SPLASH INDIANAPOLIS 2013 OCTOBER 26-31

DSM Workshop 2013

SIGPLAN

Model-driven Performance Estimation, Deployment, and Resource Management for Cloud-hosted Services

Faruk Caglar, Kyoungho An, Shashank Shekhar, and Aniruddha Gokhale

Vanderbilt University, ISIS and EECS

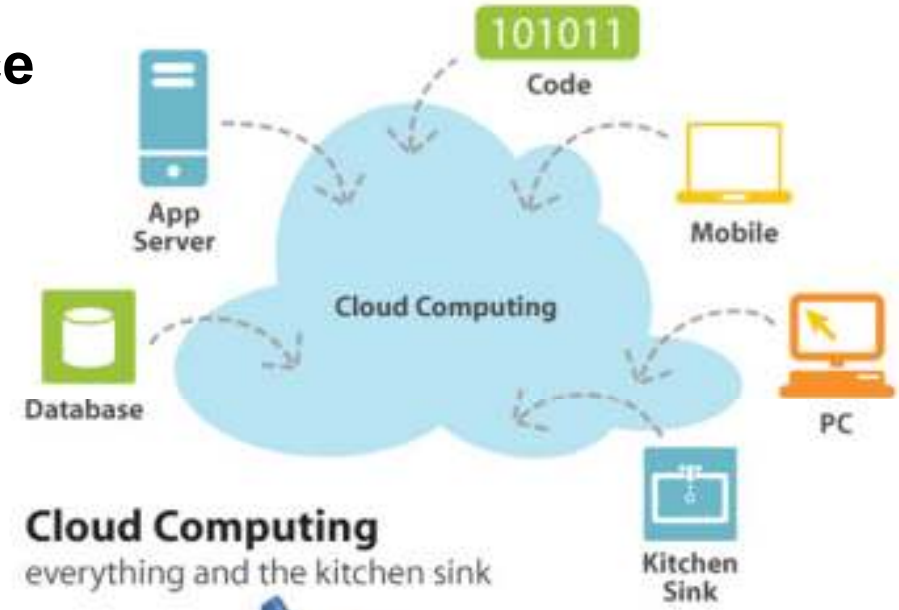
Nashville, TN, USA

VANDERBILT UNIVERSITY

INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

Context: Cloud Computing

- **Resources provided as service**
  - Resources on demand
  - "Pay-as-you-go" usage fee
  - Computing resources
    - CPUs, RAM
  - Networking resources
    - Bandwidth, network latency



- **Popular implementations**
  - Amazon Elastic Compute Cloud (EC2), Google App Engine, GoGrid, AppNexus, Emulab
  - OS, Database, RAM, CPU, Disk space, cores, load balancing, applications (e.g., Apache, Facebook servers), bandwidth, link latency

Motivation

- It is tedious and error-prone to migrate in-house applications into the cloud.

- Transitioning to the cloud should be conducted as seamlessly and easily as possible by knowing the cost and performance ahead of time

Challenges for Transitioning to Cloud

- ✓ Challenge 1: Performance and Cost Estimation?
  - What is the expected cost and performance delivered to the service?
- ✓ Challenge 2: Programming and Deployment Heterogeneity?
  - CSPs provide different APIs (e.g. Amazon EC2, GoGrid, Microsoft Azure) and different web-based user interfaces
  - Steep learning curve, API heterogeneity (DeltaCloud, libcloud, jcloud)
- ✓ Challenge 3: Resource Management?
  - Customers' responsibility
    - Determining virtual machine properties
    - Auto scaling as the demand changes
  - Must be programmed using the APIs

Our Solution Approach

Move-CAD : **M**odel-dri**ve**n Performance, **C**ost **A**nalyzer & Automated **D**eployment

Move-CAD is a two phase model-driven tool

1: Performance and Cost analysis

2: Automated Deployment

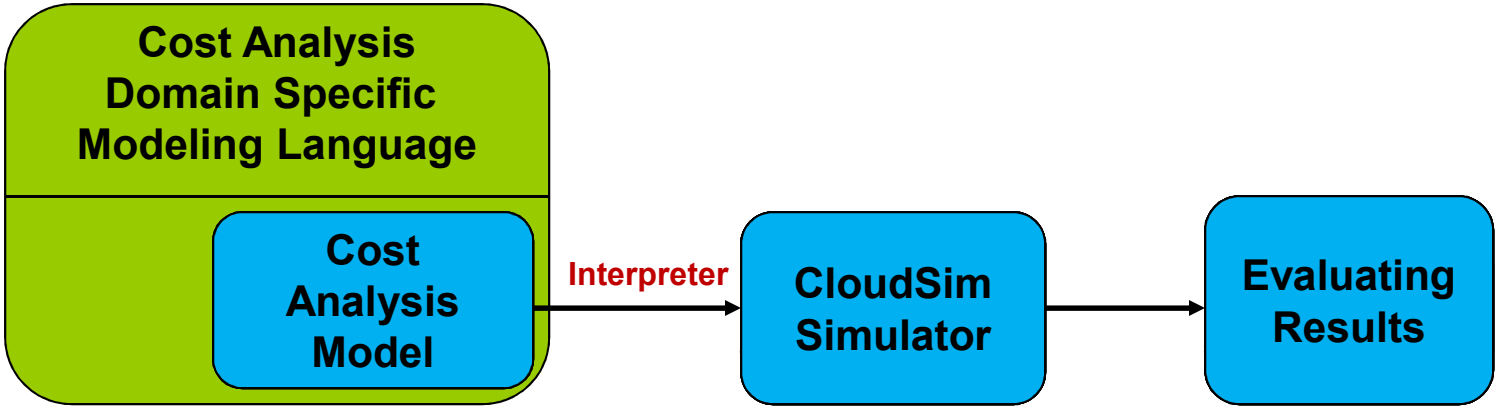
Move-CAD allows customers to analyze the cost and automatically deploying their applications on to the cloud

Tools & Platforms used in the Project

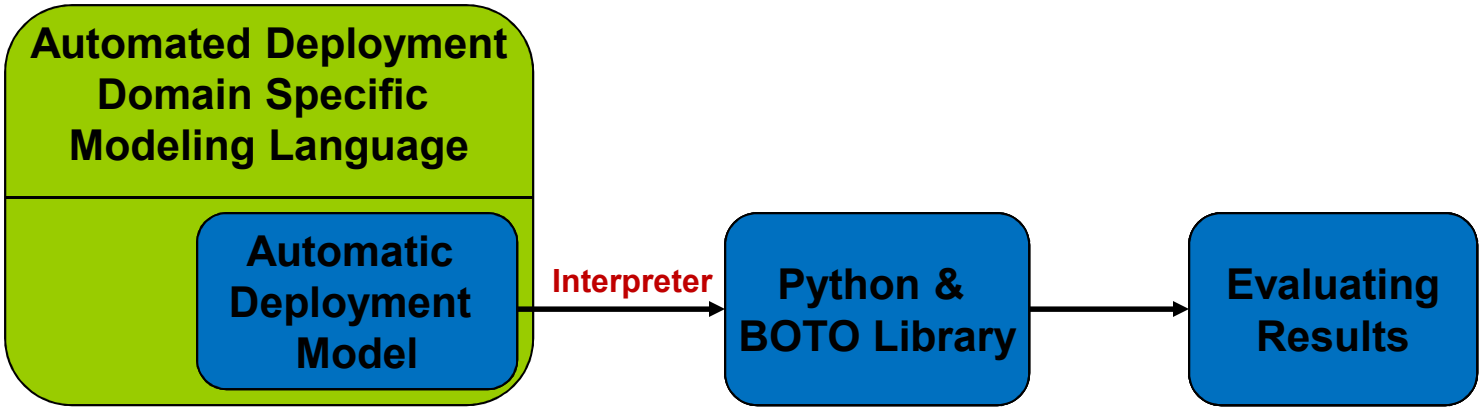
- Generic Modeling Environment (GME)
  - open-source, visual, configurable design environment for creating domain-specific modeling languages (DSMLs)
- CloudSim Simulator
  - a framework for modeling and simulation of cloud computing infrastructures and services
- Builder Object Network (BON) Interpreter w/C#
  - a framework which provides methods providing access to the objects' properties, relations, etc. in the GME Model
- Institute for Software Integrated Systems (ISIS) Cloud
  - Private cloud in ISIS

Move-CAD: System Overview

First Phase

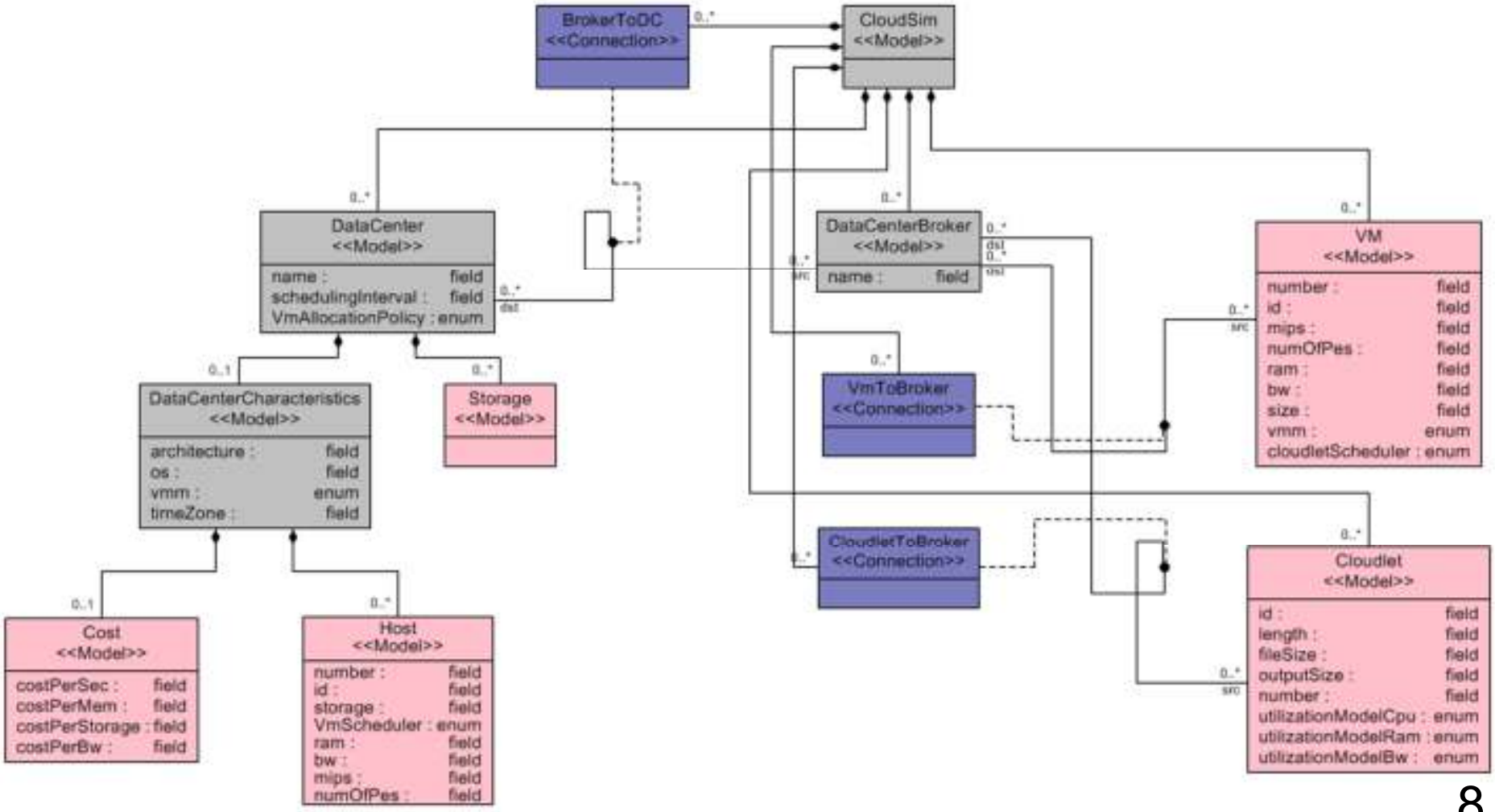


Second Phase



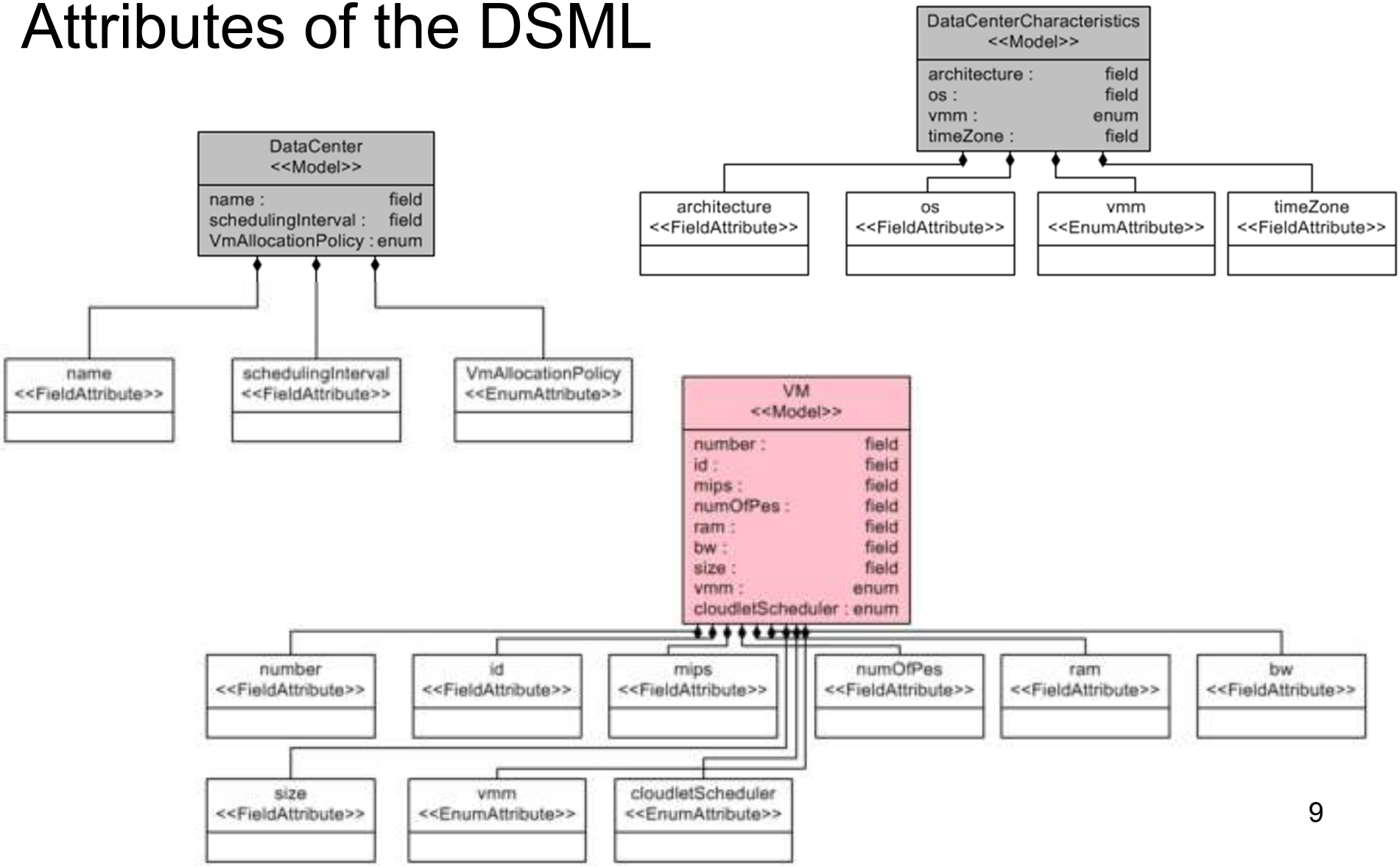
Cloud Simulation – Meta Model (1/3)

Meta model for estimating cloud-based service performance and cost



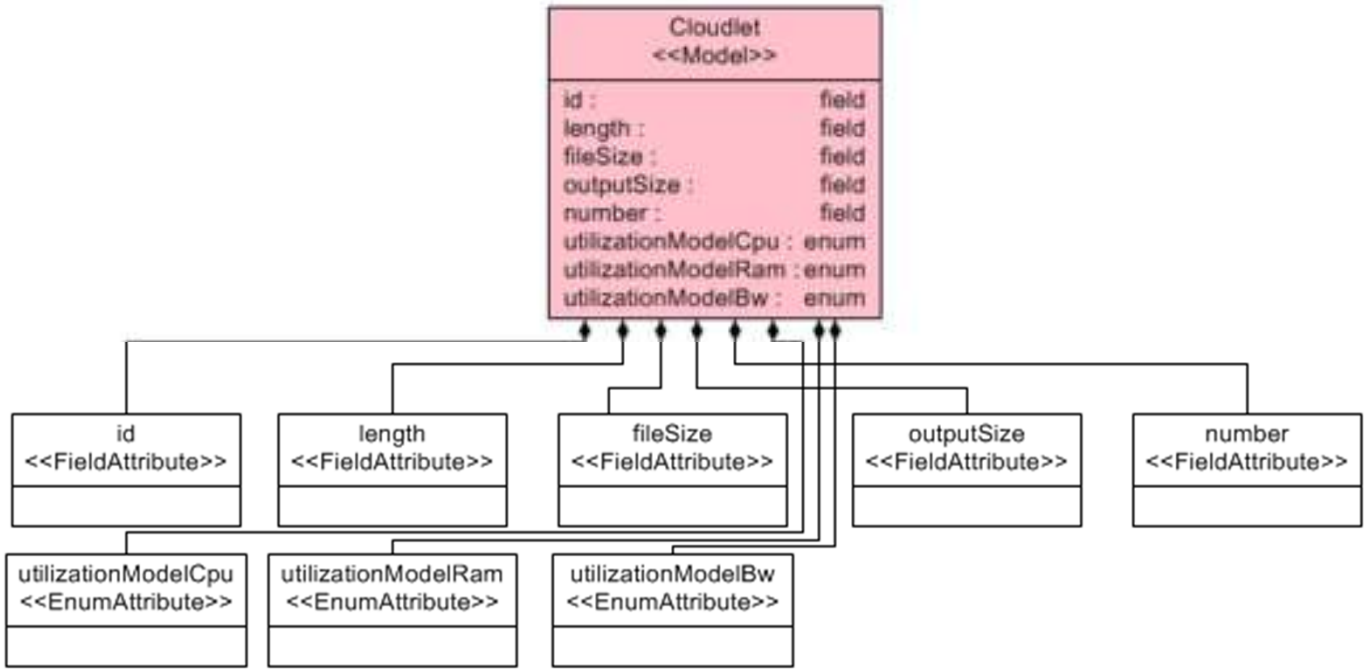
Cloud Simulation – Meta Model (2/3)

Attributes of the DSML

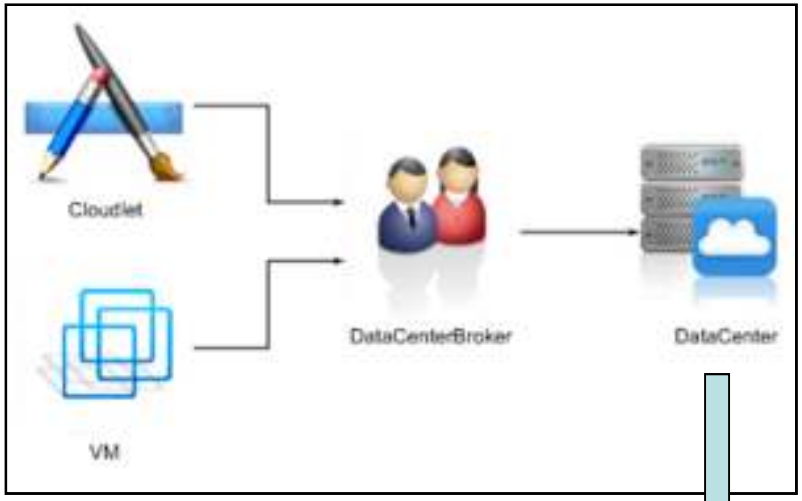


Cloud Simulation – Meta Model (3/3)

Attributes of the DSML



Cloud Simulation – Model (1/2)



FIRST: Based upon the cost and performance DSML, cloud simulation model is designed

SECOND: Attributes of each component is set including sub components

architecture	x86
os	Linux
vmm	Xen
timeZone	10

costPerSec	3
costPerMem	0.05
costPerStorage	0.1
costPerBw	0.1

number	1
id	0
storage	1000000
VmScheduler	SpaceShared
ram	2048
bw	1000
mips	1000
numOfPes	1

Cloud Simulation – Model (2/2)

Model Interpreter

```
public void Main(MgaProject project, MgaFCO currentobj, MgaFCOs
{
    GMEConsole.Out.WriteLine("Running interpreter...");

    // Get RootFolder
    IMgaFolder rootFolder = project.RootFolder;
    GMEConsole.Out.WriteLine(rootFolder.Name);

    ProcessCloudModel(rootFolder);

    StreamWriter sw = new StreamWriter("ModelBasedPricer.java");
    sw.WriteLine(this.createHeader());


    foreach (var item in this.cmdQueue)
    {
      sw.WriteLine(item.ToString());
    }

    sw.WriteLine(this.createFooter());
    sw.Flush();
    sw.Close();
}

public void ProcessCloudModel(IMgaFolder
{
```

(1)
Run Interpreter

(2)
Cloudsim
Java code
is generated
Automatically

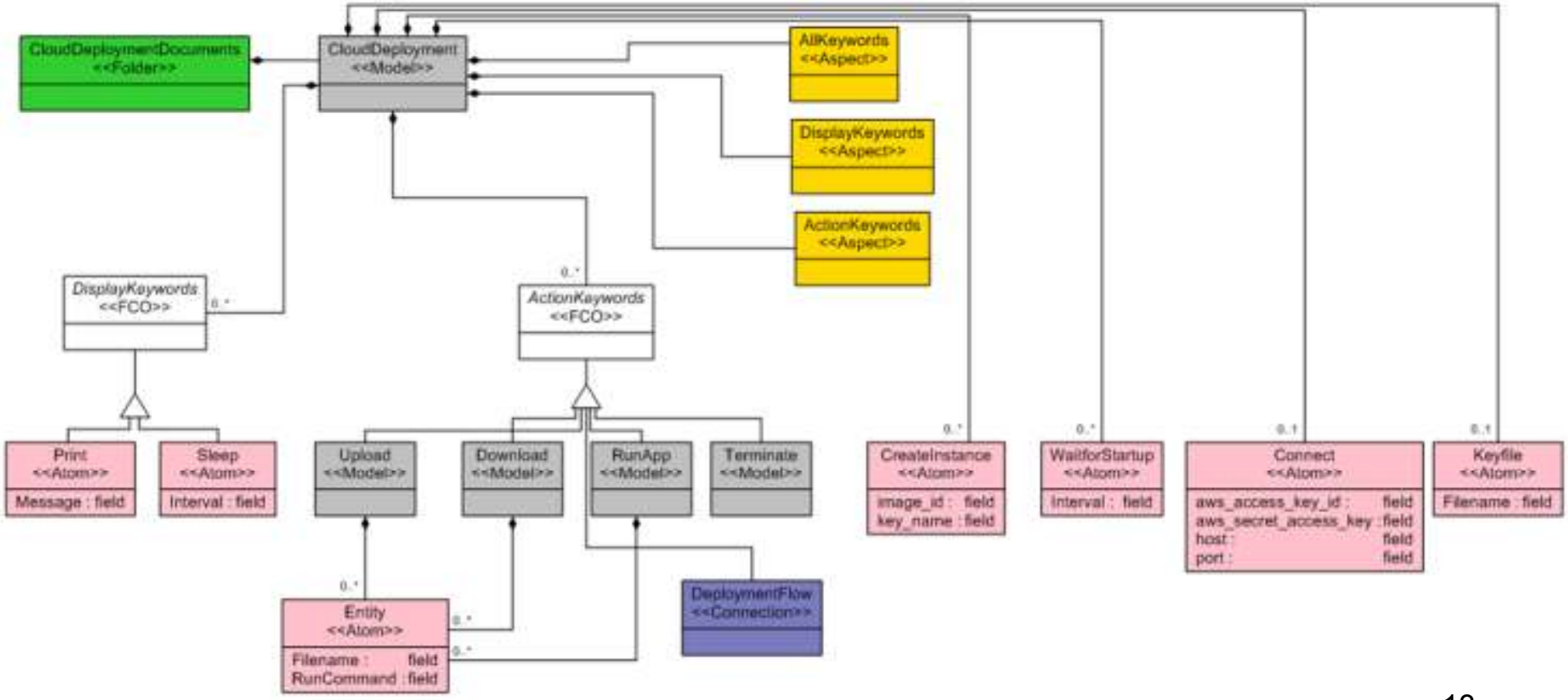
ModelBasedPricer.class
CLASS File
8.00 KB

(3)
Java code file is
Compiled
And
Run by
Cloudsim

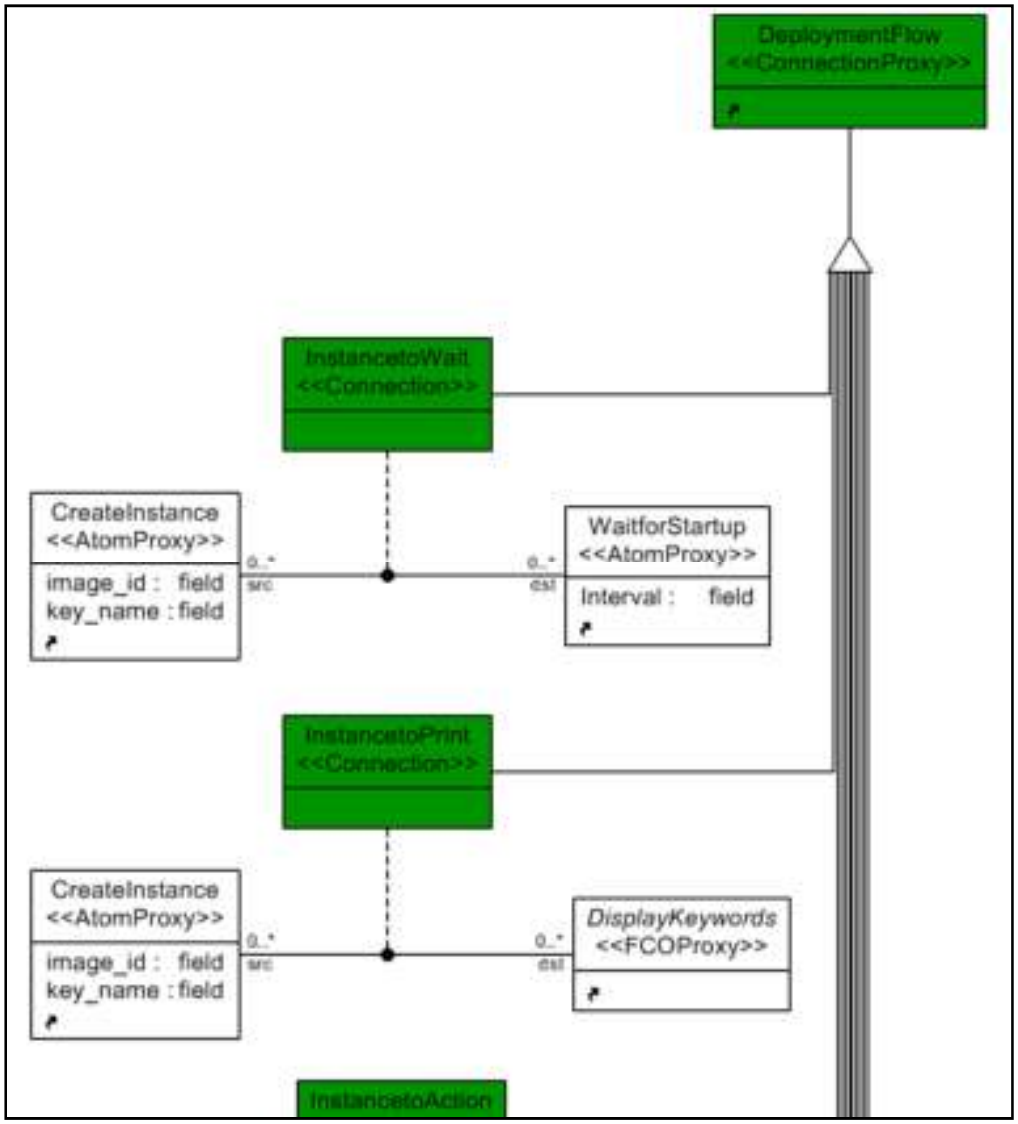
```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    0        SUCCESS        2            0            0        0.1           0.1
    1        SUCCESS        2            1            0        0.1           0.1
    2        SUCCESS        2            0            0        0.1           0.1
    3        SUCCESS        2            1            0        0.1           0.1
    4        SUCCESS        2            0            0        0.1           0.1
    5        SUCCESS        2            1            0        0.1           0.1
    6        SUCCESS        2            0            0        0.1           0.1
    7        SUCCESS        2            1            1        0.1           1.1
*****Datacenter: Datacenter_0*****
User id         Debt
3               2051.2
**********************************
```

Automated Deployment Meta Model (1/3)

The meta model of automated deployment in the cloud



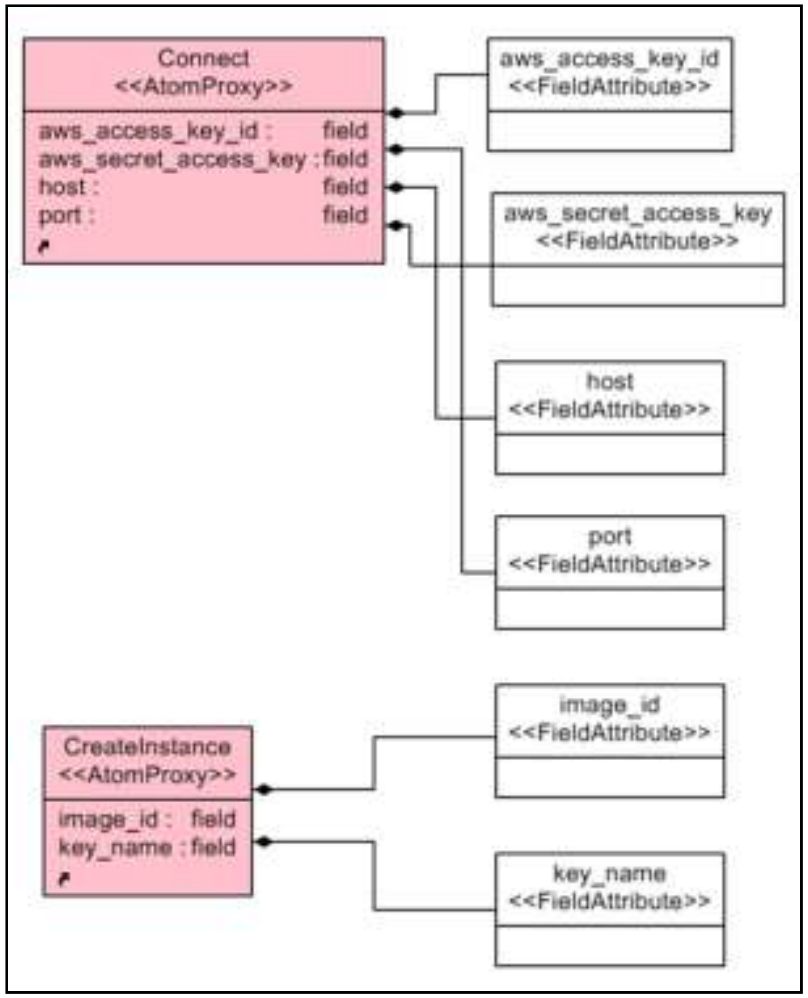
Automated Deployment Meta Model (2/3)

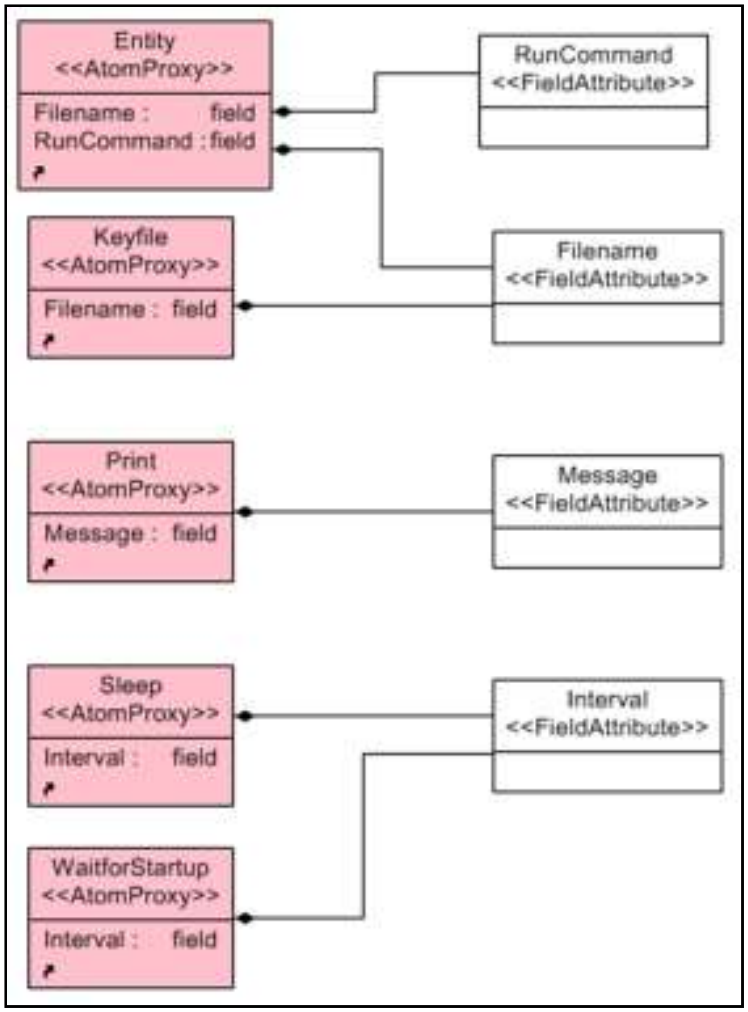


Connection structure of automated deployment in DSML

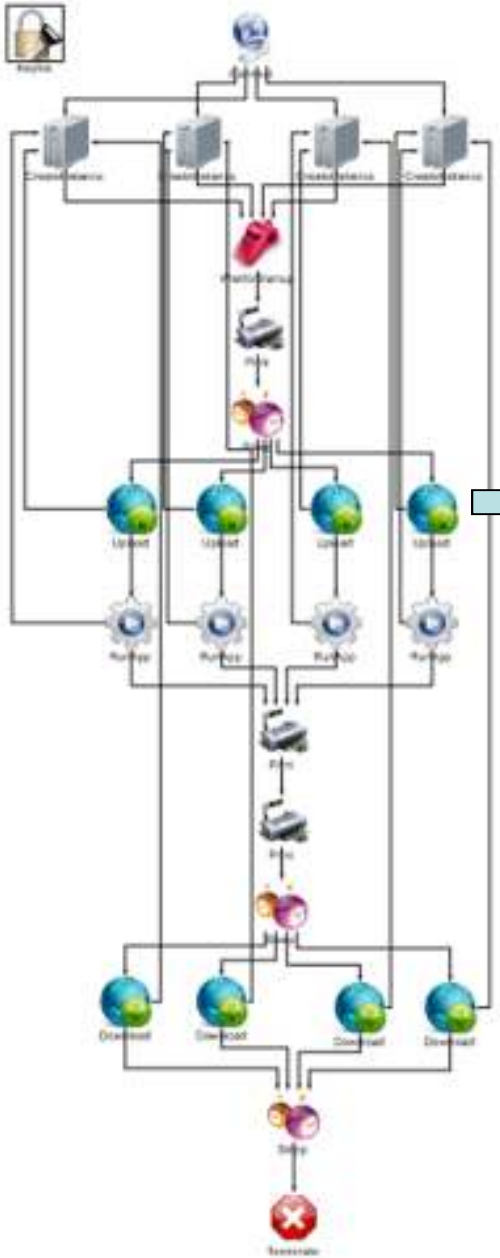
Automated Deployment Meta Model (3/3)

Attributes of the DSML





Automated Deployment Sample Model



(1)

```
public void Main(MgaProject project, MgaFCO currentobj, MgaF
{
    // Get RootFolder
    IMgaFolder rootFolder = project.RootFolder;
    ProcessCloudModel(rootFolder);

    StreamWriter sw = new StreamWriter("autodeploy.py");
    sw.WriteLine(this.CreateHeader());


    foreach (var item in this.cmdQueue)
    {
        CloudCommand cmd = item as CloudCommand;
        MgaSimpleConnection conn = cmd.Connection;
        MgaFCO src = cmd.Source;
        MgaFCO dst = cmd.Destination;
        if (src == null && conn == null)
        {
            GMEConsole.Warning.WriteLine("{0} -:- {1} -:- {2
        }
        else
        {
            GMEConsole.Warning.WriteLine("{0} -:- {1} -:- {2
        }

        switch (dst.Name)
```

(2)

(Deployment script is based on python language & boto library)

(1) After designing the deployment model

(2) Interpreter is run & Deployment script is generated.

Automated Deployment – Generated Script

```
#Copy applications to the instances created
def upload_file(hostip,fileName):
    try:
        print('Copying ' + fileName + ' over  ' + hostip)
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddP
        ssh.load_system_host_keys()
        ssh.connect(hostip, username='ubuntu',key_filenam
        ftp = ssh.open_sftp()
        ftp.put(os.path.expanduse
        ftp.close()
        ssh.close()
        print('Copying done...')
    except:
        print "Copying error: ",
```

```
def main():
    #Remove known hosts
    remove_known_hosts()


    #Connect to ISIS Cloud
    ec2Conn = EC2Connection(aws_access_key_id="c99468
                            aws_secret_access_key
                            is_secure=False, \
                            host="cloud.isis.vand
                            port=8773, \
                            region=RegionInfo(con
                            name="nova", \
                            endpoint="cloud.isis.
                            path="/services/Cloud

    print('Successfully connected to ISIS Cloud.')
```

Only showing the upload_file function and small part of the main code block generated by the interpreter

Concluding Remarks

- Presented an ongoing work on model-based simulation and model-based automated deployment in cloud
- User is completely shielded from having to learn the simulator since the MDE tool generates scripts to run the simulator.
- MDE tooling enables the user to model the deployment of their services in the cloud
- Tested our approach on a representative application scenario and deployed it in our in-house, experimental cloud environment

Future Work

- Include additional cloud service-provider APIs (e.g. GoGrid, OpenNebula, and Rackspace) in the deployment model

- Only one DSML for performance, cost analysis and automated deployment

- Directly executing the script against the APIs and having recovery mechanisms if the deployment fails at certain point.

Questions & Comments

Thank You!