating and synchronising the model instances from the data centre runtime environment. Section b) outlines how the runtime models are stored and synchronised by CactoScale. In section c) it is explained how the CACTOS Prediction Toolkit accesses the models extracted by CactoScale and how it stores them as input for simulative analyses.

a) CACTOSCALE ARCHITECTURE OVERVIEW

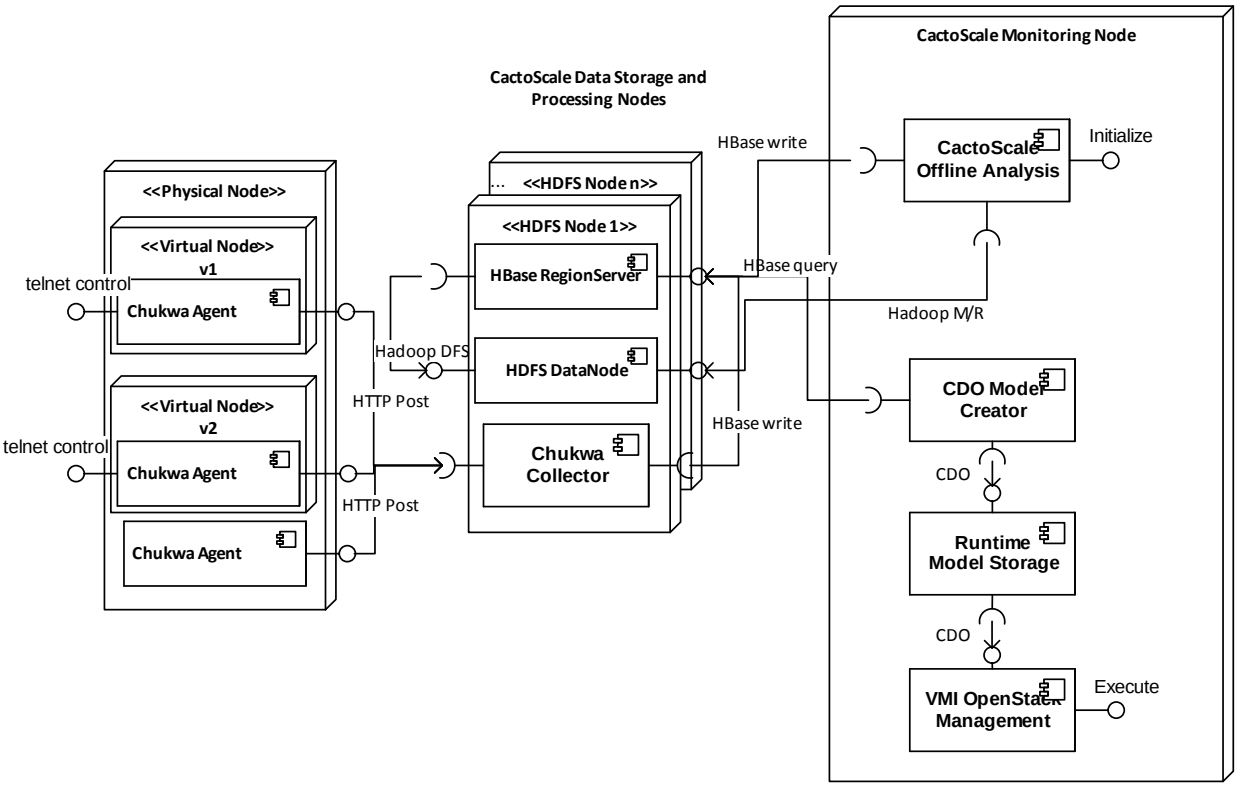

Figure 26 CactoScale Architecture

Figure 26 illustrates the architecture overview of the CactoScale tool. CactoScale provides a scalable framework for collecting and processing data from a cloud data centre. The major components of this tool can be distinguished in two categories: a) the components placed in the system being monitored and b) components which belong to the monitoring infrastructure. In the first category encompasses Chukwa agents which are distributed over the data centre nodes. A Chukwa agent consists of adaptors, which are dynamically loadable modules that run inside the agent process. There is generally one adaptor for each data source: for each file being watched or for each Unix command being executed. The behaviour of every agent can be accessed and controlled independently by connecting to the specific agent on port 9093 using telnet. The agent can utilise a monitoring tool, such as iostat, sar, Virt-Top, cigar, top, ps and Df, by using a module to collect the output.

b) RUNTIME MODEL INSTANCES

CactoScale stores the information collected at runtime in model instances of the CACTOS Cloud Infrastructure Models using the facilities provided by the *Eclipse Modeling Framework (EMF)*. EMF models can be stored in versioning systems like EMF Store, in databases via CDO Servers and serialised using the *XML Metadata Interchange (XMI)* format.

CactoScale constantly collects new information, which needs to be represented in the models of the system's current state. This state information is kept in an instance of the CACTOS Cloud Infrastructure Model which is

updated periodically. CactoOpt performs its infrastructure optimisations based on the current system state. Information on the current system state is shared using this constantly updated CACTOS Cloud Infrastructure model instance. The CactoSim Simulation Engine can access these model instances from the Runtime Model Storage.

As CactoScale continuously modifies and updates its model instance it needs to be ensured that CactoOpt accesses a model instance. For this purpose the Runtime Model Storage Component is realised via the *Connected Data Object (CDO) Model Repository* in the CACTOS Runtime Toolkit. The *CDO Model Repository* is a persistence framework for EMF models. It uses a database to maintain its model instances. In the CACTOS Runtime Toolkit, a MySQL component instance is used as part of the default solution. When CactoScale updates the model, CactoOpt can still concurrently access a consistent model state. Once CactoSim has updated the model, the changes are made available to CactoScale.

c) Simulation Model Instances

The CACTOS Prediction Toolkit is coupled with the CACTOS Runtime Toolkit in order to predict how infrastructure optimisations, changes in the user load or hardware failures will affect the Quality of Service (QoS). For this purpose, the runtime integration tooling can request the CACTOS Cloud Infrastructure Model representing the current state from the CDO Repository. The integration tooling then stores the model instance in a separate *EMF Store* model repository making it available to CactoSim simulations. The reasons for choosing a separate model repository are as follows.

First, the CACTOS Prediction Toolkit can also be used as a standalone product independent from the CACTOS Runtime Toolkit for data centre planning or scientific simulations. Therefore, it needs to be able to handle the storage of model instances independent of CACTOS Runtime Toolkit's CDO store. The simulation must additionally support models not representing the current state, e.g. because potential changes in the data centre are predicted.

Second, EMF Store is designed for offline use similar to versioning systems. The Runtime Integration Tooling and CactoSim do not need to constantly stay connected with the EMF Store. Instead, they can commit and update their copy of the model instance when necessary. Simulations are invoked significantly less frequently than the infrastructure state extraction by CactoScale. Simulated models are permanently available and allow the repetition of simulation runs as well as analysing differences and commonalities between different model versions. For these reasons EMF Store suits the storage of model instances for simulation better than the CDO Store.

4. Simulating Model Instances

CactoSim builds upon the basis of Palladio and SimuLizar to predict the QoS properties of Virtual Machines deployed in a data centre environment. SimuLizar simulates the impact that adaptations such as redeployments of components or the reconfiguration of load balancer parameters have on the QoS properties of a component-based software system. The simulations are performed for instances of the Palladio Component Model (PCM) that describe the structure, deployment and usage of a component-based software system.

In the IaaS scenario of CACTOS, all services provided to users by a deployed VM remain hidden to the data centre operator. Unlike the infrastructure models in CACTOS, Palladio assumes detailed knowledge of the application's architecture. This includes a description of provided and required services with the performance-relevant behaviour of their implementations. In order to bridge the gap between Palladio's white box behaviour modelling for individual components of an application and the black box behaviour models of CACTOS, a model

transformation is employed. The transformation is lossless since Palladio's model is more fine-grained than the CACTOS behaviour models.

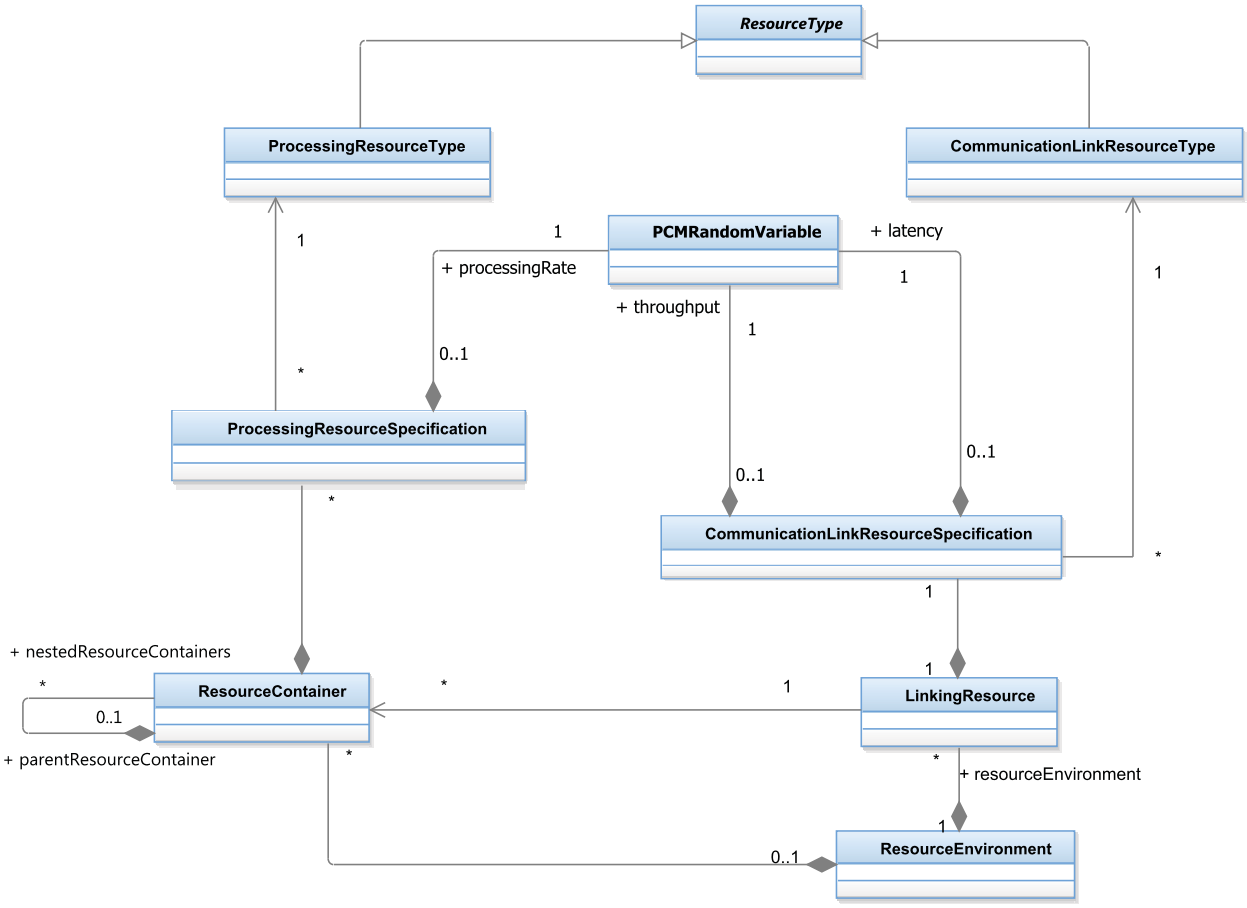

Figure 27 Palladio's Resource Environment Model (Reussner, et al., 2011)

Conversely, CACTOS' Physical Data Centre Model models the resource landscape of a data centre to a much finer degree than Palladio's *Resource Environment* Model. Figure 27 gives an overview of the Resource Environment Model of Palladio. Section IV.1 discusses the CACTOS Physical Data Centre Model in detail.

The first difference between the Resource Environment model of Palladio and the Physical Data Centre Model of CACTOS lies in the way they hierarchically structure the hardware resources. In the CACTOS model there is an explicit nesting of racks and nodes. Nodes are distinguished into Compute Nodes and Network Attached Storage nodes. Palladio only knows generic *ResourceContainers*. Resource Containers are used to model all hierarchies and nesting relationships in Palladio. Racks and both node types of the CACTOS model are mapped to these Resource Containers.

As of now, Palladio only distinguishes between *Processing Resources* and *Communication Link Resources*. In the development of CactoSim, it is planned to extend Palladio by specialised memory resources. This will enable the analysis of memory shortage effects. Processing Resources are throughput-oriented resources that are locally installed in a node. Examples for such resources are CPUs and HDDs. The properties of Palladio's Processing Resources are limited to a processing rate. Yet, all additional properties of processing units and storage of the

CACTOS model that have a meaningful influence on the simulation precision can be translated to the Palladio model.

Multi-core CPUs are translated into a set of separate Processing Resources hosted inside of a common Resource Container. For HDDs, the access delays are incorporated by subtracting their average impact on the throughput rates from the corresponding Processing Resource's throughput.

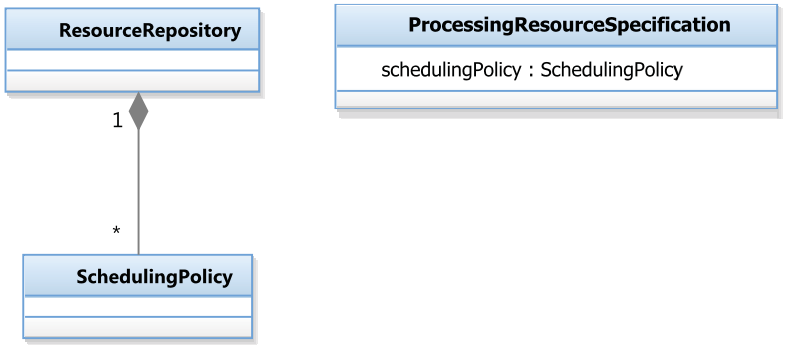

Figure 28 Specification of Resource Types in Palladio's Resource Type Model (The Palladio Component Model, 2011)

Figure 28 gives an overview on Palladio's *Resource Type Model*. The different architecture types of Processing Units can be mapped to *Scheduling Policies* that can be defined in Palladio's *Resource Type Model.* For each of these Scheduling Policies, an individual scheduler can be implemented as part of the simulation that resembles the specific processing behaviour of each architecture type.

The conversion from the modelling of network connections in the CACTOS and Palladio is carried out by joining together a set of NetworkInterconnects that is connected through Switches and aggregating it into one LinkingResource that connects multiple nodes and racks.

VIII. LICENSING

Table 16 Overview on Licenses of Tooling Used by the CACTOS tools

CACTOS tool	Used tools	Extended	License
CactoScale			
	Chukwa	Yes	Apache License, Version 2.0
	Hadoop	No	Apache License, Version 2.0
	Hbase	No	Apache License, Version 2.0
	Pig	No	Apache License, Version 2.0
	Mahout	No	Apache License, Version 2.0
	Apache CXF, Jersey, Felix, Karaf	No	Apache License, Version 2.0
	Equinox	No	Eclipse Public License, Version 1.0
CactoOpt			
	MySQL	No	GNU General Public License, Version 2.0
	Apache CXF, Jersey, Felix, Karaf	No	Apache License, Version 2.0
	Equinox	No	Eclipse Public License, Version 1.0
CactoSim			
	Palladio	Yes	Eclipse Public License, Version 1.0
	SimuLizar	Yes	Eclipse Public License, Version 1.0
CACTOS Toolkit Version 1			Eclipse Public License, Version 1.0
	Eclipse	Yes	Eclipse Public License, Version 1.0

All listed CACTOS tools and public demonstrators are released under the "Eclipse Public License, Version 1" (EPL v1). The only exception to this is background tools explicitly excluded as in the Description of Work and Consortium Agreements. EPL is an established open source license that sees practical use by many companies in a commercial context, especially for products built on top of the Eclipse platform. EPL allows redistribution of third-party code for allowing selecting widespread open source software to base on:

- Apache Software License 1.1,
- Apache Software License 2.0,
- W3C Software License,
- Common Public License 1.0,
- IBM Public License 1.0,
- Mozilla Public License 1.1,

- CDDL 1.0,
- BSD and
- MIT licenses.

Therefore EPL v1 is compatible to the tooling that is extended and used by the CACTOS toolkits. It is also ensured that central open source licenses are compatible with any results developed as part of the CACTOS project. The only exceptions to this are formed by the GNU General Public License 2.0, Lesser General Public License and the Sun Binary Code License Agreement, which are incompatible with EPL.

Some parts are directly provided back to Open Source communities with the same license used by them: The Infrastructure Models are the basis for the integration approach taken within the toolkit. These models reuse parts of the models developed in the Palladio project (Palladio - The Software Quality People, 2014). Palladio's meta-models are available under the EPL v1. In order to allow commercial users and developers as well as the open source community to utilise the Infrastructure Models as a basis for future modelling efforts, the Infrastructure Models are also released in the project SVN under the EPL license.

1. CACTOSCALE

CactoScale's tooling is released via the project SVN. CactoScale itself has been released under the EPL v1. The extended tool utilised by CactoScale is Chukwa. Chukwa's implementation has been extended by additional monitoring capabilities. With the exception of Chukwa, no other modifications to tool's source code have been made and the versions used by CactoScale are their publicly available releases. For those tools, their respective license, namely the Apache License Version 2 applies.

2. CACTOSIM

CactoSim will be made available under the EPL version 1.0. This grants users and distributors the following rights (Eclipse Public License - v 1.0):

- to copy, adapt and distribute the program in source or object code form
- to distribute the code in object code form alone under a different licence, provided that licence is compatible with the EPL
- patent rights from all contributors to use and make available the code
- to distribute works which contain the code in combination with new code modules, and to license the new code modules in any way the distributor wishes, if compatible with the EPL

It also allows for compatibility with the underlying tools that CactoSim builds upon, including Palladio-Bench (Palladio) and the SimuLizar simulation engine (SimuLizar) which are both also available under EPL version 1.0.

3. CACTOOPT

CactoOpt's core functionality including, algorithms, data models and utility tooling is licensed under the EPL v1. CactoOpt uses a MySQL database as a knowledge base employed for optimisations. It uses the binaries from the official MySQL release without any modifications to the underlying source code. The deployment using OSGi requires the listed runtime environments and bundles. Summarizing, no further restrictions on the code of the CactoOpt tool itself are imposed by the use of tools.

IX. REFERENCES

Becker, M., & Lehrig, S. (2014, July 11). *SimuLizar*. Retrieved September 3, 2014

Becker, M., Luckey, M., & Becker, S. (2013). Performance analysis of self-adaptive systems for requirements validation at design-time. *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures (QoSA '13)*.

Becker, S., Koziolek, H., & Reussner, R. (2009). The Palladio component model for model-driven performance prediction. *Journal of Systems and Software, 82*(1), 3-22.

CACTOS Consortium. (2014). *D3.1 Prototype Optimization Model.*

CACTOS Consortium. (2014). *D4.1 Data Collection Framework.*

CACTOS Consortium. (2014). *D4.2 Preliminary offline trace analysis.*

CACTOS Consortium. (2014). *D4.2 Preliminary offline trace analysis.*

CACTOS Consortium. (2014). *D5.2.1 CACTOS Toolkit Version 1.*

CACTOS Consortium. (2014). *D5.3 Operational Small Scale Cloud Testbed Managed by the CACTOS Toolkit.*

CACTOS Consortium. (2014). *D6.1 CactoSim Simulation Framework Initial Prototype.*

CACTOS Consortium. (2015). *D5.2.2 CACTOS Toolkit Version 2.*

Flexiant. (2014, August 28). *Flexiant Cloud Orchestrator Documentation - FDL Server.* Retrieved August 28, 2014, from http://docs.flexiant.com/display/DOCS/FDL+Server

Flexiant. (2014, August 26). *Flexiant Cloud Orchestrator Documentation - SOAP Documentation: Calling the SOAP Admin API and User API*. Retrieved September 3, 2014, from http://docs.flexiant.com/display/DOCS/Calling+the+SOAP+Admin+API+and+User+API#CallingtheSOAPAdminAPIandUserAPI-SOAPUser

Flexiant. (2014, August 28). *Flexiant Cloud Orchestrator Documentation - Triggers.* Retrieved August 28, 2014, from http://docs.flexiant.com/display/DOCS/Triggers

Flexiant. (2014). *Flexiant Developer Language System API Documentation.* Retrieved August 28, 20014, from http://docs.flexiant.com/display/DOCS/System+API

Flexiant. (2014, August 25). *SOAP Server*. Retrieved September 3, 2014, from http://docs.flexiant.com/display/DOCS/SOAP+server

JScience. (2006). *Amount Interface Documentation*. Retrieved 2014 йил 25-07 from JScience v4.3 JavaDoc: http://jscience.org/api/javax/measure/quantity/Quantity.html

JScience. (2006). *Quantity Interface Documentation*. Retrieved 2014 йил 25-07 from JScience v4.3 JavaDoc: http://jscience.org/api/javax/measure/quantity/Quantity.html

JScience. (2011, October 19). *JScience*. Retrieved September 5, 2014, from www.jscience.org

Palladio - The Software Quality People. (2014, September 2). *Palladio*. Retrieved September 3, 2014, from http://www.palladio-simulator.com/)

Reussner, R., Becker, S., Burger, E., Happe, J., Hauck, M., Koziolek, A., et al. (2011). *The Palladio Component Model.* Karlsruhe: Departments of Informatics, Karlsruhe Institute of Technology.

The Eclipse Foundation. (2014). *equinox - OSGi*. Retrieved September 1, 2014, from http://www.eclipse.org/equinox/

The Eclipse Foundation. (n.d.). *Eclipse Public License - v 1.0.* Retrieved September 1, 2014, from https://www.eclipse.org/legal/epl-v10.html