# Elastic Computing and Engineering Elastic Applications in the Cloud

*Summer SOC, Crete, 2 July 2013*

Schahram Dustdar and Hong-Linh Truong

Distributed Systems Group
TU Vienna

**http://dsg.tuwien.ac.at/research/viecom/**

# Acknowledgements

Includes some joint work with Kamal Bhattacharya, Muhammad Z.C. Candra, Georgiana Copil, Daniel Moldovan, Mirela Riveri, Ognjen Scekic



NOTE: The content includes some ongoing work

# Outline

- Part 1: Elastic Computing
  - Motivation for multi-dimensional elasticity
  - Quality/cost/benefits analytics
  - HBS cloud concepts
  - Conclusions
- Part 2: Engineering Elastic Applications in the Cloud
  - Programming hybrid services for solving (in)dependent tasks
  - Programming incentives
  - Controlling and monitoring elasticity
  - Conclusions
- Part 3: Demonstration of elasticity control and monitoring

# PART 2 – ENGINERING ELASTIC APPLICATIONS

# Engineering Elastic Applications in the Cloud – using hybrid service units for dependent tasks

Distributed Systems Group

# HBS Instances Provisioning

- Types of services :
  - Individual Compute Unit (ICU)
  - Social Compute Unit (SCU

Individual Compute Unit instances (iICU): iICU describe instances of HBS built atop capabilities of individuals. An individual can provide different iICU. Analogous to SBS, an iICU is similar to an instance of a virtual machine or a software.

Social Compute Unit instances (iSCU): iSCU describe instances of HBS built atop capabilities of multiple individuals and SBS. Analogous to SBS, an iSCU is similar to a virtual cluster of machines or a complex set of software services.

# HBS Instances Provisioning

**Instances Descriptions**

- iICU(CS, HPU, archtype, price, incentive, utilization,location, APIs)
- iSCU(CS,HPU, archtype, price, incentive, utilization,connectedness, location, APIs)
- Other (traditional) NFPs

**Pricing factors**

- utilization
- offering communication APIs
- connectedness

**Incentive factors**

- Based on utilization and types of tasks
- Declared by ICU/SCU
- Enforced by HBS cloud providers

# ICU/SCU Archetypes

An „archetype“ characterizes the problem domain that the ICU/SCU can solve (the type of solutions)

```
Archtype ={
      ({WebDataAnalytics,TwitterAnalytics}, DataAnalytics),
      ({DataCleansing,DataEnrichment},DataQualityImprovement)
}

Types of solutions:
            WebDataAnalytics, TwitterAnalytics, DataCleansing, DataEnrichment

Problem domains:
            DataAnalytics and DataQualityImprovement
```

# Cloud APIs for Provisioning Hybrid Services

APIs hide low-level platforms and utilize low level HBS communication interfaces

## APIs for HBS information and management

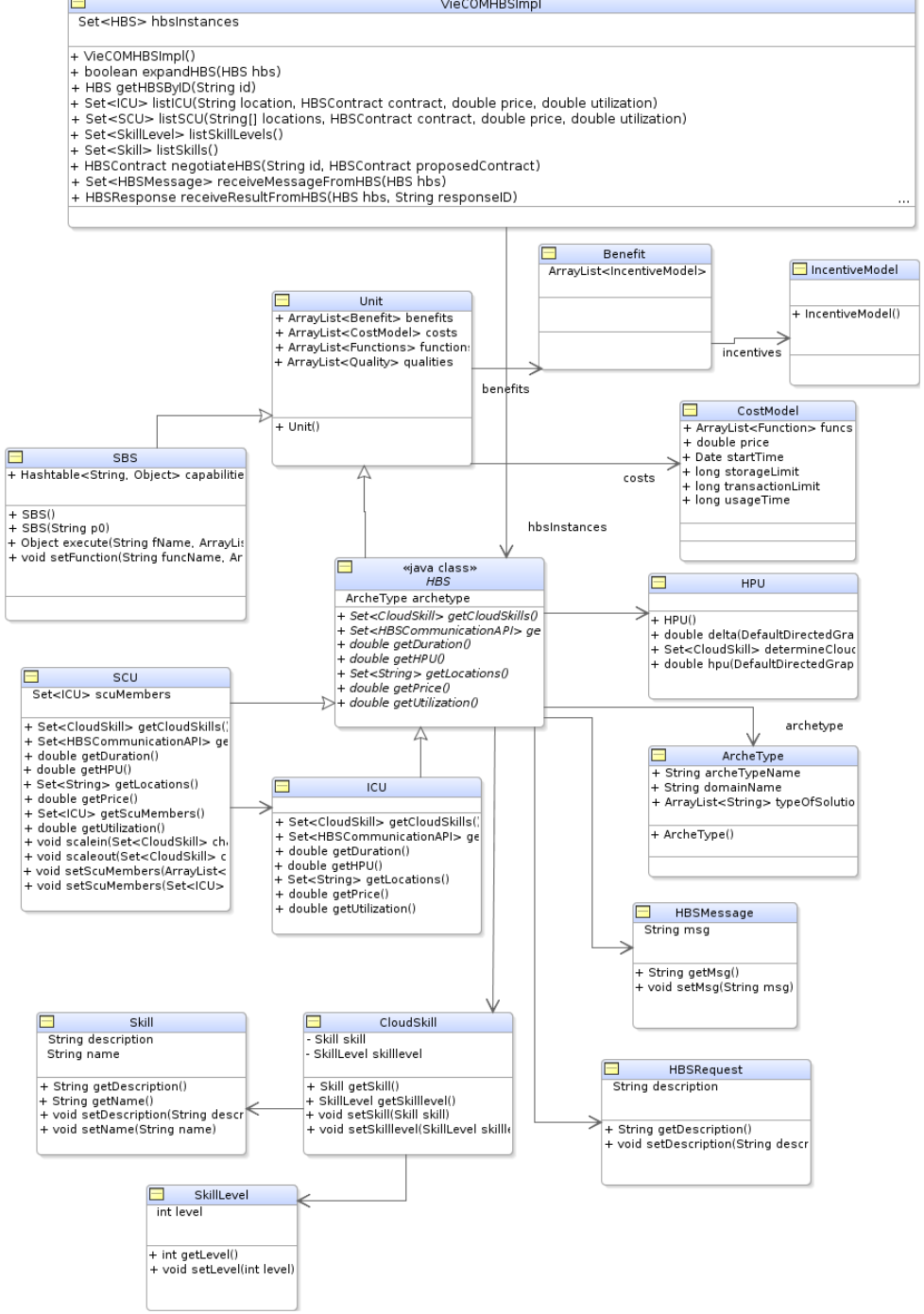
- listSkills();listSkillLevels();
- listICU();listSCU()
- negotiateHBS()
- startHBS()
- suspendHBS ()
- resumeHBS ()
- stopHBS()
- reduceHBS()
- expandHBS()

## APIs for HBS execution and communication

- runRequestOnHBS ()
- receiveResultFromHBS()
- sendMessageToHBS()
- receiveMessageFromHBS()

# Prototype (simulated environment)

Combined with Jcloud/boto for real SBS

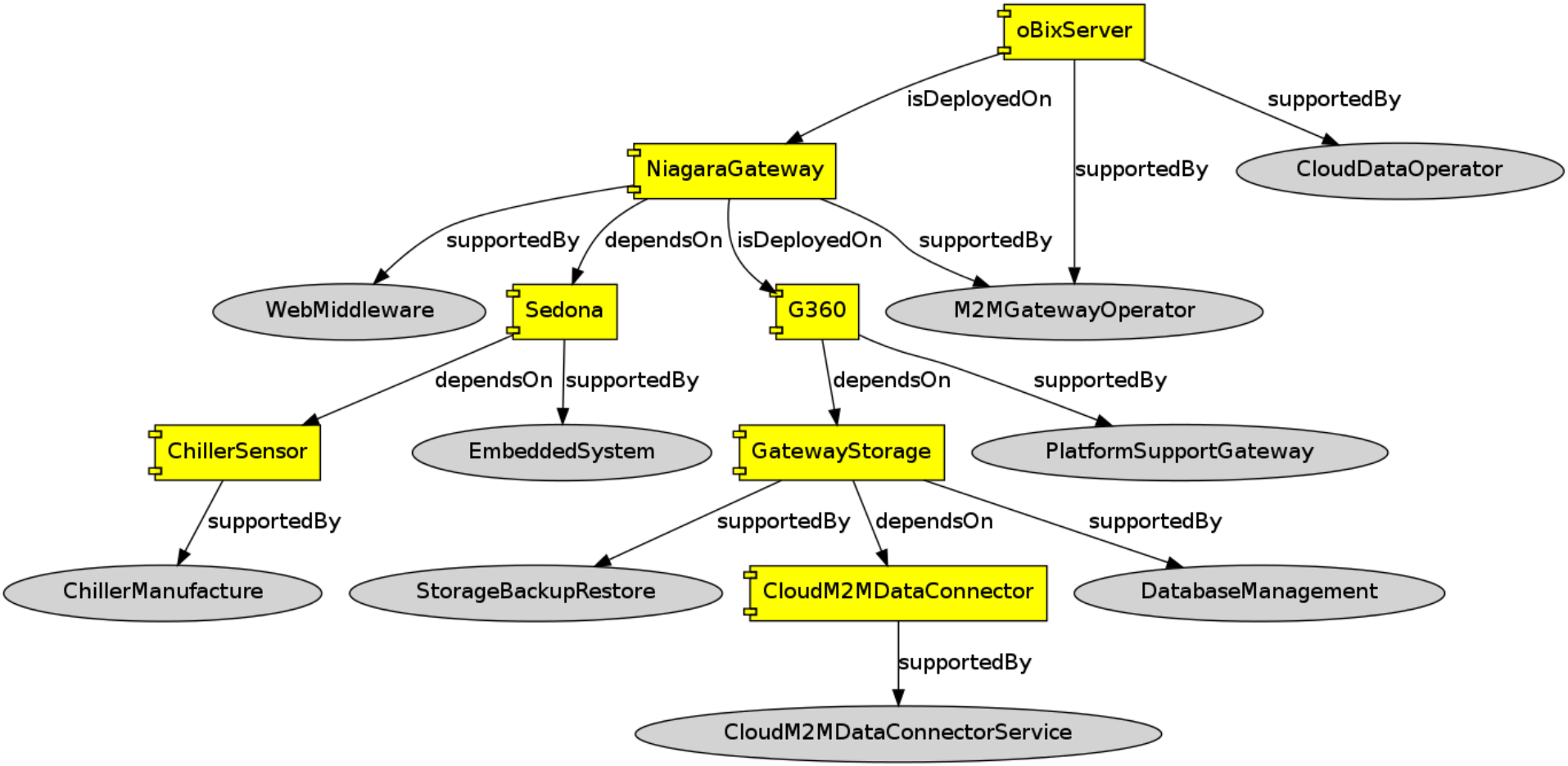


# Dependent/evoling tasks – example

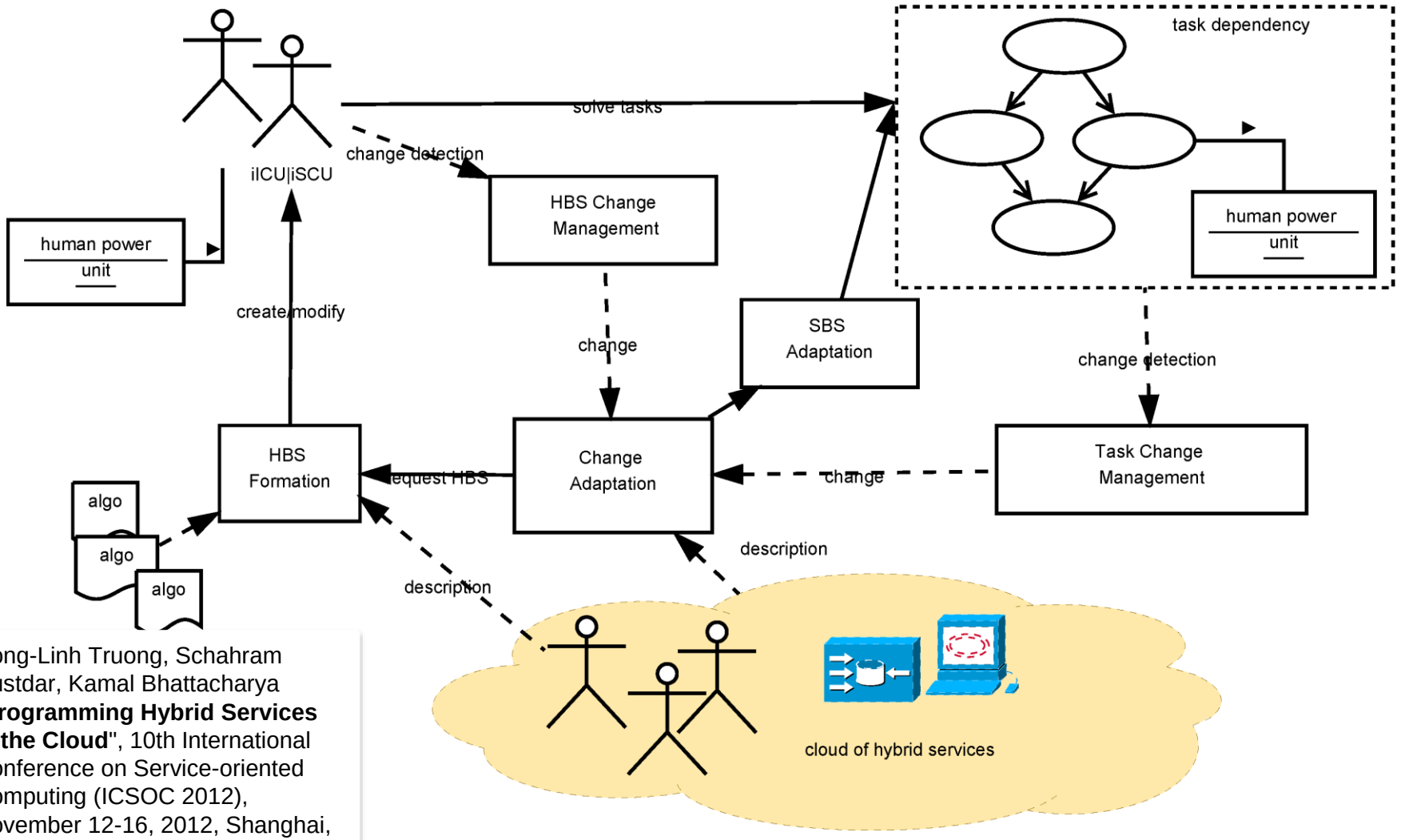
- Some problems happen in a M2M gateway in a building
  - Network problem?
  - Storage problem?
  - Something wrong in the interface to chillers?
  - M2M cloud connector  problem?
- What happens if we repair the gateway?

# Modeling HPU-aware task dependency graphs

*Link management skill constraints to tasks required HBS*



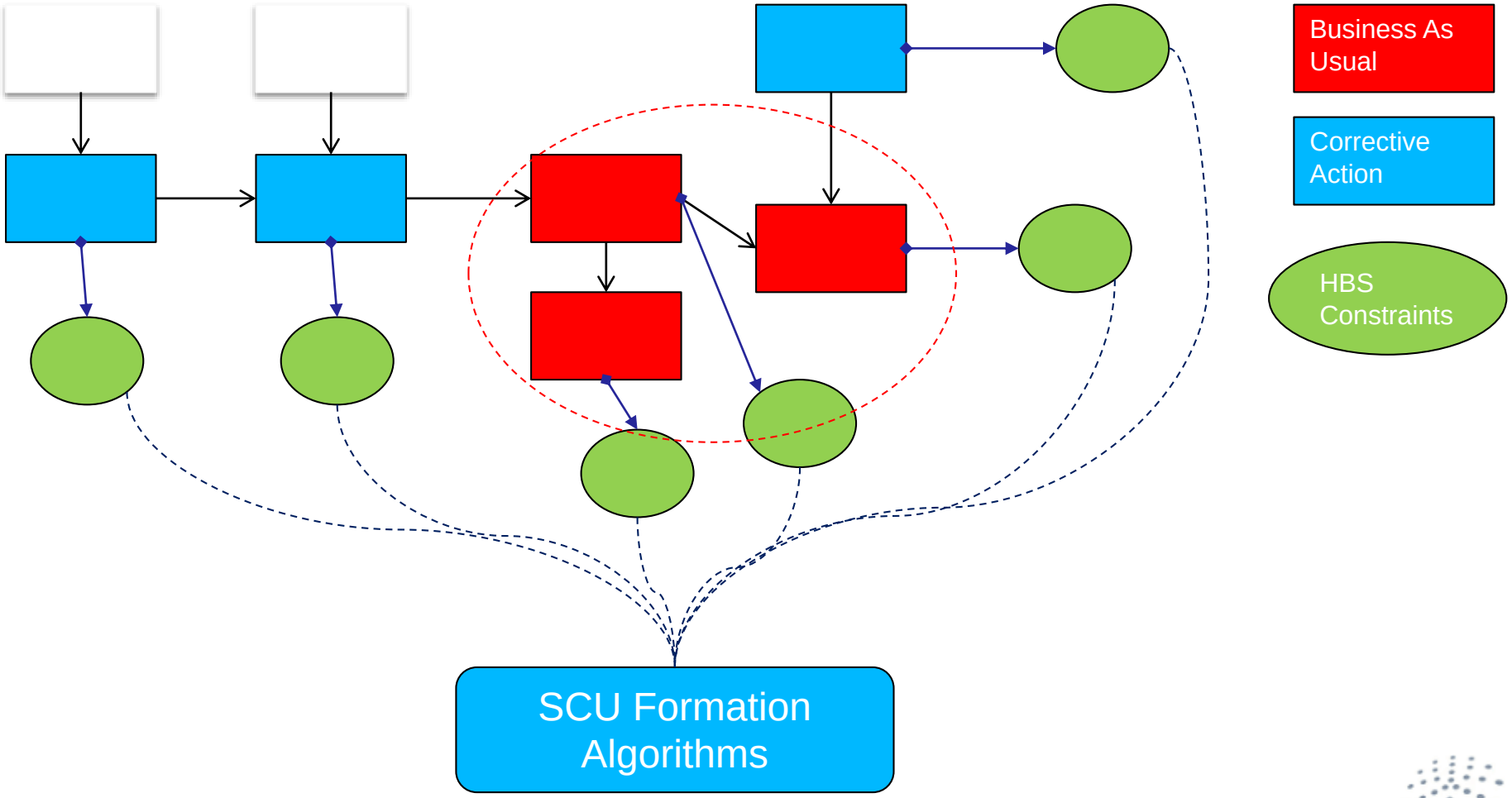
# Utilizing hybrid services for evolving/dependent task graphs




Hong-Linh Truong, Schahram Dustdar, Kamal Bhattacharya "**Programming Hybrid Services in the Cloud**", 10th International Conference on Service-oriented Computing (ICSOC 2012), November 12-16, 2012, Shanghai, China. Best Paper Award.

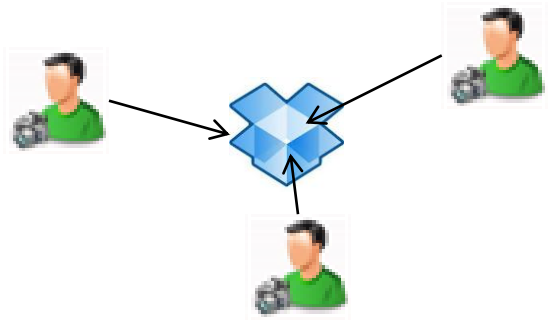
# Forming iSCUs

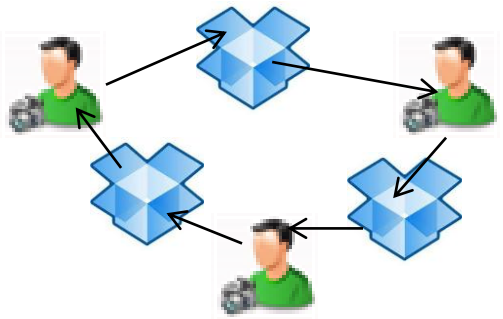
- Done by consumers or cloud providers



# Configuring iSCU

- Establish „connectedness“ based on compliance constraints and network topology
  - Addional cost might occur!
- Program SBS and HBS for the iSCU to have a complete working environment.
- Different connectedness
  - E.g., ring-based, star-based, and master-slave topologies





# Selecting HBS: Some algorithms

| Algorithms | Description |
|---|---|
| SkillWithNPath | Select $iICU$ for $iSCU$ based on only skills with a pre-defined network path length starting from the task to be solved. |
| SkillMinCostWithNPath | Select $iICU$ for $iSCU$ based on only skills with minimum cost, considering a pre-defined network path length starting from the task to be solved. |
| SkillMinCostMaxLevelWithNPath | Select $iICU$ for $iSCU$ based on skills with minimum cost and maximum skill levels, considering a pre-defined network path length starting from the task to be solved. |
| SkillWithNPathUnDirected | Similar to $SkillWithNPath$ but considering undirected dependency |
| MinCostWithNPathUnDirected | Similar to $MinCostWithNPath$ but considering undirected dependency |
| MinCostWithAvailNPathUnDirected | Select Select $iICU$ for $iSCU$ based on skills with minimum cost, considering availability and a pre-defined network path length starting from the task to be solved. Undirected dependencies are considered. |

- Several algorithms can be built based on existing team formation algorithms which do not consider dependency graphs
- Different weighted factors can be considered

# Forming iSCU by minimizing cost and considering no direction

```
DefaultDirectedGraph<Node, Relationship> dg; //graph of problems
//...
double hpu = HPU.hpu(dg); //determine
SCUFormation app = new SCUFormation( dg);
ManagementRequest request = new ManagementRequest();
//define request specifying only main problems to be solved
//....
//call algorithms to find suitable HBS.  Path length =2 and
    availability from 4am to 19pm in GMT zone
ResourcePool scu = app.
    MinCostWithAvailabilityNPathUnDirectedFormation(request, 2,
    4, 19);
if (scu == null) { return ; }
ArrayList<HumanResource> scuMembers = scu.getResources();
SCU iSCU = new SCU();
iSCU.setScuMembers(scuMembers);
//setting up SBS for scuMember ...
```

# Example of star-based iSCU using Dropbox as a communication hub

```
SCU iSCU ;
//... find members for SCU
DropboxAPI<WebAuthSession> scuDropbox; //using dropbox apis
//...
AppKeyPair appKeys = new AppKeyPair(APP_KEY, APP_SECRET);
WebAuthSession session =
        new WebAuthSession(appKeys, WebAuthSession.AccessType.
            DROPBOX);
//...
session.setAccessTokenPair(accessToken);
scuDropbox = new DropboxAPI<WebAuthSession>(session);
//sharing the dropbox directory to all scu members
//first create a share
DropboxAPI.DropboxLink link = scuDropbox.share("/hbscloud");
//then send the link to all members
VieCOMHBS vieCOMHBS = new VieCOMHBSImpl();
for (HBS hbs : iSCU.getScuMembers()) {
    vieCOMHBS.startHBS(icu);
    HBSMessage msg = new HBSMessage();
    msg.setMsg("pls. use shared Dropbox for communication " +
        link.url);
    vieCOMHBS.sendMessageToHBS(hbs, msg);
}
//...
```

# Programming a combination of HBS and SBS

e.g., preparing/managing inputs/outputs for HBS using SBS

```
//using JClouds APIs to store log file of web application server
BlobStoreContext context =
    new BlobStoreContextFactory().createContext("aws-s3","REMOVED
        ","REMOVED");
BlobStore blobStore = context.getBlobStore();
//.... and add file into Amazon S3
Blob blob = blobStore.blobBuilder("hbstest").build();
blob.setPayload(new File("was.log"));
blobStore.putBlob("hbstest", blob);
String uri = blob.getMetadata().getPublicUri().toString();
VieCOMHBS vieCOMHBS = new VieCOMHBSImpl();
//assume that WM6 is the HBS that can analyze the Web Middleware
     problem
vieCOMHBS.startHBS("WM6");
HBSRequest request = new HBSRequest();
request.setDescription("Find possible problems from " + uri);
vieCOMHBS.runRequestOnHBS("WM6", request);
```

# Change model for task graph's Human Power Unit

```
SCU iSCU ;
//...
iSCU.setScuMembers(scuMembers);
//setting up SBS for scuMember
//...
double hpu = HPU.hpu(dg); //determine current hpu
//SCU solves/adds tasks in DG
//  ....
//graph change - elasticity based on human power unit
double dHPU = HPU.delta(dg,hpu);
DefaultDirectedGraph<Node, Relationship> changegraph;
//obtain changes
Set<CloudSkill> changeCS = HPU.determineCloudSkill(changegraph);
if (dHPU > SCALEOUT_LIMIT) {
   iSCU.scaleout(changeCS); //expand iSCU
}
 else if (dHPU < SCALEIN_LIMIT) {
   iSCU.scalein(changeCS); //reduce iSCU
}
//  ...
```

# Engineering Elastic Applications in the Cloud – using HBS for independent tasks

# Independent tasks

- Requests that can be serialized into a sequence of independent tasks
  - Tasks can still be ressigned/delegated among service units

Examples: urban planning support in smart city management
Different influences on SCU formations and operations

- Techniques
  - Using Elastic Profile to specify constructs that can be used to model trade-offs and the dynamic provisioning of resources
  - Expanding/reducing SCUs using elastic profile, performance, trust, etc.

# Elastic profile for human-based services

⟨*ep*⟩ ::= `profile` ⟨*identifier*⟩ { ⟨*statement*⟩* }

⟨*statement*⟩ ::= ⟨*objects_statement*⟩
| ⟨*metrics_statement*⟩
| ⟨*activities_statement*⟩
| ⟨*behavior_statement*⟩

⟨*objects_statement*⟩ ::= `objects` { ⟨*objects_list*⟩ } ;

⟨*objects_list*⟩ ::= ⟨*object_identifier*⟩
| ⟨*object_identifier*⟩ , ⟨*objects_list*⟩

⟨*metrics_statement*⟩ ::= `metrics` { ⟨*metrics_list*⟩ } ;

⟨*metrics_list*⟩ ::= ⟨*metric*⟩
| ⟨*metric*⟩ ; ⟨*metrics_list*⟩

⟨*metric*⟩ ::= ⟨*object_identifier*⟩ `has` ⟨*metric_method_list*⟩

⟨*metric_method_list*⟩ ::= ⟨*metric_method*⟩[ (⟨*value*⟩) ]
| ⟨*metric_method*⟩[ (⟨*value*⟩) ] , ⟨*metric_method_list*⟩

⟨*behavior_statement*⟩ ::= `behavior` { ⟨*implication_list*⟩ } ;

⟨*implication_list*⟩ ::= ⟨*implication*⟩
| ⟨*implication*⟩ ; ⟨*implication_list*⟩

⟨*implication*⟩ ::= `check` [:⟨*priority*⟩] (⟨*condition*⟩) { ⟨*consequences*⟩ }

⟨*consequences*⟩ ::= ⟨*consequence*⟩ | ⟨*consequence*⟩; ⟨*consequences*⟩

⟨*consequence*⟩ ::= ⟨*metric_identifier*⟩ = ⟨*value*⟩
| `assert` ⟨*instance_identifier*⟩
| `trigger` ⟨*action_identifier*⟩(⟨*value_list*⟩)
| `throw` ⟨*exception_identifier*⟩ (⟨*value*⟩)

⟨*activities_statement*⟩ ::= `activities` { ⟨*activities_list*⟩ } ;

⟨*activities_list*⟩ ::= ⟨*activity*⟩
| ⟨*activity*⟩, ⟨*activities_list*⟩

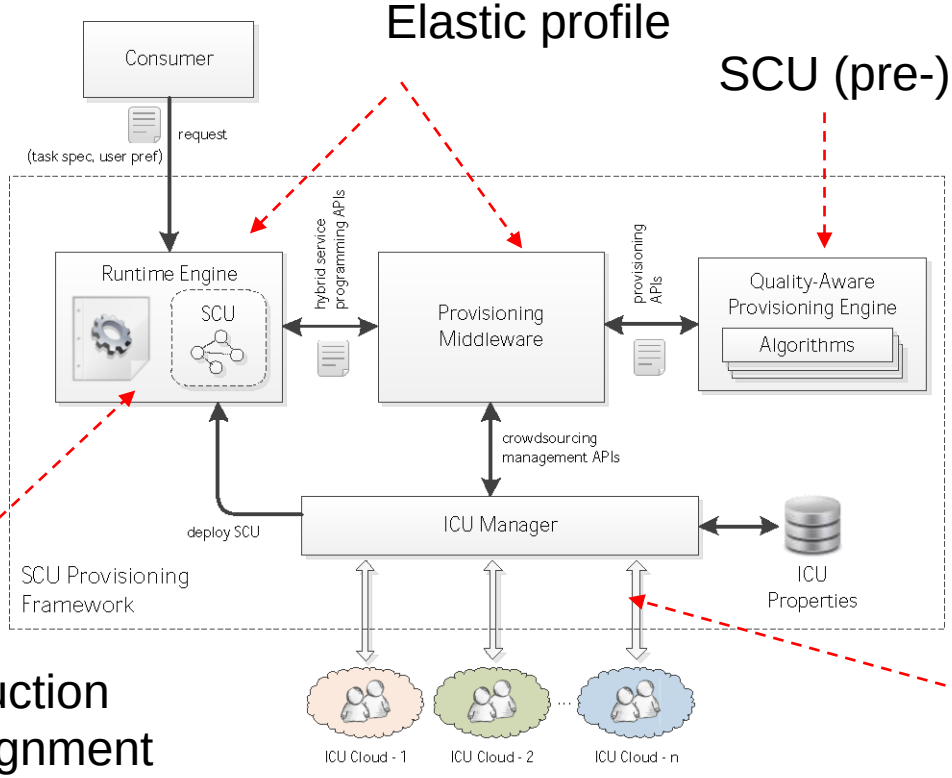
⟨*activity*⟩ ::= ⟨*activity_identifier*⟩ (⟨*activity_param_list*⟩)

Muhammad Z.C. Candra, Hong-Linh Truong, and Schahram Dustdar, "Modelling Elasticity Trade-offs in Adaptive Mixed Systems", 11th Adaptive Computing (and Agents) for Enhanced Collaboration (ACEC) Conference Track @ IEEE WETICE 2013, Hammamet, Tunisia, 17-20, June, 2013.

# Elastic SCU provisioning atop ICUs

Elastic profile

SCU (pre-)runtime/static formation




Algorithms

- Ant Colony Optimization variants
- FCFS
- Greedy

SCU extension/reduction

- Task reassignment based on trust, cost, availability

HBS cloud APIs

Mirela Riveni, Hong-Linh Truong, and Schahram Dustdar, **A Feedback Based Approach for Elasticity Coordination of Social Compute Units, June 2013, On submission**

Muhammad Z.C. Candra, Hong-Linh Truong, and Schahram Dustdar, **Provisioning Quality-aware Social Compute Units in the Cloud, June 2013, On submission**

# Engineering Elastic Applications in the Cloud – Incentive programming

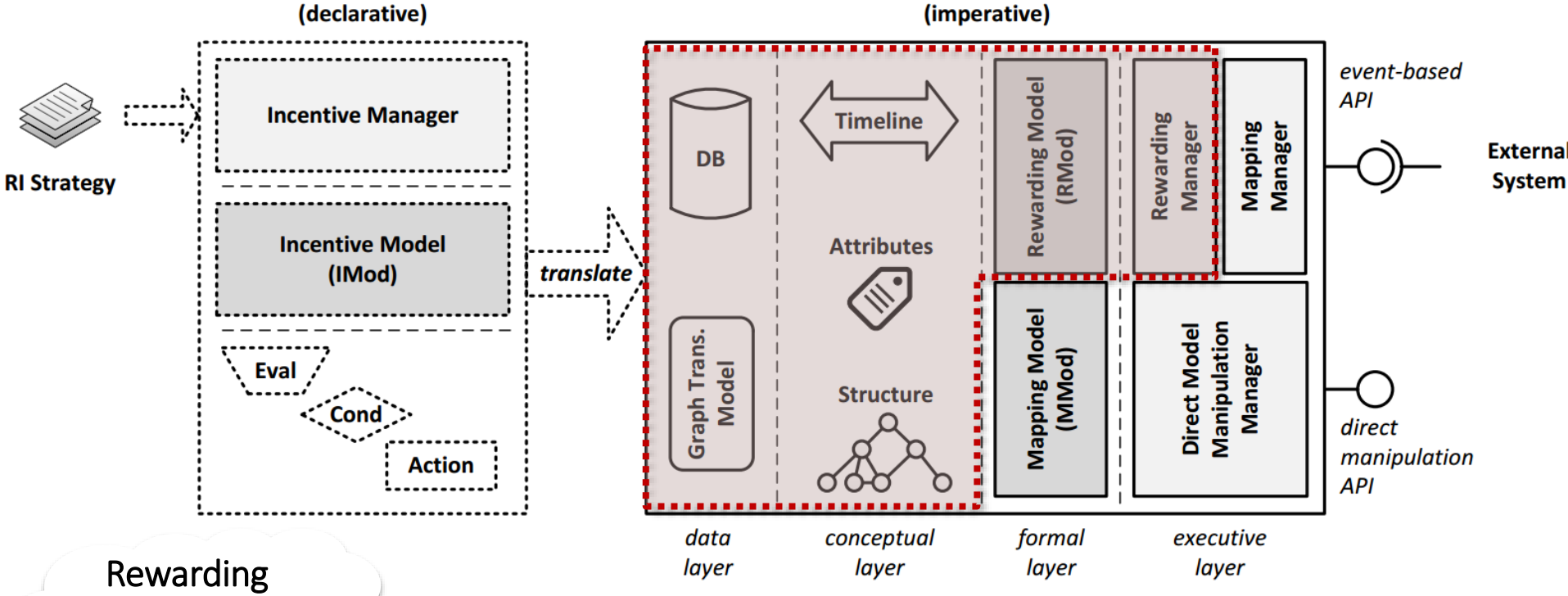
# Programming and executing Incentives

domain expert

Ognjen Scekic, Hong Linh Truong, Schahram Dustdar: Modeling Rewards and Incentive Mechanisms for Social BPM. BPM 2012: 150-155

Ognjen Scekic, Hong-Linh Truong, Schahram Dustdar, "Programming Incentives in Information Systems", 25th International Conference on Advanced Information Systems Engineering(CAiSE'13), Springer-Verlag, Valencia, Spain, 17-21 June, 2013.

# PRogrammable INCentives Framework (PRINC)




Rewarding Model (RMod)

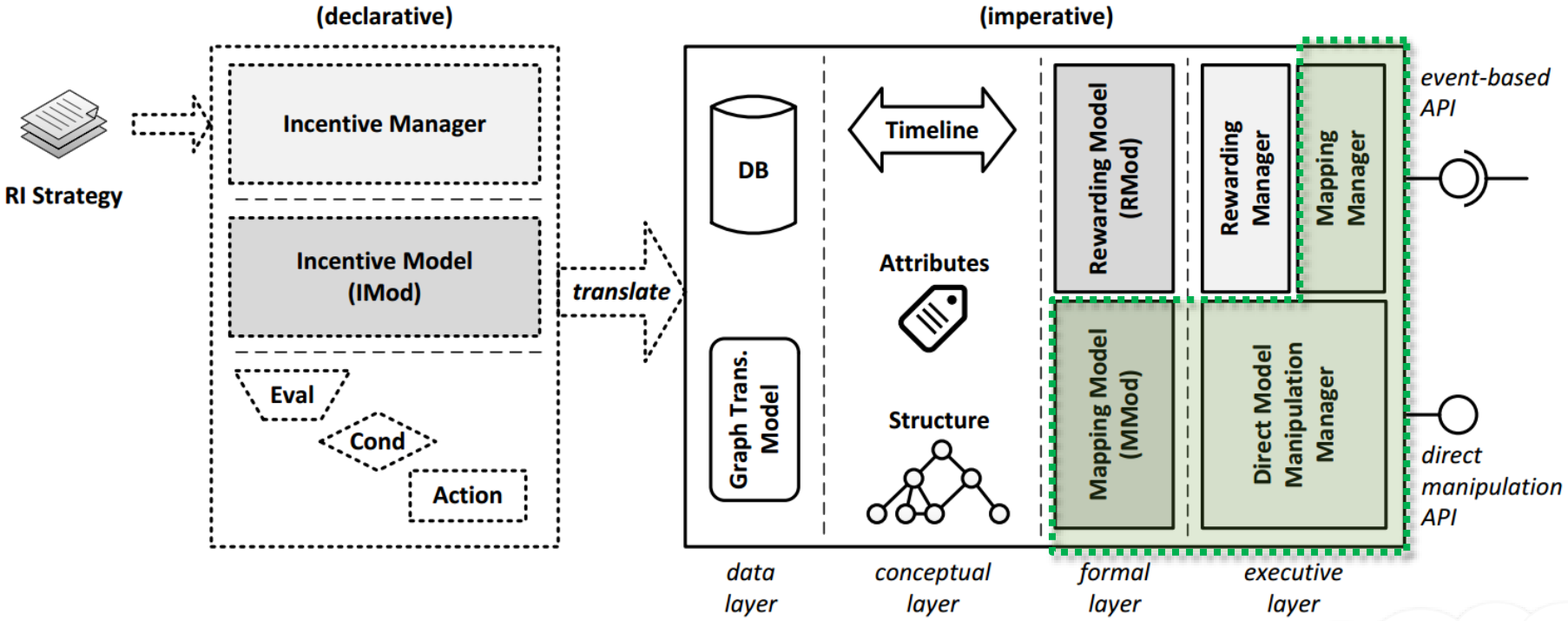
Representation of external system suitable for modeling application of incentives.

- State – Global state, individual worker attributes and performance metrics.
- Time – Records of past and future worker interactions supporting time conditions.
- Structure – Representation and manipulation of various types of relationships

# The Rewarding Model (RMod)

- Examples of mechanisms that RMod can encode and execute:
  - At the end of iteration, award each ICU who scored better than the average score of his/her immediate neighbors.
  - Unless the productivity increases to a level p within n next iterations, expand/reduce current SCU by adding highly trusted ICU or removing inefficient ICU

# PRINC Framework

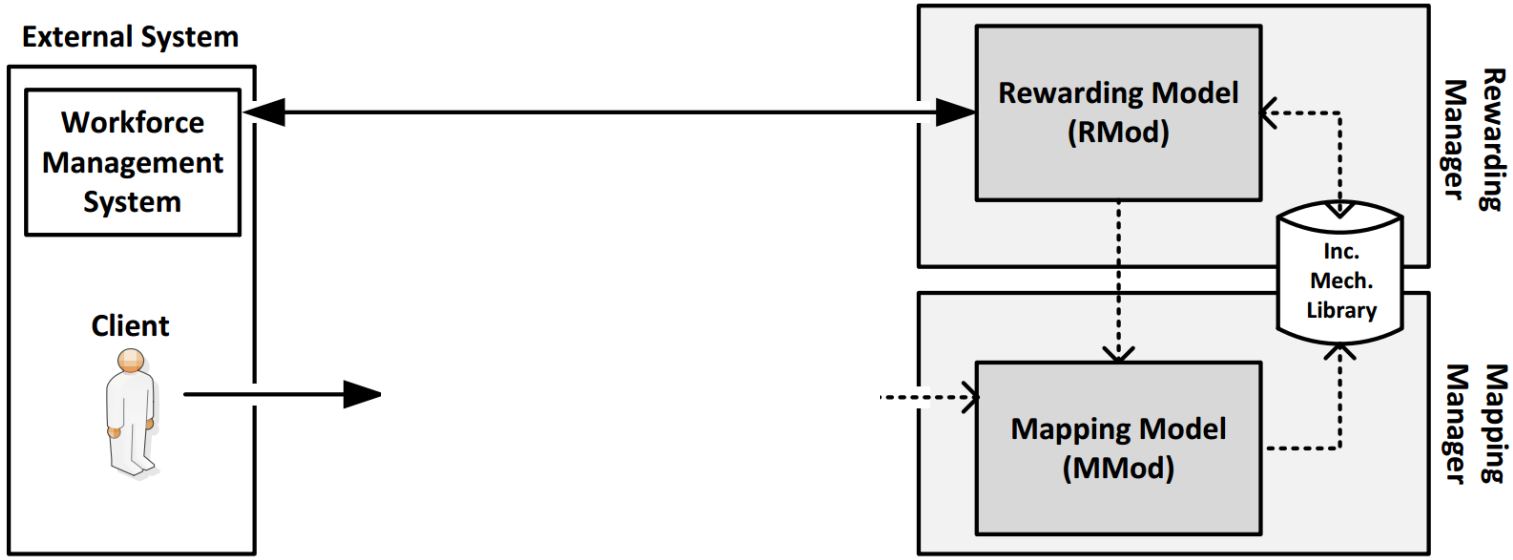



Mapping Model (MMod)

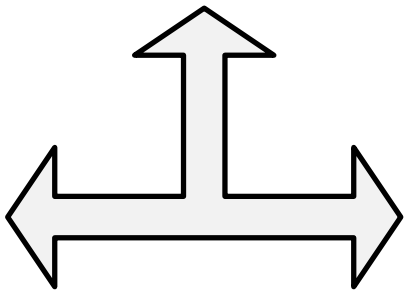
- Definition of system-specific artifacts, actions, attributes and relation types.
- Definition and parameterization of metrics, messages, structural patterns and custom incentive mechanisms.

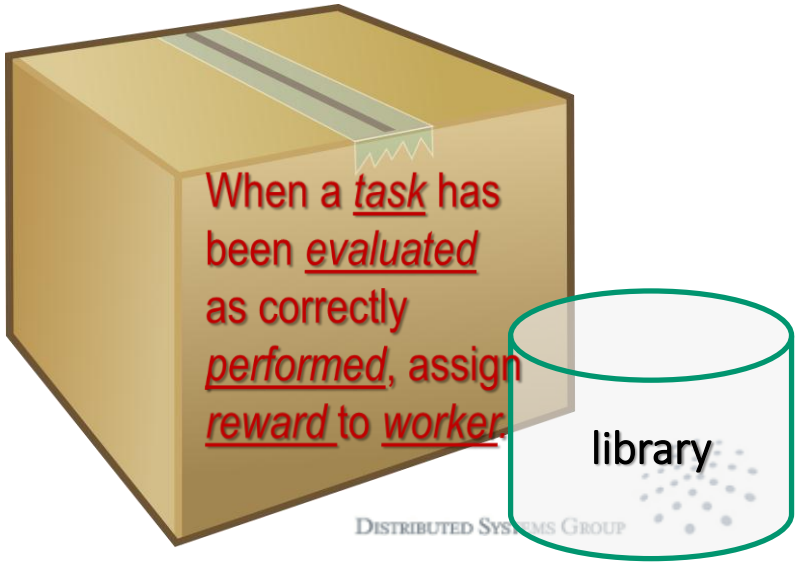
# The Mapping Model (MMod)

- Example: Adapting a general incentive mechanism for a software testing company.

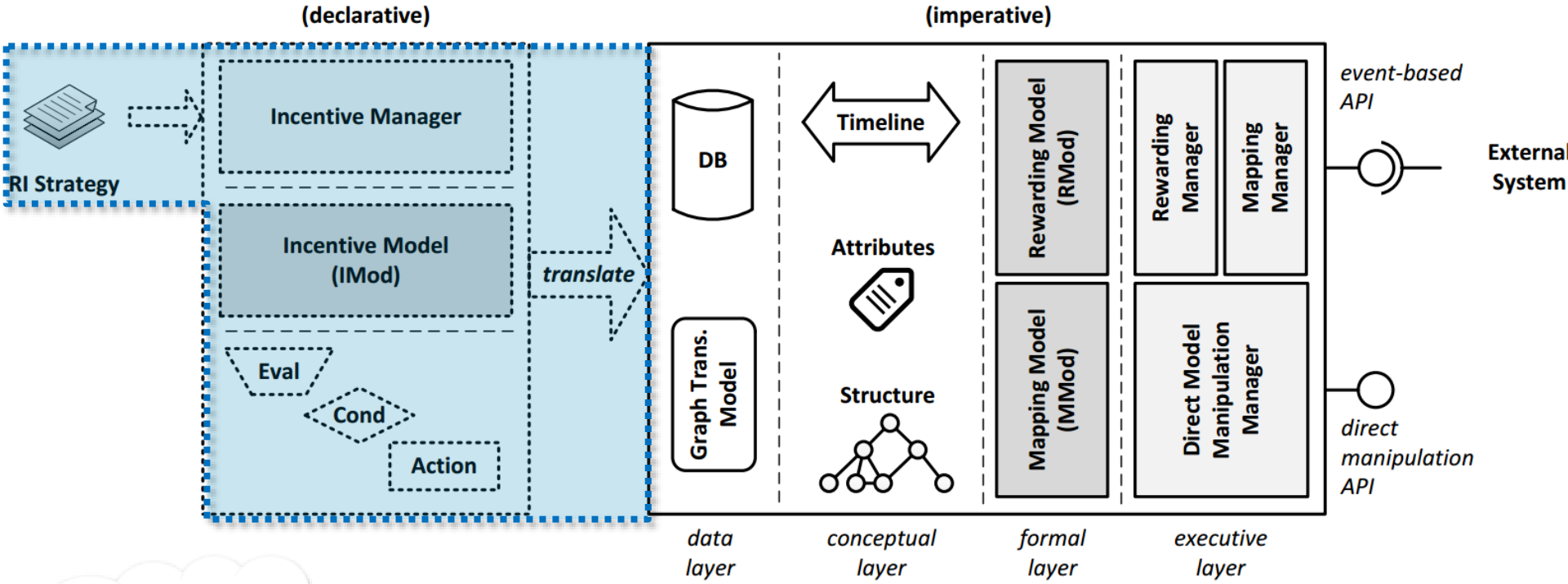








# PRINC Framework




Incentive Model (IMod)

- Declarative, domain-specific language.
- High-level, platform independent, human-friendly notation.

# Illustrating Examples

- Structural incentive mechanism rotating presidency.

```
EvtID = 47; t = END(I3); {
    Worker currMgr = MANAGER(TEAM(T5));
    Worker bestWrk = BEST_OF(TEAM(T5));
    if (currMgr != bestWrk)
      SCHD_EVT(START(I4), SET_MANAGER(T5, bestWrk));
    else
      if (DB_READ('manager', I3, T5, currMgr) == 1)
        SCHD_EVT(START(I4), SET_MANAGER(T5, BEST_OF(TEAM(T5)
            - currMgr)); //replace with 2nd best
      else DB_WRITE('manager', I4, T5, currMgr);
}
```

2.

```
rule SET_MANAGER(var mark:int, var newMgrID:int)  {
      newMgr:Employee;
      if {newMgr.marked == mark && newMgr.id == newMgrID;}
      notNewMgr:Employee;
      if {notNewMgr.marked == mark && notNewMgr.id != newMgrID;}
      <-oldRelation:ManagedBy-> notNewMgr;
      negative { notNewMgr-:ManagedBy->newMgr;}
      modify
      {
            notNewMgr -:ManagedBy-> newMgr;
            delete(oldRelation);
      }
}
```

1.

```
digraph {
n305601358 [label="n305601358:Employee"];
n1673058824 [label="n1673058824:Employee"];
n1578395868 [label="n1578395868:Employee"];
n261998859 [label="n261998859:Employee"];
n1673058824 -> n305601358;
n1578395868 -> n305601358;
n261998859 -> n305601358;
}
Current Manager = 305601358
digraph {
n261998859 [label="n261998859:Employee"];
n305601358 [label="n305601358:Employee"];
n1673058824 [label="n1673058824:Employee"];
n1578395868 [label="n1578395868:Employee"];
n1673058824 -> n1578395868;
n305601358 -> n1578395868;
n261998859 -> n1578395868;
}
New Manager = 1578395868
```

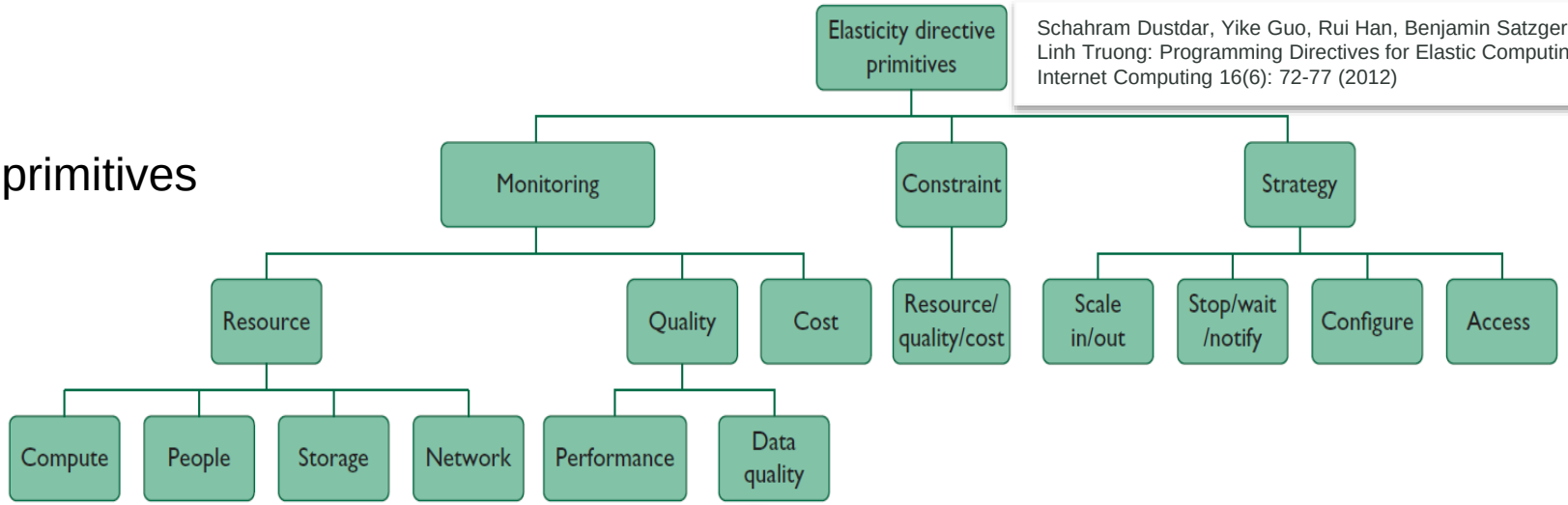
3.

# Engineering Cloud Applications -- modeling and controlling multi-level elasticity of cloud services

Distributed Systems Group

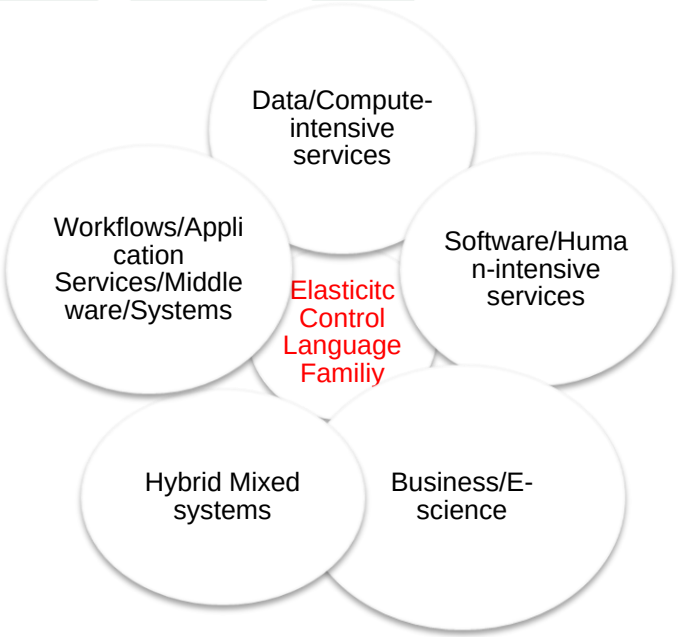
# Specifying and controling elasticity

Basic primitives




Schahram Dustdar, Yike Guo, Rui Han, Benjamin Satzger, Hong Linh Truong: Programming Directives for Elastic Computing. IEEE Internet Computing 16(6): 72-77 (2012)

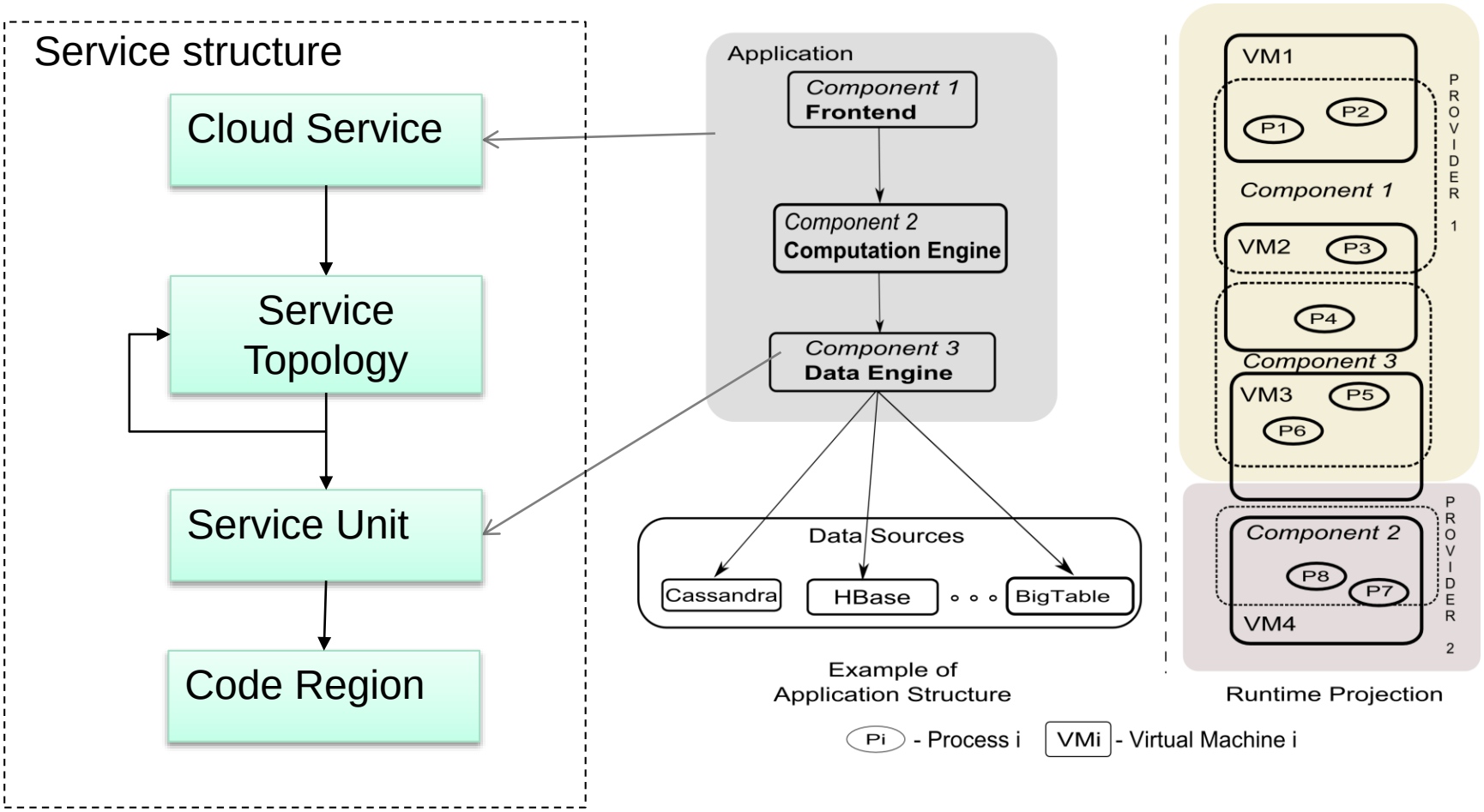
Domain-specific/
Customized features



# SYBL -- Simple Yet Beautiful Language

- Stimulated by directive programming models
  - Goals: easy to use, high-level, multiple levels of control
- Language for elasticity requirements specification
- Possible users: cloud provider, application owner, application developer, software provider
- Targeted to data/compute intensive cloud services

# Multi-level elasticity needed



# SYBL main concepts (1)

- „Monitoring“

  Directives for describing what needs to be monitored and under what conditions

$$M_i := \texttt{MONITORING}\ varName = x_j \mid$$
$$\texttt{MONITORING}\ varName = formula(x_1 ... x_n)$$
$$where$$
$$x_j \in c, c \in \ ApplicationDescriptionInfo$$

Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, "SYBL: an Extensible Language for Controlling Elasticity in Cloud Applications", 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 14-16, 2013, Delft, the Netherlands

# SYBL main concepts (2)

- „Constraint“

  Directive for describing what needs to true and under what conditions

$$C_i := \texttt{CONSTRAINT}\ p\ \in\ formula_i(x)\ rel\ formula_j(y)$$
$$where$$
$$x, y\ \in\ ApplicationDescriptionInfo$$
$$rel\ \in\ \{\leq, \geq, \neq, =\}$$

# SYBL main concepts (3)

„Strategy“

Directive for describing how to achieve certain goals and under what conditions

$$S_i := \texttt{STRATEGY CASE } [Condition : Action] |$$
$$\texttt{WAIT } Condition \mid \texttt{STOP} \mid \texttt{RESUME} |$$
$$\texttt{EXECUTE } strategyName\ parameter_{1...n}$$
$$where$$
$$Condition : DefFunctions \rightarrow \{true, false\}$$

# SYBL main concepts (4)

Other constructs: predefined functions and environment variables

| Function | Description |
|---|---|
| GetEnv | Current cloud infrastructure environment |
| Violated | Checks whether the constraint sent as parameter is violated |
| Enabled | Checks whether an elasticity specification is enabled or not |
| Priority | Returns the priority of an elasticity specification |

| Environment variable | Description |
|---|---|
| optimal_cloud_provider | The cloud provider that the decision components finds to be best suited |
| compute_bid | The current bid for the current cloud provider |
| total_cost | The cost - depends on the level at which variables are being referenced |

# Examples of SYBL elasticity requirements

```
#SYBL.CloudServiceLevel
Mon1 MONITORING rt = Quality.responseTime
Cons1 CONSTRAINT rt < 2 ms. when nbOfUsers < 1000
Cons2 CONSTRAINT rt < 4 ms. when nbOfUsers < 10000
Cons3 CONSTRAINT totalCost < 800 Euro
Str1 STRATEGY CASE Violated(Cons1) OR Violated(
Cons2): ScaleOut
Priority(Cons1)=3, Priority(Cons2)=5

#SYBL.ServiceUnitLevel
ComponentID = Component3; ComponentName= DataEngine
Cons4 CONSTRAINT totalCost < 600 Euro

#SYBL.ServiceUnitLevel
ComponentID = Component2 ComponentName= ComputingEngine
Cons5 CONSTRAINT cpuUsage < 80%

#SYBL.CodeRegionLevel
Cons6 CONSTRAINT dataAccuracy>90% AND cost<400
```

# SYBL and Implementation

- Current SYBL implementation

  in Java using Java annotations

  ```
  @SYBL_CloudServiceDirective(monitoring=„“,constraints=„“,strategies=„“)
  ```

  in XML

  Specific xml schema

  ```
  <SYBLElasticityDirective><Constraints><Constraint
    name=c1>...</Constraint></Constraints>...</SYBLElasticityDirective>
  ```

- Other possibilities

  C# Attributes

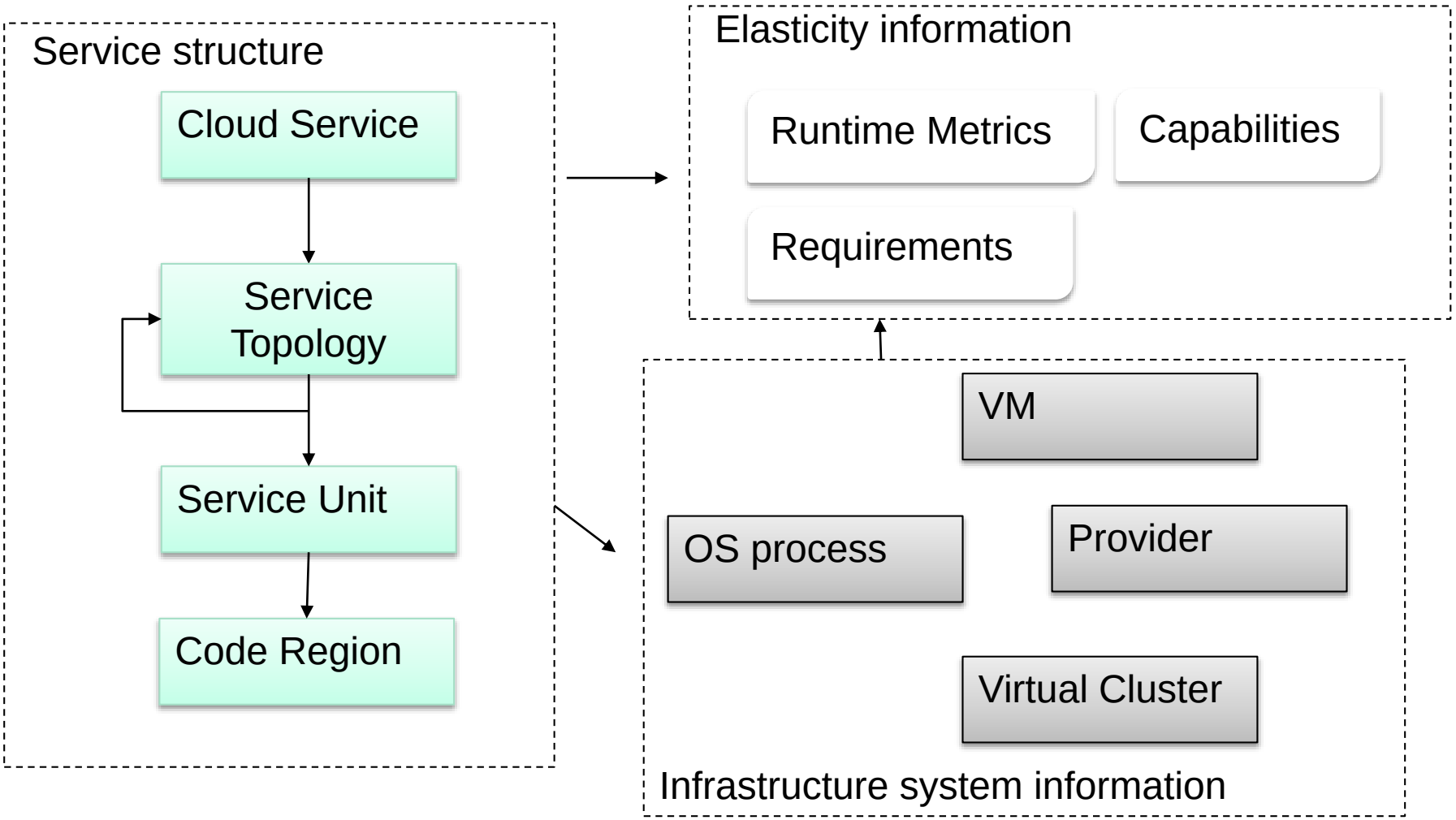
  ```
  [SYBLElasticityAttribute(monitoring=„“,constraints=„“,strategies=„“)]
  ```

  Python Decorators

  ```
  @SYBLElasticityDecorator(monitoring,constraints,strategies)
  ```

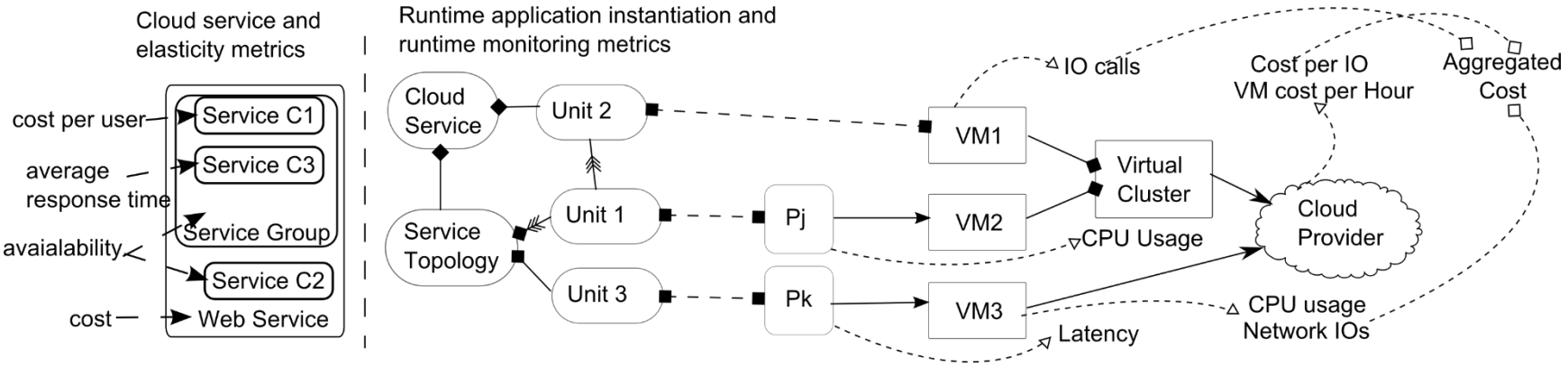
  ...

# Controling the elasticity



# Complex mapping and generation actions for enforcing elasticity (1)

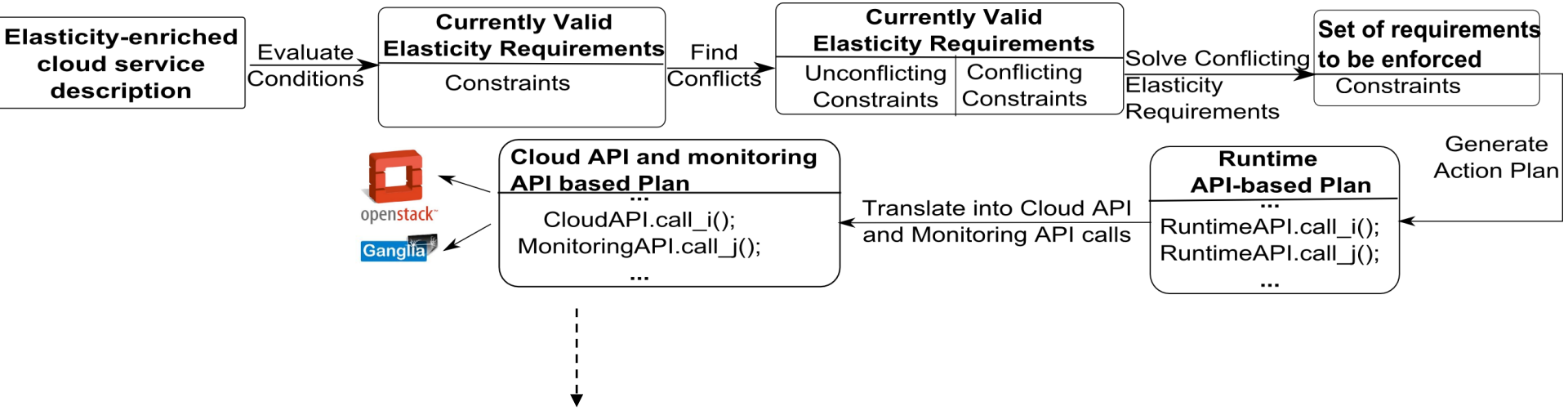
Constructing and maintaining the elastic cloud service dependengy graph




Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, "**Multi-level Elasticity Control of Cloud Services**", June 2013, On Submission.

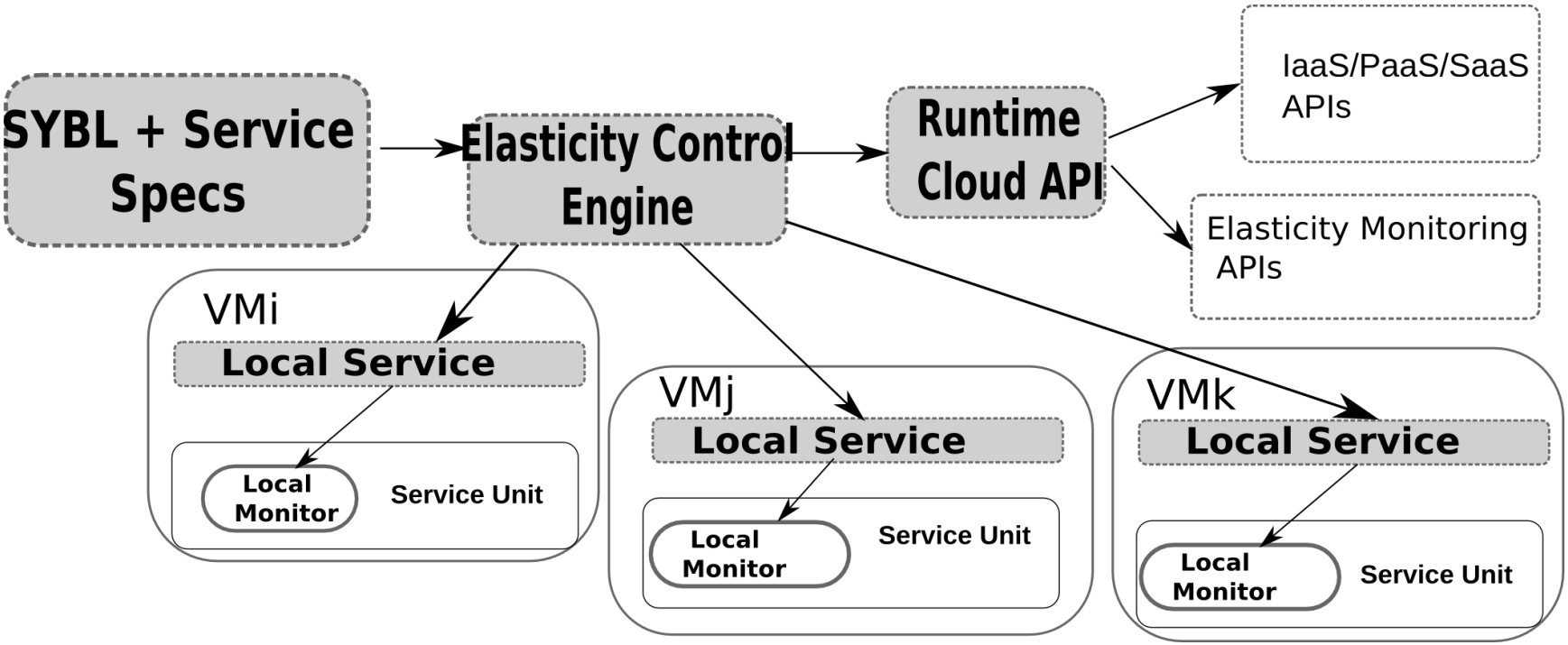
# Complex mapping and generation actions for enforcing elasticity (2)

Steps in enforcing elasticity




Cloud providers/tools must support higher and richer APIs for elasticity controls

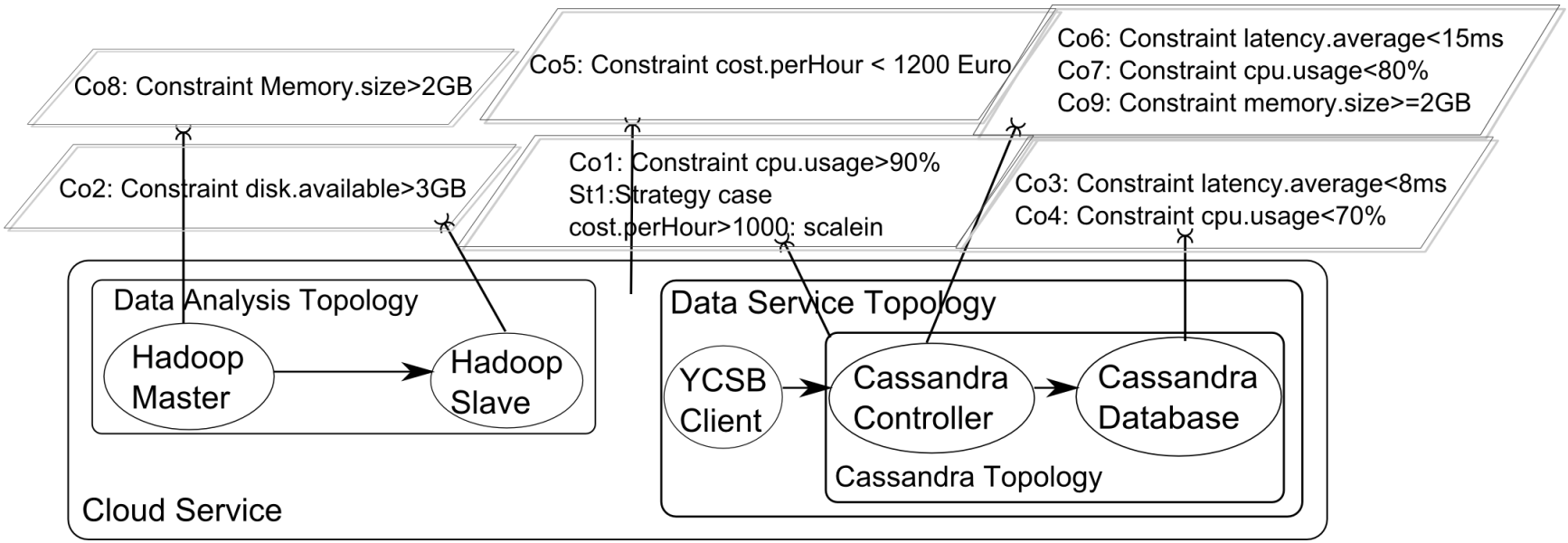
# Elasticity Control as a Service




Currently, we support non-shared computational resources (VM)

# Examples of Elasticity Controls

A service provider deploys its cloud service to an IaaS infrastructure

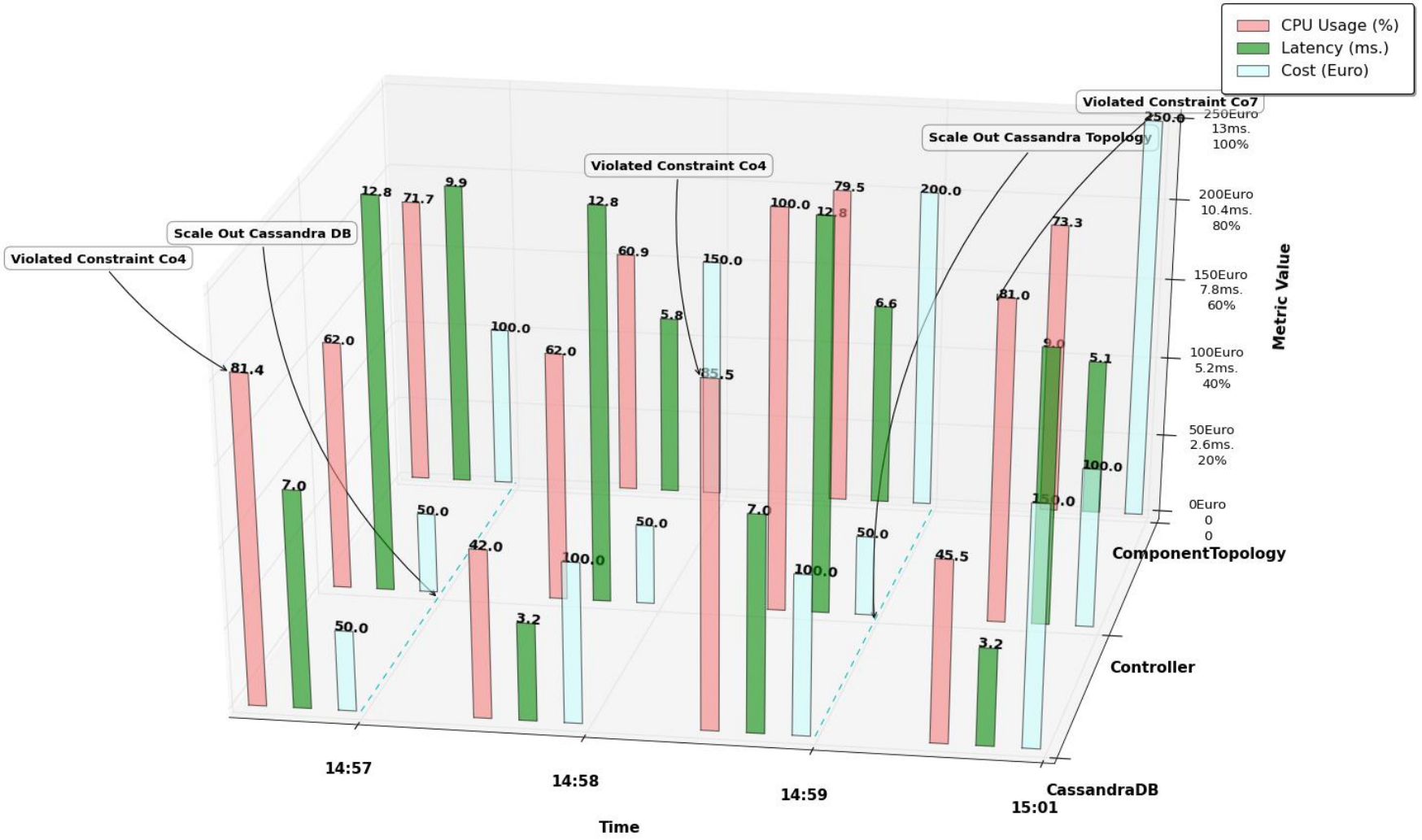



| Configuration | Controllers | DB Nodes | Total execution time | Cost |
|---|---|---|---|---|
| Config1 | 1 | 3 | 578.4 s | 0.48 |
| Config2 | 1 | 6 | 472.1 s | 0.91 |
| Config3 | 2 | 2 | 382.4 s | 0.42 |
| Config4 | 3 | 7 | 372.2 s | 0.72 |

Service unit level

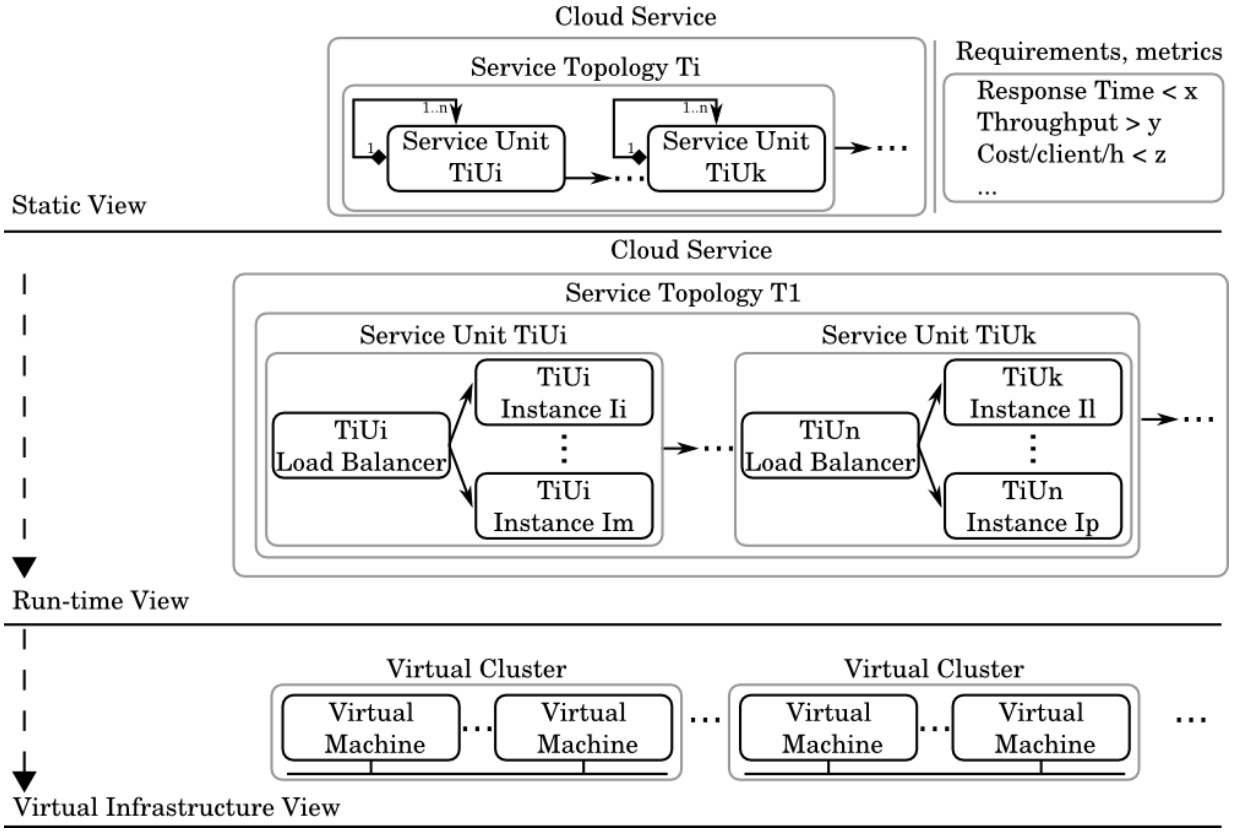
Service topology level

# Elasticity actions and metrics



# Engineering Cloud Applications – elasticity monitoring and analysis

# The complexity of elasticity monitoring




Elasticity Requirements

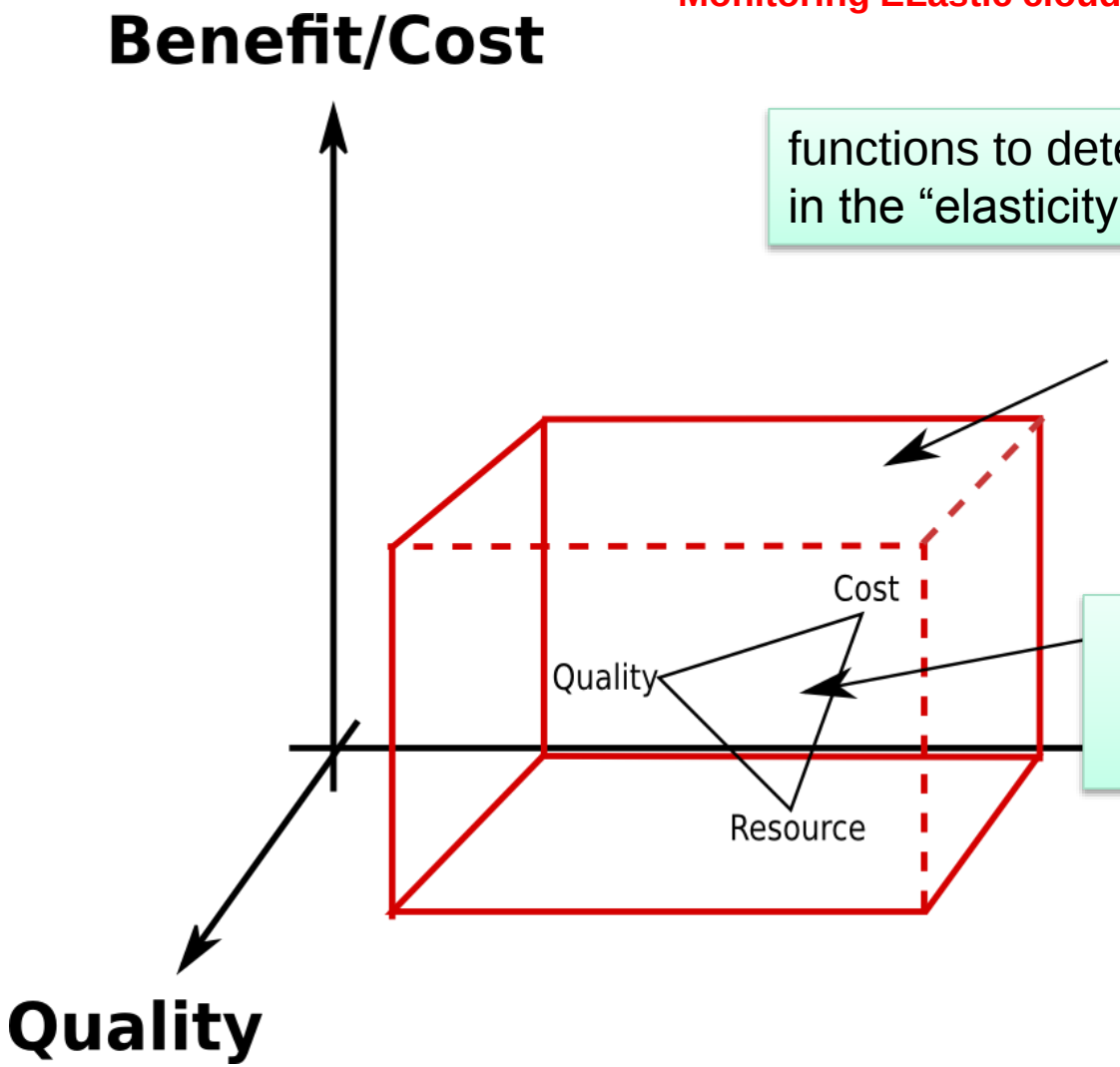
Elasticity Boundaries

Refined based on runtime views

How to detect and characterize the elasticity behaviors?

# Elasticity Model for applications

Moldovan D., G. Copil,Truong H.-L., Dustdar S. (2013). **MELA - Monitoring ELastic cloud Services. On Submission**



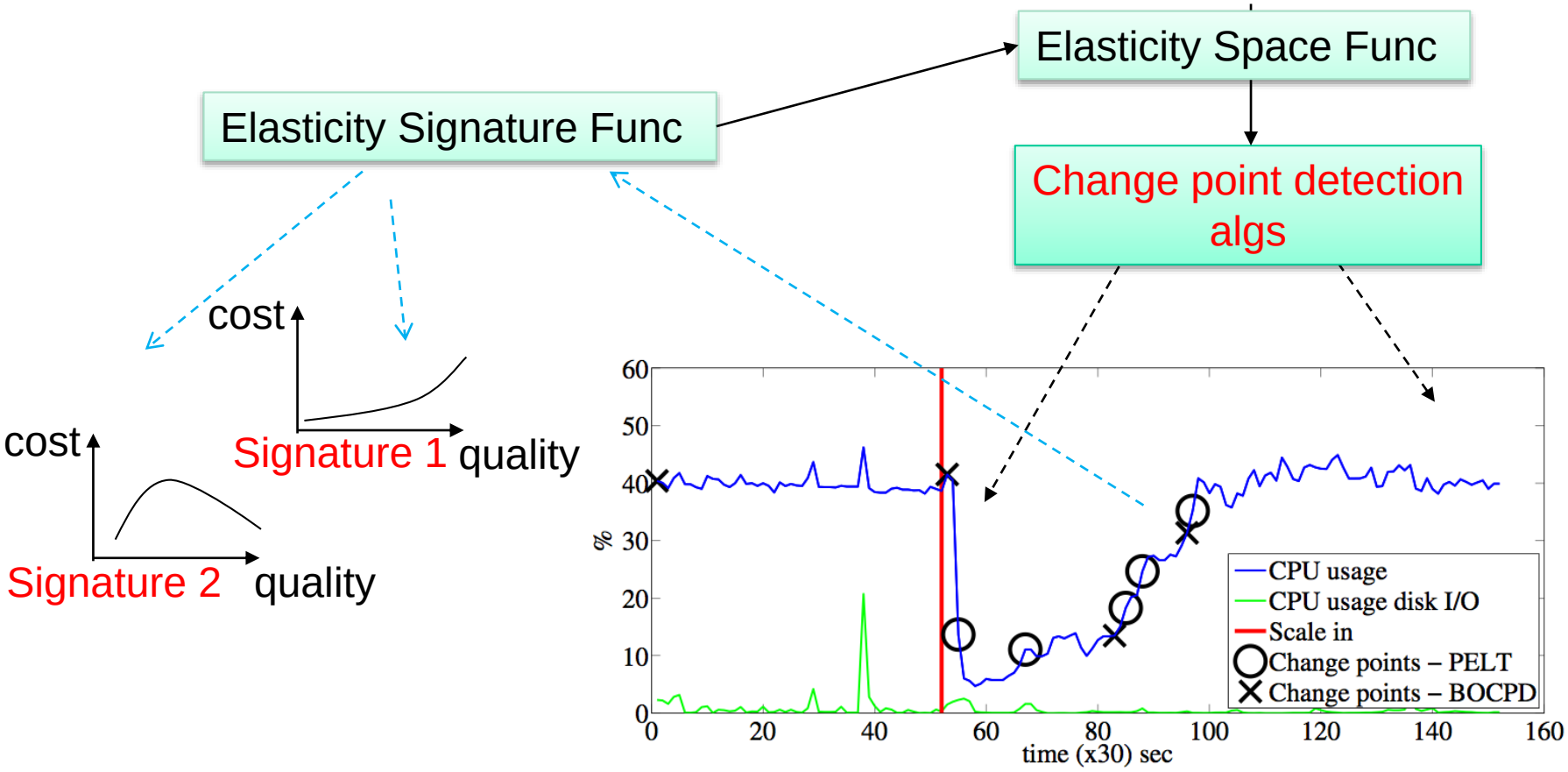

functions to determine if a service unit/service is in the “elasticity behavior”

Elasticity Space

Elasticity Signature

functions to characterize the elasticity behavior from a general/particular view

# Examples of functions for Elasticity Space and Signature

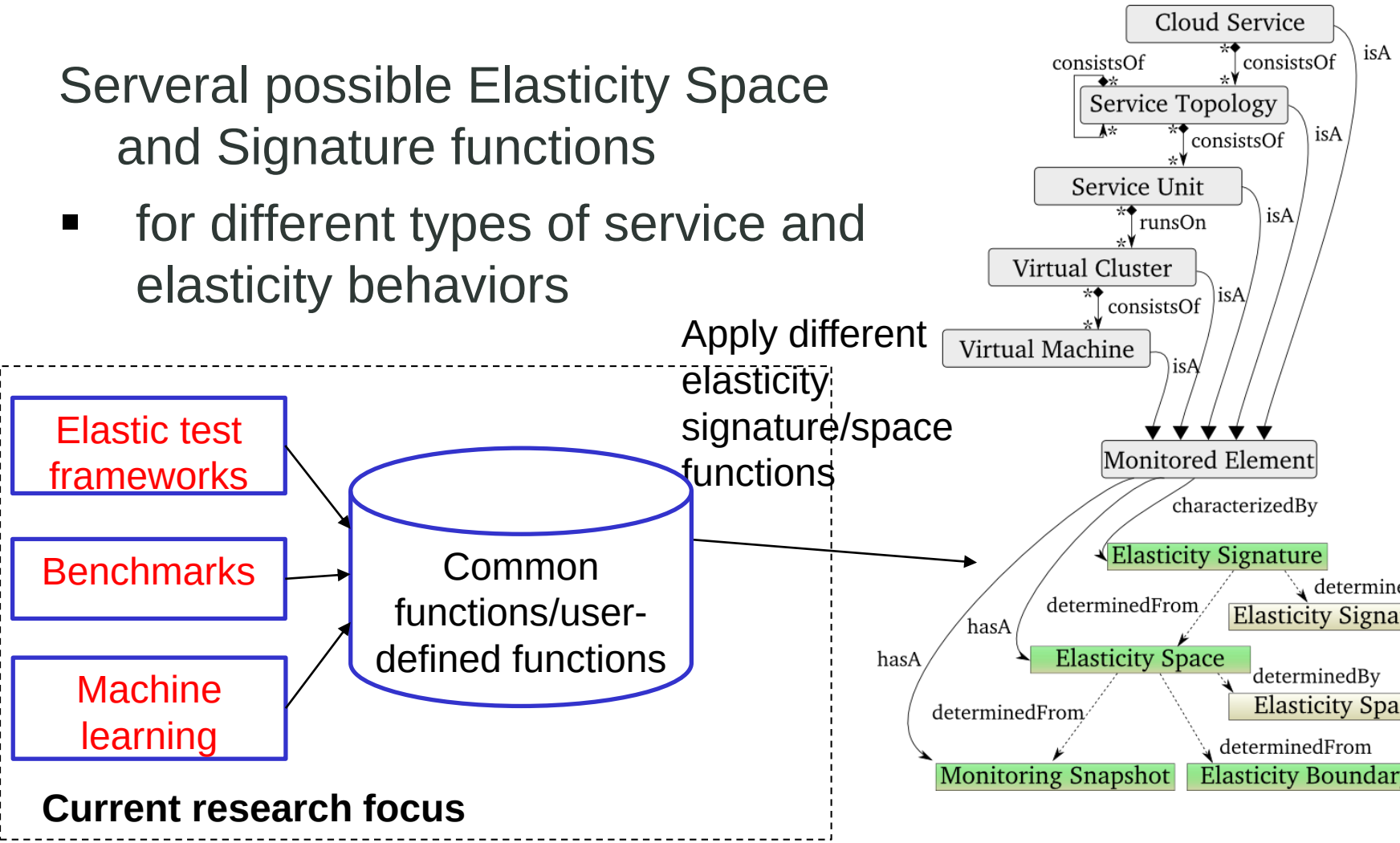



Alessio Gambi, Daniel Moldovan, Georgiana Copil, Hong Linh Truong, Schahram Dustdar: On estimating actuation delays in elastic computing systems. SEAMS 2013: 33-42

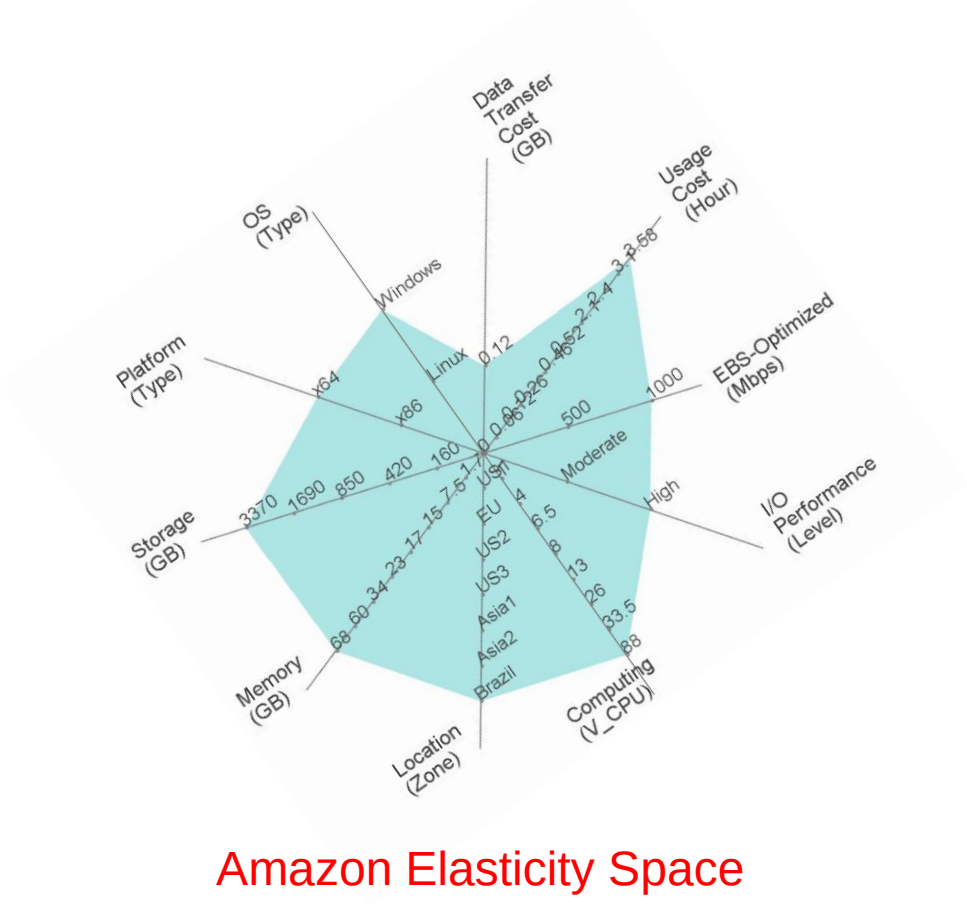
# Multi-level monitoring and analysis of cloud services

Serveral possible Elasticity Space and Signature functions

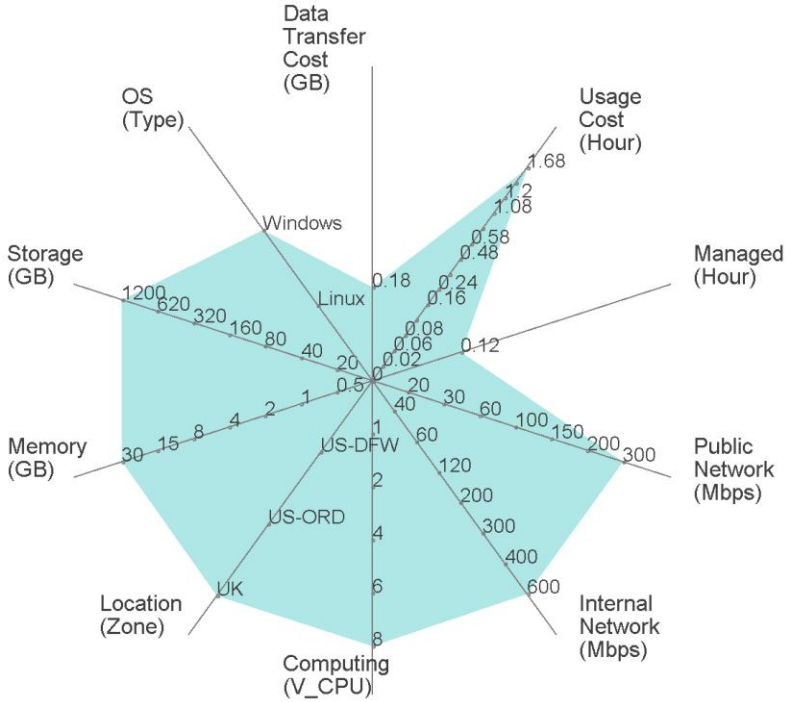
- for different types of service and elasticity behaviors



# Elasticity Space for Cloud Infrastructure

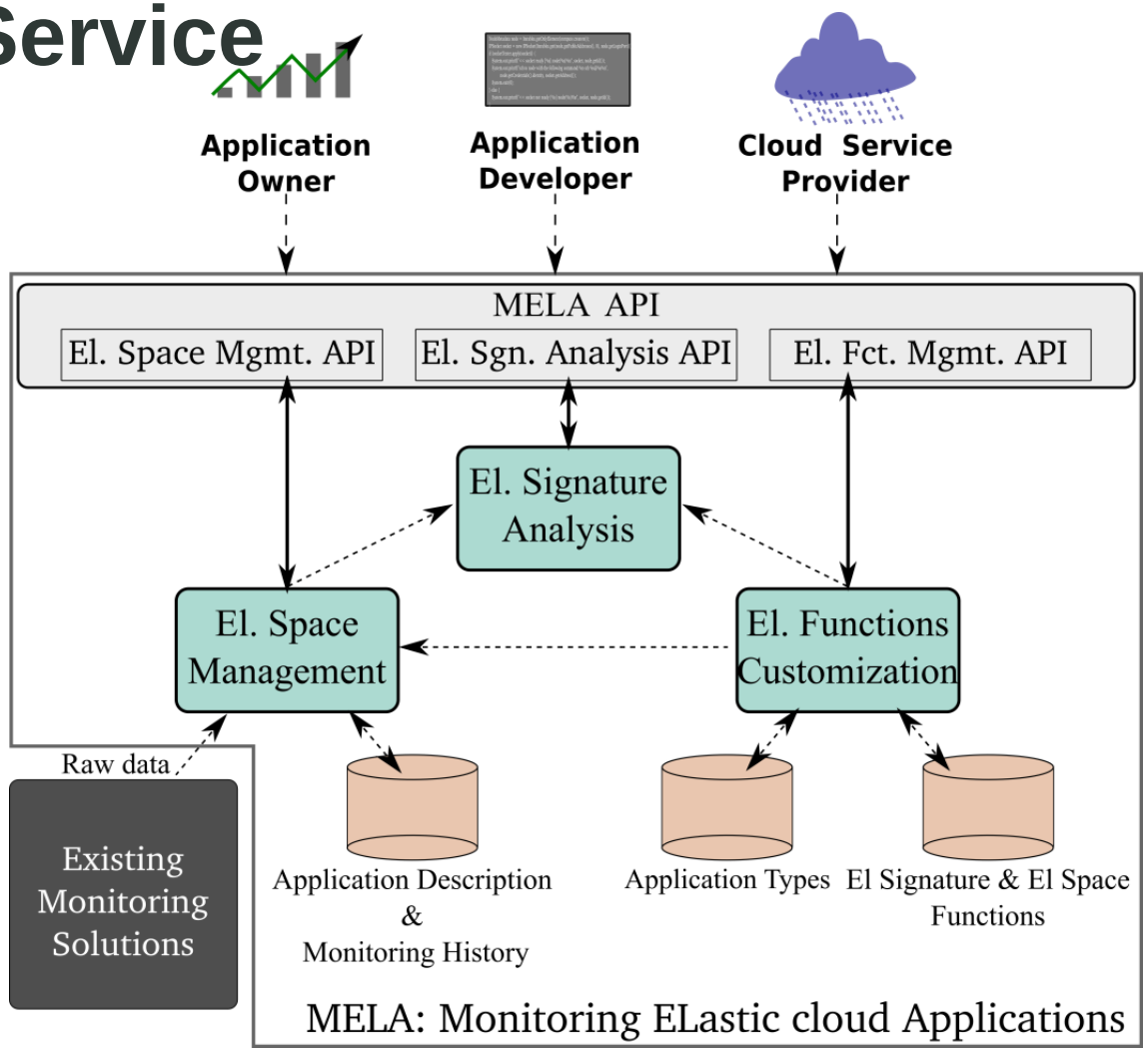



Amazon Elasticity Space




Rackspace Elasticity Space

# MELA -- Elasticity Monitoring as a Service




**Daniel Moldovan, Georgiana Copil, Hong-Linh Truong, Schahram Dustdar, MELA - Monitoring ELastic cloud Services. June 2013, on Submission.**

# Conclusions (1) – Engineering Elasticity

- The evolution of underlying systems and the utilization of different types of resources under different models for elasticity requires
  - Complex, open hybrid service unit provisioning frameworks
    - Different strategies for dealing with different types of tasks
  - quality issues for software, data and people in an integrated manner for different perspectives

- We are just at an early stage of developing techniques for engineering elastic applications wrt multi-dimensional elasticity

# Conclusions (2) – Engineering Elasticity

- Service engineering analytics of elastic systems
  - Programming hybrid compute units for elastic processes
  - Elasticity specifications and reasoning techniques
  - Elasticity spaces analytics
- Application domains
  - „Social computer“ and smart cities (FP 7 FET Smart Cities and PC3L)
  - Computational science and engineering (FP 7 CELAR)

# Thanks for your attention!

Hong-Linh Truong

Distributed Systems Group
TU Wien

dsg.tuwien.ac.at