# Introduction

## Motivation and Objectives

Worldwide data centers electricity consumption accounts for almost 2% of the world production and is expected to overcome the 40% of Total Cost of Ownership of worldwide IT by 2012[1] . The US Environmental Protection Agency estimated that in 2006 the servers and datacenters power consumption accounted for 61 billion kWh (**!! De introdus in acronime**) , about 1.5 % of total U.S electricity consumption and for a cost of $4.5 billion[3].

The same organization points out that the average data center is only 30% efficient, with 70% of the electricity lost due to inefficiencies of power and heat dissipation, along with powering cooling equipment [4]. The environmental impact of datacenter expansion is of great importance, every server using 7,000 kWh of electricity and indirectly generating four tons of carbon dioxide emissions per year [2].

This has lead to the need of finding environmental friendly methods for managing datacenters while maintaining performance. This new research trend has been called Green IT or Green Computing. The philosophy of Green IT is designing and using computer resources in an environmentally friendly way. This work has as aim reducing the overall number of servers used worldwide and further minimizing the number of powered on servers by applying server consolidation.

The main goal of this research is to develop an agent-based framework for dynamically adaptive datacenters which achieves server consolidation by combining server and task virtualization with a self-adapting datacenter controller. The framework should also provide a datacenter self-healing environment control mechanism. A secondary goal is finding new ways and methods of tradeoff between QoS and power consumption for further increasing server consolidation.

## Contributions

This project seeks to provide the necessary tool support for datacenter virtualization, consolidation, environment management and to develop new methods of negotiating between QoS and Power Consumption. There are both theoretical and practical contributions made by this project.

**Theoretical contributions**

* Develop a reinforcement-learning policy-based algorithm for autonomic self-healing environments. Adapt the previously mentioned algorithm for data center task management.
* Develop methods for negotiating between QoS (!!!!!!!!ACRONIME) requirements and Server Optimum Load Values (implying power consumption).

**Practical contributions**

* + - Create a datacenter management framework using policy-based mobile agents.
    - Develop tools for monitoring datacenter resources usage and environment sensors.
    - Find and evaluate appropriate technologies for implementing a self-adapting datacenter based on the framework mentioned above.

## Publications

The research conducted for this project has generated the following publications. The report chapters which include material from these publications are listed for each publication.

* + - **A Reinforcement Learning based Self-healing Algorithm for Managing Context Adaptation** [5] presents a reinforcement learning based algorithm use for finding the optimum sequence of actions which enforce some user-specified policies. A proof of concept implementation is also presented, which uses an X3D (**!!! De pus in lista de achronimes)** representation of a smart environment in which a user click on an object to change its state. The algorithm detects if a user-specified policy is broken, searches for the best (having the highest reward) sequence of actions which “repairs” the context and executes them. (**!!!!** Material from this paper can be found in **chapter2onpage7,Chapter4onpage31andChapter5onpage47**
    - **An Autonomic Algorithm for Energy Efficiency in Service Centers** [6] (!!!!**De modificat daca nu I acceptat)** is an adaptation of the algorithm presented in the previous paper to handle datacenter task management. The old algorithm is still used for monitoring the datacenter environment. Both algorithms are used in a proof of concept implementation which uses a simulated datacenter in which tasks are dynamically added and removed, triggering consolidation actions.
    - **!!!! THESIS OVERVIEW?**

## Background

This chapter provides an overview of the background theory and results in the area of Artificial Intelligence, Self-Adapting Systems and Green Computing specific to the requirements of this project.

The following topics are discussed:

* **Self-\* systems – de completat**
* An introduction to Intelligent Agents and Pervasive Computing
* An overview of Artificial Intelligence Learning, Knowledge Representation and Reasoning
* An introduction to Green Computing, Virtualization and Server Consolidation
* An overview of existing negotiation and bargaining solutions
* An overview of existing self-adapting systems

## Intelligent Agents and Pervasive Computing

Pervasive computing, also named as **everywhere computing** or **ubiquitous computing** [6] is a computing paradigm in which information processing has been integrated in everyday life by means of small networked processing devices. These devices link communications and computing infrastructure to everyday life settings and commonplace tasks [7].

Agents are defined as anything perceives its environment through sensors and performs actions on that environment through actuators [7].

### Agent Environment

By their nature, the environments are roughly classified by [8] in:

**Fully observable and partially observable environments:** As the name suggests, in fully observable environments the agent has all the needed information, while in partially observable not everything about the context is known.

**Deterministic and Stochastic environments**: In a deterministic environment the next state is determined entirely based on the current state and the action executed by the agent.

**Episodic and Sequential environments**: in an episodic environment there are independent episodes, while in sequential ones the next world state is always influenced by the previous one.

**Static and Dynamic environments**: In a static environment the world does not change while the agent thinks what to do.

**Discrete and Continuous environments**: Discrete environments have a finite number of states.

**Single-Agent and Multiple-Agent environments**: In multiple-agent environments agents compete for a set of resources.

### Agent Types

Based on the complexity of the agent’s reasoning process the agents are classified in 5 categories [8]:

**Simple reflex agents:** Act only based on the current input data, ignoring the context history. The agent’s actions are determined by simple rules. Such an agent can succeed only in fully observable environments.

**Model-based agents:** This type of agents maintains an internal model of the world in order to be able to reason over partially observable environments. The model holds previously observed information that now is not observable anymore and reacts just as the simple reflex agent to input data.

**Goal-based agents:** Because in complex situations just reacting will not lead to the desired outcome this type of agent has appeared. These agents know some **goal states** and will try to reach those states.

**Utility–based agents:** Knowing about just the goal does not provide a measure of degree of how expensive is a set of actions the concept of utility is introduced. A **utility function** is used to map each an environment state to a desirability value, thus the agent being capable of comparing two courses of action and choose the one with the highest utility.

**Learning agents:** Are the most complex and can operate in unknown environments. These agents learn about the surrounding world as they go and continuously adapt themselves.

## Artificial Intelligence Learning, Knowledge Representation and Reasoning

### Knowledge Representation

Any intelligent agent, even reflex agents use knowledge about the environment in their reasoning process. The process of knowledge engineering is a daunting one due to the complexity of the surrounding world.

**Ontologies** are the state of the art mechanism for knowledge representation. An ontology can be defined as a set of concepts from a particular domain together with the relationships between those concepts. These concepts are grouped in categories by the use of inheritance, thus providing a flexible and extensible means of representing the surrounding world.

### Learning

The main idea behind learning is to use the gathered world data for improving the decision process, allowing an intelligent agent to improve its behavior trough study of its own experience. The goal of machine learning is for the intelligent system to be able to recognize complex patterns and make intelligent decisions based on them. For achieving this goal, machine learning has to borrow methods and concepts from other fields such as statistics, probability theory, data mining, pattern recognition and others.

### Reasoning

Reasoning in computer science can be regarded as the process of finding in the input data support for some other concept. The most developed research area in this direction is in automated theorem proving algorithms, which are used in finding support for theorems based on the input data. Other research area as reasoning under uncertainty is of major importance in designing intelligent agent systems because the real world is not discrete and conclusions need to be drawn without knowing every outcome of every action.

## An introduction to Green Computing, Virtualization and Server Consolidation

Green Computing or Green IT is a computing paradigm that refers to environmental sustainable IT. All hardware and software components must use as little energy as possible and must have as small as possible impact on the environment. Another reason why this trend is important is because implementing its concepts reduces the total cost of ownership by reducing the energy cost.

### Virtualization and Server Consolidation

A major problem in today’s datacenters is under usage of resources [3][4]. Due to the lack of dynamic datacenter management based on the current or incoming load each datacenter must have all servers online in the event that the traffic increases .Also, being prepared for the largest traffic possible leads to the need of having a large number of servers. These two facts combined generate a problem called **server sprawl**, a situation in which multiple, under-utilized servers take up more space and consume more resources than can be justified by their workload.

**Server Consolidation** has as goal solving this problem by finding means of reducing the number of idle servers. There are several methods for server consolidation, some based on more powerful servers, some on virtualization.

**Virtualization** is a technique for running several operating systems on a single system. By employing virtualization many servers can be transferred to dedicated virtual machines and run on a more powerful system (Fig 1), thus increasing server consolidation. In virtualized environments, the virtual machines run on a “virtual machine OS” called a **hypervisor** which manages the resource allocation and distribution among the virtual machines. A major advantage of this approach is that a virtual machine’s resources can be modified depending on the load. Also, with modern hypervisors, virtual machines can be migrated from one server to another without any visible downtime to increase server utilization or in the event of a hardware failure.

Figure 1: Server Consolidation trough Virtualization

## An overview of existing negotiation and bargaining solutions

A further improvement to server consolidation is the use of negotiation techniques to find a tradeoff between power consumption and QoS. Although QoS requirements are of most importance, in some case a tradeoff is needed due to hardware failure or even unjustified power consumption. For example if a decrease in 5% of CPU requirements for the entire datacenter would allow, after virtual machine rearrangement, for one server to be turned off or send to low power state, then a negotiation technique can be used to find the best value to decrease the CPU for each virtual machine.

The research area of distributed negotiation or multi-agent negotiation has a lot of work associated to it. From this work is clear that negotiation is an important issue in almost any distributed or intelligent system, being present from web services negotiation to grid resource allocation and with the help of this paper, in datacenter consolidation efforts. Web service discovery and invocation benefit from negotiation as described in [8], where a tradeoff between QoS and Cost of Service is achieved trough exchange of less desirable tokens with more desirable ones between parties. Each party provides with its option and also with alternatives and the parties exchange ownership of those alternatives. In [9], the authors present relaxed-criteria negotiation for Grid resources allocation. In this approach, under intense pressure the negotiation criteria are relaxed in order to reach a consensus. A similar technique is used in [10] where a logic programming based framework is used for creating counter proposals.

The research work in the field of multi-issue negotiation can be split into three categories after their understanding of the best negotiation result: minimum loss, maximum gain and the more general utility function maximization. An approach that fits in the first category is [11], which presents a negotiation method which searches for envy-free states. [12] fits in the second category, searching for joint gains as negotiation result. An improvement over existing negotiation techniques is brought by [13] which describes a Nash bargaining solution involving multiple interdependent issues, as encountered in many real-world scenarios.

(!! DACA SE POATE SA SCOT PAPERU ALA CU FUZZY SA BAG PE AICI K AM NEVOIE DE EL SA DEMONSTREZ K I NOU CE AM FACUT)

## An overview of existing self-adapting systems

The work in the area of self-adapting systems is usually centered on the use of ontologies for representing context information and policies for representing goals. This leads to the need of having a reliable mechanism for gathering data from the surrounding environment and representing it in a manner that supports reasoning. Such a mechanism is presented in [15] under the form of an event-driven publish/subscribe architecture for data gathering, processing and event creation. Events are created based on the input data and are used by subscribers to monitor different areas or activities within the surrounding context.

Under the pressure to make computing eco friendly an important research branch in self-adapting systems is creating energy-aware systems that are capable of reducing power consumption and maximize resource usage. One such approach is [16], in which the authors design server provisioning and load dispatching algorithms that minimize energy consumption without affecting user experience. [17] further develops dynamic load management by introducing look ahead control. This approach is based on virtualized server environments and implements a dynamic resource management framework which takes into account the virtual machines switching costs. It has as result an average of 26% power reduction while maintaining QoS requirements.

# Problem Statement and Goals

## Problem Statement

The purpose of this book is to present a self-adapting framework based on mobile intelligent agents which uses ontologies for context representation and policies for goal description. This solution must provide a general extensible platform for building context-aware self-adapting systems applicable to any field, not just energy-aware datacenters. The framework should provide a uniform mechanism of data gathering, context representation and policy enforcement. The policy enforcement engine should detect when the context is in a state which violates at least one policy. It should use a reinforcement learning algorithm to search for the best course of action which brings the context back in an accepted state. The expected reward of being in the resulting sate must be the highest between reachable acceptable states. The reward mechanism must be flexible and should capture accurately the differences in severity between different unacceptable states based not only on the number of broken policies, but also on the distance to the closest acceptable state and the number of sub rules in the policy which are violated. For completing the environment management, the proposed framework must provide a flexible and extensible mechanism for actuators management.

## Problem Goals

1. Create a suitable context representation
2. Build a context management suite
3. Apply the context management solution to a self-healing environment
4. Extends and adapt the context management solution for datacenter management

In order to achieve this goals there are some important sub-goals to be met.

1. **Create a suitable context representation**
   1. Study the RAP[18] context model
   2. Discover the context object types and their relationships
   3. Build a context ontology representation. // using Protégé [19].
2. **Build a context management suite**
   1. Create an information gathering mechanism:

* Create an information gathering module for the System Under Test(SUT) endpoint
* Create a context information consumption mechanism for querying the SUT information gathering modules.
  1. Create an extensible actuator management mechanism:
* Insert the actuators in the context ontology representation
* Design a self-adapting algorithm which can handle in real time actuator list modifications.
  1. Build a logging mechanism for better context monitoring and debugging
* Create a PDF log writer and a log display window
  1. Build a distributed context management framework based on mobile agents
* Study intelligent mobile agents
* Define agents and their behaviors
* Choose an inter-agent communication mechanism
  1. Build a 3D context representation:
* Study the available 3D technologies
* Find a synchronization method between the 3D representation and the real context
* Make the representation interactive: interactions with the 3D context must generate actuator actions.
  1. Provide additional visual system information:
* Provide a real-time plot representing the algorithm running time
* Provide a visual representation of the context policies, sensors and their values

1. **Apply the context management solution to a smart environment**
   1. Create web services for the smart environment sensors
   2. Define policies, actuators and associated actions
   3. Test the context management solution under context patterns
   4. Extensively test the system using the interactive 3D representation
2. **Extends and adapt the context management solution for server cluster management**
   1. Create web services for interacting with the server operating system
   2. Adapt the context representation

* Split the policies into two categories : Energy policies and QoS policies
* Augment the context representation with information particular to datacenters
  1. Find techniques for improving the energy efficiency in datacenters
* Find a mechanism for dynamic task migration
* Find a mechanism for improving server consolidation
  1. Study datacenter management technologies
  2. Provide system information visually:
* Adapt the existing 3D representation
* Implement server resource monitors
* Create a visual representation of the tasks(both deployed and undeployed) for better task tracking
  1. Test the system on a real world server cluster
  2. Develop a workload generating tool for extensive system test.

# Analysis and Design

De inserat descrierea capitolului

CRED K ASTA LA PARTEA DE IMPLEMENTARE.

Due to the joint development effort done by me and a colleague of mine, Copil Gergiana in designing the system’s core, the **reinforcement based self-healing algorithm** [5] and associated concepts, the algorithm and concepts description will appear both in this book and in [19]. Although in the rest of the project the work can be clearly divided, the algorithm was developed together, the concepts where defined together and so it is impossible to perform a distinction between our works in this case. The entire **4.1.The Context Model** and **4.2.The Reinforcement Learning Algorithm** subchapter is joint development effort, elements contained in it being also described in [19].

## The context model

The context model used by the self-adapting solution presented in this book is based on the RAP (Resources, Actors, Policies) model presented in [18]. This model was chosen because is general and powerful enough to be used in any self-adapting system.

### Context Ontology Representation

The core ontology representation only extends the Resources element of the RAP model with Sensor and Actuator concepts. This distinction between resources is needed as the system needs to know what element returns context information and what element can influence the context.



Figure 2: RAP Ontology Representation

Because the datacenter context needs more specific information, another ontology representation is defined and used together with the previous basic representation. The previous representation is used for environmental representation and the next ontology is used to represent the datacenter physical and software components. Also, due to the fixed number of actions possible in a datacenter, the generality of the representation is reduced by creating a concrete class for each possible action. The reduction is generality is balanced by an increase in efficiency, eliminating the search for all possible actions at each step in the algorithm.



Figure 3: Datacenter Ontology Representation

### Policies

The policies consist of a set of acceptable sensors values. In order to have a fine-grained view over any context situation and be able to differentiate between the severity of two different context situations, weights are used. Each policy has a weight associated to it which represents the importance of the policy in the overall context. Furthermore, each resource associated to a policy has a weight representing the importance of the sensor in that particular policy.

### Context Entropy

For understanding the concept of context state entropy the state must be defined. A context state is a snapshot of the execution environment taken at a specific moment in time.

For evaluating the degree of context state deviation from the acceptable value the concept of context state entropy is used [5]. The context state entropy is used to represent the desirability of the system to be in a particular state. Based on the definition of the policy, the entropy is defined as the product between the policy weight and the sum of products between the associated resources weight and their distance from the acceptable value:

|  |  |
| --- | --- |
|  | (1) |

dDE PUS UNDEVA IN LISTA DE ECUATII SAU CEVA. SAU FIGURI> NUSH

, where:

- **:** the weight of i-th policy

- **:** the importance of resource j in policy i

- **:** is a value marking the degree to which resource j respects policy i.

### Inter-Independent Resources Group

The time required to find the best sequence of actions that brings the system in an accepted state depends on the number of policies and on the number of resources attached to a policy. For improving the performance of the system the concept of Inter-Independent Resources Group (IIRG) is introduced. In order to understand the IIRG concept the dependency relation must be defined first. A resource is dependent on another resource when a change in the value of one of the resources triggers a change in the value of the other resource. Furthermore, the dependency relation is of two types: direct dependency and indirect dependency. Direct dependency between two resources and occurs when a change in the value of triggers a change in ’s value. Indirect dependency between two resources and occurs when a change in ’s value triggers a chain of changes in the values of several resources () which eventually affects . The dependency relation is represented under the form of a dependency matrix with has as entries the length of the dependency between the resources (1 for direct dependency and path length +1 for indirect dependency). A “0” in the dependency matrix means no dependency.

Table 1: Resource Dependency Matrix example

|  |  |  |  |
| --- | --- | --- | --- |
|  | R1 | R2 | R3 |
| R1 | 0 | 0 | 0 |
| R2 | X | 0 | 0 |
| R3 | Y | 0 | 0 |

From this it is defined that two resources are inter-independent if there is no direct dependency between them and if the length of the indirect dependency chain is over a certain limit. By separating the resources in IIRGs solutions for context repair can be searched in parallel over the IIRGs, thus reducing the system’s search space and improving the running time.

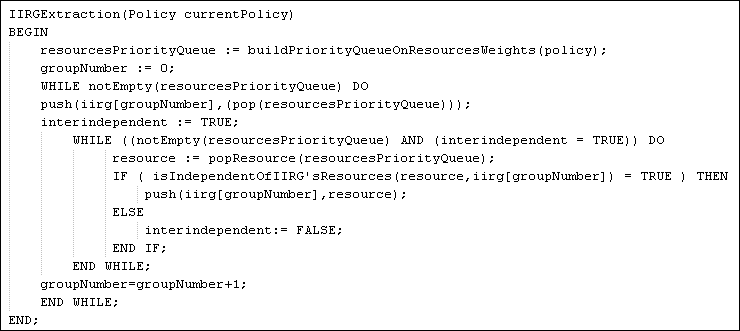


Figure 4: IIRG Extraction Pseudo-Code

For a better understanding of the IIRG extraction process a trace is provided on an example where it is assumed that the resources weights follow this rule : weight(R2) >weight (R3) >weight (R1).



Figure 5: IIRG Extraction Algorithm Trace

### RAP instantiation for the self-healing system

For using the RAP model for implementing a self-healing environment its elements must be mapped onto real world entities.

Each sensor can have associated one or more actuators which have an influence on it. Each actuator has at least one action defined, each action having its effect represented by a **+ value (increment action), - value (decrement action)** or **value (set value action).**

Table 2: Self-healing environment RAP mapping

|  |  |  |
| --- | --- | --- |
| <R,A,P> entity | Self-Adapting context entity | Details |
| Resources | **Sensors** | * Smart room sensors |
| **Actuators** | * Components capable of influencing the environment |
| Actors | **Actions** | * Each actuator has associated actions |
| Policies | **User specified policies** | * Acceptable sensor values combinations for the environment |

### RAP instantiation for the self-adapting datacenter

The previous RAP instantiation is still used in the self-adapting datacenter for environmental control but a new mapping is needed for task management. Due to the reduced number of possible datacenter actions (excepting environmental control) the new mapping is simpler.

Table 3:Self-Adapting Datacenter RAP Mapping

|  |  |  |
| --- | --- | --- |
| <R,A,P> entity | Self-Adapting context entity | Details |
| Resources | **Servers** | * All datacenter servers |
| Actors | **Actions** | * Deploy task * Move task * Send server to hibernate * Wake up server |
| Policies | **Energy policies** | * Servers key performance indicators |
| **QoS policies** | * Amount of requested resources by tasks |

## Reinforcement Learning Algorithm

For finding the best sequence of actions to bring the system in an acceptable state an algorithm based on reinforcement learning was developed. The advantage of basing this approach on reinforcement learning is the generality provided by this type of learning. Any real world situation can be represented in terms of <action, reward> pairs. Also, by manipulating the expected reward the mobile agent that uses this algorithm can take any desired course of action.

The desired context states are described using policies. Based on these policies the algorithm computes the entropy value for the current context state. If the entropy value is above 0 (or above a user-defined threshold) the search for the repairing sequence of actions is triggered. For each broken policy, the algorithm takes each broken resource and tries to change its value in an acceptable one by executing actions on associated actuators. Each associated action executed is simulated, the resulting state’s entropy is computed and the state is placed in the states priority queue, sorted by entropy. The next state to be expanded is the top of the queue, which holds the best state found so far.

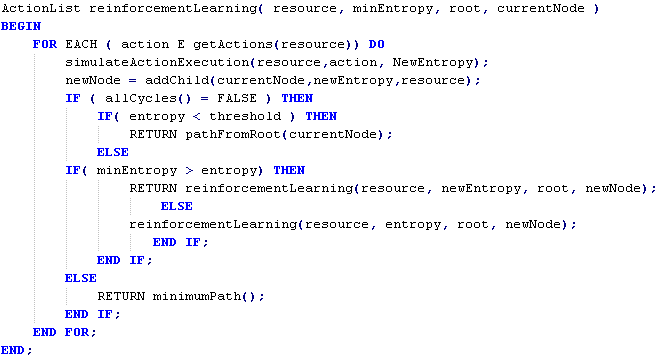


Figure 6: Reinforcement Learning Algorithm DE REFACUT EVENTUAL PSEUDOCODU

Due to the generality of the reinforcement learning approach, the previously described algorithm can be applied easily both to a self-healing smart environment and to a self-adapting datacenter.



Figure 7: Self-Healing system Reinforcement Learning Process

Due to the real-time performance required from real world datacenters, in the datacenter version the algorithm can be stopped at any given moment in time and the best action sequence found so far is used. This enables the system to stop searching for the perfect solution if a certain time period has passed and just use the best that was found so far.



Figure 8: Datacenter Reinforcement Learning Process

## The Conceptual Architecture

Having an agent-based architecture, the system mainly consists of four agents all interacting with the same ontology: Context Model Administering Agent (CMAA), Context Interpreting Agent (CIA), GUI Agent and Reinforcement Learning Agent (RLA). The CMAA reads a context description and loads the information into the shared ontology. CIA periodically queries the sensor data and refreshes the context information. The GUI Agent is responsible of the graphical context representation. RLA is the most important agent. It detects when the context is broken and searches for actions to bring it back in an acceptable state.

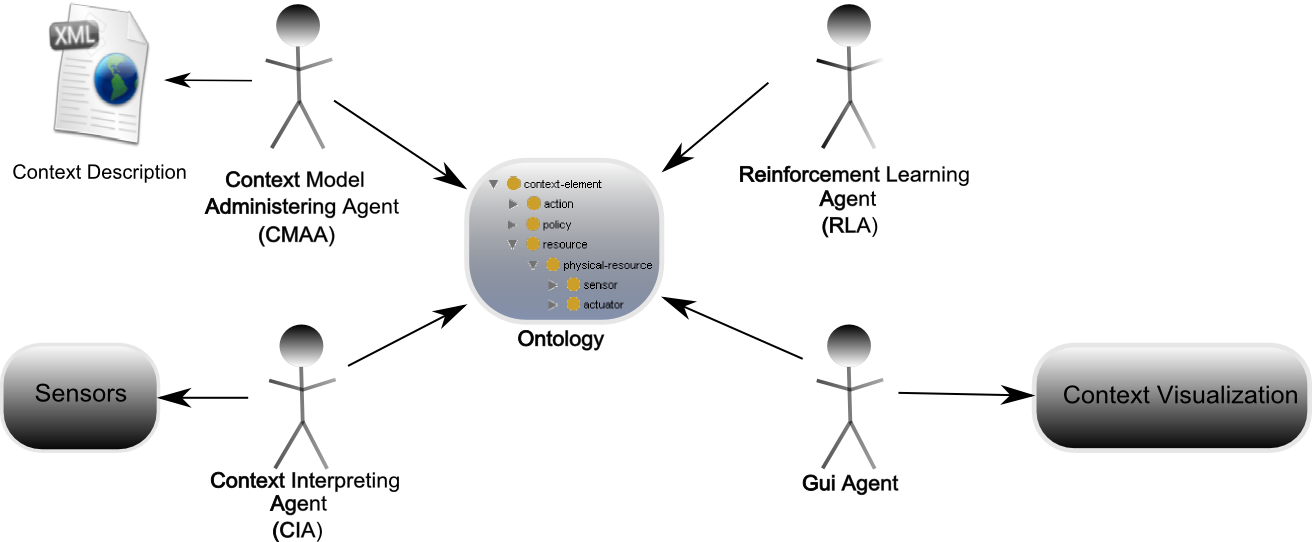


Figure 9: System Conceptual Architecture

# Implementation

# Related Work

# Conclusions

# References

[1] Green **Active Management of Energy in IT Service centres:** Performance Computing Center Stuttgart (HLRS), <http://www.hlrs.de/organization/av/isis/research/distributed-systems/games/>

[2] M. Poniatowski: **Foundations of Green IT: Consolidation, Virtualization, Efficiency, and ROI in the Data Center,** Prentice Hall, 2009, ISBN-10: 0137043759, ISBN-13: 9780137043750, 978-0137043750, pages xiii  
[3] **VMware Whitepa**per : How VMware Virtualization Right-sizes IT Infrastructure to Reduce Power Consumption, <http://www.vmware.com/files/pdf/WhitePaper_ReducePowerConsumption.pdf>, pages 4

### [4] EPA [Data Center Report to Congress -FINAL 7-25-07](http://www.federalnewsradio.com/pdfs/EPADatacenterReporttoCongress-August2007.pdf), <http://www.federalnewsradio.com/pdfs/EPADatacenterReporttoCongress-August2007.pdf>,

Pages 6

[5] T.Cioara, I.Anghel, I.Salomie, M.Dinsoreanu, C.Georgiana and **M.Daniel**: **A Reinforcement Learning based Self-healing Algorithm for Managing Context Adaptation,** First International Workshop on Communication, Collaboration and Social Networking in Pervasive Computing Environments (PerCol 2010)

[6] Ubiquitous Computing, <http://en.wikipedia.org/wiki/Ubiquitous_computing>

[7] S.Russell, P.Norvig : **Artificial Intelligence A Modern Approach**, Second Edition, Prentice Hall,ISBN-10: 0137903952, ISBN-13: 978-0137903955

[8] P.C.K. Hung, H. Li: **Web Services Discovery based on the Trade-off between Quality and Cost of Service: A Token-based Approach**

[9]K.M. Sim: **Relaxed-criteria G-negotiation for Grid Resource Co-allocation (Position Paper)**,

[10] C. Sakama, K. Inoue: **Negotiation by Abduction and Relaxation**

[11] Y.Chevaleyre, U.Endriss, S.Estivie, N.Maudet : **Reaching Envy-free States in Distributed Negotiation Settings**

[12] Q.B.Vo, L.Padgham: **Searching for Joint Gains in Automated Negotiations Based on Multi-criteria Decision Making Theory**

[13]K.Fujita, T.Ito, .Klein: **Finding Nash Bargaining Solutions for Multi-issue Negotiations: A Preliminary Result**

[14]D. Baker, D. Georgakopoulos, M. Nodine and A. Cichocki: **From Events to Awareness**

[15] R.M.Bahati, M.A.Bauer : **Adapting to Run-time Changes in Policies Driving Autonomic Management**

[16] G.Chen, W.He, J.Liu,S.Nath,L.Rigas,L.Xiao,F.Zhao: **Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services**

[17] D.Kusic,J.Kephart, J.E.Hanson, N.Kandasamy and G.Jiang:**Power and Performance Management of Virtualized Computing Environments Via Lookahead Control**

[18] I.Salomie, T.Cioara, I.Anghel, M. Dinsoreanu : **RAP – A Basic Context Awareness Model**

[19] REFERINTA LICENTA GEORGIANA

**Protégé**, <http://protege.stanford.edu/>