Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 15(3) September 2014

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 15, Number 3, September 2014

TABLE OF CONTENTS

DOI 10.12694/scpe.v15i3.1015
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON MODELING AND SIMULATION APPLICATIONS OF SOFTWARE AGENTS

Dear SCPE readers,

The aim of this Special Issue is to introduce to the readers a selection of papers from the 3rd Workshop on Applications of Software Agents WASA 2013 in the area of agent-based modelling and simulation. WASA2013 was held in Sinaia, Romania, during October 11-13, 2013. The WASA 2013 workshop was organized within the framework of the 17th International Conference on System Theory, Control and Computing ICSTCC2013. The aim of the WASA series of workshops is to contribute to the advancement of technologies and applications of software agents' by bridging the gap between the theory and practice of software agents.

15 papers were accepted for presentation at WASA2013. Among them, 3 papers were selected, after further extension and additional review, for inclusion in this Special Issue on Modeling and Simulation Applications of Software Agents.

The article Formal Modelling and Simulation of a Multi-Agent Nano-Robotic Drug Delivery System by Marina Ntika, Petros Kefalas, and Ioanna Stamatopoulou is in the area of agent-based modelling and simulation. The authors propose an agent-based simulation model for targeted drug delivery with the help of nano-robots. The simulation model is developed using X-machines, while the simulation itself was achieved using NetLogo. The paper reports interesting experimental results that were obtain by applying the proposed agent-based simulation model, thus supporting the usefulness of formal multi-agent system for the better understanding of collaborative nano-robotic technologies for drug delivery in unhealthy tissues.

The article A System Supporting the Evaluation of the Operational Effectiveness of Naval Tasks Based on Agent Simulation by Davide Anghinolfi, Alberto Capogrosso, Massimo Paolucci, and Perra Francesco is in the area of agent-based modelling and simulation. The authors propose an agent-based simulation model denoted as denoted as Operational Evaluator Model, for the multi-dimensional analysis of alternative configurations of military naval units to support the design of naval vessels. After the careful consideration of several existing frameworks for agent-based simulation, the authors propose their own framework denoted as ABSF. The paper contains convincing experimental results obtained by simulation of a specific Anti-surface Warfare use case that support the advantages of ABSF over other agent-based simulation frameworks.

The article ABVE-Construct: An Agent-Based Virtual Enterprise Model for Civil Engineering by Mihaela Oprea is in the area of agent-based enterprise modelling. This paper proposes a generic framework for virtual enterprises development denoted as VE-Frame. Based on this framework, a new agent-based virtual enterprise model for the civil engineering domain denoted as ABVE-Construct, is formulated. The ABVE-Construct ontology and design model are developed using OWL and Protg, as well as using Prometheus Design Tool. The proposed framework and model are evaluated by considering a specific case study of a residential building construction task.

We would also like to thank all the reviewers for their restless reviewing effort and valuable feedback and all the authors who submitted papers to WASA2013 and to this Special Issue.

Costin Bădică and Mirjana Ivanović

DOI 10.12694/scpe.v15i3.1016
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. 201–216. http://www.scpe.org

A SYSTEM SUPPORTING THE EVALUATION OF THE OPERATIONAL EFFECTIVENESS OF NAVAL TASKS BASED ON AGENT SIMULATION*

DAVIDE ANGHINOLFI† ALBERTO CAPOGROSSO‡ MASSIMO PAOLUCCI§ AND PERRA FRANCESCO¶

Abstract. This paper considers a system for supporting the multi-dimensional analysis of alternative configurations of military naval units, which is currently the object of a research and development project between Orizzonti Sistemi Navali, an Italian ship design company, and the University of Genova (Italy). The project aims at providing a decision making aid in the early phase of the vessels' design. In particular, the agent-based simulation framework included in such a system is presented and its characteristics are compared with a set of four existing frameworks, i.e, MANA, NetLogo, Stage and MASON. Agent-based simulation is specifically exploited in the early design phase for the evaluation of the vessel efficiency in a set of reference naval tasks. The comparison is performed considering a common reference scenario and a set of qualitative and quantitative key performance indicators. The results obtained show both the effectiveness and efficiency of the proposed agent-based simulation environment for the considered application case.

Key words: Agent-based simulation, decision support systems, military naval tasks.

1. Introduction. During the early design phase of military naval vessels, it is of fundamental importance being able to evaluate their operational effectiveness with respect to the set of the main naval tasks in which such vessels will be engaged. This capability, indeed, allows to consider the possible different alternative design configurations from a multi-dimensional standpoint, associating with each of them a *Measure of the Effectiveness* (MOE) of a specifically configured vessel in the main scenarios in which it is intended to operate, besides a number of key performance indicators as, for example, the cost.

This kind of analysis is investigated in a research project currently carried on in collaboration with Orizzonte Sistem Navali (OSN), an Italian company with great experience in total naval ship design and integrated support services. The project aims at supporting the design of naval vessels through the development of a system, denoted as Operational Evaluator Model (OEM), providing the evaluation of the operational effectiveness of particular ship configurations for a set of specified operating scenarios [1] [2]. This paper presents the main characteristics of such an operational evaluation support system, specifically focusing on the new simulation framework included in the OEM.

In general, simulation can significantly improve the design phase of naval units and specifically, in the considered context, Agent-Based Simulation (ABS) can play a fundamental role [3] [4] [5] [6] [7]. Therefore, one of the aims of the project consists in exploiting ABS in order to support the naval designers in their exploration of the space of alternative solutions and in the identification of a suitable ship configuration able to satisfy the customer requirements with good performance in terms of both operational effectiveness and costs. It must be underlined that, in the context of this project, the purpose of simulation is neither that of supporting the operational decisions of the commander, nor that of evaluating the effectiveness of operational tactics; differently, here simulation is used to provide ship designers and stakeholders with insights on what could be the expected behaviour of a vessel configured to satisfy certain requirements (e.g., maximum speed, presence of a helicopter on board, kinds of weapons). This means that the simulation specifically focuses on the ship configuration, whereas other aspects that in a real scenario may influence the performance of an operational vessel, such as, the meteo conditions or human factor as the commander skill, are considered fixed or not explicitly modelled.

This paper first presents in Section 2 the architecture and main functionalities of the OEM, as well as the characteristics of the developed ABS framework. In order to compare such framework with other existing ABS environments, a case study is introduced in Section 3 consisting in the so-called Anti-surface Warfare (ASuW). In particular, such a scenario assumes a set of patrol vessels in charge of protecting a strategic facility, e.g., a port located in a gulf, from the possible attack of small fast boats. Note that, in the last years, this scenario

*This work was supported by Orizzonte Sistemi Navali S.p.A.
†DIBRIS Department, University of Genova, Genova, Italy, (davide.anghinolfi@unige.it).
‡DIBRIS Department, University of Genova, Genova, Italy, (alberto.capogrosso@unige.it).
§DIBRIS Department, University of Genova, Genova, Italy, (massimo.paolucci@unige.it).
¶Orizzonte Sistemi Navali S.p.A., Genova, Italy (francesco.perra@orizzontesn.it).

has been usually associated with the defence from terrorist attacks. The patrol vessels use the radar to detect the threats and their actions are coordinated by a command centre on land; the patrol vessels have missiles and a gun to defend themselves from the missiles launched by the small fast boats, and their mission is considered successful if they are able to neutralize all the threats before these latter reach the facility. In Section 4 the implementation of this scenario with the new ABS Framework (in the following denoted as ABSF) included in the OEM is illustrated. In addition, in Section 5 a set of performance indicators are defined to facilitate the comparison of the different simulation environments. Such indicators assign to each of the considered frameworks a rating providing a qualitative or quantitative evaluation, having applied each environment to the same reference naval operational scenario.

The different simulation environments considered in the analysis presented in Section 6 of this paper are the followings:

- *MANA* (Map Aware Non-uniform Automata), developed by the Operational Analysis Section of the New Zealand Defense Technology Agency (DTA), is released for academic research purposes; the tool is used in a number of studies, such as the management of urban warfare, maritime surveillance and coastal patrol [8].
- *NetLogo*, developed at the Center for Connected Learning (CCL) of the Northwestern University (USA) by Uri Wilensky, is an open source visual environment that is particularly suitable for the analysis of complex natural and social phenomena, as well as military tasks. NetLogo provides a simple programming language in order to characterize the behavior of the agents and to specify the evolution of the simulation, and it is equipped with a graphics viewer that allows the users to visualize the evolution of the simulation and to interact with the agents [9].
- *Stage*, developed by Presagis Inc., is a commercial visual environment that allows to develop models of very complex war scenarios in terms of the involved platforms (avionics, naval, land, etc.) as it includes highly accurate models of sensors (radar, sonar, Missile Warning Set, etc.) and weapons (missiles, guns, etc.) [10].
- *MASON*, developed at Evolutionary Computation Laboratory of the George Mason University, is an open source function library developed in Java. MASON is designed to serve as the basis for a wide range of multi-agent simulation tasks, ranging from swarm robotics to machine learning to social complexity environments [11, 12, 13].

A previous version of the comparative analysis of the first three different ABS environments is provided in [14], where a number of limitations of such environments pointed out the need for a new ABS framework for the operational evaluation of military naval tasks. In this paper such an analysis is extended with the inclusion of MASON. It must be observed that NetLogo and MASON are general purpose environments, whereas MANA and Stage have been specifically designed for the simulation of military scenarios. As revealed by the comparative analysis, only NetLogo and MASON appear suitable for appropriate modelling the chosen reference scenario. Therefore, in Section 7 the statistical equivalence of the simulation models implemented using ABSF, NetLogo and MASON is demonstrated. Even if MASON tool overcomes most of the limitations highlighted in [14], in Section 8 the higher quality of the new ABS framework here proposed is confirmed. Finally, Section 9 reports some concluding remarks.

2. The OEM architecture and functionalities. The Operational Evaluation Model (OEM) is a decision support system aiming at the multi-dimensional analysis of alternative ship design configurations, in particular, with the capability of estimating the effectiveness of the considered vessel in a set of operational scenarios, and of taking it into account in the overall evaluation. OEM has been developed in the programming language C# and it includes in an integrated environment a framework for ABS modelling and simulating military naval scenarios. The overall architecture of such a support system is shown in Fig. 2.1. The environment is composed by five main modules: Model Definer, Experiment Designer, Model Simulator, Data Analyser and Decision Support System. The Data Analyser and the Decision Support System modules are not yet implemented in the current version of OEM, but they will be the object of the next developments of the project. The following of this section provides a description of each module.

2.1. Model Definer. The Model Definer (MD) is a visual development tool that allows the definition of simulation models by exploiting an UML-like formalism. In particular, through MD a user, the Simulation

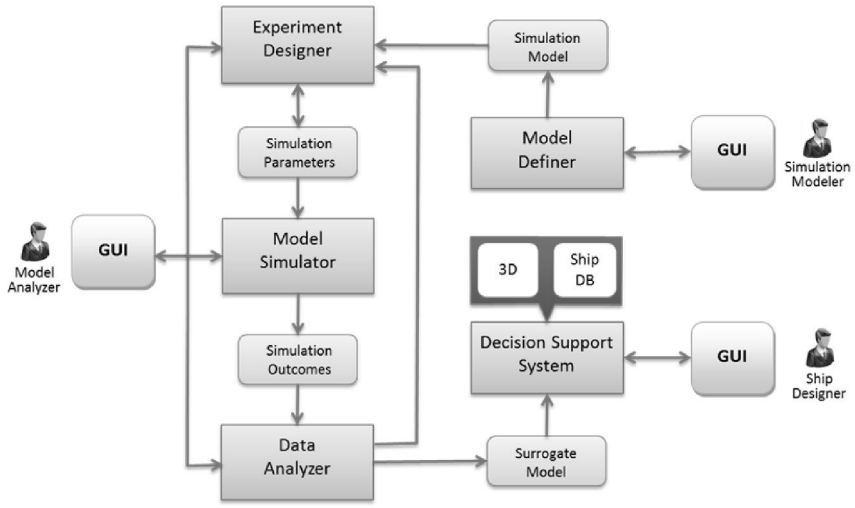

FIG. 2.1. *The overall architecture of the Operational Evaluation Model.*

Modeler in Fig. 2.1, can specify an ABS model by means of the following three types of diagrams:

- *class-diagram*, used to define both *entities*, i.e., the prototypes of the agents that can be included in the simulation models, and the set of *messages* that such agents can generate during the simulations; the user can define a new agent prototype by specifying:
  - the *Unique name* for the prototype;
  - the *Attributes* that represent the *knowledge* of the entity (e.g., the maximum speed for a naval unit);
  - the *States* that represent the operating modes of the entity (e.g., possible states for a naval unit could be Idle, Patrolling, Following and so on);
  - the *Message traps* that determine the *behaviours* to be activated when receiving messages;
  - the *Behaviours* corresponding to a set of activity-diagrams associated with the various states and Message traps;
- *object-diagram*, used to specify the instances of the agents deriving them from the available entities (e.g., having defined a Naval Unit entity, instances of agents as Unit1 or Unit2 with specific characteristics can be derived). To define a new instance of an agent the modeler user must specify:
  - the *Unique name* for the agent;
  - the *Values* for the Attributes included in the referenced agent prototype;
- *activity-diagram*, used to define the behaviour of the agents. An activity diagram consists in the sequence of actions that an agent must perform when it is in a specific state or when it receives a specific message. The ABSF included in the OEM provides a library of generic blocks (e.g., *Change state* or *Send message*) and a library of naval blocks for ship simulations (e.g., *Get radar detected agents* or *Follow route*).

The kind of agent model that the ABSF allows to define is quite general; different classes of agents (e.g., reactive, goal driven or belief-intention-desire agents) can be specified with the above mentioned model elements, provided that an appropriate set of building blocks is used to generate the agents' behaviour. MD generates an output file that can be successively translated in a C# source program and then compiled. In this way the activity of designing and implementing a simulation model is greatly simplified without worsening the computational efficiency of the executable simulation code finally generated.

2.2. Experiment Designer. The Experiment Designer (ED) is a visual tool for the definition of simulation campaigns. In particular, it allows to specify the set of scenario configurations to be tested by exploiting some Design Of Experiments algorithms [15] (e.g, Full Factorial or Nearly Orthogonal Latin Hypercube [16]).

2.3. Model simulator. The Model Simulator (MS) is the core of the environment as it allows the synchronous evolution of the agent behaviours. MS has two different use modes:

- *graphic mode* – the real-time evolution of the simulation is graphically visualized in a window so that the user can observe and validate the behaviour of different agents. The window also includes controls for slowing down or speeding up the simulation, for suspending or resuming it, as well as zooming the graphic simulation area. A log including the fundamental simulation events occurred is also provided;
- *batch mode* – this mode allows to compress the time required to simulate the single experiments not providing any graphical representation of the simulation. The interface requires the specification of a configuration input file, in order to execute multiple experiments for a large number of the ship configurations, and the number of repetitions for each configuration needed to obtain a suitable statistical relevance.

2.4. Data Analyser. The Data Analyser (DA) is the module devoted to execute the statistical analysis of the results collected through the batch simulation mode with the purpose of generating a proper surrogate model. The surrogate model is an analytic function, e.g., a polynomials or a neural network, that returns a value of a MOE as a function of the values of a set of input parameters. The generic structure of the surrogate model is:

$$MOE = f(p_1, ..., p_n) \tag{2.1}$$

where n is the number of parameters. An example of polynomial surrogate model (without cross-parameters) of the m-th order with n parameters is:

$$\begin{aligned} f = a_{11}p_1 + a_{21}p_2 + ... + a_{n1}p_n + a_{12}p_1^2 + a_{22}p2^2 + ... + a_{n2}p_n^2 + \\ + ... + a_{1m}p_1^m + a_{2m}p_2^m + ... + a_{nm}p_n^m \end{aligned} \tag{2.2}$$

where $a_{11}, ..., a_{nm}$ are the unknown coefficients.

To estimate the unknown coefficients, DA implements a set of algorithms e.g. *Ordinary Least Squares* for polynomial structures and *Back Propagation* for neural network structures.

2.5. Decision Support System. The Decision Support System (DSS) module provides a set of tools for multi-criteria decision support in order to perform the analysis of the alternative design configurations for a naval unit. DSS references the list of admissible ships (i.e., corresponding to design configurations that can actually be developed) contained in a Ship Database, and it is able to provide a 3D graphic representation of the vessel corresponding to a selected configuration by invoking an appropriate external 3D visualization module. DSS provides users with a tool for the multi-dimensional comparison of the alternatives, in particular the Pareto-optimal ones, in order to support the selection of the most suitable one for the stakeholders' requirements.

2.6. Graphic User Interface. The first four modules (i.e,. MD, ED, MS, DA) share an unified Graphic User Interface (GUI) that allows the user to define a simulation model, specify the campaign of experiments, simulate the model, validate the simulation outcomes and generate an appropriate surrogate model. Such model is then provided as input to the DSS module that will be equipped with a separate GUI, denoted as Dashboard, that will be developed in the next prosecution of the OEM project. The users of the unified GUI should be expert in the naval domain, i.e., they should know the the details of the tasks to be modelled and simulated. Differently, the Dashboard users play the role of the Ship Designer, that is, they are responsible of selecting one or more vessel configurations that will be considered for the following steps of the new naval unit development.

Fig. 2.2 and Fig. 2.3 illustrate two views of the unified GUI associated with the ED and MS modules. Fig. 2.2 shows the design parameters tab where the user can specify DOE factors, random or constants parameters. In case of DOE factors and random parameters upper and lower limits must be set and the kind of method for the generation of the experimental campaign must be chosen (e.g., full factorial or NOLH). Fig. 2.3 shows the simulation view tab where the user can select the type of simulation (batch or graphic), and the simulation input parameters. For example, in case of graphic simulation the user must specify the configuration number

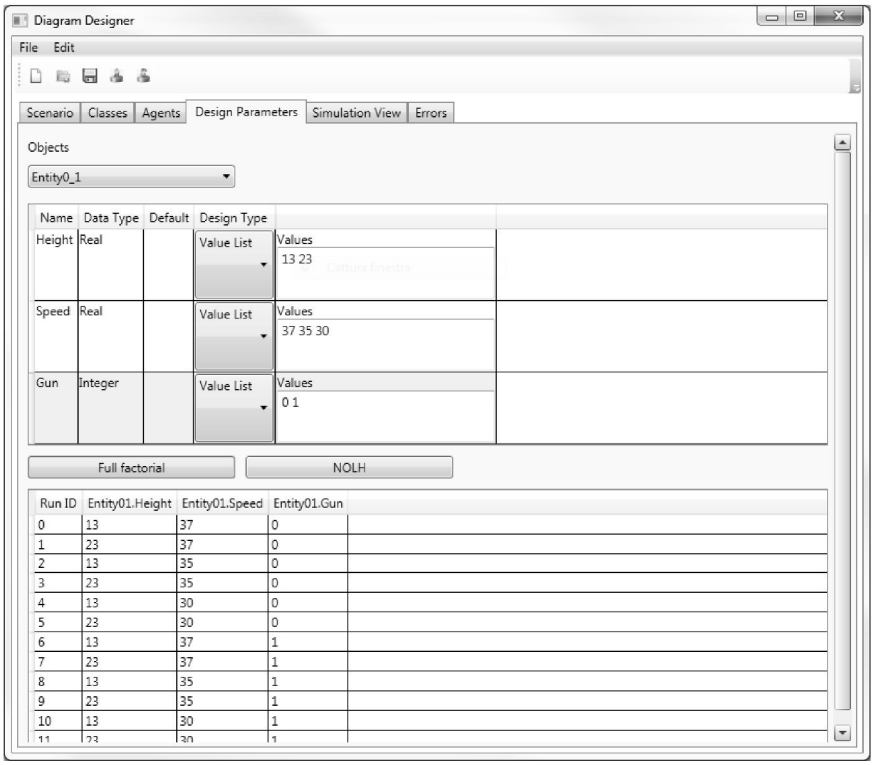

FIG. 2.2. *The design parameter tab of the ED module.*

to simulate, the simulation speed and the dimension of the simulated area (note that the possible geographic coordinates are defined as scenario parameters). The interface allows to change the simulation speed during its execution, as well as pausing or stopping the run.

3. The Anti-Surface Warfare Scenario. In this section the main features of the ASuW scenario used as reference for the ABS framework comparison is introduced (for reader's convenience, the list of the acronyms used for this scenario is provided in Tab. 3.1). The ASuW scenario consists in considering a facility in a gulf (i.e., a port) that must be protected against threats, represented by a group of Small Fast Attack Crafts (SFAC) [17].

TABLE 3.1
ASuW acronyms table.

Acronym	Meaning
ASuW	Anti Surface Warfare
OPV	Offshore Patrol Vessel
SFAC	Small Fast Attack Craft
SAM	Surface-to-Air Missile
SSM	Surface-to-Surface Missile

The port is guarded by a set of Offshore Patrol Vessels (OPV) patrolling the area. The OPVs are equipped with a radar, so that they can detect the presence of the SFACs, and with ASuW weapons used to contrast the threats, which consist in a fixed number of surface-to-surface missiles (SSM) and a gun. The SFACs are assumed to be equipped with four SSM missiles used to attack the OPVs. Specifically, in this scenario it has been assumed that a SFAC launches a SSM against an OPV as soon as it realizes to be engaged or followed by it. On the other hand, the OPVs have three surface-to-air (SAM) missiles that are utilized, together with the gun, to neutralize the attacking missile.

The OPVs are coordinated by an ASuW Commander that influences their behavior. The mission of the

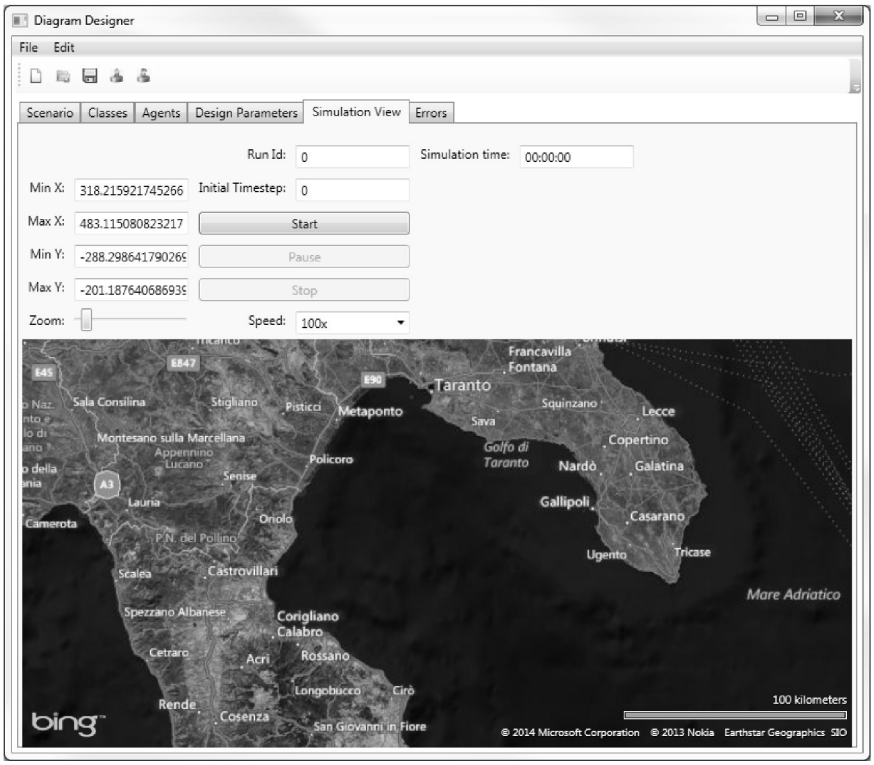

FIG. 2.3. *The simulation view tab of the MS module.*

OPVs is considered successful if the port is defended, i.e., if all the threats are neutralized before reaching their target (the port). In this scenario a helicopter can be included in the OPVs configuration; in this case, the helicopter is assumed to perform the function of radar picket, i.e., it stays on a fixed position with the purpose of improving the threat detection ability of the defence system.

The ASuW scenario specifically considered in this paper includes nine SFACs and three OPVs. The gulf subject to possible attacks is schematically represented as a rectangular area of about 50×100 nautical miles (nm), situated in the Gulf of Taranto (Italy) so that the coasts are not affecting the operations. The scenario is shown in Fig. 3.1 where the horizontal line in the upper (north) side of the area represents the SFAC target.

In Fig. 3.1 the three OPVs patrolling routes (consisting of straight stretches between successive patrol points) are represented, as well as the area in which the SFACs are assumed to be initially located when the simulation starts and the position of the helicopter, if present. The SFAC attack and OPV contrast strategies are summarized as follows:

- the OPVs patrol the area, according to given routes, whereas the helicopter, if present, maintains a fixed position;
- each SFAC moves from its starting position following a vertical trajectory (from south to north) at constant speed until reaching the target line;
- the SFACs start from a random position within the area shown in Fig. 3.1;
- the OPVs and the helicopter, if present, can detect the threats by radar sensor;
- if a threat is detected, its position is communicated to the ASuW Commander who assigns the OPVs to the SFACs. Specifically, the SFACs to be engaged are ordered according to their distance from the target line (*threat evaluation*);
- a list of at most three SFACs is assigned to each OPV; in particular, each SFAC is assigned to the nearest OPV;
- each OPV directs towards the first target in its list; if the list is empty, the OPV continue its patrol

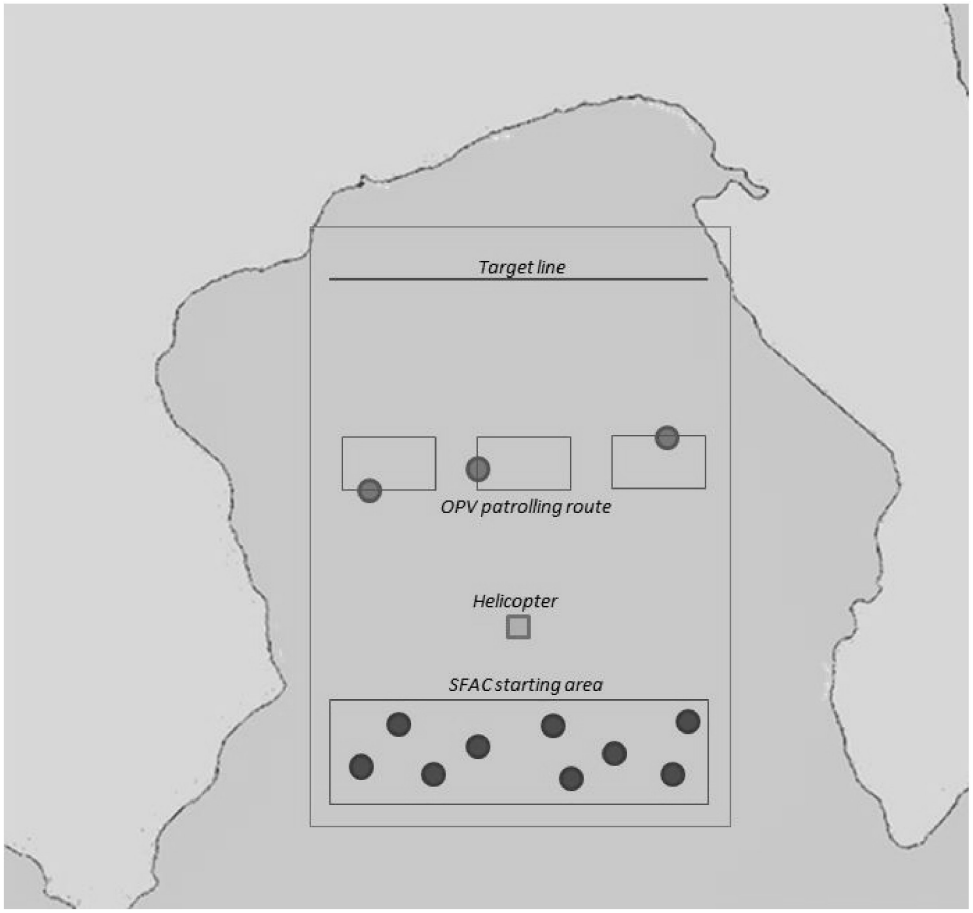

FIG. 3.1. *Geometric representation of the considered ASuW scenario.*

route;

- as soon as a SFAC realizes to be engaged or followed by an OPV, it launches a SSM against the OPV;
- the OPV engages with weapons a SFAC that is within its intercepting range in two cases:
  (a) whenever is attacked by a SSM from the SFAC or
  (b) whenever the SFAC is at sight range (in this latter case OPV can classify the SFAC as a threat).
  The preferred weapon system used by the OPV are the SSM missiles; if the OPV runs out of missiles it engages with the gun. A threat is neutralized with a probability given by a function depending on the kind of weapon and on the distance between the launcher and the target;
- if OPV is attacked by a SSM, it launches one or possibly two SAM to neutralize it. If this defence action fails, OPV tries to contrast the missile with the gun.

The OPV radar range depends on the radar height and on SFAC height. In particular, denoting with h_{OPV} and h_{SFAC} the height respectively of the OPV radar and of the SFAC, the OPV radar range RR is given by

$$RR = 2.22 \times \left(\sqrt{h_{OPV}} + \sqrt{h_{SFAC}}\right) \tag{3.1}$$

Moreover, the turn rate ($\dot{\theta}$) of the OPV (that influences its maneuverability) depends on the OPV length (L) and speed (V), as follows:

$$\dot{\theta} = \frac{V}{R} \tag{3.2}$$

where $R \cong 2L$.

The weapon success probability functions have been provided by Italian Navy officers and they depend on the type of weapon. Fig. 3.2 and Fig. 3.3 plot the success probability function respectively for the gun and the SSM missile.

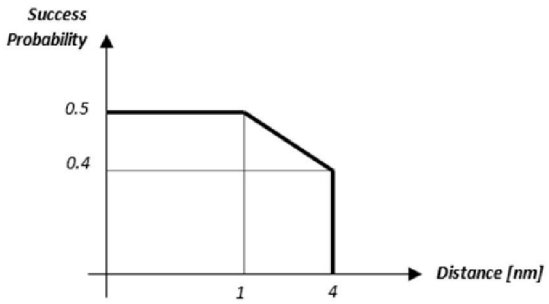

FIG. 3.2. *Success probability function for the gun.*

The probabilities for SAM missiles to destroy an SSM have been assessed by an analytic study of the anti-air warfare scenario. Tab. 3.2 reports a subset of such data, showing the probability that the first SAM destroy the SSM (PK1) and the one for the second SAM (PK2) as a function of the SAM and SSM types, and the OPV radar height (expressed in meter). Note that the value zero for PK2 denotes the impossibility to launch the second SAM in time to contrast the attacking SSM.

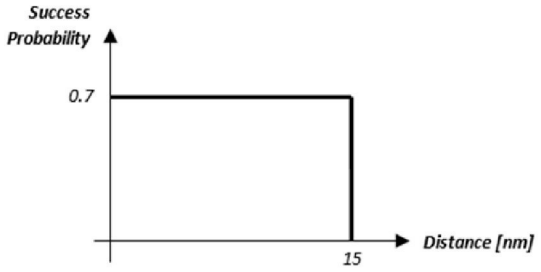

FIG. 3.3. *Success probability function for an SSM missile.*

TABLE 3.2
An excerpt of the success probabilities for SAM contrasting SSM.

SSM	SAM	Radar Height	Pk1	Pk2
1	1	18	84.8	65.9
1	1	19	85.6	66.7
1	2	18	67.2	60.9
1	2	19	66.9	61.8
1	1	18	60.7	0.0
...	...	...	...	...

4. Implementation of the ASuW scenario. This section illustrates how the ASuW scenario can be defined in the OEM system, in particular providing an idea of the kind of interaction of a Simulation Modeler user with the MD module.

The elements of the ASuW scenario that show an autonomous behaviour are modelled as agents. Therefore, in this scenario four classes of agents are introduced: the OPV, the SFAC, the Commander and the (launched) Missile. The missile agents are created during the simulation whenever an OPV or SFAC agent lunches an available missile, whereas the other classes of agents are created during the initialization of the scenario. Note that an element such as the helicopter, that, in a different scenario, could be modelled as agent, here simply plays the role of a sensor providing information to the OPVs and the Commander.

Fig. 4.1 and 4.2 give two examples of views of the unified GUI that are needed by the MD module. Specifically, Fig. 4.1 shows the class diagram tab that allows the definition of the entity and the message traps, whereas Fig. 4.2 shows the activity diagram tab used to define the behaviour of the agents. Any new item in both interfaces can be defined by dragging the item prototype from the toolbox in the left of the window to the workspace and then setting the appropriate properties. The items in the toolbox of the class diagram tab represent the prototypes of classes and messages, whereas the ones in the toolbox of the activity diagram tab correspond to the building blocks available to formalize the agent actions.

The Fig. 4.1 shows the definitions of the classes (*OPV*, *Commander*, *Missile*, *SFAC*) and the messages (*EnemiesDetected*, *FollowEnemies*, *HitByMissile*) for the ASuW scenario as it appears in the class-diagram tab of the MD GUI. The OPV class is characterized by two different states, i.e., *Patrol* and *Follow*, respectively associated with the patrolling behaviour, which makes the OPV reaching in sequence a number of patrolling points, and the following behaviour, which makes the OPV direct towards a target (this behaviour is implemented by the workflow shown in Fig. 4.2). The Commander class has a single state *ProcessEnemies* associated with its threat evaluation and assignment behaviour. The SFAC class has two states, *Idle* and *MoveToTarget*, the first needed to model the possible delay before starting to move towards the target and the second associated with the SFAC mission of reaching the target line. Finally, the Missile class has a state *Follow* associated with the behaviour that moves it towards the target and a *TargetReached* state, assumed when the missile finishes its trajectory hitting or missing the target.

The model includes three kind of messages: the *EnemiesDetected* message, sent by an OPV to the Commander when the OPV detects a SFAC; the *FollowEnemies* message, sent by the Commander to an OPV in order to assign the threat; the *HitByMissile* message, send by a missile to its target when this latter is reached. Whenever a recipient receives a message it invokes the corresponding message trap behaviour. As an example, when an OPV receives a *FollowEnemies* message it starts its *Follow routine*.

For each class of agents at least one behaviour must be modelled. For example, in Fig. 4.2 the behaviour prescribing an agent to follow a given target is shown. The behaviours are modelled using *workflows*. A workflow starts always with a *Start* block and finishes always with a *End* block. Considering the workflow in Fig. 4.2, at each time step the agent:

1. updates its position;
2. calculates the distance to the target and stores this value in a local variable;
3. checks if the distance is lower than a threshold, and in the positive case the agent changes its state to *TargetReached*, otherwise the agent remains in its current state.

After the definition of the classes and the messages, a simulation model can be specified by creating the instances of the agents involved in the simulation. This can be done in the agent tab of the MD GUI, where an Object Diagram can be generated. As an example, Fig. 4.3 shows the agents included in the ASuW scenario that includes a Commander, three OPVs and five SFACs. Note that in this diagram there are no missile agents as these latter are only created at run time. In the agent tab the user must provide the initial values of the attributes for each agent instance (e.g., for OPV_1 PatrolSpeed is set to 16 kn).

5. Performance Indicators. In order to compare the considered simulation environments, both qualitative and quantitative key performance indicators (KPIs) have been defined. The qualitative KPIs aim at defining the ability of the simulation environment to comply with specific characteristics, whereas the quantitative KPIs provide some numerical indexes as detailed in the following.

5.1. Qualitative indicators. The following KPIs are used to evaluate the environment's capacity to implement simulation scenarios with a degree of complexity comparable to the considered ASuW scenario.

1. *Possibility to execute Monte Carlo simulations.* This is possible if the environment allows operating with random variables.

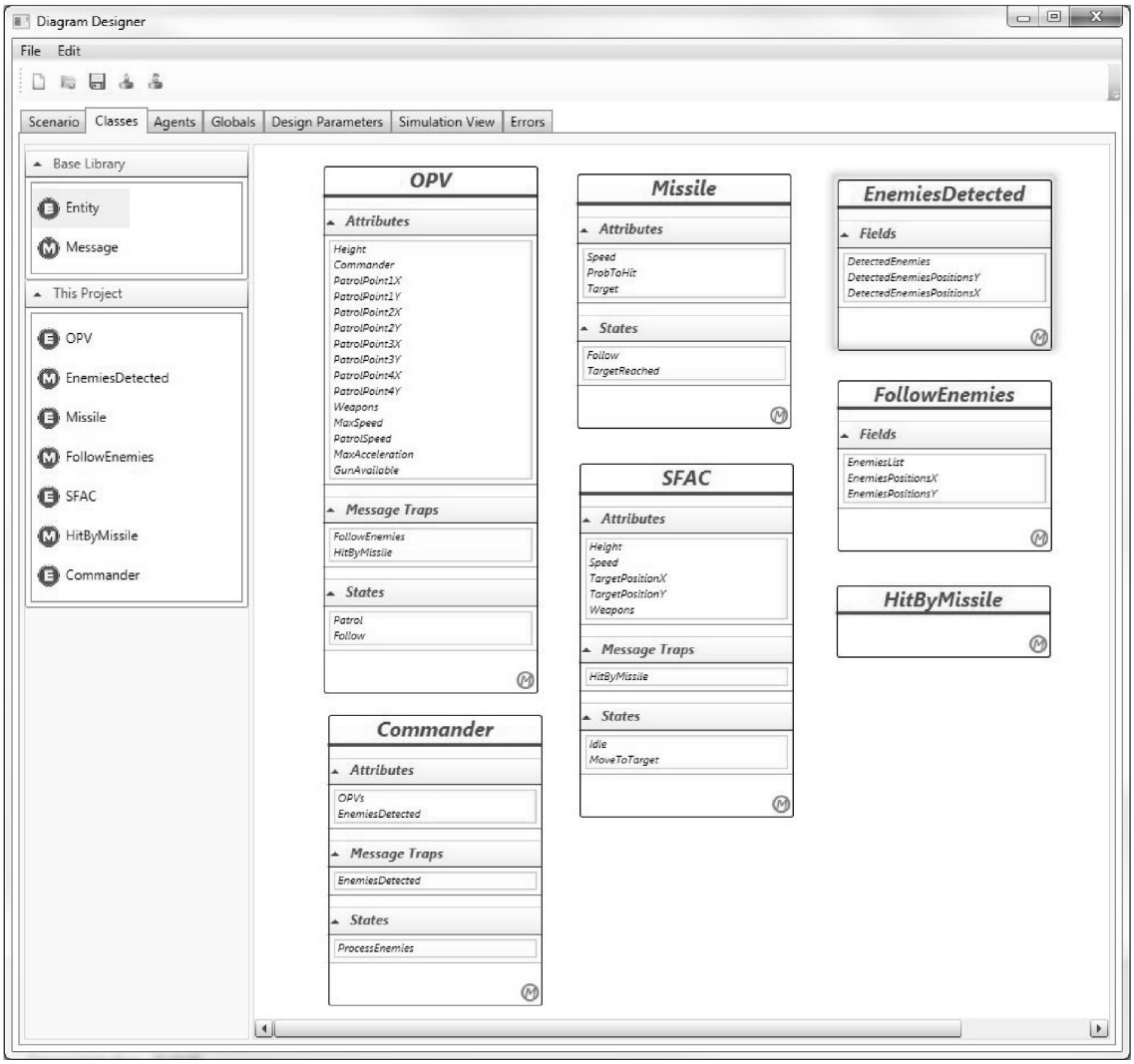

FIG. 4.1. *Definition of classes (OPV, Commander, Missile, SFAC) and messages (EnemiesDetected, FollowEnemies, HitByMissile) involved in the ASuW scenario with MD Module of OEM.*

2. *Possibility of running multiple experiments in batch mode.* This property makes it possible to execute extended experimental campaigns; specifically, the environment must allow reading the parameters characterizing the different experiments from some data structure or file, as well as writing the results in an appropriate way.
3. *Presence of an integrated visual development environment.* Such a tool reduces the complexity of defining new simulation model and the time needed for the development.
4. *Possibility to implement complex agents.* This is a qualitative judgement of what is the level of the complexity in the definition of the agents allowed by the environment, specifically in term of the type of data managed by the agents and of the elaborateness of the agent behaviour. As an example, the index takes into account the possibility of including complex data types (such as lists of structures) in the agent attributes. In the specific case of the ASuW scenario this index is specified considering the ability of the environment to define and manage the list of SFAC assigned to each OPV by the Commander.
5. *Possibility to define complex coordination politics for the agents.* With reference to the ASuW scenario, this index expresses the possibility of defining the ASuW Commander agent able to coordinate the

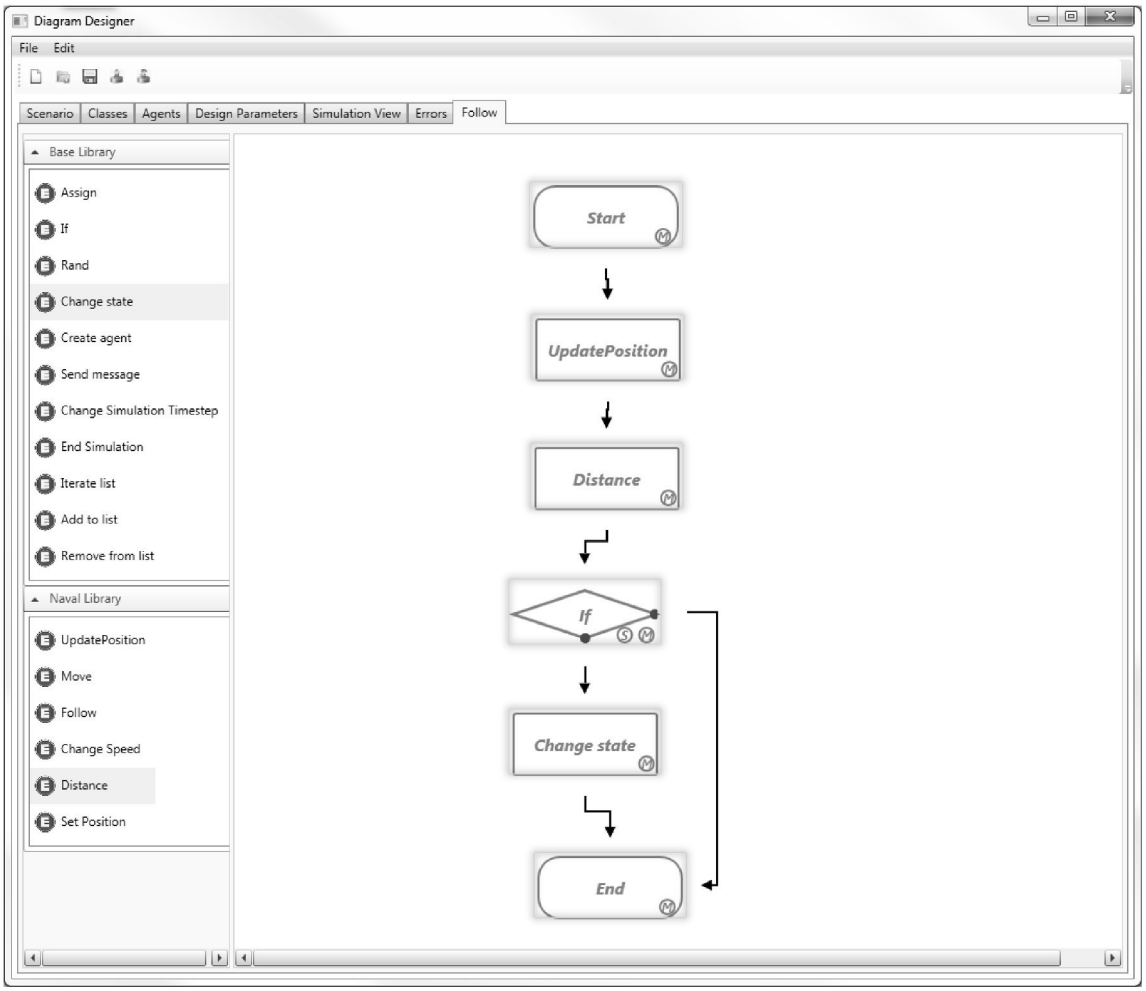

FIG. 4.2. *An example of behaviour definition with the MD Module.*

OPVs actions according to the complex threat evaluation and assignment rules.

6. *Possibility to implement and manage the communication between agents.* In the considered scenario this feature is needed to allow the ASuW commander communicating with the OPVs.
7. *Availability of language primitives of an appropriate level.* In the ASuW it is necessary to use mathematical equations, for example to implement the turn rate and the radar range.

5.2. Quantitative indicators. The following numerical KPIs can be adopted to compare the models developed by means of the different environments.

1. *Number of code lines.* This can be considered an index of the complexity of the development of the models, as well as of development time (note that this index is estimated by considering only the code needed to define agents and their behaviour);
2. *Average time to perform a single run* (in seconds);
3. *Average time to perform 1000 runs* (in seconds);

6. Comparison among MANA, NetLogo, Stage and MASON. This section discusses the features and eventual limitations emerged by comparing the ways in which the ABS frameworks, different from the ABSF included in the OEM, model and simulate the ASuW scenario.

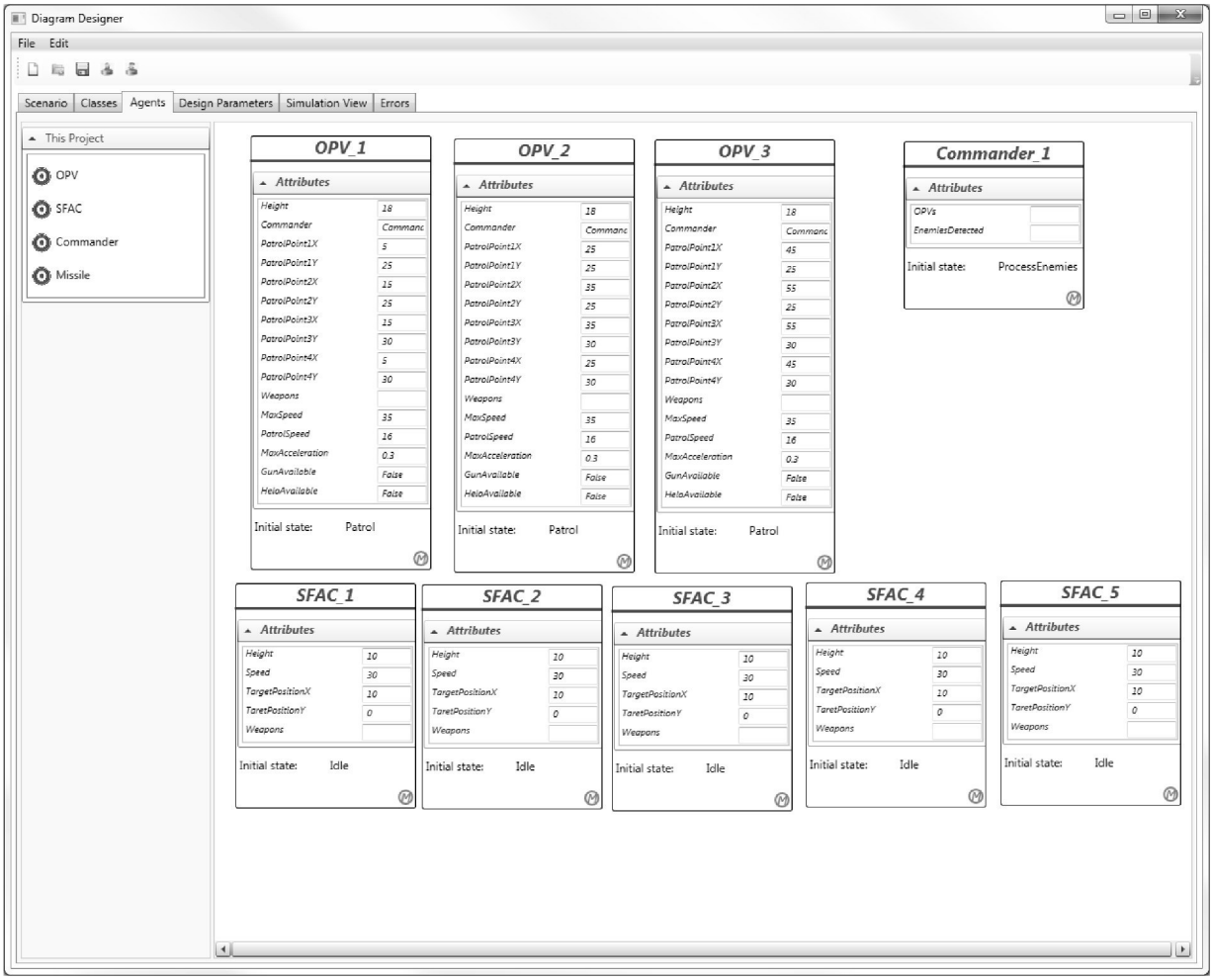

FIG. 4.3. *Definition of the Agents involved in the ASuW scenario with MD Module of OEM*

- MANA - It allows to define the topological characteristics of the scenario, such as the type and size of accessible areas and the presence of hot-points (i.e., starting points, target points and patrol points) [18]. The agent characteristics are defined by drop-down menus and scroll bars; for example, the tendency of the agents to direct towards either the next patrol point or the nearest threat, the speed and the agent icon. Agents can be equipped with armaments and sensors and their associated hit/detect probability can be specified by a data table as a function of distance. The MANA environment allows to execute a sequence of different runs of the scenario. The main limitation of MANA (confirmed also by a personal communication with one of its designers) is that it does not provide a programming language for describing complex agent behaviours. This made impossible to implement the scenario ASuW as described. In particular, the ASuW agent Commander assignment rules, the OPV air-air missiles defence behaviour, and the evaluation of algebraic expressions (i.e., the ones needed for the rate of turn and the radar range) cannot be implemented.
- NetLogo - The scenario is defined by a graphical interface that integrates a development environment and an automated analyser to simplify and speed up the design of the scripts used to specify the agent behaviours. The NetLogo scripting language is a high-level language, interpreted by the provided simulation engine, that includes a large set of functions for configuration, movement and communication. This language allows the management of pseudo-random variables, as well as the input and output files,

so making the execution of multi-run simulation batches possible. An interesting feature is that the graphic interface can be deactivated during the multi-run simulation, making in this way the simulation quite fast. Summing up, NetLogo includes all the features needed to implement the ASuW scenario.

- Stage - It has a graphical environment that allows to create very sophisticated environments. Stage enables creating simulation models with realistic details, as it provides very detailed models of ships, vehicles, aircrafts, sensors and radar. The agent behaviour can be defined by an interpreted script language inspired by the C language. Stage integrates a development environment and an automated script analyser. The agents communicate using a standard communication protocol. In addition, the execution of a simulation can be distributed in a computer network to parallelize the computational burden. Stage definitely allows implementing the ASuW scenario. However, two main limitations were observed that prevent the use of Stage in batch mode: it is quite hard (at least for conventional designers or users) to simulate excluding the graphic output; it is not possible to perform multiple simulations.
- MASON - The scenario and the agent behaviours must be defined by coding in Java language. The framework supplies some features like a very high performance random numbers generator and some high efficiency data-structures. It supplies also some features to implement a customizable graphic interface. As an alternative, the simulation can run in batch mode, i.e. without interface, to speed-up it. In addition, the execution of a simulation can be distributed in a computer network to parallelize the computational burden. Therefore, MASON includes all the features needed to implement the ASuW scenario.

TABLE 6.1
KPI comparison for the simulation environments

	MANA	**NetLogo**	**Stage**	**MASON**
Qualitative Indicators				
Monte Carlo simulations	×	√	√	√
Batch multiple experiments	√	√	×	√
Visual development environment	√	√	√	√
Complex agent definition	×	√	√	√
Complex agent coordination politics	×	√	√	√
Communication among agents	×	√	√	√
Appropriate language primitives	×	√	√	√
Quantitative Indicators				
Source lines of code	–	1000	700	1300
Average time for single run (s)	–	0.6	480	0.015
Average time for 1000 runs (s)	–	754	–	20

Tab. 6.1 summarizes the main features observed (here a cross denotes the lack of the corresponding feature, whereas a dash the impossibility to collect the data). Observing the quantitative KPIs, the number of code's lines needed by Mason to implement the agent behaviours is greater than the one of NetLogo: this indicates that NetLogo provides an higher level language compared to the one provided by Mason. The low number of lines of code required for the implementation of the scenario in Stage is due to the fact that the environment has been developed specifically for the management of military simulations.

Finally, the batch run time results are shown only for NetLogo and MASON, as MANA does not allow the appropriate design for the considered scenario and Stage does not permit the batch execution mode. Comparing the simulation times obtained using Mason and NetLogo, it can be noticed that the first is about 40 times faster than the second. This result is due to the fact that NetLogo uses a scripting language, while Mason uses Java language which is semi-compiled.

7. Equivalence test for the compared scenario models. After modeling the ASuW scenario with ABSF, it was possible to compare it with Mason and NetLogo, as these latter emerged as the only environments able to appropriately simulate such a naval task scenario. However, the comparison is sensible only after having verified that the models developed with the three different frameworks are able to produce equivalent results from the statistical standpoint. Therefore a set of experiments was performed to verify the equivalence of the ABSF, MASON and NetLogo models for the ASuW scenario. In particular, 24 possible OPV configurations

were analyzed and for each of them 1000 repetitions were executed, each of which producing a binary value (representing the success/failure in defending the target) as simulation outcome. The tested configurations correspond to all the combinations of the values for the parameters shown in Tab. 7.1.

TABLE 7.1
The tested OPV configurations

Parameter	**Values**
OPV Height m	{13, 23}
OPV Speed kn	{37, 35, 30}
Helicopter presence	{true, false}
Gun presence	{true, false}

The binary outcomes obtained from the models were divided into 20 groups, each consisting of 50 elements. For each group the relative OPV success frequency was calculated. As an example, Tab. 7.2 shows the success frequencies obtained for the configuration characterized by *OPV Height* = 13m, *OPV Speed* = 37kn, *Helicopter presence* = true, *Gun presence* = true.

TABLE 7.2
The relative success frequency for a tested configuration.

Groups	**ABSF**	**NetLogo**	**MASON**
1	94.1	88.2	74.5
2	88.2	90.2	82.3
3	92.1	86.3	88.2
4	82.3	92.2	90.2
5	94.1	78.4	80.4
6	82.3	86.3	90.2
7	86.3	84.3	84.3
8	92.2	90.2	82.3
9	92.2	66.7	84.3
10	80.4	78.4	98.0
11	92.2	96.1	94.1
12	90.2	86.3	96.1
13	82.3	82.3	90.2
14	82.0	88.0	88.0
15	88.2	92.2	90.2
16	90.2	95.1	84.2
17	88.2	86.3	76.5
18	84.3	92.2	86.3
19	86.3	92.2	78.4
20	81.8	100.0	93.9

Successively, the hypotheses needed to correctly perform the test of analysis of variance (ANOVA) were verified on the obtained data. In particular, the well-known Jarque-Bera test was used to verify the hypothesis of normality of the data series and the Welch test to verify the hypothesis homoscedasticity. Then, the ANOVA test was used to validate the hypothesis of equivalence of the models. Fig. 7.1 shows the box-plots obtained from the ANOVA test. From Fig. 7.1 it can be observed how the three boxes are almost overlapped, so that it can be concluded that the three implemented models can be considered statistically equivalent for the configurations shown in Tab. 7.2. The equivalence of the three models was also confirmed for the other configurations in Tab. 7.1.

8. Comparison between ABSF, NetLogo and MASON. Having established the equivalence of the models produced by ABSF, NetLogo and MASON, the three frameworks were evaluated over the KPIs defined in Section 5. Tab. 8.1 reports the obtained results for the quantitative KPIs, provided that the three considered environments have the same qualitative characteristics.

Observing the number of lines of code it is possible a first conclusion: ABSF does not require any coding to characterize the considered scenario, i.e., the involved agents and their behaviours, but the design of the scenario is performed using the GUI of the MD module.

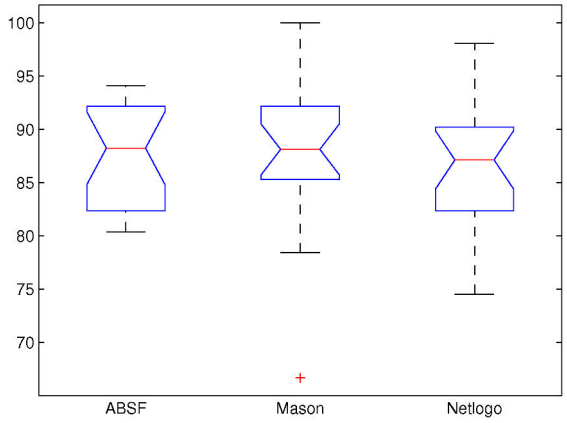

FIG. 7.1. *The box-plot produced by ANOVA*

TABLE 8.1
KPI comparison of ABSF, NetLogo and MASON

Quantitative Indicators	**ABSF**	**NetLogo**	**MASON**
Source lines of code	0	1000	1300
Average time for single run (s)	0.003	0.6	0.015
Average time for 1000 runs (s)	5	754	20

Observing the comparison among the simulation times needed by ABSF, NetLogo and MASON, it can be observed that the average times for 1000 runs obtained by ABSF were about 4 times lower than those obtained by MASON, and 150 times lower than those obtained by NetLogo. It is worth recalling that for the three models the simulation experiments were run on the same computer and they were executed without any graphical interface to minimize the computational burden.

9. Conclusions. This paper presents the main features of a project carried on in collaboration with OSN, an Italian company specialized in total naval ship design, aiming at supporting the early phase of military naval unit design. In particular the OEM support system should enable human designers and stakeholders to evaluate different configurations also from the point of view of the operational effectiveness of the resulting ship in a set of operating scenarios.

This paper devotes particular attention to the agent-based simulation framework included in the OEM and it provides evidence in favour of its suitability as a tool for rapidly modelling military naval scenarios and efficiently simulating them. To this end, the main characteristics and limitations of a set of agent-based simulation environments have been analysed, using as a reference a specific Anti-surface Warfare scenario. As a result of such an analysis, the quality of ABSF as excellent simulation tool was proved, since it is able to overcome the main limitations of the other simulation environments considered and to outperform, in terms of ease of development and run execution speed, MASON, the best one among the competitors.

The high quality computational performance showed by ABSF are actually very important, as it allows to decrease the time needed for exploring the space of the possible configurations of the naval units, so facilitating a deeper alternative analysis. Differently from MASON, the Model Definer module included in the proposed framework lets the user define scenarios without the need to know or learn any programming language, or, as is the case for NetLogo, any scripting languages.

Finally, it can be noted that the use of libraries of generic blocks allows ABSF to be able not only to perform simulations of naval units but also of different contexts. In other words, ABSF, through appropriate verticalizations of the block libraries, can be considered a generic simulation environment.

REFERENCES

[1] M. Tiah, *n Analysis Of Small Navy Tactics Using A Modified Hughes Salvo Model*, thesis, Naval Post Graduate School, 2007.
[2] W. P. Hughes, *Fleet Tactics: Theory and Practice*, Naval Institute Press, Annapolis, MA, 2000.
[3] R. L. Axtell, and J. M. Epstein, *Agent-Based Modeling: understanding our creations*, The Bulletin of the Santa Fe Institute, Winter (1994), pp. 28–32.
[4] K. Decker, and V. Lesser, *Quantitative Modeling of Complex Environments*, Computer Science Department, University of Massachusetts Technical Report (1994) pp. 93–21.
[5] A. Vag, *First generation multi-agent models and their upgrades*, Interdisciplinary Description of Complex Systems, Vol. 2, No. 1, 2004, pp. 95–103.
[6] B. Heath, R. Hill and F. Ciarallo, *A Survey of Agent-Based Modeling Practices*, Journal of Artificial Societies and Social Simulation, Vol. 12, No. 4, 2009.
[7] M. T. Cioppa, and W L. Lucas, *Military Applications Of Agent-Based Simulation*, Proc. Winter Simulation Conference, 2004.
[8] M. K. Lauren and R. T. Stephen, *Map-aware Nonuniform Automata (MANA) a New Zealand Approach to Scenario Modelling*, Journal of Battlefield Technology, Vol. 5, No. 1, 2009.
[9] U. Wilensky, *NetLogo* `http://ccl.northwestern.edu/netlogo/`, (Accessed: July 30, 2013).
[10] Presagis Inc., *Presagis Stage*, `http://www.presagis.com/products_services/products/modeling-simulation/simulation/stage/`, (Accessed: July 30, 2013).
[11] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, *A Multi-Agent Simulation Environment*, Simulation: Transactions of the society for Modeling and Simulation International, Vol. 82, No. 7, 2005, pp. 517–527.
[12] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan, *A New Multi-Agent Simulation Toolkit*, Proc. SwarmFest Workshop, 2004.
[13] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, *MASON: A Java Multi-Agent Simulation Library*, Proc. Agent 2003 Conference, 2003.
[14] D. Anghinolfi, A. Capogrosso, M. Paolucci and F. Perra, *An agent-based simulator for the evaluation of the measurement of effectiveness in the military naval tasks*, Proc. 17th International Conference System Theory, Control and Computing (ICSTCC), 2013, pp. 733–738.
[15] D. C. Montgomery, *Design and analysis of experiments*, Wiley, Hoboken, NJ, Year 2009.
[16] D. Bingham, R.R. Sitter and B. Tang, *Orthogonal and nearly orthogonal designs for computer experiments*, Biometrika, Vol. 96, No. 1, 2006, pp. 51–65.
[17] N. E. Wissel, *Surface Combatant Readiness To Confront A Sea Control Navy*, thesis, Naval Post Graduate School, 2008.
[18] A. W. Gill, *Improvement to the Movement Algorithm in the MANA AgentBased Distillation*, Journal of Battlefield Technology, Vol. 7, No. 2, 2004, pp. 19–22.

Edited by: Costin Bădică
Received: April 24, 2014
Accepted: Sept 21, 2014

DOI 10.12694/scpe.v15i3.1017
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. 217–230. http://www.scpe.org

FORMAL MODELLING AND SIMULATION OF A MULTI-AGENT NANO-ROBOTIC DRUG DELIVERY SYSTEM

MARINA NTIKA*, PETROS KEFALAS†, AND IOANNA STAMATOPOULOU‡

Abstract. Targeted drug delivery with the use of nanorobots, a yet mostly theoretical but very promising future concept, is anticipated to become a significant ally in cancer treatment. The way that nanorobot systems are currently envisaged by researchers is such that they exhibit autonomous and collaborative behaviour that can be uniquely captured by multi-agent systems. In this paper, we investigate this hypothesis by describing the process of formally modelling a simple agent-based system for a simulation of targeted drug delivery. We propose a system comprising different types of nanorobots, and evaluate the effects of various parameters on the final outcome. The data that were retrieved from the corresponding simulation runs, are in support of our hypothesis, demonstrating that nanorobotic drug delivery systems can be effectively simulated by utilising intelligent agent technology.

Key words: multi-agent system, simulation, drug delivery, nanorobots, formal modelling

AMS subject classifications. 68T42, 68U20

1. Introduction. Multi-Agent Systems have established their applicability in studying complex systems that involve a number of heterogeneous resources working collaboratively towards solving a common problem, despite the fact that each individual might have partial information about the problem and limited capabilities. As stated by Brooks in [4, 3], Multi-Agent Systems (MAS) may potentially exhibit "*emergent intelligence*" commonly found in biologically inspired systems.

In parallel, Nanotechnology, currently one of the most important and fast growing areas in modern science, focuses on manipulating matter with dimensions similar to the ones of biological molecules. Many scientific fields, such as Physics, Material Sciences, Chemistry, Chemical Engineering, Microbiology and Genetics, are currently enjoying the fruits of nanotechnological research [10], which has already led to various commercially available applications, but what appears to be amongst the most promising endeavours is the development of nanotechnological constructs targeted for medical use. One quite attractive but also challenging such application is the targeted drug delivery, which is anticipated to be a revolutionary strategy for overcoming serious barriers in health impairments, and nowadays, with the rapidly increasing nanotechnology advances, is regarded as one of the most promising research topics worldwide.

Combining the aforementioned research areas, our contribution focuses on the formal modelling and development of a MAS simulation of nanorobots that deliver an appropriate pharmaceutical substance to unhealthy human body tissue. Inspired by the fact that *in silico* experimentation has established its importance to the research community over the past decade, with mathematical models, algorithms and computer simulations having proven to be valuable allies in studying various and diverse problems in the fields of Biology and Medical Sciences [31, 24], we opted to simulate a simple targeted drug delivery system.

The paper is organized as follows: Section 2, briefly discusses the current status of nanotechnology in medicine and the way computer simulations are employed in medical nanorobotics, with special focus on Multi-Agent System simulations. In Sections 3 and 5 our proposed system is presented and the formal modelling, simulation design and implementation processes are described. Section 6 includes the experimental analysis and the results of the simulation runs. In Section 7 the contemporary approaches in manufacturing robots in the microscale are presented, and finally, we conclude with a short discussion for further work in Section 8.

*South-East European Research Centre (SEERC), Research Centre of the International Faculty of Sheffield, CITY College, 24 Proxenou Koromila Str. 546 22, Thessaloniki, Greece(mantika@seerc.org).

†The University of Sheffield International Faculty, CITY College, 13 Tsimiski Street, 546 24 Thessaloniki, Greece(kefalas@city.academic.gr)

‡The University of Sheffield International Faculty, CITY College, 3 Leontos Sofou, 546 26 Thessaloniki, Greece(istamatopoulou@city.academic.gr)

2. Background and Similar Work.

2.1. Nanotechnology in a Nutshell. The prefix "nano-" means a billionth, hence, when it is added in front of the word "meter" it denotes a factor of 10^{-9}, resulting in dimensions comparable to atomic diameters. Consequently, Nanoscience studies the properties of objects at the nanoscale, and Nanotechnology focuses on manipulating matter in structures with at least one dimension sized from 1 to 100 nanometers. Both fields have a multidisciplinary core, involving researchers of diverse backgrounds and with variant skills. Current advances in the domain have been receiving much attention for the past two decades, both from the Academia and the Industry, largely due to the fact that nanostructures exhibit unique properties and characteristics [29, 26, 53], profoundly different than the ones that have been observed at the macroscale.

A plethora of applications in a wide range of fields are currently available, spanning from flexible digital screens to self-cleaning surfaces, from nanoparticle comprised catalysts [48] to carbon nanotubes acting as chemical sensors [51] etc. However, the potential of nanotechnological constructs targeted for medical use is of special interest.

Nanoparticles, being considered as the building blocks for nanotechnology, are of great scientific interest, due to the unique properties they often possess. Those properties are mostly attributed to the far larger surface area of nanostructured materials compared to masses built from larger-scale blocks. From a biomedical viewpoint, their size is what makes them a very attractive option for a variety of applications, among which drug-delivery and biological and chemical sensing being the most predominant ones. When used for drug-delivery systems, the nanoparticles carry a payload that consists of an appropriate pharmaceutical substance or other types of payload, i.e. proteins, genes etc. This payload is encapsulated in a biocompatible and possibly biodegradable coating. Due to their small size, nanoparticles can evade immune system detection and cross the blood-brain barrier gaining access to the brain. Nanoparticles suitable for drug-delivery, such as dendrimers, nanocrystals, polymeric micelles, lipid nanoparticles and liposomes [15] are already being manufactured. Those nanostructures exploit their inherent biological characteristics, and are based on molecular and chemical interactions to achieve the target-specific delivery or their biochemical sensing. In in-vitro experiments [25, 43], the target recognition is performed by administering these nano-vehicles into a cell-culture or some other suitable substrate, whereas in in-vivo [27, 16] experiments the nanoparticles are injected into the area of interest, either the bloodstream or at specific animal body sites. The nanoparticles can only recognize their target and bind to it when they randomly come in contact with it. Additionally, the process of releasing their payload lies purely on the rate by which their protective surface biodegrades.

Notwithstanding the fact that research in this domain is still in its infants, scientists already envision a far more effective schema, where the nanodevices will attain additional properties, such as communication and navigation. These nanodevices can be also characterized with the term "nanorobot".

2.2. Simulation in Medical Nanorobotics. Although a plethora of nanotechnological applications are already commercially available [34], what is considered as a fundamentally important sector, human health, is yet to enjoy substantial benefits. This is partially due to the very nature of medical research, which requires lengthy periods of time, usually at least a decade, and adherence to strict protocols, before any new developments are encompassed. Under these circumstances, computer simulations offer a great advantage towards accelerating both the basic and applied research processes.

Computer simulations can offer an advantageous insight into medical nanorobotics. On one hand, appropriately designed simulations provide a method for thoroughly investigating the properties and dynamics of virtually any conceptual system. Data can be collected and possible behavioural patterns may be discovered, even before the physical elements comprising the system are technologically available. On the other hand, such simulations can produce valuable information on the way those physical elements must be built.

Recognising the beneficial implications of biomedical computer simulations, a number of studies have been published during the last decade. Unmat et al. [52] discuss the use of Molecular Dynamics, Molecular Kinematic and Monte-Carlo simulations to assist in predicting the performance of bio-nanocomponents on a molecular level, such as energy and force calculations, calculating all feasible conformations of a biomolecule etc. Sharma et al. [46] discuss the potential of combining Molecular Dynamics simulations with Virtual Reality to portray more accurately complex molecular structure models, that include unique nano-properties such as interatomic forces, electrostatic fields around molecules etc.

Going further to study the possible coordination and control of a swarm of nanorobots that are inserted in body tissue, Calcavanti et al. have developed the Nanorobot Control Design (NCD) simulator [5]. In [5], the NCD simulator is used to investigate a fractured coronary artery. The environment of the fractured artery was initially simulated using FLUENT, a finite-volume based Computational Fluid Dynamics (CFD) package [14]. The simulation in FLUENT produces a set of output data, such as the temperature heterogeneity that is known to exist over inflamed plaque surfaces. Those data are subsequently fed as input to NCD, along with several other parameters produced by FLUENT, i.e. blood flow velocities in different parts of the arteries, cytokines and adhesion molecule concentrations that are of interest etc. NCD then produces the visual representation of the nanorobots operating environment. The authors demonstrate the applicability of NCD by using it to determine the optimal values of trigger events that would cause a nanorobot to perform its pre-defined task. The nanorobots, as envisioned by the authors, comprise molecular sorting rotors and robotic arms, biocompatible exteriors and chemotactic sensors for distinguishing between different molecules and detecting possible obstacles in their route.

NCD is used to further simulate other nanorobotic systems for medical applications, such as the potential benefits that could rise by using nanorobots in laparoscopic cancer surgery [8]. Nowadays, the use of sophisticated robotics machinery in laparoscopic cancer surgeries is widespread, with the daVinci surgical system [17] being the cutting-edge technology used. DaVinci comprises a surgeon console, a patient-side cart and an image-processing or insufflation stack. It is proposed by the authors to add nanorobots as a component, in order to achieve advanced high-precision mapping of cancerous areas that the surgeon must dissect. CMOS manufacturing technology is suggested, with addition of chemical sensors to detect chemical changes. The authors' simulation focuses on the chemical signal detection capabilities of 50 nanorobots inside a vessel, and on the manner that the vessel size and the relative position of the nanorobots to the vessel walls can affect measurements. Additional NCD simulations may be found in [6, 7].

Following this brief discussion on the impact that computer simulations may have on refining the characteristics of future nanorobots, it is purposeful to investigate what type of model is appropriate for capturing the dynamics of such a system.

2.3. Nanorobots as a Multi-Agent System. The characteristics that future nanorobots for medical applications should possess are described in [38], to include biocompatibility and power to function, communication and navigation. A variety of nanorobotic design and control required functionalities is discussed in [52], including amongst them swarm behaviour and bio-nanointelligence. The authors attribute the capability of making decisions to the latter term. It can be therefore stated that, overall, nanorobot coordination is of paramount importance, as nanorobots will need to coordinate their actions in a decentralized manner, to behave cooperatively, to be programmable and able to process information, all combined towards achieving a common goal.

Setting aside the issue of biocompatibility, which falls mainly under the scope of biological research, the remaining features, namely power supply, communication, navigation and coordination, have to be considered when selecting the software approach that would be suitable for modelling a system comprising nanorobots. Such a system can be conceptualized as a number of autonomous entities, each having own individual features and properties, which work collaboratively towards achieving a common goal. Following this line of reasoning, a multi-agent system appears to be an appropriate approach that captures this type of collaborative behaviour.

Holland and Melhuish [19] focused on investigating minimal agents' systems, by simulating agents with minimal sensing, mobility and computational capabilities either homing a static beacon or following a moving one. Several algorithms are implemented and compared, with or without agent interaction, in an effort to study the possible collective behaviour of micro- and nano-robots. One of the interesting findings in this work is the capability of secondary swarms in tracing the attraction source. Towards that direction, Melhuish and Welsby in [32], specifically studied the impact of secondary swarming by implementing some of the algorithms from [19] in real blimp robots that were carrying 96g of Helium as a payload, and found significantly increased performance in homing a static beacon when incorporating collective behaviour compared to employing purely individualistic strategies.

A multi-agent system simulation is used in [41] that includes an indirect communication model (i.e. agents with no direct communication but able to communicate with a base station), and to investigate its effect on the

efficiency of locating a target within a complex and dynamically changing environment, comprising biological cells that deform over time. A multi-agent simulation is designed for cancer monitoring using nanorobots [42]. Using NetLogo [54] three simple bio-inspired strategies of nanorobot target finding in complex fluid environments were implemented and results are presented by comparing the average times of locating the target. The results of the ran experiments indicated that all strategies were effective in locating the target within the pre-set time limit, with the "wandering" strategy, i.e. aimlessly moving agent, adjusting direction when it senses the target, producing the best finding times.

3. A Nanorobotic Targeted Drug Delivery System. In [37] we presented a simulation of a number of nanorobots with chemical sensing and communication capabilities, that searched, targeted and eventually destroyed a tumour. In particular, our scenario's environment includes a number of cancerous cells forming a tumour within a human body. The tumour is located in close proximity to some vascular supply and its approximate location is assumed to be known via existing imaging techniques, therefore an appropriate injection site near the area of interest is presumably selected. A number of nanorobots are subsequently injected to the bloodstream with the purpose of precisely locating the tumour, and then delivering their pharmaceutical payload so as to destroy the tumour cells.

For our simulation, we opted on simulating three different types of simple nanorobots, each group having one main capability. Namely, some "tracking" nanorobots, bearing appropriate chemotactic sensors that can trace the gradient of some chemical marker, some "barrier degrading" nanorobots carrying payload that can help penetrate the vessel wall, and some "drug delivery" nanorobots carrying payload that can destroy the cancerous cells. Our simulated nanorobots are presumed to have limited communication capabilities, that is they are able to send and receive messages and are also limited in energy. Overall, this suggested scenario incorporates the four key features of future nanorobots i.e. power to function, communication, navigation and coordination. A schematic representation of the scenario is shown in Fig. 3.1.

4. Modelling and Design. In order to map the entities described in section 3, the following roles were attributed to three different types of agents: *Tracking*, *Barrier Degrading* and *Drug Delivery*. We used finite state machines extended with memory (X-Machines) [22] to formally design the agents behaviour and plans. The following sections describe the process of formal modelling for our proposed system.

4.1. Formal Modelling with X-Machines. An *X-machine* ($\mathcal{X}$) is defined as: $\mathcal{X} = (\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$ [20], where:

- Σ and Γ are the input and output alphabets.
- Q is a finite set of states.
- M is a (possibly) infinite set called memory.
- Φ is a set of partial functions φ; each such function maps an input, a memory value and an emotional state to an output and a possibly different memory value, $\varphi : \Sigma \times M \times E \to \Gamma \times M$.
- F is the next state partial function, $F : Q \times \Phi \to Q$, which given a state and a function from the type Φ determines the next state. F is often referred to as a state transition diagram.
- q_0 and m_0 are the initial state and initial memory.

X-machines are state-based machines extended with a memory structure. This particular feature makes modelling more intuitive and leads towards refinement of the model and eventually implementation. In contrast to memory-less state machines, the transitions between states are not triggered by inputs alone, but by functions that accept an input and the memory values and produce an output and new memory values. Again, this leads nicely towards the final implementation.

There are significant advantages by applying formal modelling mainly concerned with verification and testing. In particular, X-machines have significant advantages over other methods because of their strong legacy of theory and practice in: modelling potential for dynamically structured MAS [47], refinement, animation and simulation [40], testing methods that prove correctness [20] with tools for automatic test generation [11], and model checking for verification of properties [13]. It has been demonstrated that X-Machines and its extensions are particularly useful for modelling biological and biology-inspired MAS [23].

4.2. Simulation Design. *Tracking nanorobots (T-bots)* are designed as reactive agents, that are initialized in a state of moving randomly. They are able to sense their environment, and more particularly to accept as

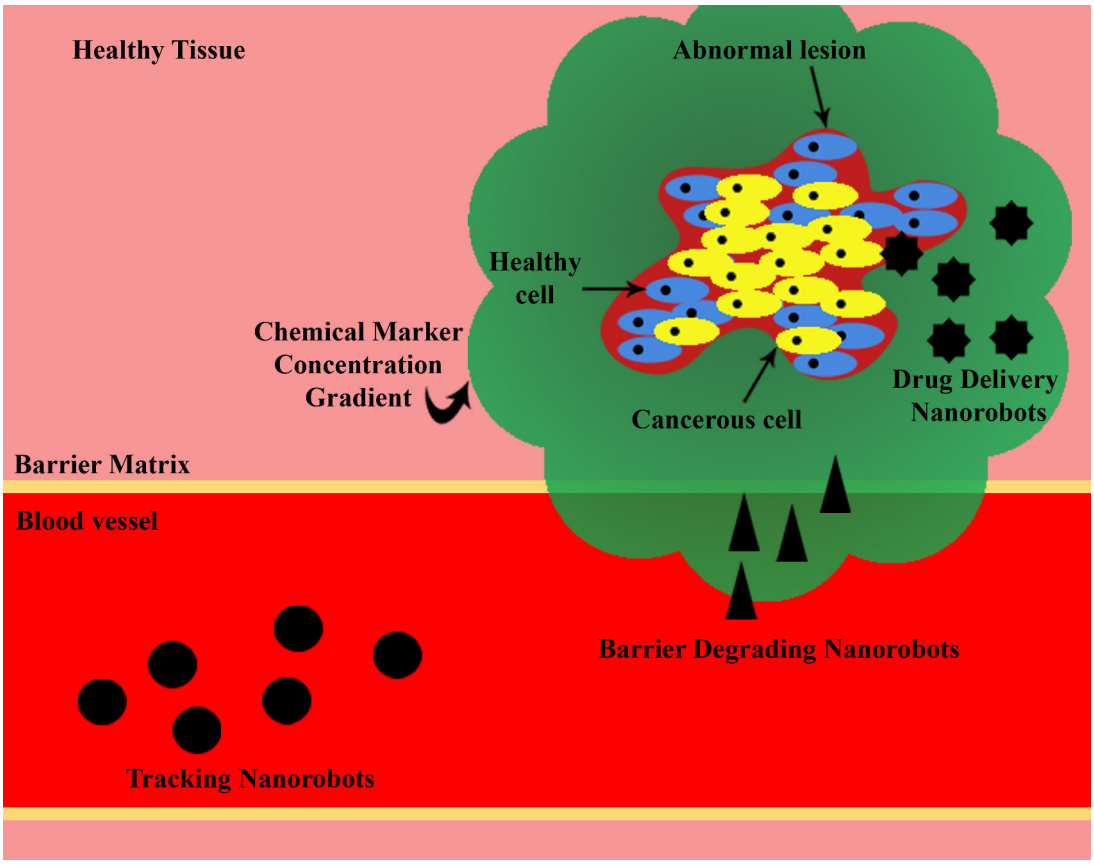

Fig. 3.1: Schematic representation of the targeted drug delivery scenario

input their current location and the protein levels in their surroundings. Once the protein level detected exceeds a predefined value considered as *Threshold1*, they send a message to a random *Barrier Degrading nanorobots (B-bots)* in the area, transmitting the corresponding coordinates. They continue moving, until they sense the second protein level value, considered as *Threshold2*, and then they broadcast a message to all *Drug Delivery nanorobots (D-bots)* in the area. From initialization and onwards, the *T-bots* are considered to spend a portion of their initial energy with their every action, namely moving and transmitting messages. Once their initial energy has ran out, the agents die and are removed from the system. The corresponding X-Machine model is depicted in Fig. 4.1. The diagram is an illustration of F (the next state partial function).

The rest of the formal definition falls outside the scope of this paper. However, briefly:

- $\Sigma = \{empty\_space : (\mathbb{R}, \mathbb{R}), protein\_level : \mathbb{R}\}$ is the input alphabet;
- Γ is the output alphabet (any actions performed onto the environment and any messages sent to other agents);
- $Q = \{moving, protein\_detected, dead\}$ is the set of states.
- $M = ((x, y), Energy, SensingRadius, ProteinTheshold1, ProteinThreshold2)$ is the memory holding the position of the agent, its energy which degrades over time, the radius in which the protein can be detected and the protein thresholds which trigger the message passing.
- $\Phi = \{move - randomly, energy - exchausted, detect - protein, continue - searching, inform - B - bots, inform - D - bots\}$ is the set of functions φ; for instance the function $energy - exhausted$ is defined as: $energy - exhausted(empty\_space, ((x, y), 0, r, bs, ds, t1, t2)) \rightarrow (\text{"}outoforder\text{"}, ((x, y), 0, r, bs, ds, t1, t2)))$. Similarly, the rest of the functions are de-

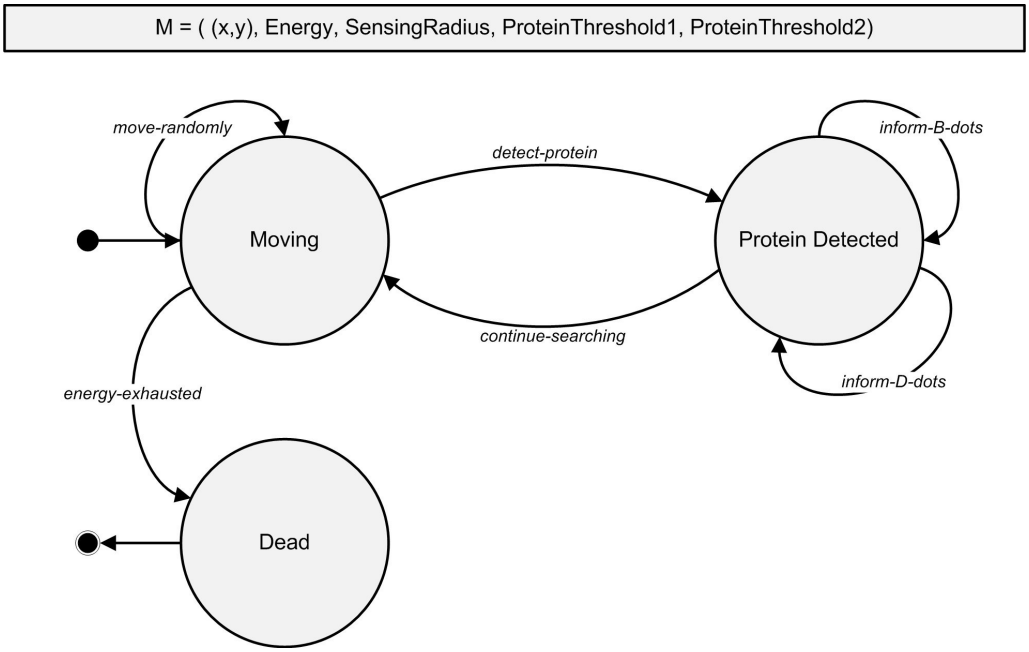

Fig. 4.1: X-Machine model of *Tracking nanorobots (T-bots)*

fined.

- $q_0 = moving$ and an example of initial memory might be $m_0 = ((4,8), 15, 2, b1, b2, b3,$ $d6, d7, d8, 0.14, 0.17)$ are the initial state and initial memory.

B-bots are initialized as idle, and awaiting for a message from the *T-bots*. Upon receiving this message, they start moving towards the coordinates that are included in the message they received. Their target location is considered to be near a vessel, since tumours need blood supply to grow and in fact stimulate the growth of new vessels towards them by producing chemical signals, a process known as angiogenesis. Once the *B-bots* reach their target location they move around in a pre-defined radius, searching for a vessel cell. When they detect it, they release their payload and degrade it. Upon doing so, they also broadcast a message to the *D-bots*, transmitting their current location that implies that a pathway has been created for the *D-bots* to pass through and head to the tumour. Similarly with *T-bots*, the *B-bots* are assumed to have an initial energy that declines over time with their actions, and once it is exhausted, the agent dies. The corresponding X-Machine model is shown in Fig. 4.2.

Finally, the *D-bots* are initialized idle and waiting for a message which is initially received by the *B-bots*. Upon receiving the coordinates, the agents move towards the target. By then, the *T-bots* have also passed through the pathway in the vessel and have accessed the tumour, which means that they have also sent messages with the coordinates where the higher protein level has been detected. Therefore, the *D-bots* receive a number of target locations, and once again start moving towards them. Their architecture is hybrid, comprising a reactive layer, meaning that if upon moving they come into proximity of a cancerous cell, they release their payload and destroy it. There is also a BDI-type layer, which involves stacking the coordinate data and creating respective goals of reaching the target cell and destroying the cell in location. The corresponding X-machine model is shown in Fig. 4.3.

4.3. Design Constraints and Limitations. Biological systems are inherently complex, and any attempt to model them is accompanied by applying several layers of abstraction. One of the major simplifications of our model is that it does not take into consideration the real environment of the nanorobots in the bloodstream. In reality, the blood itself consists of blood-cells, free macromolecules, etc., elements that are in fact considered to be moving obstacles that the nanorobots should avoid. Additionally, when focusing on the nanoscale, electric

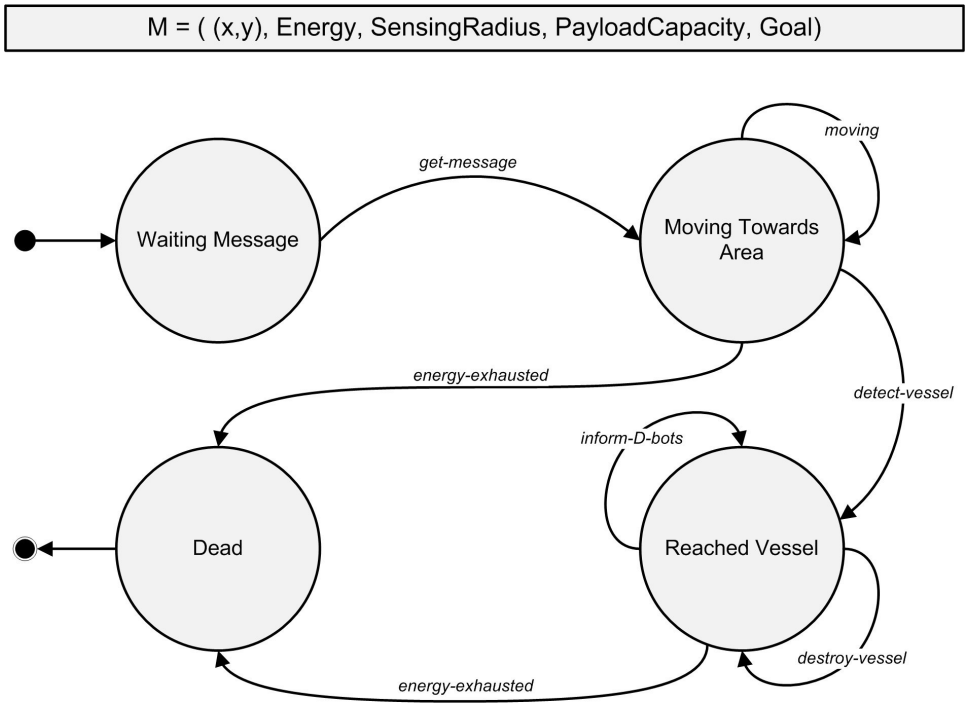

Fig. 4.2: X-Machine model of *Barrier Degrading nanorobots (B-bots)*

forces are of paramount importance. Finally, when moving in the blood, the nanorobots' movement will be affected greatly by the blood flow.

Furthermore, the physical quantities that are involved in our design, such as energy, time, payload capacity etc. are in our simulation treated as unit-less entities. If such a system was ever to be implemented, all these parameters should be tuned to correspond to some real measurable quantities. This includes any constraints that might be attributed to the limited resources that real nanorobots might have once developed.

Overall, it is important to stress out that our simulation does not attempt to mimic a real nanorobotic drug delivery system, since nanorobots are still theoretical constructs, and there is no way to predict their individual characteristics in a low-detail level. However, our work focuses on demonstrating the applicability of MAS simulations to model a drug delivery system and in doing so, takes into consideration only a limited number of parameters and functionalities.

5. Simulation of our model. The rapid growth of Agent Based Modelling is accompanied by an increase in the number of software platforms that are utilized for building agent based simulations [2]. Some of the most popular ones are Swarm [33], Repast [36], MASON [28], FLAME [9] and NetLogo [54], the latter being our selected choice amongst the rest of candidates because it is simple and powerful enough to meet our aims. Our choice would have been different if the simulation was meant to increase in scale and functionality.

NetLogo is both a programming language and a programmable modelling environment for simulating multi-agent systems, with a built-in graphical user interface for immediate visualization of the simulation. Language-wise, NetLogo supports agents with a rather extensive vocabulary of built-in primitives. Our criteria for selecting NetLogo as the development framework include the following:

- Given that the vast majority of our system's functional requirements involve spatial movement, the most fundamental consideration was to utilize a platform that would natively support spatial agent characteristics.
- Based on the user requirements, manipulating the model's parameters and collecting the simulation results should be accomplished in a user-friendly manner.

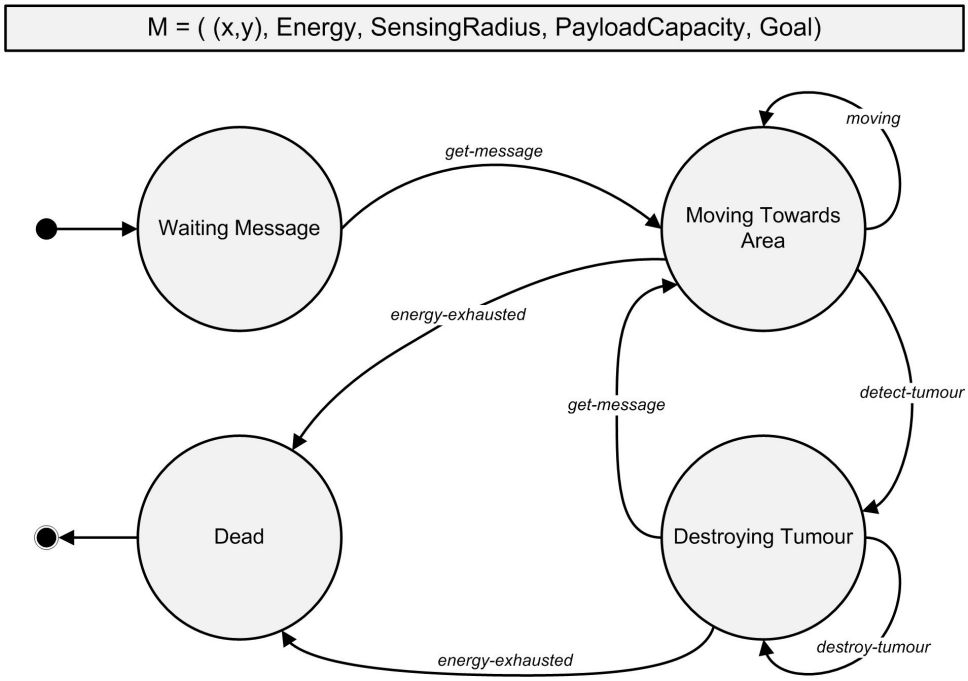

Fig. 4.3: X-Machine model of *Drug Delivery nanorobots (D-bots)*

- Availability of runtime visualization was desirable, since this would provide the additional benefit of observing behaviours that are not necessarily reflected on the numerical data collection.
- Agent communication should also be supported, as well as the implementation of both reactive and proactive agent architectures.

NetLogo is a platform primarily intended for constructing social and natural phenomena models, it thus meets the first three criteria but does not support neither agent communication nor proactive architectures [45]. Two NetLogo libraries [44] were used, one that includes procedures and reporters tailored to implement message passing, and thus agent communication may be achieved and one for implementing some form of proactive behaviour. The former library includes primitives for creating, sending and receiving a message that follows closely the FIPA-ACL [50] message format.

6. Results and Experimental Analysis. The simulation graphical interface is presented in Fig. 6.1. A simple experiment was conducted. In a scenario closely mapped to Fig. 3.1, the environment was set-up with the upper area representing the body tissue, the middle (brown) area representing the vessel wall and the bottom (red) area representing the blood within the vessel. Within the body tissue area, a number of tumour-cells (purple) depicting the tumour were created in a random position, and the remaining area was filled with body-cells (pink). Subsequently, the area surrounding the tumour was given a *protein concentration* property, with a value gradually declining with distance from the tumour walls. A user-defined number of each type of agents, namely *Tracking T-bots*, *Barrier Degrading B-bots* and *Drug Delivery D-bots*, were randomly initiated in the bottom left side of the simulation area, which was selected to represent the injection site. Three possible experiment outcomes were identified:

1. *Tumour eliminated*: no more tumour cells left in the simulation world;
2. *Failure to eliminate tumour*: no more *D-bots* left but at least one tumour cell exists;
3. *Failure to complete experiment*: one or more *D-bot* fail to transition from idle state because all *B-bots* fail to degrade even one vessel cell.

By using the integrated NetLogo BehaviorSpace tool, a number of experiments were executed for different sets of parameter values, and with 500 simulation runs for each experiment. Although normally a thorough

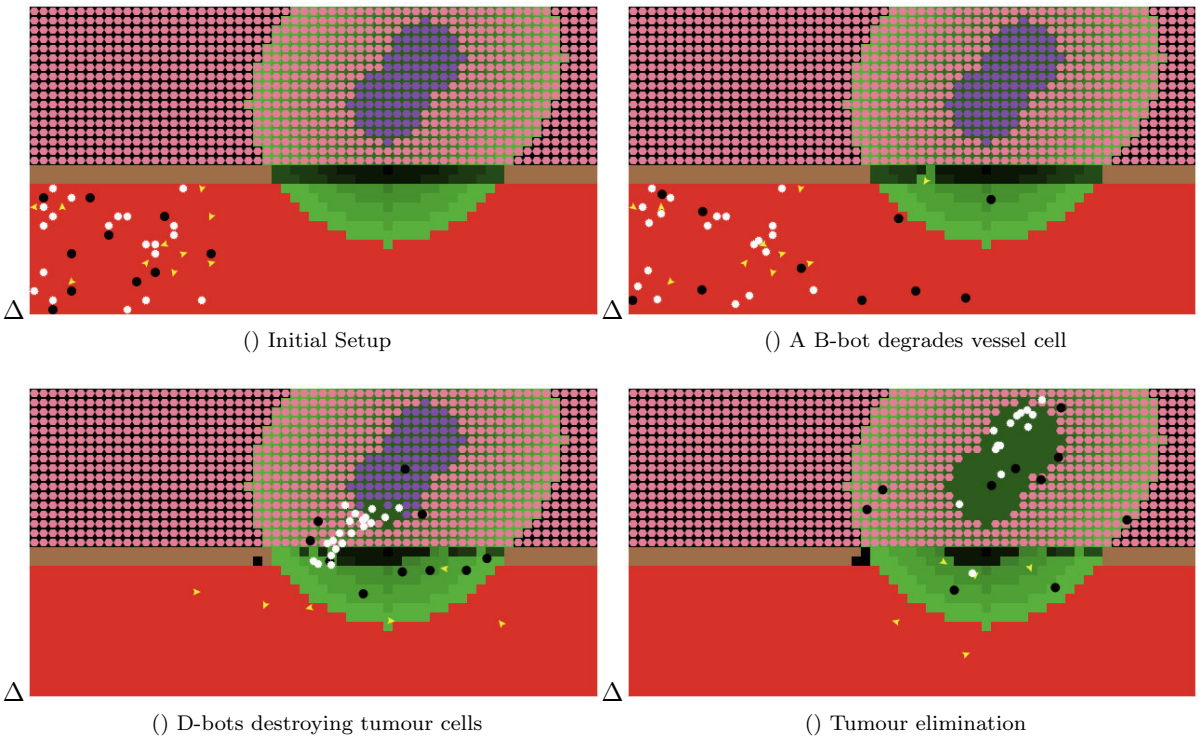

Fig. 6.1: Simulation Interface: The graphical interface developed to run the simulation

statistical analysis would be required, at these early stages we decided to run simple experiments in order to visualize the agent models created as well as to verify certain aspects of the experiments which were intuitively predicted. The stochastic elements in these experiments are simply based on the random setup of the agents as well as on their random move. Apart from this, individual agents were designed to operate in a deterministic way which however in collaboration results in an emergent behaviour.

The experiments were grouped into four categories, one for each set of parameters, i.e. number of *T-bots*, number, payload capacity and sensing radius of *B-bots*, number and payload capacity of *D-bots* and energy. The purpose was to estimate how different values for each of these entities would result in one of the three possible outcomes. For each experiment in a group, the relative frequencies of occurrence of the three possible end-results were calculated.

The results were qualitatively assessed and interpreted under two perspectives. On one hand, some conclusions were drawn that indicate the effect of the parameter values on the designed simulation outcome. On the other hand, the results were considered as likely offering some insight on how to improve the design of the system, hence, produce a more efficient or error-free model. Lacking any real-life experimental data to compare our results with, the model itself was validated against the main objective of tumour destruction.

The first conclusion that was drawn is that the system was able to accomplish the set goal of tumour elimination. As the results indicated, even with a low numbers of *T-bots* and *B-bots*, and even with a low *D-bot* payload capacity and sensing radius, it was still possible to eliminate the tumour. Nevertheless, the highest percentages of tumour elimination were achieved for high numbers of active agents. Various experiments have verified the intuition one might have about the importance of the three parameters in the simulations. For instance, failure to eliminate the tumours is related to the number of agents of each type. This is apparent when we incrementally increase the number of agents of each type, as shown in Fig. 6.2. However, the results indicate that the effect of increasing numbers of *T-bots* and *B-dots* becomes important for low numbers of

agents. The corresponding curves reach a plateau, and numbers higher than do not seem to lead to higher success percentages. Similarly, regarding the *D-dots*, a minimum threshold appears to exist, in the sense that with less than 8 *D-bots* the percentages of successful outcomes were close to zero. A steep increase appears after above between 8 and 10 *D-bots*, and once again, a plateau seems to emerge for slightly higher numbers. This observation suggests that there is a narrow band of appropriate values for the number of *D-bots*. Naturally, this observation is strongly correlated to the specific model, however, this phenomenon is a fitting example of how MAS simulations may offer advantageous insight into better understanding and possibly improving complex systems.

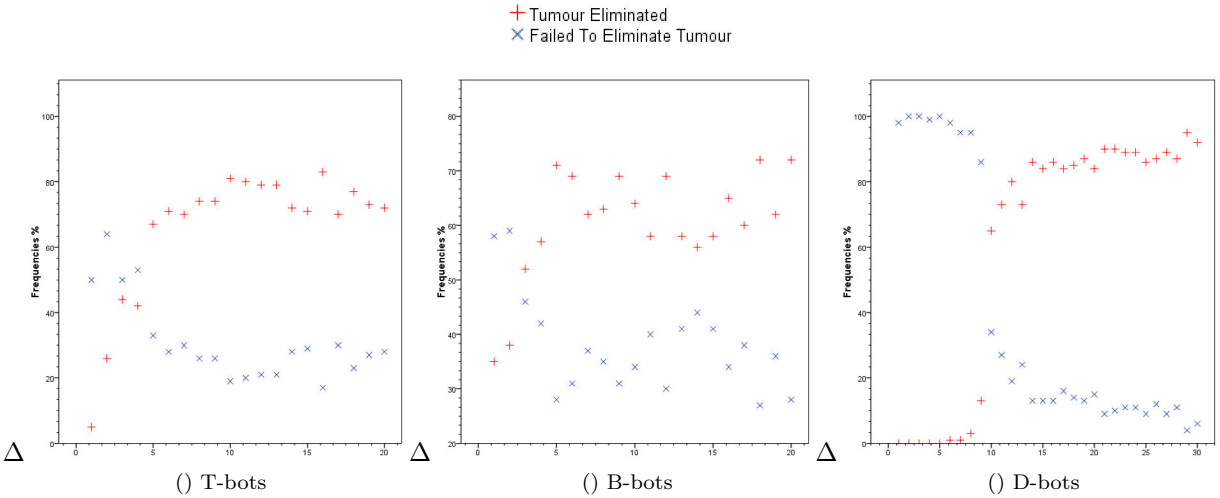

Fig. 6.2: Frequencies of outcomes plotted against numbers of agents

Further observations include:

- The effects of the *B-bot* sensing radius and payload capacity and the *D-bot* payload capacity were investigated. Regarding the *B-bots*, it appeared that even with a maximum value for their number, the percentage of tumour elimination is very low. This was attributed to the fact that —in the specific design— for the *B-bots* to penetrate the vessel wall they needed to degrade two consecutive vessel cells. With the payload capacity and the sensing radius set at the minimum value, tumour elimination was an extremely rare occurrence.
- It was also estimated that the *B-bot* payload parameter affected the outcome substantially more than the sensing radius. Indeed, when the payload was minimum, even with a high sensing radius, the "Tumour eliminated" outcome was quite rare, with recorded frequencies ranging from 1.83% to 3.33%. On the contrary, a minimum percentage of 15% was recorded when the payload was set to a medium value. This effect, however, is not observed for the maximum value for the sensing radius, implying that its significance lessens from a certain point onwards. Even with both parameters set on their highest values, tumour elimination was recorded in roughly 65% of the cases. This result, compared to the percentages of the control experiments (medium values for all parameters), supports the argument that the number of *D-bots* affects the outcome more strongly than the *B-bot* payload and sensing radius.
- The *D-bot* payload appeared to contribute greatly to the outcome, where an increase reaching almost 88% for successful tumour elimination was observed when the corresponding parameter was set from low to medium. However, this increase became minimal between medium and high values for the same parameter, implying that its contribution had reached a plateau. This observation could offer insight on determining the optimum payload capacity, should such a system be realized in real-life.
- A set of experiments were designed specifically to estimate the effect of the energy parameter, that would be of special interest should such a system ever exist in reality. The results are shown in Fig. 6.3. The

percentage frequencies of occurrence of the "Tumour Eliminated" and "Failed to Eliminate Tumour" outcomes were plotted against the corresponding values of the energy parameter, while maintaining average numbers. A correlation may be observed between the two sets of variables, indicating that as the energy values were incrementally increased, the probability for tumour elimination increased as well, whereas the probability for failure to eliminate the tumour decreased. Additionally, for energies lower than 100, the agents could not achieve tumour elimination, whereas for energies higher than 600, the percentage of success seems to remain unaffected. Once again, this indicates that MAS simulations could possibly be used to identify lower and upper limits in the values of parameters of interest, should such a system ever be realized. The line for the "Failure to Complete Experiment" outcome is not plotted, as it has no practical value. However, the percentage of its occurrence may be directly deduced from the graph.

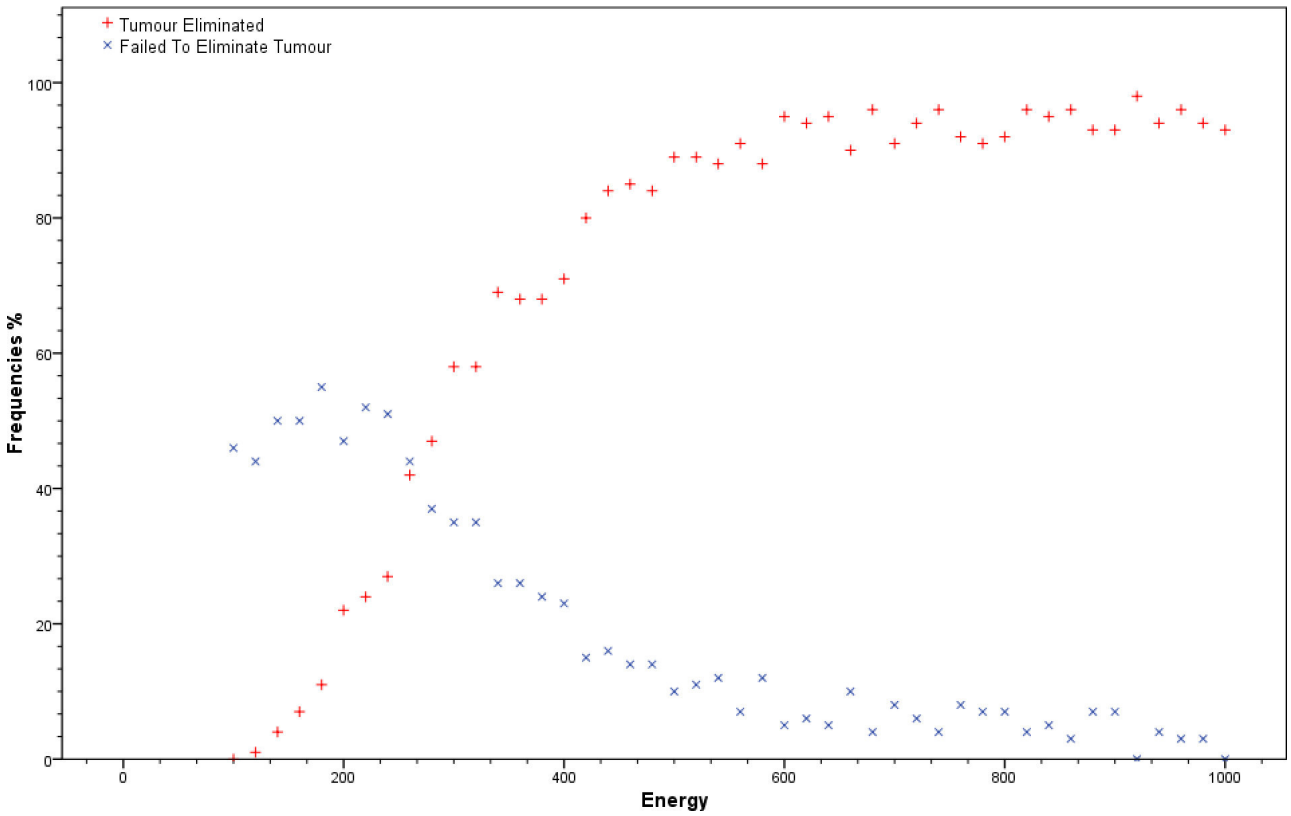

Fig. 6.3: Frequencies of *Tumour Eliminated* and *Failed To Eliminate Tumour* outcomes plotted against the energy values.

Regarding the system's overall functionality, the experiments served as an indirect validation against the set requirements. A successful outcome, i.e. a "Tumour eliminated" result was reproducibly achievable. The observed "Failed to complete experiment" high percentage in experiments with all parameters set to their lowest values was anticipated. It was a design decision to favour energy preservation over experiment completion concerns, by initializing both *B-bots* and *D-bots* awaiting for a message before exiting the initial idle state. This lead to the possibility of these agents staying inactive, in the case that not enough *T-bots* managed to sense the protein gradient. Nevertheless, the observed percentages in subsequent experiments with high values for the agent parameters, suggest that a better design should perhaps be considered, to combine both energy preservation with optimum experiment completion occurrences.

7. Nanorobotics: Simulation versus Reality. Currently, the nanorobots as envisaged in our system, are purely theoretical artifacts. Nanorobots are a distant and ambiguous concept. However, researchers involved in the domain insist that studying and attempting to mimic nature's molecular machines, which have been working and optimizing their functionalities over millions of years, might be the best approach towards fabricating real nanorobots. A roadmap proposed in [30] describes the following steps, yet far from reality:

- Developing bio-nanocomponents, such as bio-nanosensors and bio-nanomotors;
- Assembling the bio-nanocomponents into bio-nanorobots;
- Developing the concepts for nanorobot collaboration (*bionanoswarms*), establishing the methods for distributive intelligence, programming and control;
- Developing and implementing automatic fabrication methodologies.

Technology nowadays allows for fabrication of micro- rather than nano-components. Spiral-shaped devices with lengths of approximately 50 μm, that are able to swim in a controlled fashion under weak magnetic field application have already been constructed [55]. Fabrication of a prototype autonomous robot of size $3.9 \times 3.9 \times 3.3$ mm^3, with communication, locomotion and energy storage capabilities is possible [12]. The first controlled vertical flight of a 19 mm sized microrobot has been announced [39]. These technological advances are merely mentioned to report the current status of microrobotic fabrication, indicating that manufacturing capabilities become more limited as size decreases.

However, what is of significant interest from a MAS viewpoint is the prospects of nanoro-botic communication and collaboration. To achieve nanorobotic collaboration, strategies for communication among nanorobots must be designed. Communication by means of acoustic waves has been recently investigated by Hogg and Freitas, and findings were reported in [18]. Spherically shaped robots of three different radii; 0.5 μm for the size of an isolated nanorobot, 5 μm for nanorobot aggregates that would address tasks requiring greater than individual capabilities, and 50 μm for the size of tissue-embedded signal repeater stations were mathematically investigated. It was concluded that communication could occur at 10^4 bits/s, over distances of 100 μm, depending on available power and on safety constraints (i.e. nanorobot heating).

Another approach towards communication is once again inspired by nature, namely chemical signalling. This method for communication has been observed both in the macro-world (ants that release and sense pheromone trails [21]) and in the microworld (bacteria that produce and respond to signalling molecules called *autoinducers*, in a process called *quorum sensing* [49]).

The issue of coordination will require far more elaborate mechanisms. *Swarm intelligence* may currently be investigated only via computer simulations. The decentralized movement coordination by implementing an enhanced version of common flocking algorithms is investigated [1]. A framework based on an adopted variation of the Artificial Bee Colony algorithm is also proposed [35]. It is evident that the actual fabrication of nanorobot-swarms will require the realisation of individual nanorobot assembly first, but researchers are committed to explore such models in anticipation of the technological breakthrough.

8. Conclusions and Future Work. We demonstrated how Multi-Agent Systems can assist in simulating drug delivery in unhealthy tissues by a set of collaborating nanorobots. Despite the fact that such a system is far yet from becoming reality, MAS appeared to be an applicable solution in modelling and identifying a number of key-issues. A targeted drug delivery system was designed, including formal models of agents that work together in order to treat a tumour located inside some bodily tissue. A simulation based on the formal models was developed in order to demonstrate how various parameters affect the overall treatment.

Further work would include improvement on the design of MAS. This could include enhancing the agent communication, from simple message passing to implementing some existing agent interaction protocols, i.e. introducing some form of negotiation. Another improvement includes refining the parameters to be investigated in a more pragmatic manner. Blood elements might be added as obstacles in the nanorobot movement, and blood flow may also be considered. Furthermore, inclusion of physical properties, such as blood viscosity, tissue density and electric current can result in a more detailed system.

REFERENCES

[1] G. Al-Hudhud, *On Swarming Medical Nanorobots*, International Journal of Bio-Science and Bio-Technology, 4 (2012), pp. 75–90.

[2] R. J. Allan, *Survey of agent based modelling and simulation tools*, Tech. Report DL-TR-2010-007, Science and Technology Facilities Council, oct 2010.

[3] R. A. Brooks, *Intelligence without reason*, in Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), J. Myopoulos and R. Reiter, eds., Sydney, Australia, 1991, Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, pp. 569–595.

[4] ———, *Intelligence without representation.*, Artificial Intelligence, 47 (1991), pp. 139–159.

[5] A. Cavalcanti, L. Rosen, L. Kretly, M. Rosenfeld, and S. Einav, *Nanorobotic challenges in biomedical applications, design and control*, in Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on, dec. 2004, pp. 447 – 450.

[6] A. Cavalcanti, L. Rosen, B. Shirinzadeh, M. Rosenfeld, S. Paulo, and T. Aviv, *Nanorobot for treatment of patients with artrery occlusion*, Platelets, 20 (2007), pp. 1–10.

[7] A. Cavalcanti, B. Shirinzadeh, T. Fukuda, and S. Ikeda, *Nanorobot for brain aneurysm*, The International Journal of Robotics Research, 28 (2009), pp. 558–570.

[8] A. Cavalcanti, B. Shirinzadeh, D. Murphy, and J. A. Smith, *Nanorobots for laparoscopic cancer surgery*, in Computer and Information Science, ACIS International Conference on, IEEE Computer Society, 2007, pp. 738–743.

[9] S. Chin, *FLAME (FLexible Large-scale Agent Modelling Environment)*, 2014.

[10] M. Coccia, U. Finardi, and D. Margon, *Current trends in nanotechnology research across worldwide geo-economic players*, The Journal of Technology Transfer, 37 (2012), pp. 777–787.

[11] D. Dranidis, K. Bratanis, and F. Ipate, *JSXM: A tool for automated test generation*, in Software Engineering and Formal Methods, Springer, 2012, pp. 352–366.

[12] E. Edqvist, N. Snis, R. C. Mohr, O. Scholz, P. Corradi, J. Gao, A. Diguez, N. Wyrsch, and S. Johansson, *Evaluation of building technology for mass producible millimetre-sized robots using flexible printed circuit boards*, Journal of Micromechanics and Microengineering, 19 (2009), p. 075011.

[13] G. Eleftherakis and P. Kefalas, *Model checking safety critical systems specified as X-machines*, Matematica-Informatica, Analele Universitatii Bucharest, 49 (2000), pp. 59–70.

[14] Fluent Inc., *FLUENT*. `http://www.fluent.com`.

[15] R. A. Freitas, *Pharmacytes: an ideal vehicle for targeted drug delivery*, Journal of Nanoscience and Nanotechnology, 6 (2006), pp. 2769–75.

[16] T. Hamaguchi, T. Doi, T. Eguchi-Nakajima, K. Kato, Y. Yamada, Y. Shimada, N. Fuse, A. Ohtsu, S.-i. Matsumoto, M. Takanashi, and Y. Matsumura, *Phase I Study of NK012, a Novel SN-38Incorporating Micellar Nanoparticle, in Adult Patients with Solid Tumors*, Clinical Cancer Research, 16 (2010), pp. 5058–5066.

[17] E. J. Hanly, M. R. Marohn, S. L. Bachman, M. A. Talamini, S. O. Hacker, R. S. Howard, and N. S. Schenkman, *Multiservice laparoscopic surgical training using the davinci surgical system*, American Journal of Surgery, 187 (2004), pp. 309–315.

[18] T. Hogg and R. A. F. Jr.), *Acoustic communication for medical nanorobots*, Nano Communication Networks, 3 (2012), pp. 83 – 102.

[19] O. E. Holland and C. R. Melhuish, *Getting the most from the least: Lessons for the nanoscale from minimal mobile agents*, in Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, C. G. Langton and K. Shimohara, eds., MIT Press, 1997, pp. 59–66.

[20] F. Ipate and M. Holcombe, *An integration testing method that is proved to find all faults*, International Journal of Computer Mathematics, 63 (1997), pp. 159–178.

[21] D. E. Jackson and F. L. W. Ratnieks, *Communication in ants.*, Current biology : CB, 16 (2006), pp. R570–4.

[22] P. Kefalas, M. Holcombe, G. Eleftherakis, and M. Gheorghe, *Intelligent Agent Software Engineering*, IGI Global, July 2002.

[23] P. Kefalas, I. Stamatopoulou, I. Sakellariou, and G. Eleftherakis, *Transforming Communicating X-machines into P Systems*, Natural Computing, 8 (2009), pp. 817–832.

[24] J. King, M. Lees, and B. Logan, *Agent-based and continuum modelling of populations of cells*, December 2006.

[25] A. Kumari, S. K. Yadav, and S. C. Yadav, *Biodegradable polymeric nanoparticles based drug delivery systems*, Colloids and Surfaces B: Biointerfaces, 75 (2010), pp. 1 – 18.

[26] D. L. Leslie-Pelecky and R. D. Rieke, *Magnetic properties of nanostructured materials*, Chemistry of Materials, 8 (1996), pp. 1770–1783.

[27] L. Li, F. Tang, H. Liu, T. Liu, N. Hao, D. Chen, X. Teng, and J. He, *In vivo delivery of silica nanorattle encapsulated docetaxel for liver cancer therapy with low toxicity and high efficacy*, ACS Nano, 4 (2010), pp. 6874–6882.

[28] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, *MASON: A multiagent simulation environment*, SIMULATION, 81 (2005), pp. 517–527.

[29] J. Martín, J. Nogués, K. Liu, J. Vicent, and I. K. Schuller, *Ordered magnetic nanostructures: fabrication and properties*, Journal of Magnetism and Magnetic Materials, 256 (2003), pp. 449 – 501.

[30] C. Mavroidis, A. Ummat, and A. Dubey, *Bio-Nanorobotics*, in Biomimetics, CRC Press, Nov. 2005, pp. 201–227.

[31] A. D. McCulloch and G. Huber, *Integrative Biological Modelling In Silico*, John Wiley & Sons, Ltd, 2008, pp. 4–25.

[32] C. Melhuish and J. Welsby, *Gradient ascent with a group of minimalist real robots: implementing secondary swarming*, in Systems, Man and Cybernetics, 2002 IEEE International Conference on, vol. 2, Oct 2002, pp. 509–514.

[33] N. Minar, R. Burkhart, C. Langton, and M. Askenazi, *The Swarm simulation system: A toolkit for building multi-agent simulations*, Working Papers 96-06-042, Santa Fe Institute, June 1996.

[34] Nano.gov, *Benefits and applications — nano*. http://www.nano.gov/you/nanotechnology-benefits, 2014. On-line; Accessed October 20, 2014.

[35] T. Nantapat, B. Kaewkamnerdpong, T. Achalakul, and B. Sirinaovakul, *Best-so-far ABC based nanorobot swarm*, in Proceedings of the 2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics - Volume 01, IHMSC '11, Washington, DC, USA, 2011, IEEE Computer Society, pp. 226–229.

[36] M. North, T. Howe, N. Collier, and J. Vos, *A declarative model assembly infrastructure for verification and validation*, in Advancing Social Simulation: The First World Congress, S. Takahashi, D. Sallach, and J. Rouchier, eds., Springer Japan, 2007, pp. 129–140.

[37] M. Ntika, P. Kefalas, and I. Stamatopoulou, *Multi-agent system simulation of nano-robotic drug delivery in tumours of body tissues*, in System Theory, Control and Computing (ICSTCC), 2013 17th International Conference, Oct 2013, pp. 777–782.

[38] G. M. Patel, G. C. Patel, R. B. Patel, J. K. Patel, and M. Patel, *Nanorobot: a versatile tool in nanomedicine.*, Journal of Drug Targeting, 14 (2006), pp. 63–67.

[39] N. O. Pérez-Arancibia, K. Y. Ma, K. C. Galloway, J. D. Greenberg, and R. J. Wood, *First controlled vertical flight of a biologically inspired microrobot*, Bioinspiration & Biomimetics, 6 (2011), p. 036009.

[40] I. Petreska, P. Kefalas, and M. Gheorghe, *A framework towards the verification of emergent properties in spatial multi-agent systems*, Proceedings of the Workshop on Applications of Software Agents, (2011), pp. 37–44.

[41] C. A. Piña García, E. J. Rechy-Ramírez, and V. A. García-Vega, *Using an alternative model in a complex environment for nanorobotics navigation*, in 16th International Conference on Computing (CIC), nov. 2007.

[42] ———, *Comparing three simulated strategies for cancer monitoring with nanorobots*, in MICAI 2008: Advances in Artificial Intelligence, A. Gelbukh and E. F. Morales, eds., vol. 5317 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 1020–1030.

[43] H. Redhead, S. Davis, and L. Illum, *Drug delivery in poly(lactide-co-glycolide) nanoparticles surface modified with poloxamer 407 and poloxamine 908: in vitro characterisation and in vivo evaluation*, Journal of Controlled Release, 70 (2001), pp. 353 – 363.

[44] I. Sakellariou, P. Kefalas, and I. Stamatopoulou, *Enhancing NetLogo to simulate BDI communicating agents*, in Proceedings of the 5th Hellenic conference on Artificial Intelligence: Theories, Models and Applications, SETN '08, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 263–275.

[45] I. Sakellariou, P. Kefalas, and I. Stamatopoulou, *Teaching intelligent agents using NetLogo*, in Proceedings of the Informatics Education Europe III Conference (IEEIII 2008), A. Cortesi and F. Luccio, eds., 2008, pp. 209–221.

[46] G. Sharma, C. Mavroidis, and A. Ferreira, *Virtual reality and haptics in nano- and bionanotechnology*, in Handbook of Theoretical and Computational Nanotechnology, M. Rieth and W. Schommers, eds., vol. X, American Scientific Publishers, 2005, ch. 40, pp. 1–33.

[47] I. Stamatopoulou, P. Kefalas, and M. Gheorghe, *OPERAS: a formal framework for multi-agent systems and its application to swarm-based systems*, in Proceedings of the 8th International Workshop on Engineering Societies in the Agents World (ESAW'07), A. Artikis, G. O'Hare, K. Stathis, and G. Vouros, eds., 2007, pp. 208–223.

[48] B. Steele, *'Ordered' catalyst boosts fuel cell output at lower cost*. http://www.nanowerk.com/news2/newsid=27135.php, 2012. [Online; accessed 20-April-2014].

[49] M. E. Taga and B. L. Bassler, *Chemical communication among bacteria*, Proceedings of the National Academy of Sciences of the United States of America, 100 (2003), pp. 14549–14554.

[50] The Foundation for Intelligent Physical Agents, FIPA, *The history of FIPA*. http://www.fipa.org/subgroups/ROFS-SG-docs/History-of-FIPA.htm. On-line; Accessed November 11, 2012.

[51] A. Trafton, *Super-sensitive and small: New MIT detector uses nanotubes to sense deadly gases*. http://newsoffice.mit.edu/2008/nanotube-0605, 2008. [Online; accessed 20-April-2014].

[52] A. Ummat, G. Sharma, A. Dubey, and C. Mavroidis, *Bio-nanorobotics: State of the art and future challenges*, in Tissue Engineering and Artificial Organs, The Biomedical Engineering Handbook, J. D. Bronzino, ed., London, UK: CRC Press, 2006, ch. 19, pp. 1–42.

[53] R. Valiev, *Nanostructuring of metals by severe plastic deformation for advanced properties*, Nature materials, 3 (2004), pp. 511–516.

[54] U. Wilensky, *NetLogo*. http://ccl.northwestern.edu/netlogo/, 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

[55] L. Zhang, J. J. Abbott, L. Dong, B. E. Kratochvil, D. Bell, and B. J. Nelson, *Artificial bacterial flagella: Fabrication and magnetic control*, Applied Physics Letters, 94 (2009), pp. 064107 –064107–3.

Edited by: Costin Bădică
Received: July 21, 2014
Accepted: Sept 21, 2014

DOI 10.12694/scpe.v15i3.1018
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. 231–248. http://www.scpe.org

ABVE-CONSTRUCT: AN AGENT-BASED VIRTUAL ENTREPRISE MODEL FOR CIVIL ENGINEERING

MIHAELA OPREA *

Abstract. Modelling efficient virtual enterprises (VEs) represents a real challenge in the context of dynamic, complex and extending business markets. Among the modelling approaches proposed for VEs, multi-agent systems are more suitable due to their basic characteristics of flexibility, modularity, distributivity as well as some characteristics such as adaptability and learning, that could be added in order to improve the overall VE performance and scalability. The application of VEs and agent-based VEs were proposed for several industrial domains: automotive industry, food industry, electronics and telecommunications, transportation, robotics, for manufacturing and supply chain problems solving. This paper presents a generic framework for virtual enterprises development, *VE-Frame*, and an agent-based virtual enterprise model, *ABVE-Construct*, for the civil engineering domain, derived from the *VE-Frame*. Also, a case study of using the model for a building construction scenario, and the model adaptation options for other domains of applications are discussed.

Key words: Virtual Enterprise, Agent-Based Virtual Enterprise Model, Ontology, Civil Engineering

AMS subject classifications. 68T42

1. Introduction. The increased global competition in most domains (such as industrial and commercial) with dynamic, complex and extending business markets represents a challenge for the adoption of new ICT-based solutions, especially in virtual environments provided by the Internet, where real world organizations, enterprises and institutions can have a virtual presence and can join into virtual enterprises for common business goals. Domains such as automotive industry, electronics and telecommunications have adopted this kind of solution. Virtual enterprises can offer a very good opportunity for dynamic cooperation between partners enterprises in order to increase the business profit based on better adaptability to the changing business markets.

Several research projects were developed worldwide during the last two decades, focusing on the study of VEs at the modelling level, as well as at the implementation level, in real world applications and in other collaborative networks (e.g. in education and research). Examples of such projects are PRODNET, VEGA, MASSYVE, BIDSAVER, ECOLEAD, SPIKE, BIVEE and NIIIP. The current research focuses in two principal directions: development and testing of new applications (real world VEs testbeds), and new or improved modeling methods, development methodologies and frameworks. Other research issues are concentrated on new techniques and methods for particular tasks solving (e.g. VE partners search and selection, VE formation, contract negotiation).

Three main modelling approaches were proposed so far for VEs [28]: the object-oriented business process modelling, the holons and the intelligent agents-based modelling, and the ontological modelling. Among the proposed models for VEs, multi-agent systems seems to be the best choice for modelling and for the implementation of real world applications (geographically distributed) [6]. An agent-based virtual enterprise (ABVE) takes all the advantages provided by intelligent agents used in distributed applications. Therefore, multi-agent systems are more suitable due to their basic characteristics of flexibility, modularity, distributivity (e.g. geographically distribution, tasks distribution), as well as some characteristics that could be added (such as adaptability and learning) in order to improve the ABVE performance and scalability.

In this paper it is proposed a generic framework for virtual enterprises development, *VE-Frame*, based on ontologies, and an agent-based virtual enterprise model, *ABVE-Construct*, for the civil engineering domain. We have started the development of the model from an early research work reported in [23], which presented an ABVE VIRT_CONSTRUCT, for private houses construction, and we have extended the work reported in [22] with the VE development framework, *VE-Frame*, a detailed description of the *ABVE-Construct* model, and a discussion about a case study of using the model for a building construction scenario and the model adaptation options for other domains of applications.

*Department of Automation, Computers and Electronics, Petroleum-Gas University of Ploiesti, Ploiesti, Romania (mihaela@upg-ploiesti.ro)

The building construction task can be seen as a manufacturing problem, where the business goal is to provide the building according to the customers requirements. In this case, the typical structure of a VE includes apart from the main construction enterprise, the construction materials suppliers as well as other partner enterprises (e.g. enterprises executing the installations and the domestic services, i.e. utilities connection). The application of VEs in civil engineering can improve the overall building construction business quality.

New challenges arose in the last years in this domain: supply chain management for a better construction activity performance [16], building energy performance analysis, zero carbon buildings construction [33], intelligent buildings construction, computer-aided 3D building modeling, simulation of building construction activities in order to make re-design or revision/refinement design and to set for example, a safe workplan analyzing various risks (e.g. risk of collisions on the construction site), and a real-time context aware computing, as well as the new business opportunities arising in the virtual markets for developing virtual enterprises in buildings construction.

The rest of the paper is organized as follows. *Section 2* discusses some generalities about the virtual enterprise and a brief review on related work. The VE development framework, *VE-Frame*, and the agent-based virtual enterprise model, *ABVE-Construct*, are introduced in *Sect. 3*. A case study of using the model for a building construction scenario, and the model adaptation options for other domains of applications are discussed in *Sect. 4*. The final section concludes the paper.

2. Background and related work. A virtual enterprise represents a temporary alliance between some networked enterprises (VE partners) that have a common business goal and will to cooperate. The VE partner enterprises share skills and competencies in order to achieve the VE goal and to exploit the business opportunities of fast changing markets. The lifecycle of a virtual enterprise has three main phases: the VE creation, the VE operation and evolution, and the VE dissolution. The VE creation includes VE initiation, VE partners search and selection, VE contract negotiation etc. The business process is performed during VE operation. The VE evolution is determined by some events that can occur during VE operation, as for example, the change of the business plan and the replacement of a VE partner. The VE dissolution is performed when the VE finishes its business process or the VE operation is stopped as a decision of all VE partners.

On the VE formation success depends the future VE business performance. Most of the research work reported in the literature is focused on this topic.

2.1. VE Partners search and selection. The VE partners are found by using some partner search tools as for example, recommending systems (e.g. yellow pages agents, facilitators) or the internal list that the main contractor keeps, based on its previous work experience.

Various criteria are used for partners enterprises selection such as trust, commitment, successful cooperation history, reputation, experience, risk, reliability, skills, cost, time, distance, value-added services, environmental issues etc. These criteria are divided in two classes: non-negotiable and negotiable. Examples of non-negotiable issues are past cooperation performance, reputation, technology, trust, commitment, production capacity. These issues are usually qualitative criteria used during the pre-selection phase when potential partners are selected from interested partners. Negotiable issues such as price, quality, service, delivery are used during the final selection phase when the VE partners are selected from the potential partners, based on the content of their bids. In the last years more researchers have started to tackle the VE partners selection as a multi-criteria decision making problem, for multiple products and/or services. The partners are ranked by using a specific method. An example is given by the TOPSIS method [15] that ranks the alternative solutions according to their distances from the ideal alternative and the negative ideal alternative.

Among the variety of techniques and approaches that were used for VE partners selection and VE formation, we mention some of them.

VE partner's selection: agent interaction protocols [27], the TOPSIS method [15], a fuzzy extension of the TOPSIS method [9], the use of particle swarm optimization (PSO), a metaheuristic, inspired from nature [35], a partners selection approach for multiple products / services by taking into account the synergy effects (e.g. complementarities and substitutability) between products / services [32], a new method based on vague sets and PSO that uses as selection criteria factors such as the satisfaction degree, due date, cost, precedence of tasks [14].

VE formation: a direct reputation model [2], an organization-based coalition formation, based on a novel distributed algorithm improved by a reinforcement learning technique that optimize the decision made by the agents inside the VE [1], a dynamic formation of a VE for the transportation domain [5], and for the environmental management [3].

The current trends in the VE development research include apart from the application of metaheuristics in the VE formation, various advanced negotiation techniques (see e.g. [21] for the negotiation in the VE formation process, and [25]), as well as the inclusion of some environmental issues in the VE partners selection criteria, as for example, the green enterprise, i.e. an enterprise that uses or produces green materials, green components, new energy friendly products, that applies advanced technology of products recycling and disposal, with pollution control on air emission, waste water, as well as low energy consume, by saving electric and heat energy etc, and also, the product eco-design. A recent research work on the development of an ecological VE is reported in [31], where the focus is made on the ontological involvement in the VE.

2.2. Methodologies, frameworks and platforms for VE and ABVE development. A design methodology is presented in [13] that facilitates coordination and development of context-aware workflow management systems for the agent-based virtual enterprise. Other methodologies for VE development include those reported in [11] and [17]. An architecture-driven modeling approach for VE engineering is given in [28]. The implementation of a generic life-cycle assessment (LCA) methodology for ecological sustainable development applied in civil engineering (i.e. environmental improvement in the building sector) is discussed in [10]. More research efforts are directed to the design and implementation of various platforms for collaborative networks, as support tools of the VE development infrastructure or for electronic institutions. For example, an early work reported in [18] introduced Agora, an infrastructure for cooperative work support in multi-agent systems, that was used for VE, while recent works introduce a service-oriented framework for virtual organizations support: a collaborative network in [26], the SOAVE platform in [19], the PaaS framework in [29], and the VirtES, a VE simulator in [8]. Some methodological issues about the development of agent-based VEs are described in [6, 17, 18, 23] etc.

Few methodologies and frameworks that were reported in the literature provide a domain-independent view on VE or ABVE development by taken into account the main phases of software development.

Two major artificial intelligence approaches were applied in the civil engineering domain: knowledge based systems and multi-agent systems. From the knowledge based systems that were reported so far in the literature, we have selected one that has a possible applicability in the implementation of a VE as a multi-agent system. In [20] it is introduced an integrated knowledge based system, ICEMS, for alternative design decisions, materials selection and cost estimating for pre-design analysis in civil engineering. An ICEMS like system can be integrated in an ABVE, at least, for the building design phase. Regarding the multi-agent systems approach, it was applied so far to VE modeling in the building construction domain (see e.g. [9], [24], and [34]). Both approaches can be used in the development of an ABVE in the civil engineering domain.

Our research work was concentrated on the development of a generic framework for VEs development, *VE-Frame*, based on ontologies, and of an agent-based VE model for the civil engineering domain, *ABVE-Construct*.

3. The development of an agent-based virtual enterprise. Virtual enterprises can be modelled in a natural way as multi-agent systems, being distributed applications. Before doing an agent-based modelling, we can develop the VE by using a certain methodology or framework. We have designed a generic framework for VE development, *VE-Frame*, that can be adapted for the agent-based model or any other VE model. The main characteristics of the framework are flexibility and adaptability. Also, it is enough general in order to be used for any domain of application and with any VE type.

3.1. VE-Frame - a generic framework for VE development. The generic framework, *VE-Frame*, for VE development was derived from the main phases of software development: analysis, design, implementation and testing. Details related to the VE are provided for each phase and we have included a new phase, the conceptualization of the business domain (i.e. ontology development), that is important for the development of a VE in the next phases: VE Design and VE Implementation. *Figure 3.1* shows the workflow of the generic framework *VE-Frame*.

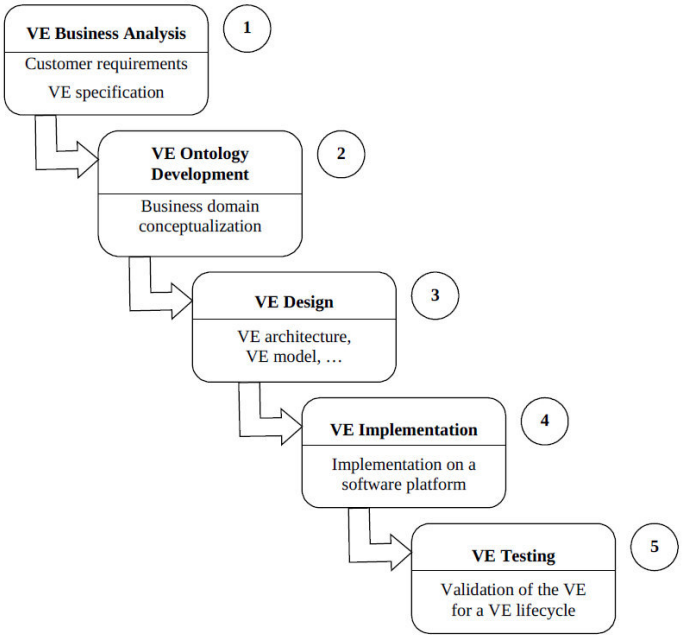

Fig. 3.1: The workflow of the generic framework, VE-Frame

The main steps for the application / adaptation of the framework to a domain are: VE business analysis, VE ontology development, VE design, VE implementation and VE testing. They are detailed as follows.

1. *VE business analysis* i.e. *problem analysis* (business analysis starting from the selected business opportunity with customer requirements and provides the VE specification)
   - identification of the business domain and the main business goal (i.e. the VE goal), decomposing the goal into sub-goals and correspondingly, the main business task into sub-tasks, provide a preliminary form of the business work plan;
   - resume the functionalities, roles and list of responsibilities, the information flow (with inputs/ outputs), the business rules, norms, legislation, a preliminary form of the contract between the customer and the main contractor;
   - identify the activities, the working environment, the needed resources (e.g. skills, competencies, abilities, personnel, data, equipment and tools, materials) etc;
   - identify the main needed enterprises (future main contractor / VE coordinator enterprise and VE partners) that will cooperate in order to fulfill the business goal, describe the list of requirements for each enterprise (coordinator enterprise or the main contractor enterprise and partner enterprise), and provide a preliminary VE model;
2. *VE ontology development* (conceptualization of the business domain)
   - choose an existing VE ontology for the business domain or develop a VE ontology starting with the general VE ontology (with terms specific to any VE), and adding the domain specific VE ontology; - during all the phases of VE lifecycle, the partners, the coordinator and the customer will use in their interactions concepts from the VE ontology which is a shared resource;
3. *VE design* (organizational model, general and detailed architecture, design or choose the main strategies and procedures used in main contractor / coordinator and partners search and selection, contract and sub-contract negotiation, VE coordination mechanism, VE operation and evolution, VE dissolution etc)

- design the VE organizational model or choose an existing one (e.g. explicit consortium, internal consortium, sub-contracting and partnership);
- design the VE architecture based on the VE organizational model and the preliminary VE model (step 1);
- design or choose the partners search and selection procedure (e.g. auctions), the contract and sub-contract negotiation models, the coordinator and partner enterprises coordination mechanism, and other strategies and protocols needed for the good functioning of the VE;
- design each enterprise (VE coordinator, VE partner) from the VE (starting from the main VE goal and the sub-goals and sub-tasks that are associated with that enterprise), including the databases and non-legacy software;

4. *VE implementation*
   - choose a software platform (e.g. one with incorporated VE infrastructure) or design and implement one;
   - implement the VE on the software platform;
5. *VE testing*
   - validate the developed VE for the VE lifecycle: creation, operation, evolution and dissolution.

The organizational model of the VE must be chosen in step 3 (*VE design*) according to the application domain. Four main organizational forms are possible (as stated in [6]): explicit consortium, internal consortium, sub-contracting and partnership. Each organizational model imposes certain types of relations and responsibilities between the VE partners. Thus, the VE Coordinator is not necessary in all possible models.

The coordinator enterprise will perform the phases of VE initiation, VE operation and evolution, and VE dissolution, when the VE will end its activity. Before dissolution, the coordinator enterprise will assess and register the partners performance in order to be used by the partners selection tools in future VE creation.

In the case of an agent-based VE, the steps 3 and 4 will be adapted for the use of intelligent agents. For example, in step 3 it will be chosen or designed the architecture of each type of agent from the ABVE model (e.g. deliberative architectures such as BDI, reactive architectures or hybrid architectures). In step 4 it will be chosen an agent-based platform (usually, a FIPA compliant one).

3.2. The agent-based VE. In an agent-based model of a virtual enterprise, each enterprise has associated at least one agent that represents the enterprise in the virtual enterprise. The types of relations between the VE partners have an influence onto the ABVE model. For example, depending on the organizational model that was chosen, a VE Coordinator is necessary or not. In our ABVE model we have considered that the organizational model requires a VE Coordinator (as e.g. partnership, sub-contracting), responsible for the agents coordination and for the success of the VE business.

The VE has a goal that is achieved by a set of activities (i.e. tasks) which are performed by a set of roles (i.e. a kind of behaviours). In an ABVE, the agents fills the roles. *Figure 3.2* shows the roles diagram (UML) for an agent-based VE. The main sub-roles of an ABVE are coordinator and partner.

An ABVE can be defined as a set of the following entities: the ABVE coordinator agent (CoordAgent) and the list of ABVE partners agents (PartnersAgents), the common business goal (BGoal), the business work plan (BWorkPlan), the ABVE infrastructure (usually, a multi-agent platform), and the shared ABVE ontology (containing all terms used in the messages exchanged by the agents of the ABVE, and the relations between terms):

$$\text{ABVE} = \{\textit{CoordAgent}, \textit{PartnersAgents}, \textit{BGoal}, \textit{BWorkPlan}, \textit{ABVE-Infrastructure}, \textit{ABVE-Ontology}\}$$

The infrastructure of an ABVE includes the coordination and communication mechanisms, as well as the resources, databases and knowledge bases. The coordinator and partners search and selection mechanisms, the contract negotiation strategy are also included.

3.3. The ABVE-Construct model for civil engineering. In this section it is introduced the *ABVE-Construct* model that is defined by a generic architecture, the roles diagram, a description of each agent with the associated tasks, the ontology, the ABVE lifecycle algorithm, the ABVE creation algorithm, the partners selection mechanism, and the ABVE functioning diagram.

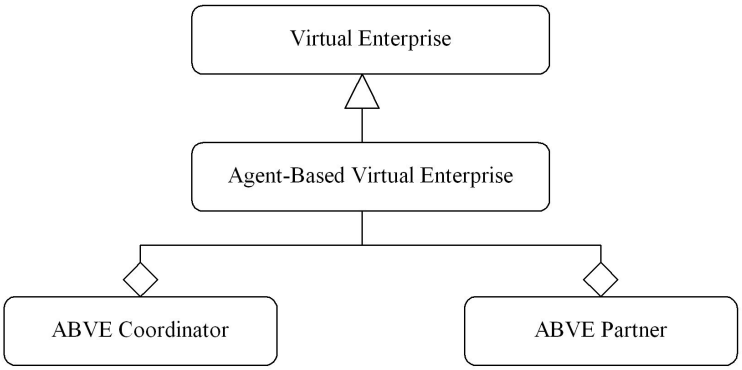

Fig. 3.2: The roles diagram for an ABVE (UML)

3.4. The ABVE-Construct Generic Architecture. For the civil engineering domain, the business goal is the building construction, and the specific VE requirements list is derived from the customer list of requirements. We have chosen as VE organizational model the sub-contracting model, which reflects better the reality. *Figure 3.3* shows the generic architecture of a VE applied in civil engineering. We have considered a simplified version of the building construction problem in which there are four types of enterprises: the main construction enterprise (E1), contracted by the customer, a building designer enterprise (E2), an enterprise doing the installations and construction utilities connection (E3), and a suppliers enterprise (E4).

The agent-based virtual enterprise model associates to each of the four enterprises, and to the customer a software agent. So, the whole agent-based system has five agents. Each agent of the ABVE (i.e. except the customer agent) has a database and a knowledge base. In the case of the civil engineering domain, the databases include, for example, data about various construction materials (such as bricks, concrete blocks, panels, wood tiles, carpet, ceramic tiles). The knowledge bases contain rules from the civil engineering domain, that are used by the agents when reasoning and executing their tasks.

3.5. The Roles Diagram. *Figure 3.4* shows the roles diagram (in UML) for the *ABVE-Construct* model. Examples of roles are Customer, Designer Enterprise, Suppliers Enterprise. A role is described by the position in the ABVE and a list of responsabilities. For example, the Designer Enterprise role has the position of ABVE partner enterprise being subordinated to the Construction Enterprise role and its list of responsabilities includes design, re-design, revision and refinement of building architect drawings.

3.6. The Agents Description. Agents are associated with roles. Each agent is described by a set of percepts, actions, goals, environment (PAGE description) as shown in *Table 3.1.*

The main contractor / ABVE coordinator will decompose the business goal, Building Construction, in subgoals and subsequently in a set of four main tasks T1, T2, T3, T4 that will be distributed to the ABVE partners for execution during ABVE operation. These tasks are T1 = building design; T2 = supply materials, equipment and other resources; T3 = building construction execution; T4 = installations execution and utilities connection. A task is composed by a set of activities included in the task execution plan. Each activity has time constraints, resource requirements and some relations with other activities (e.g. precedence relations). The ABVE partner enterprises will decompose their main task in subtasks. Some examples of tasks and sub-tasks performed by the agents of the *ABVE-Construct* model are given in *Table 3.2.* The Construction Agent will decompose the task Building Construction Execution (T3) in seven sub-tasks: foundation work, bricklaying and roof skeleton work, ceiling and floor work, mortar work, carpentry work, roof work and painting.

3.7. The ABVE-Construct Ontology. The *ABVE-Construct* model uses an ontology derived from a generic ontology of a VE as discussed in [12]. The ontology integrates a basic ontology with terms specific to any virtual enterprise and a specialized ontology with terms from the civil engineering domain that were extracted

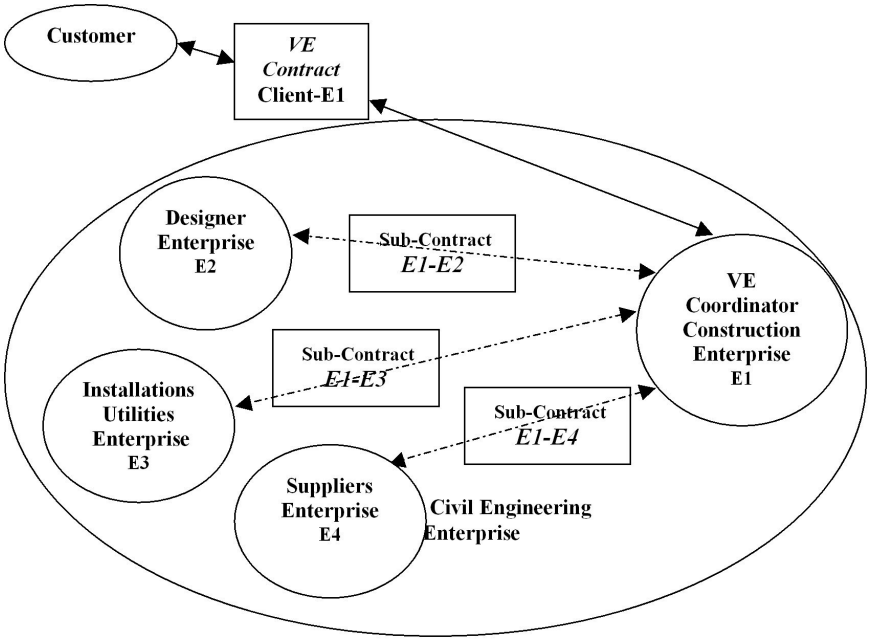

Fig. 3.3: The virtual enterprise generic architecture for civil engineering

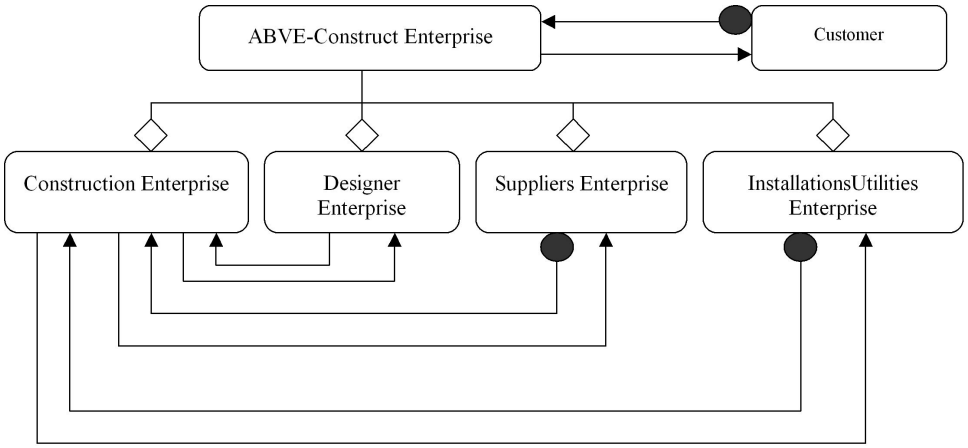

Fig. 3.4: The UML roles diagram for the ABVE-Construct model

mainly from [7]. In this specialized ontology we have added terms from other domains such as building design, building installations and domestic services. A first version of the ontology was developed in Protégé [37], a Java-based ontology editor. Examples of terms from the *ABVE-Construct* ontology are as follows. Basic VE terms: *Activity*, *Resource*, *VE-Coordinator*, *VE-Partner*, *Personnel*, *Tools*, *Equipment*, *BusinessGoal*. Specialized terms: *Residential_Building*, *ArchitectDrawings*, *EngineerDrawings*, *Construction_Materials*, *Architect*, *Engineer*, *bricks*, *BuildersPlant* etc. Examples of relationships between terms are: *isa* (is a), *ako* (a kind of), *part_of*, *has*. *Figure 3.5* shows a screenshot from the *ABVE-Construct* ontology. Each term, viewed in Protégé as a class, is described by a set of slots (i.e. attributes) that have a name, cardinality, type and other facets. Also, the role (abstract, concrete), a documentation (usually, a definition of the term) and a set of constraints (if any) are specified. Details of the class *hallow_brick_10cm* are provided in *Fig. 3.5*.

Table 3.1: The PAGE description for the agents of the ABVE-Construct model

Agent Name	**PAGE Description**			
	Percepts	**Actions**	**Goals**	**Environment**
Construction Agent (Main Contractor)	Customer Requirements List Contract	Pre-Design, Contract Negotiation Partners Search and Selection Sub-contract Negotiation ABVE Formation, Operation, Evolution and Dissolution Building Construction Execution ABVE Coordination ABVE Management	Building construction according to the customer and contract requirements	Multi-agent platform Internet
Designer Agent	Customer Requirements List Engineer Drawings Sub-Contract	Design, Re-Design Revision, Refinement	Building design according to the customer and subcontract requirements	Multi-agent platform Internet
Suppliers Agent	Materials list, Equipment and other Resources Sub-Contract	Provide Construction Materials, Equipment Tools and other Resources necessary to building construction	Supply all needed materials, equipment and other resources according to the sub-contract requirements	Multi-agent platform Internet
Installations Utilities Agent	Building Drawings Sub-Contract	Installations Execution Utilities Connection	All instalations execution and all utilities connection according to the building drawings and sub-contract	Multi-agent platform Internet

3.8. The ABVE-Construct Lifecycle Algorithm. We have formalized the main phases from the *ABVE-Construct* lifecycle under the form of an algorithm, written in pseudocode and shown in *Fig. 3.6.* The main steps of the algorithm are: the ABVE coordinator selection (main contractor selection), the business pre-planning, contract negotiation between the coordinator agent and the customer agent, the ABVE creation, the ABVE operation and evolution and the ABVE dissolution. The selection of the coordinator construction enterprise is realized by using a specific selection mechanism (named SelectM), composed by a Contract Net Protocol (CNP) [30], and a negotiation [25]. In the first phase of the ABVE lifecycle, the ABVE is created and configured. In the operation phase, the ABVE performs the business plan in order to achieve its goal (building construction). The evolution of the ABVE may arise during ABVE operation when exceptional events occur as e.g. the change of the business plan and the temporary incapacity of a partner, events that will lead to partner roles changing or the replacements of a partner. The dissolution phase is initiated when the ABVE achieved its goal. Before dissolution, the coordinator enterprise will assess and register (e.g. in its databases and in some public databases) the partners performance in order to be used by the partners selection tools in

Table 3.2: Examples of tasks and sub-tasks performed by the agents of the ABVE-Construct system.

AGENT NAME	TASKS/SUB-TASKS
Construction Agent (Coordinator Agent)	**Sub Tasks** specific to **Task**: *Building Construction Execution* **TE1-1**: *Foundation work* **TE1-2**: *Bricklaying and roof skeleton work* **TE1-3**: *Ceiling and floor work* **TE1-4**: *Mortar work* **TE1-5**: *Carpentry work* **TE1-6**: *Roof work* **TE1-7**: *Painting,...*
Designer Agent	**TE2-1**: *Building design* **TE2-2**: *Re-design* **TE2-3**: *Refinement,...*
Installations Utilities Agent	**TE3-1**: *Serverage* **TE3-2**: *Heating installation work* **TE3-3**: *Electricity installation work* **TE3-4**: *Telecommunications installation work* **TE3-5**: *Utilities connection*
Suppliers Agent	**TE4-1**: *Supply construction materials* **TE4-2**: *Supply equipment and other resources,...*

the future ABVE creation. A similar assessment and registration can be done by the customer regarding the main constructor enterprise performance.

3.9. The ABVE Creation Algorithm. The ABVE creation phase is detailed in *Fig. 3.7*. After decomposing the business goal in the major subgoals, for each subgoal it is made the search of the interested partners, the selection of the potential partners, the negotiation of the subcontract with these latter ones, and finally, the selection of the agreed partner according to the negotiation results.
The contract between the customer and the main contractor (the ABVE coordinator agent) is defined as follows.

Contract(Customer, Coord)={*BGoal, BWorkPlan, NegotiatedPrice, DutiesObligations, LiabilitiesDefinition, OtherClauses*}

In our case, the contract is an agreement between the customer and the coordinator enterprise (ConstructionAgent) regarding the specific common goal of building construction, and the plan of actions to accomplish it (i.e. the business work plan) as well as the negotiated price for the whole building construction work. The actions included in the business work plan are grouped in tasks that are allocated to the *ABVE-Construct* partners. The contract specifies the duties and obligations of the involved parties. The same is for the subcontracts agreed between the coordinator enterprise and the partners (sub-contractors). The *ABVE-Construct* structure includes the list of partners enterprises with their corresponding subcontracts agreed between each of them and the ABVE coordinator. When the ABVE is initiated by the coordinator, interested partners send their bids that are analyzed by the coordinator and a negotiation will start with the potential partners till an agreement is reached. *Figure 3.8* shows the ABVE-Construct creation phase. The steps 2.1, 2.2 and 2.3 from the ABVE-Creation algorithm are graphically represented in this figure for all the subgoals. The cluster formation (i.e. interested partners search), and the potential partners selection is realized via a Contract Net Protocol. The subcontract is negotiated, agreed and signed after multiple negotiation rounds (with proposals and counterproposals).

3.10. The Partners Selection Mechanism. The partners selection mechanism is based on a Contract Net Protocol [30] and an integrative negotiation of the subcontract with rounds of proposals and counterpropos-

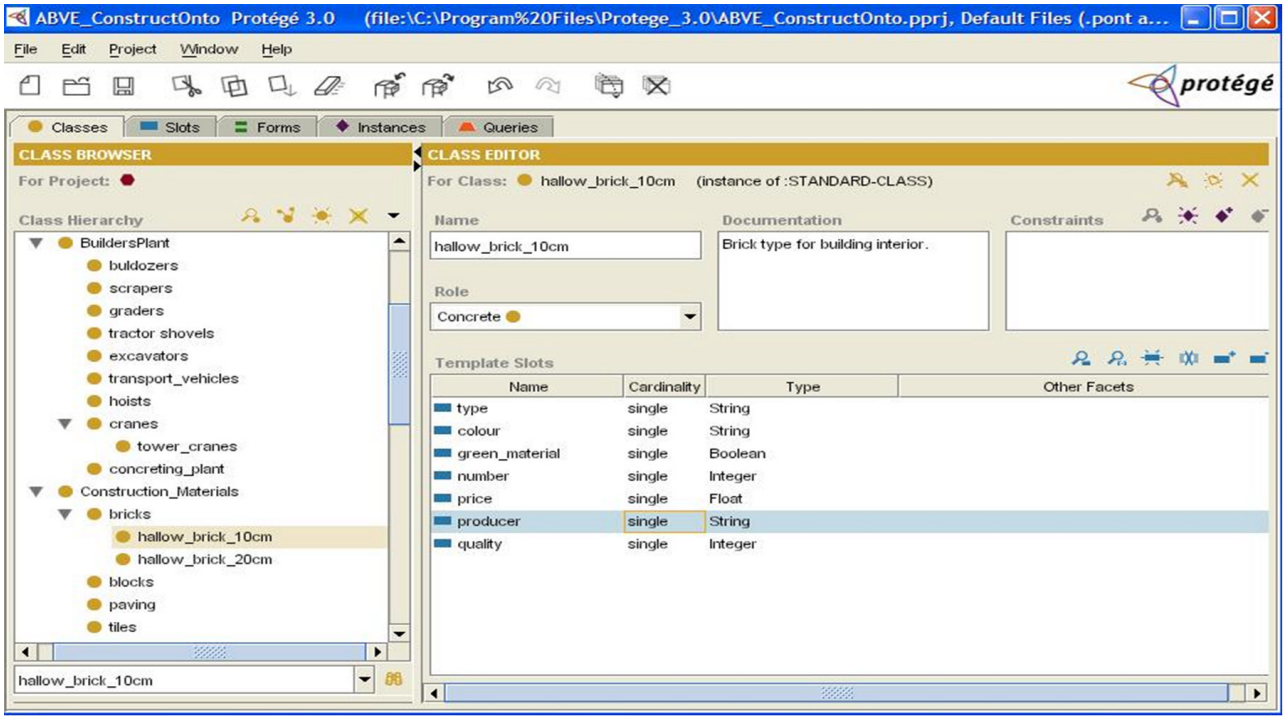

Fig. 3.5: Screenshot from the ABVE-Construct ontology (in Protégé)

als [25] (*Fig. 3.8*). *Figure 3.9* shows an example of an auction-based coordination model during *ABVE-Construct* partners selection.

In our *ABVE-Construct* model we have considered that for the selection of the VE main contractor (the VE Coordinator) and for the partners selection it was applied the same mechanism: a CNP (for coordinator / partners selection from the interested ones) and an integrative negotiation (for contract / sub-contract negotiation). In case the CNP fails, a new call for proposals (CFP) is launched (eventually, with some changes based on the feedback from the first one). In case the contract / sub-contract negotiation fails, it is started a negotiation with the next ranked potential partner (if any, selected during CNP) or a new CFP is launched.

At the end of the ABVE-Creation procedure run, the agent-based VE, *ABVE-Construct* is set and prepared to start the next phase, that of ABVE operation, i.e. of the building construction according to the contract signed between the customer and the main contractor / coordinator enterprise.

3.11. The ABVE Functioning Diagram. The main steps of the ABVE-Construct functioning are as follows:

START ABVE-Construct Business

1. Construction Enterprise selection and contract negotiation → done by the Customer Agent
2. Partners search and selection; sub-contract negotiation; Pre-design of Engineer Drawings; Business work plan setting → done by the Construction Enterprise Agent
3. Designing, Re-designing, Revision / Refinement → done by the Designer Enterprise Agent
4. Supply construction materials, equipment and other resources → done by the Supplier Agent
5. Building Construction work plan execution → done by the Construction Enterprise Agent
6. Installations work plan execution; Utilities connection → done by the InstallationsUtilities Enterprise Agent
7. Building reception → done by the Construction Enterprise Agent and the Customer Agent

Algorithm *ABVE-Construct_lifecycle* (Input, Output)

Input: BGoal, BC_ReqList
Output: Building

Main Agents: CustomerAgent, ConstructionAgent
Agent-Based Virtual Enterprise:
ABVE-Construct = { AGENTS = {ConstructionAgent, DesignerAgent, SupplierAgent, InstallationsUtilitiesAgent } }
Functions: *CoordinatorSelection(…), BusinessPrePlanning(…), Negotiation(…), ABVE_Creation(…), ABVEOperationEvol(…), ABVE_Dissolution(…)*

Begin
/* select the coordinator enterprise (main construction enterprise) according to the business goal (BGoal) and the customer list of building construction requirements (BC_ReqList) by using a selection mechanism (SelectM) */

1. ConstructionAgent = *CoordinatorSelection*(BGoal, BC_ReqList, SelectM);

/* business process pre-planning done by the coordinator agent (ConstructionAgent) and the customer agent (CustomerAgent) - the initial business work plan (BWP) is created */

2. BWP = *BusinessPrePlanning*(ConstructionAgent, CustomerAgent)

/* contract negotiation */

3. Contract = *Negotiation*(CustomerAgent, ConstructionAgent, BWP);

/* creation of the ABVE-Construct agent-based virtual enterprise */

4. ABVE-Construct = *ABVE-Creation*(ConstructionAgent, Contract, BGoal, BWP, BC_ReqList);

/* operation and evolution of the ABVE-Construct */

5. Building = *ABVE-OperationEvol*(ABVE-Construct, Contract, BWP);

/* dissolution of the ABVE-Construct when the business goal (Building Construction) is achieved */

6. *ABVE-Dissolution*(ABVE-Construct);

return Building;
End.

Fig. 3.6: The generic algorithm for the ABVE-Construct lifecycle

8. Final payment → done by the Customer Agent
STOP ABVE-Construct Business

Figure 3.10 shows the diagram of the *ABVE-Construct* functioning.

An example of an interaction diagram for a scenario of the *ABVE-Construct* lifecycle is given in *Fig. 3.11*. The messages exchanged between the agents, the tasks executed by agents, the ABVE creation and the ABVE dissolution phases are marked in the diagram. The interactions are chained in chronological order. The tasks are represented as dashed rectangles.

3.12. ABVE-Construct System Design. We have used for the *ABVE-Construct* system design the Prometheus Design Tool (PDT) version 2.5 [36], a graphical editor, written in Java, which supports the agent-based systems design within the Prometheus methodology (described in the handbook [4]). The main phases of the methodology are: system specification, architectural design and detailed design. Each phase provides a set of diagrams and textual descriptors in terms of actors/agents, percepts, messages, scenarios, protocols, goals depending on the type of diagram. The *ABVE-Construct* business goal, *Building Construction Business Goal*, was decomposed in four major subgoals (the root node is an AND node): *Building Design*; *Supply Construction Materials, Equipments and Resources*; *Building Construction Execution*; *Installations Execution and Utilities*

Algorithm *ABVE-Creation* (Input, Output)

Input: ConstructionAgent, Contract, BGoal, BWP, BC_ReqList
Output: ABVE-Construct

Agent-Based Virtual Enterprise: ABVE-Construct = {{ConstructionAgent, DesignerAgent, SupplierAgent, InstallationsUtilitiesAgent}}
Functions: *GoalDecomposition(...)*, *PartnersSearch(...)*, *PartnersSelection(...)*, *Negotiation(...)*

```
Begin
 /* performs business goal decomposition in n major subgoals */
  1. SubGoals = GoalDecomposition(BGoal)
 /* search and selection of the partners enterprises for each business major subgoal */
  2. For  *each SubGoal = SubGoals[i], i=1, .., n  do
          /* apply a Contract Net Protocol and subcontract negotiation */
          /* search the interested partners (IPartnersList) – cluster formation */
          2.1 IPartnersList = PartnersSearch(SubGoal, BC_ReqList)
          /* select the potential partners (P_Partners) – ABVE-Construct formation */
          2.2 P_Partners = PartnersSelection(IPartnersList)
          /* negotiation of the SubContract */
          2.3 SubContract = Negotiation(ConstructionAgent, P_Partners, BWP)
          2.4 ABVE-Construct.Partner[SubGoal] = SubContract.SelectPartner
          2.5 ABVE-Construct.Partner[SubGoal].SubContract = SubContract
  3. ABVE-Construct.Coordinator = ConstructionAgent
  4. ABVE-Construct.Coordinator.Contract = Contract
  5. ABVE-Construct.BWorkPlan = BWP;
  6. ABVE-Construct.Coordination = ContractNetProtocol;
  7. ABVE-Construct.Goal = BGoal
 return ABVE-Construct;
End.
```

Fig. 3.7: The ABVE-Creation algorithm

Connection. Figure 3.12 shows the ABVE-Construct goal overview diagram.

The *ABVE-Construct* system roles diagram is given in *Fig. 3.13* and the *ABVE-Construct* system overview diagram is shown in *Fig. 3.14*.

4. Case study. We have considered as a case study a specific building construction task: residential building construction, with the following scenario. Before the creation of the ABVE, the customer (a person or an institution) will announce the business goal (i.e. residential building construction) with all the customer requirements given in *Fig. 4.1*, and will receive bids from the interested civil engineering enterprises as main contractors.

The customer will select the agreed construction enterprise, and the next steps of the ABVE-Construct lifecycle will be run according to the algorithm given in *Fig. 3.6*. In this section we discuss the Designer Enterprise selection from the *ABVE-Construct* creation phase. *Figure 4.2* shows the interaction diagram for this case study. We have represented the selection of the Construction Enterprise and of the Designer Enterprise.

In the first phase, the Construction Enterprise selection, the Customer Agent will announce the building construction task as a business goal, with the customer list of requirements (e.g. *Fig. 4.1*). Three interested construction enterprises were chosen: A1, A2, A3. During the second phase, the ABVE creation is initiated.

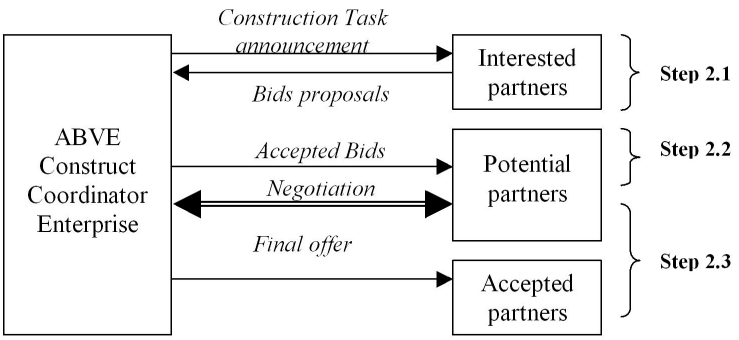

Fig. 3.8: The ABVE-Construct creation (Contract Net Protocol and negotiation)

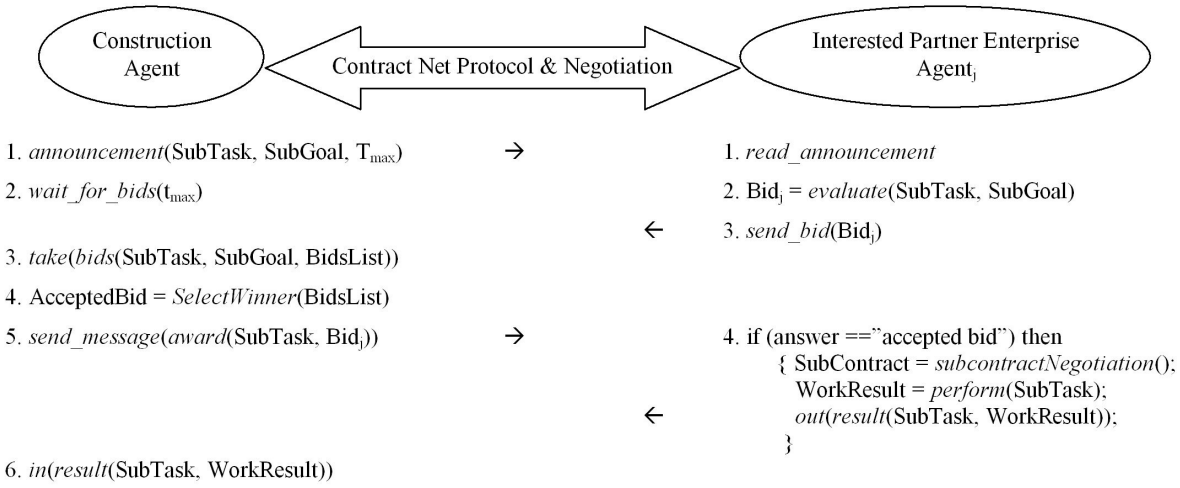

Fig. 3.9: The ABVE-Construct subtasks allocation - Partners selection via CNP and Negotiation

For the first sub-task, Building Designer Enterprise selection two interested enterprises were chosen: P1, P2.

The ABVE Coordinator list of preferences for partners selection includes the following criteria: partners reputation, performance, advanced technology use, price, lead time, quality, delivery, service, other criteria (e.g. environmental and energy issues). An example of the first announcement for the Building Designer Enterprise selection is given as follows:

```
ABVE-Announcement
        goal(BuildingConstruction, residential)
           roles(BuildingDesigner, Skills)
           (requirements requirementsList))
```

where

```
Skills: List_of_DesignerSkills(CAD, EcologicalDesign)
        experience: >= 10 years
        availability:
            Start_date: June 1, 2013
            End_date: July 31, 2013
```

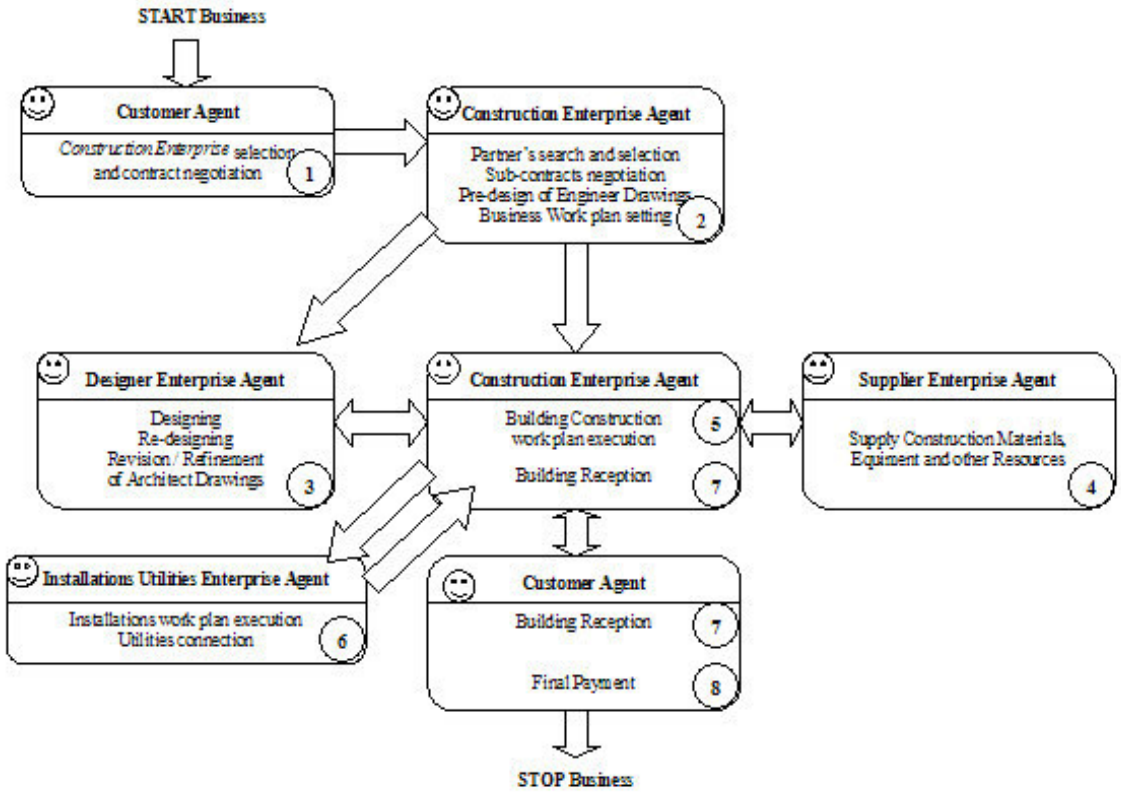

Fig. 3.10: The diagram of the ABVE-Construct functioning

cost_per_hour < 70
performance rating: [1, 10], >= 7
commitment: [1, 10], >= 7

The two interested partners send bids with values for the required attributes, as for example:

bid_skill(P2_Designer,
role(BuildingDesigner)
attributes (skills ([CAD, EcologicalDesign])),
experience_no_years(15),
performance_rating(8),
commitment(8)).

For our scenario, the ABVE-Construct system included the enterprises: A1 (Construction Coordinator Enterprise), and P2 (Building Designer Enterprise). In a similar way, the other partners are selected by the ABVE Coordinator.

In this case study, the VE Coordinator and partners selection mechanism followed the procedure described in *Figs. 3.8 and 3.9*, i.e. a CNP for the selection of potential VE Coordinator / VE partner, and an integrative negotiation for contract / sub-contract negotiation.

4.1. The ABVE-Construct Model Adaptation. The *ABVE-Construct* model can be adapted and extended to be used in other domains of application. The main changes options that depend on the application domain are: the extension of the roles diagram with the specific enterprises (main contractor / coordinator,

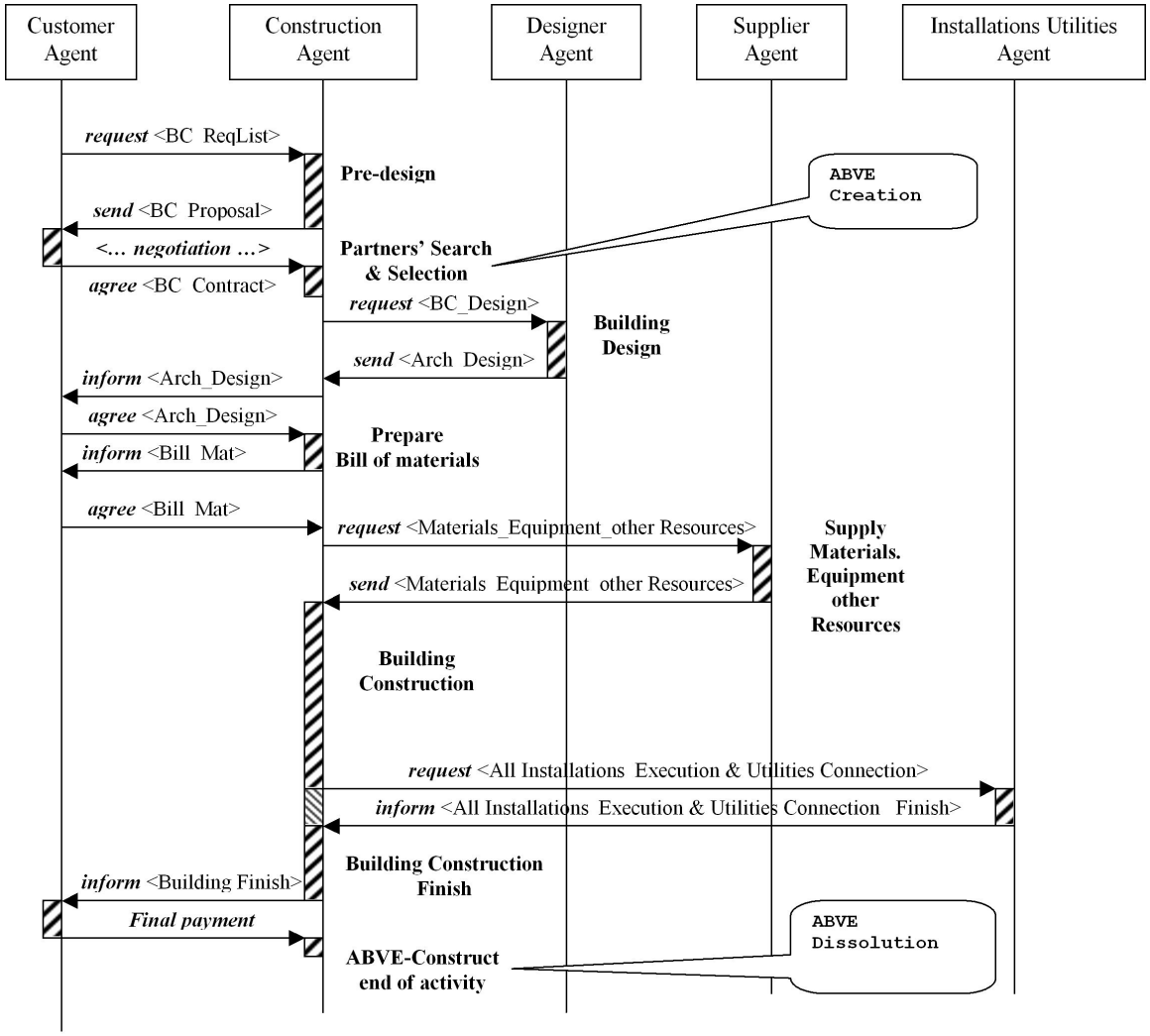

Fig. 3.11: The interaction diagram for a scenario of the ABVE-Construct lifecycle

partners), the description of the agents and tasks (set according to the new roles diagram), the inclusion of the domain specialized ontology, the inclusion of the specific ABVE functioning diagram, the adaptation of the two algorithms (ABVE lifecycle, ABVE Creation) to the new domain. The generic architecture will include the new enterprises while the organizational form could be kept or changed according to the new business domain. The partners selection mechanism may need no changes, being domain independent.

5. Conclusion. We have presented a generic framework, *VE-Frame*, for virtual enterprises development, and an agent-based virtual enterprise model, *ABVE-Construct*, for civil engineering, that was derived from *VE-Frame*. The framework was generated by following the main phases of software development: analysis, design, implementation and testing. *VE-Frame* is based on ontologies, a basic ontology for any VE, and specialized ontologies with terms from the domains of building design, building construction execution, building installations execution and domestic services connection. The *ABVE-Construct* model is defined by a generic architecture with a sub-contracting organizational form, a set of diagrams (roles, ABVE functioning), a PAGE description of the agents with their associated tasks, the algorithms for the ABVE lifecycle and ABVE Creation, the ABVE-Construct ontology, a partners selection mechanism based on CNP and integrative negotiation. The model

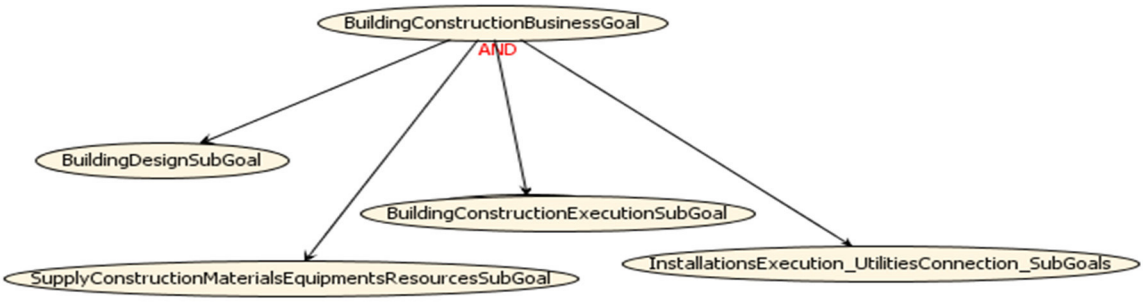

Fig. 3.12: The ABVE-Construct goal overview diagram (PDT)

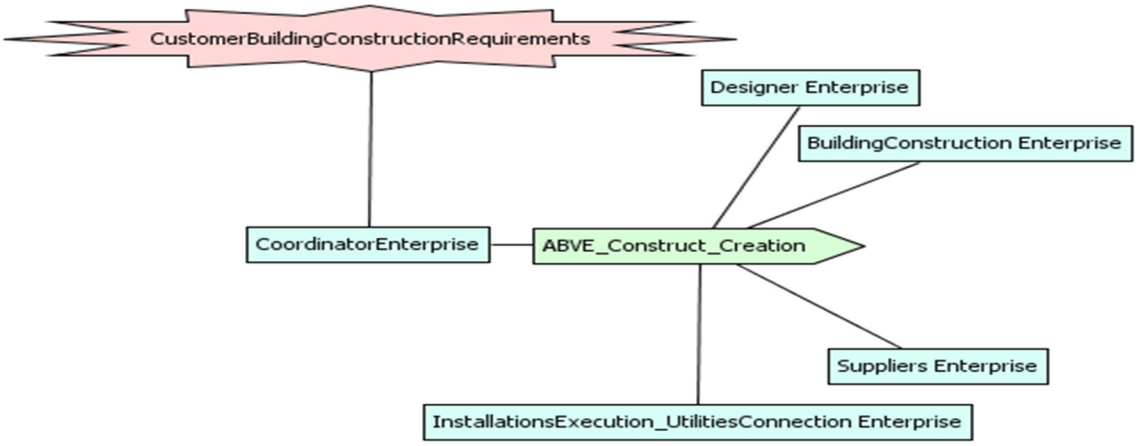

Fig. 3.13: The ABVE-Construct system roles diagram (PDT)

can be adapted and extended to other application domains such as pharmaceutical industry, food industry, agribusiness industry.

REFERENCES

[1] S. Abdallah and V. Lesser, *Organisation-based coalition formation*, in Proc. of the 3rd Int. Joint Conf. on Autonomous Agents & Multi-Agent Systems, vol. 3, 2004, pp. 1296–1297.

[2] A. Avila-Rosas and M. Luck, *A direct reputation model for virtual organization formation*, in Proc. of CEEMAS 2005, LNAI 3690, 2005, pp. 460–469.

[3] C. Bădică, S. Ilie, M. Kamermans, G. Pavlin, A. Penders, and M. Scafeş, *Multi-agents systems, ontologies and negotiation for dynamic service composition in multi-organizational environmental management*, in Software Agents, Agent Systems and Their Applications, NATO Science for Peace and Security Series - D: Information and Communication. Security, IOS Press, vol. 32, 2012, pp. 286–306, http://dx.doi.org/10.3233/978-1-60750-818-2-286.

[4] F. Bergenti, M. -P. Gleizes, and F. Zambonelli, *Methodologies and software engineering for agent systems - The Agent-oriented Software Engineering Handbook*, Kluwer Academic Publishers, 2004.

[5] D. Cabrera-Paniagua, G. Herrera, C. Cubillos, and M. Donoso, *Towards a model for dynamic formation and operation of virtual organizations for transportation*, Studies in Informatics and Control, 20 (2011), pp. 255–264.

[6] L. M. Camarinha-Matos and H. Afsarmanesh, *Virtual enterprise modelling and support infrastructure: applying multi-agent system approaches*, Multi-Agent Systems and Applications, LNAI 2086, (2001), pp. 335–364.

[7] R. Chudley and R. Greeno, *Building construction handbook*, seventh edition, Elsevier, 2008.

[8] G. Davoli, S. A. Gallo, and R. Melloni, *VirtES (Virtual Enterprise Simulator): A proposed methodology for enterprise simulation modeling*, in B. Vallespir and T. Alix (eds), APMS 2009, IFIp AICT 338, 2010, pp. 374–380.

[9] J. Du and M. El-Gafy, *Virtual organizational imitation for construction enterprises: Agent-based simulation framework for exploring human and organizational implications in construction management*, in Journal of Computing in Civil

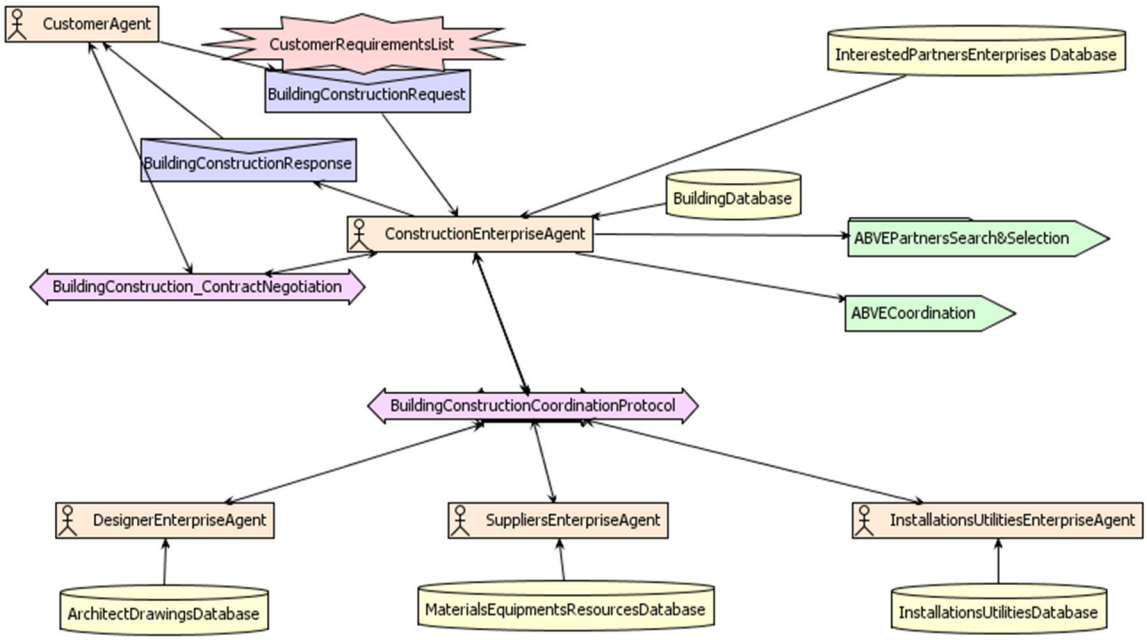

Fig. 3.14: The ABVE-Construct system overview diagram (PDT)

Engineering, 26, 2012, pp. 282–297.

[10] M. Erlandsson and M. Borg, *Generic LCA-methodology applicable for buildings, constructions and operation services-today practice and development needs*, Building and Environment, 38 (2003), pp. 919–938.

[11] B. Fidler and Š. Valčuha, *Modelling methodology for development of virtual organization's supporting systems*, Journal of Achievements in Materials and Manufacturing Engineering, 25, 2007, pp. 85–90.

[12] M. S. Fox, M. Bărbuceanu, and M. Gruninger, *An organisation ontology for enterprise modelling: Preliminary concepts for linking structure and behaviour*, in Computers and Industry, 29, 1996, pp. 123–134.

[13] F. -S. Hsieh and J. -B. Lin, *Context-aware workflow management for virtual enterprises based on coordination of agents*, Journal of Intelligent Manufacturing, (2012), doi 10.1007/s10845-012-0688-8.

[14] B. Huang, C. Gao, and L. Chen, *Partner selection in a virtual enterprise under uncertain information about candidates*, Expert Systems with Applications, 38 (2011), pp. 11305–11310.

[15] C. L. Hwang and K. Yoon, *Multiple attribute decision making: methods and applications*, Springer, 1981.

[16] J. Irizarry, E. P. Karan, and F. Jalaei, *Integrating BIM and GIS to improve the visual monitoring of construction supply chain management*, Automation in Construction, 31 (2013), pp. 241–254.

[17] E. A. Kendall, M. T.Malkoun, and C. Jiang, *A methodology for developing agent based systems for enterprise integration*, IFIP Working Conference of TC5 Special Interest Group on Architectures for Enterprise Integration, 1995.

[18] M. Matskin, O. J. Kikeluten, S. B. Krossnes, and O. Saele, *Agora: An infrastructure for cooperative work support in multi-agent systems*, in T. Wagner, O. Rana (eds), Infrastructure for Agents, Multi-Agents, and Scalable Multi-Agent Systems, Springer, LNCS 1887, (2001) pp. 28–42.

[19] A. Motro and Y. Guo, *The SOAVE Platform: A service-oriented architecture for virtual enterprises*, in L.M. Camarinha-Matos, L. Xu, and H. Afsarmanesh (eds): PRO-VE 2012, IFIP AICT 380, 2012, pp. 216–224.

[20] A. Mohamed and T. Celik, *An integrated knowledge-based system for alternative design and materials selection and cost estimating*, Expert Systems with Applications, 14, 1998, pp. 329–339.

[21] E. Oliveira, A. P. Rocha, *Agents' advances features for negotiation in electronic commerce and virtual organisation formation process*, in European Perspectives on Agent Mediated Electronic Commerce, Springer, 2000.

[22] M. Oprea, *The development of an agent-based virtual enterprise for civil engineering - a preliminary report*, in Proc. of the 17th Int. Conf. on System Theory, Control and Computing, WASA-4, 2013.

[23] M. Oprea, *A case study of agent-based virtual enterprise modelling*, in Proc. of CEEMAS 2005, LNAI 3690, 2005, pp. 632–635.

[24] M. Oprea, *Applications of Multi-Agent Systems*, in R. Reis (ed), Information Technology - Selected Tutorials, Kluwer Academic Publishers, 2004, pp. 239–270.

[25] M. Oprea, *Coordination in an agent-based virtual enterprise*, Studies in Informatics and Control, 12, 2003, pp. 215–225.

[26] A. L. Osório, H. Afsarmanesh, and L. M. Camarinha-Matos, *A service integration platform for collaborative networks*, Studies in Informatics and Control, 20, 2011, pp. 19–30.

CUSTOMER REQUIREMENTS LIST:
BUSINESS GOAL: BUILDINGCONSTRUCTION
BUILDING TYPE: RESIDENTIAL;
SITE AREA: 200 m^2;
LOCATION: I3-J10 - CITY MAP – BUCHAREST
DEADLINE: 30/04/2014;
PLANNING AND COST ESTIMATING: AFTER DETAILED DESIGN;
QUALITY: FIRST CLASS;
BUILDING FIRST PLAN SKETCH: PLAN SKETCH - VERSION 1, …
    NUMBER OF FLOORS: 1;
    GROUND FLOOR:
        1 ENTRANCE HALL, 1 DINING ROOM, 1 STUDY ROOM,
        1 BATHROOM, 1 KITCHEN, 1 PANTRY;
    FIRST FLOOR:
        3 BEDROOMS, 1 BATHROOM, 1 HALL;
    STAIRCASE: 1;
ROOF TYPE: PITCHED;
HVAC: CENTRAL HEATING;
ENVIRONMENTAL & ENERGY ISSUES:
    ECOLOGICAL BUILDING: YES;
        GREEN MATERIALS, GREEN PRODUCTS, …
    REDUCED ENERGY CONSUME: YES;
        SAVING ELECTRIC AND HEAT ENERGY;
    EXTERIOR ISOLATION: YES;
    ALTERNATIVE GREEN ENERGY SOURCES: YES, SOLAR;
FLOOR COVERING: WOOD TILES, CERAMIC TILES;
INTERIOR: HALLOW BRICK 10cm;
EXTERIOR: HALLOW BRICK 20cm;
OTHER PREFERENCES: …

Fig. 4.1: Example of customer requirements list

[27] S. A. PETERSEN AND M. MATSKIN, *Agent interaction protocols for the selection of partners of virtual enterprises*, in Proc. of CEEMAS 2003, LNAI 2691, 2003, pp. 606–615.
[28] A. PRESLEY, J. SARKIS, W. BARNETT, AND D. LILES, *Engineering the virtual enterprise: An architecture-driven modeling approach*, The Int. Journal of Flexible Manufacturing Systems, 13, 2001, pp. 145–162.
[29] B. L. SADIGH, H. Ö. ÜNVER, AND S. E. KILIÇ, *Design of a multi-agent based virtual enterprise framework for sustainable production*, in G.D. Putnik and M.M. Cruz-Cunha (eds): ViNOrg 2011, CCIS 248, Springer-Verlag, 2012, pp. 186–195.
[30] R. G. SMITH, *The contract net protocol: High-level communication and control in a distributed problem solver*, IEEE Transactions on Computers, 29, 1980, pp. 1104–1113.
[31] X. WANG, T. N. WONG, AND G. WANG, *An ontological intelligent agent platform to establish an ecological virtual enterprise*, Expert Systems with Applications, 39, 2012, pp. 7050–7061.
[32] C. YU AND T. N. WONG, *Multiple products partner selection model of virtual enterprise based on multi-agent systems*, in Proc. of World Congress on Engineering, 2011, II.
[33] G. ZAPATA-POVEDA AND C. TWEED, *Official and informal tools to embed performance in the design of low carbon buildings. An ethnographic study in England and Wales*, Automation in Construction, 37, 2014, pp. 38–47.
[34] A. ZARLI AND P.POYET, *A framework for distributed information management in the virtual enterprise: The VEGA project*, 1999.
[35] Q. ZHAO, X. ZHANG, AND R. XIAO, *Particle swarm optimization algorithm for partner selection in virtual enterprise*, Progress in Natural Science, 18, 2008, pp. 1445–1452.
[36] PROMETHEUS DESIGN TOOL, *http://www.cs.rmit.edu.au/agents/pdt*.
[37] PROTÉGÉ-2000, *http://protege.stanford.edu*.

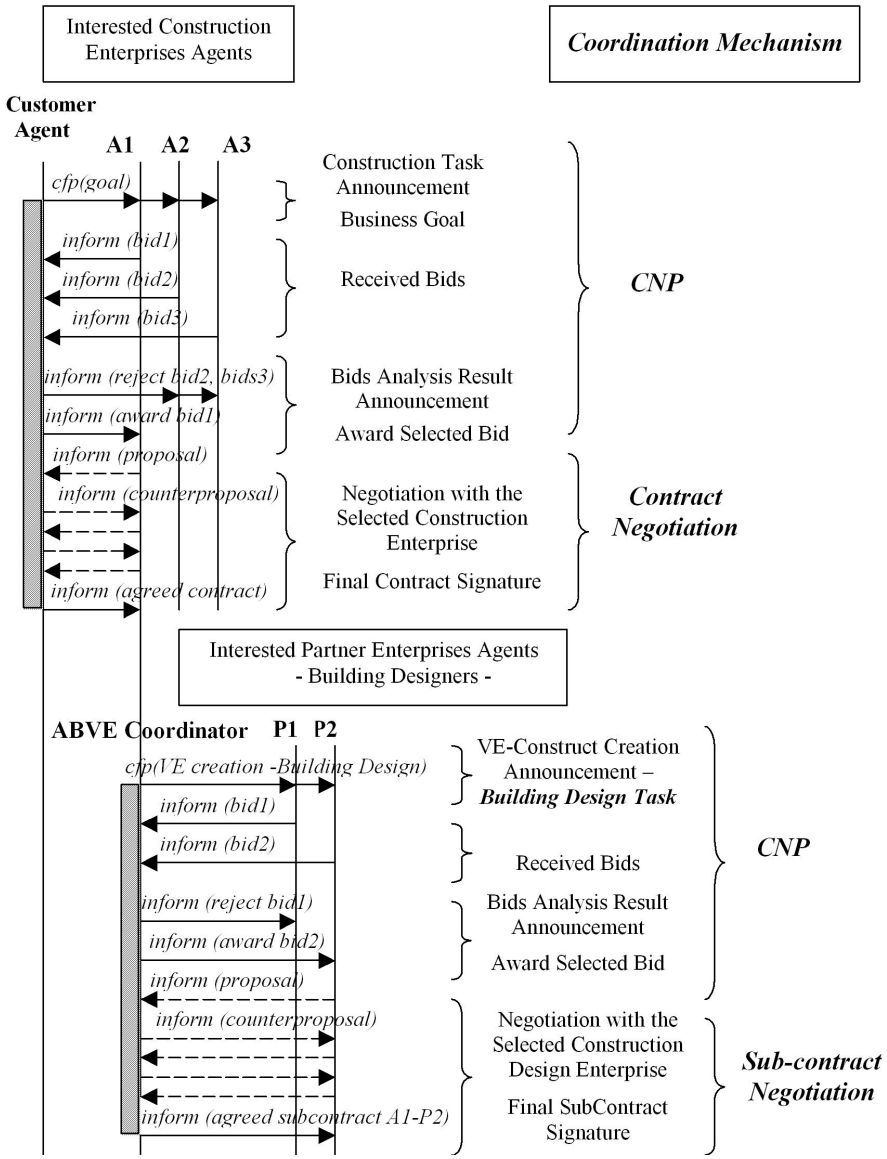

Fig. 4.2: The interaction diagram for the Construction Enterprise selection and ABVE creation Building Designer Selection

Edited by: Costin Bădică
Received: May 9, 2014
Accepted: Sept 21, 2014

DOI 10.12694/scpe.v15i3.1019
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. 251–270. http://www.scpe.org

PORTABILITY IN CLOUDS: APPROACHES AND RESEARCH OPPORTUNITIES

DANA PETCU* AND ATHANASIOS V. VASILAKOS†

Abstract. The migration towards Cloud environments is still hindered by several barriers. One of them is the low portability of the applications that are consuming Cloud services. This paper intends to provide an image of the state of the art in this particular topic and to identify the potential paths to follow in order to solve the problem. The main concerns are the portability reasons, scenarios, taxonomies, measurements, requirements, and current technical solutions (through open standards, open application programming interfaces, semantics and model-driven engineering). A research agenda is following the current analysis of the state of the art.

Key words: Cloud computing, Portability, Multi-Clouds

AMS subject classifications. 68M14, 68W15, 68U01

1. Introduction. Portability is know as being the ability to use software components on multiple hardware or software environments. In particular, the portability in Cloud or portability between Clouds, or shortly Cloud portability in this paper, is expected to ensure that an application, service or data works in the same manner regardless of the consumed Cloud services and has a common method of interaction with these services.

The Cloud portability problem has been recognized already from the beginning of Cloud computing (e.g. in [1]). More recently, the IDC report from 2012 [2] points that portability is one of the most important obstacle hindering increasing Cloud adoption. However, the work on Cloud portability is gaining now momentum, as demonstrated by recent standardization initiatives for Cloud portability (e.g. TOSCA [3]). Cloud providers are nowadays also concerned about portability (e.g. IBM and HP are supporting to OpenStack initiative, while Amazon and Microsoft are investing in virtual machine portability [4]).

Several research papers, proof-of-concept implementations, or commercial products, have dealt until recently with the portability problem. However, Gonidis et al [5] observed that there is no extensive survey to describe in detail the problem space of Cloud application portability and how current solutions are mapped to that space. They underlined that such a study is essential in order to understand the root causes of the problem and the desirable characteristics of solutions to this problem. A first step has been done by them in [5], and we intend to continue in this paper their survey, as well as our earlier attempts in the same direction. More precisely, Petcu et al [6] proposed also a classification of the portability levels, while earlier Petcu [7] has underline the difference between interoperability and portability.

The novelty of this paper is two-fold. First, it gathers in one place the latest solutions. Second, it identifies the missing pieces needed to solve the problem. Therefore, the paper results can be useful to researchers in Cloud computing who need an identification of relevant studies, as well as to practitioners interested in understanding the available methods, techniques and tools and their maturity level in supporting portability in Cloud environments.

The structure of the paper is as follows. Section 2 is discussing the need for portability. Section 3 is defining the problem and the requirements. Section 4 investigates the current solutions. Section 5 is identifying the current barriers, challenges, open issues and the future solution path finding. Section 6 draw the conclusions.

2. The need for Cloud portability. Cloud computing is currently recognized as a computing model based on ubiquitous network access to a shared and virtualized pool of computing capabilities (like processing, network, storage, message queuing) that can be rapidly provisioned [8]. However this conceptual model is implemented by a large number of service providers in very different ways. The current software stacks of Cloud services are heterogeneous and the provided features that are often incompatible between different service providers. This diversity is an obstacle with respect to demands such as promoting portability and preventing vendor lock-in.

* West University of Timişoara, and Institute e-Austria Timişoara, Romania, (petcu@info.uvt.ro).
† University of Western Macedonia, Greece, (vasilako@ath.forthnet.gr).

The portability is requested by reasons varying from optimal selection regarding utilization, costs or profits, to technology changes, as well as legal issues. We discuss them shortly in what follows.

2.1. Economical reasons. Cloud portability is needed from an economical point of view for at least two reasons:

1. Customer: protection of the end user investments in developments;
2. Market: development of a Cloud eco-system and market.

Optimizing the operational cost is often the reason that is behind the interest of the Cloud service customer in Cloud portability. Rewriting the services and applications is usually required to comply with the change of the Cloud service. The porting process of customer application and data is also often triggered by changes of the offers of the providers.

The Cloud service providers can be interested to enhance their own Cloud resource and service offers to be more attractive in a highly dynamic market mainly by improving the quality of their resources and services. However agreements with other providers are possible (or even overtakes) in the case they provide services with particularities not provided elsewhere or complementary resources. The porting process of services is usually triggered by the operational changes.

If the Cloud portability is ensured, third parties are able to be developed, acting as intermediaries between multiple customer and multiple Cloud providers, enabling deployments according the customer requirements to the appropriate Cloud and adaptation to the current status and consumption conditions of the multiple Cloud services. The porting process of data, applications or services is usually triggered at customer request or automated by the third party.

2.2. Technical reasons. Cloud portability is needed from a technical point of view for at least two reasons:

1. Concept: to exploit the advantage of elasticity and pay-as-you concept;
2. Continuity: to ensure continuity in application and service functionality.

The portability between Private and Public Clouds is essential in realizing the vision of the Hybrid Cloud that handles the peaks in service and resource requests addressed to a Private Cloud using external resources of a Public Cloud. The porting process is usually triggered on demand basis.

The Cloud service providers are interested to use other provider services to ensure backup-ups to deal with disasters or scheduled inactivity. The porting process is usually triggered according to the providers agreements.

The replication of data and applications consuming services from different Clouds can be used to ensure their high availability. The porting process is usually triggered at the design phase.

2.3. Legal reasons. Cloud portability is needed from a legal point of view for at least two reasons:

1. Constraints: data, application and service protections according to the locations or national laws;
2. Free will: avoid the dependence on only one external provider.

Changes in a country legislation or changes in the customer location can trigger the need to port the software assets from a Cloud environment to another.

The Cloud non-portability of application, data, applications or services relaying upon a certain Cloud provider faulty service can produce a cascade effect on the activities of its service dependent customers. In this context the portability is necessary to reach a good recovery time objective.

Moreover, public institutions using auctions for acquiring services are reluctant to be engaged in contracts which are not offering proper services or guarantees in case of incidents.

3. Problem definition. As stated earlier, portability is the ability to move software assets among different runtime platforms without having to rewrite them partly or fully (enhancement of the definition given by Lenhard and Wirtz [9]). If the Cloud portability is achieved, data, application or service components can be moved and reused regardless of the operating system, storage or application programming interface (API).

In what follows we discuss some scenarios and attempts for classifications of various cases as well as the basic terminology.

3.1. Portability scenarios. The real world scenarios are highlighting two main categories of portability scenarios encountered in current Cloud service market (mentioned first by Ranabahu and Sheth [10]):

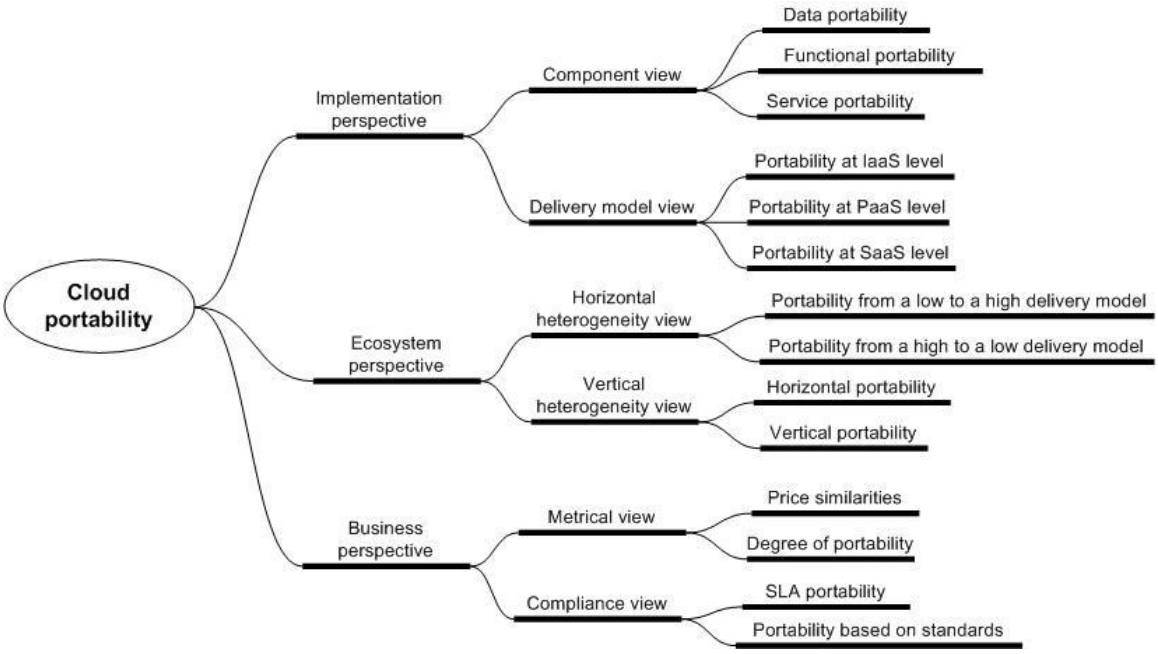

Fig. 3.1: Cloud Portability Taxonomy

1. Porting legacy applications, data or services: the legacy software assets are currently requiring a significant effort to be invested in porting them into in Cloud environment due to the dependence on particular technologies and data organization, on both sides, the initial environment as well as the Cloud environment.
2. Porting Cloud native applications, data or services: even when applications, data or services are written from scratch in a Cloud environment, they are usually locked and targeted for a specific Cloud, and the effort for porting in a different Cloud is usually a one time exercise.

Porting between Clouds of all or a part of existing services, applications or data is usually done one time. However a third party running services on multiple Clouds and offering unique entry points to various service customers is interested to ensure that the porting process is reversible, fast and semi-automated.

The most challenging scenario for portability is that in which the Cloud applications are distributed across several administrative domains of different providers simultaneously, and, moreover, at least data (if not even application and service components) are ported from one Cloud environment to another.

A less discussed scenario is the reverse portability from the Cloud environment towards the own premises resources (even partially, lets say only the data part). This is critical when the service provider changes the API or imposes API restrictions or depreciate a service, so that all depenedent services fall (domino effect).

3.2. Portability taxonomy. Figure 3.1 introduces a taxonomy of Cloud portability based on the studies that are detailed bellow.

Kolb and Wirtz [11] categorized Cloud portability from three different perspectives: business, ecosystem and implementation (while the study was done for platform-as-as-service level, it is generally valid also for the other service delivery models):

1. Implementation perspective: is related to portability constraints that are implementation-specific requirements or restrictions, e.g. deployment descriptors, restricted usage of runtime APIs or specific management API calls.
2. Ecosystem perspective: describes concrete portability constraints including application specific dependencies like runtimes and specific services.

3. Business perspective: is the most abstract perspective and includes business relevant, non-functional and abstract constraints on portability, like pricing, compliance or SLAs.

From the implementation perspective, depending on the software assets that are ported, the Cloud portability can be classified in three categories (first mentioned by Oberle and Fisher [12]):

1. *Data portability:* enables the re-use of data components between Clouds. It is based on import and export functionality of Cloud data services for data structures.
2. *Functional portability* or *application portability* refers to the Cloud service agnostic definition of application functionality.
3. *Service portability* or *platform portability* is the ability to add, reconfigure and remove Cloud resources on the fly, independent on the Cloud provider.

Data portability is achieved when application data can be retrieved from one provider and imported into an application hosted by another provider. To reach this type of portability, platform-independent data representation is needed to be done in a short-term, as well as specific target representations and code for the application's data access layer. On long term, data portability depends on standardization of import and export functionality of data and its adoption by the providers that are the market key-players.

Functional or application portability requires the definition of the application's functionality details in a vendor-agnostic manner. A particular issue is the application portability between development and operational environments. Cloud computing is bringing development and operations closer together, the two being integrated as DevOps. This requires that the same environment is used for development and operation, and the application is portable between development and operation environments. If different environments are used the application portability can become an important issue.

Service or platform portability is based on Cloud-agnostic control APIs allowing Cloud resources to be added, configured, or removed in real time by humans or programmatic, based on factors like current outages. In the case of service or platform portability there are two scenarios:

1. Service component portability: service or platform components are ported between Clouds, without involving the applications and data on top of them;
2. Machine image portability: bundles containing applications and data and their services or platforms are ported between Clouds.

The portability expectations are different for each service delivery model:

IaaS: The main scenario is related to the migration of applications, data and services to a new IaaS environment.

PaaS: The aim is to minimize the amount of application rewriting while preserving or enhancing controls, and a successful data migration.

SaaS: The main scenario supposes that the Cloud customer is substituting software applications with new ones. The aim is to preserve or enhance the application functionality.

For example, in the case of PaaS, specific points of conflict emerging when attempting to deploy an application to multiple platforms are the followings (according Gonidis et al [5]): programming languages and frameworks, platform-specific services, data stores and platform specific configuration files.

The Cloud service eco-system and market has currently a high degree of heterogeneity. This heterogeneity is classified by Ranabahu and Sheth [10] and as being vertical or horizontal. In the first case the heterogeneity appears within the same delivery model. For example, in IaaS case, raw infrastructure resources are exposed as services and various resources are exposed differently between IaaS providers. In the horizontal case, the heterogeneity appear within different service delivery models. For example, a typical PaaS requires the use of specialized libraries and run-times with limited functionalities. A IaaS native application assuming the control over operating system requires extensive changes when moving into a PaaS environment. Note that the split according vertical and horizontal views is not yet commonly accepted, as the same terms are reused in other contexts in solving portability issues. In particular, Jonnalagedda et al [13] distinguish two major cases:

1. *Vertical portability* as the capability of an application intended for a Private Cloud to be portable to a Public Cloud. The restriction is that the application runs on the same technology stack on both platforms.
2. *Horizontal portability* as the ability to port an application from one technology stack to another, staying

at the same level of abstraction but changing the technology provider.

3.3. Measuring the portability. The porting process is build from various tasks:

1. rewrite and recompile codes in case of the applications and services;
2. re-structure data and applications;
3. re-configure services;
4. transfer data, applications, services or machine images between Clouds.

If these tasks are straightforward or automated, the degree of portability is high. If these tasks are requiring timely human effort, the degree of portability is low.

In particular, according to the service delivery model the portability is evaluated based on:

IaaS portability of the virtual machines (VMs) and of the configurations across different run-time environments.

PaaS openness degree of the source programming languages for application development, openness degree of the data formats, coupling degree of the coupled services, platform-dependence degree of the abstraction layers for messaging and queuing services.

SaaS openness of the source code, openness of standard data formats, integration technologies and application server or operating system.

Lenhard and Wirtz [9] have define software quality metrics in order to quantify the degree of portability of an executable service-oriented process. They have map existing metrics for program portability to service and process-oriented programs and they have combined the metrics with empirical data on language support in various runtimes. Only program elements that are supported by a majority, or all, runtimes are considered by Lenhard and Wirtz to be portable. Note that their measure of the portability of executable process definitions take only the runtimes for the analyzed process into account. If all runtimes available support all of the language elements available in the same manner with respect to semantics, then any compilable application of service is supposed be portable to any runtime. In such context, Lenhard and Wirtz stated that the first step towards measuring the portability of a process is to calculate the degree of portability for each language element and its configuration: this degree can be identified by the number of runtimes that support a language element. This measurement procedure can be currently applied only partially to portability in PaaS and IaaS environments.

3.4. Cloud portability requirements. Figure 3.2 enumerates several requirements in order to ensure the portability between Clouds. The technical requirements are split into three categories according the stage of the application (functionality), data or service in their life-cycle.

3.5. Cloud portability relationship with Cloud interoperability. The interoperability and portability are often used in conjunction when Cloud adoption barriers are mentioned. They are often seen as one and the same concept due to the fact that many of their technical solutions are the same. However the two concepts are different.

Cloud interoperability is the ability of diverse Cloud services, data and applications to work together. It is mainly a concern for Cloud service providers to ensure the proper functionality of their services with the external world (even with services of other Cloud providers). Usually it assumes a previous knowledge or agreement of the systems with which the current one will inter-operate. Therefore the issue appears mainly in the case of Federations of Clouds, that are based on agreements between Cloud providers. The customer of the Cloud service is usually not aware of the Cloud interoperability facilities of the Cloud provider which whom is in agreement. Moreover, the interoperability degree between Clouds is usually measured at runtime. Data synchronization is for example a concern that is encountered in Cloud interoperability and is not a concern for Cloud portability. A recent comprehensive study on Cloud interoperability was provided by Toosi et al [14].

On opposite site, Cloud portability is a concern that appears mainly at design and deployment stages of an application, data or service. The aim is to minimize the human efforts in re-design and re-deployment of application, data and services when moving from one Cloud to another.

4. Current solutions for Cloud portability. Cloud portability is currently achieved through open standards, protocols, widely used APIs or through abstraction layers which decouple application development from specific target Clouds.

Fig. 3.2: Portability Requirements

4.1. Types of solutions. Gonidis et al [5] and Silva et al [15] suggested that there are three types of solutions for the Cloud portability problem:

1. adoption of existing or emerging standards and protocols (like TOSCA, CDMI, OCCI, OVF);
2. usage of intermediary layers (like jClouds or mOSAIC);
3. adoption of semantics and model-based solutions.

These solutions are complementary rather than contradictory. Moreover, regardless of the solution type, a common approach is creating an abstraction layer that seeks to hide the differences between various Cloud services. The abstraction layer is agnostic to the Cloud services by hiding or wrapping the proprietary technologies of particular Cloud providers. Moreover the abstraction layer prevents developers from being bound to specific programming languages or data stores. Adapters are used to support the interaction between the abstraction layer and the target service.

4.2. A time-line perspective. The technical solutions for the portability have evolve in time, and three main stages are visible:

Stage 1: The virtual machine portability: the export and import of virtual machines, with the introduction of Open Virtualization Format (OVF) standard.

Stage 2: The networked virtual machine portability: virtual machines together with their network settings are moved between Clouds.

Stage 3: The data, application, and services portability: the portability issues are focusing on the storage and as well as APIs and grouping on-demand.

4.3. Open standards. The classical way to bust the portability is the adoption of standards and open source. Surely they are important mostly for the Cloud providers and service developers, and not in a similar degree, for the consumers.

Tables 4.1 and 4.2 are pointing towards the most important standards for Cloud portability, and, respectively, Cloud standard initiatives. Loutas et al [16] have done a complex survey of the efforts by the industry, standardization groups, and research community, at the level of 2011. More recently Lewis [17] provided a non-exhaustive list of standards-related efforts for Cloud computing, discussing also the role of standards in Cloud computing. He is also arguing that the standards do not make sense beyond IaaS, taking into account the fact the service model often involves value-added features (taking the form of application software, in the case of SaaS, or the form of platforms or libraries, in the case of PaaS).

The most prominent standard in Table 4.1, initially intended for IaaS layer, is Open Cloud Computing Interface (OCCI). OCCI is a specification for remote management of Cloud service infrastructure. It allows the development of tools for common tasks including deployment, monitoring, and autonomic scaling. Its API supports three concepts: compute, network and storage. The current OCCI standard claims to be applicable for all layers, but it is still missing a concrete proposal besides the IaaS management API (according Kolb and Wirtz [11]).

Other open standards are OVF and CDMI. OVF describes virtual appliances to be deployed across various virtualization platforms. CDMI specifies the functional manner on how applications operate with data residing on Cloud environments.

The Topology and Orchestration Specification for Cloud Applications (TOSCA), the result of standardization effort from OASIS, is the most recent entry in Table 4.1. Binz et al [3] presented how TOSCA enables the portable and standardized management of Cloud services. The composite applications and their management in a modular way is done based on service templates that contain service's topology and its operational aspects. TOSCA specifies the meta-model for types of services as well as the templates (more specific, the language for defining them). TOSCA enables application developers to model re-occurring tasks explicitly into plans using a workflow language supported by a management environment. The operations are described in WSDL, REST, or scripts. The last ones are implementing particular management operations on a specific node. These operations can be external services or their implementation can be included in the service template (as an implementation artifact). The portability is dependent on the workflow language and engines that are used.

An on-going initiative is the Guide for Cloud Portability and Interoperability Profiles (CPIP): the IEEE project P2301 (from Table 4.2), is developing a portability standard. It proposes to group different interfaces, operation conventions or tile formats of the Cloud elements into logical profiles.

4.4. Open-source libraries and deployable services. The use of abstraction layers and adapters has been widely adopted by libraries, tools and platforms aiming to enable Cloud portability. An abstraction layer hides the differences between providers and exposes a uniform semantics and syntax.

According Hill and Humphrey [18], the APIs that are allowing Cloud portability are classified in three categories:

1. multiple independent implementations, like Eucalyptus versus EC2, AppEngine versus AppScale;
2. runnable on multiple Clouds, but not through independent implementations, like the several implementations of MapReduce;
3. separators of the application into application-logic layer and Cloud layer (Table 4.3).

Representative for the first category, AppScale is an open source software system implementing a PaaS. It is API-compatible with Google App Engine (GAE): it allows the execution of GAE applications on-premise or over public Cloud infrastructures, without any code modification [19].

The latest category (and most general) requires a large time investment by a developer to create the layers, and, later on, to maintain them as the APIs change. Several open APIs offer already a thin abstraction level

Table 4.1: Standards, reference architectures, open group proposals

Type	Acronym	Link	Usefulness for portability
Standard	OCCI	occi-wg.org	It is designed to be a management API for IaaSs and includes tools for deployment, monitoring and autonomic scaling.
	OVF	www.dmtf.org/standards/ovf	Describes virtual appliances for the deployment across heterogeneous platforms, like the ones using different hypervisors, allowing its users to deploy their virtual appliances in various Clouds
	CDMI	cdmi.sniacloud.com/	Used for data management, it specifies how applications create, retrieve, update, delete data in the Cloud
	CIMI	dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.1.pdf	It addresses the management of the lifecycle of infrastructure services. Basic resources of IaaS are modeled in order to provide to the consumer a management access to an implementation of IaaS. It focus on the portability between Cloud implementations that support the specification. A REST-style protocol is used.
	TOSCA	http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html	Specification language to describe service components and their relationships. Its service template uses a combination of topology and orchestration and describes what is needed to be preserved across deployments in various environments in order to enable inter-operable deployment and management of services when the applications are ported over alternative Cloud environments.
Architecture	IETF	tools.ietf.org/html/draft-khasnabish-cloud-reference-framework-01	Proposed a draft of Intra-Cloud and Inter-Cloud reference framework, documenting basic functions or layers to support the general requirements of Cloud services. The framework can be used to standardize the interfaces between the functions/layers.
	DMTF	dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf	A reference architecture for Cloud services that describes key components, such as actors, interfaces, data artifacts, profiles and the interrelationships among these components.
Open groups	The Open group	www.opengroup.org/getinvolved/workgroups/cloudcomputing	The Cloud Work Group within Open group collaborate on standard frameworks and models for eliminating vendor lock-in.
	CSCC	www.cloud-council.org	Has released a guide for SLAs that highlights the critical elements of a SLA for Cloud and provides guidance on what to expect and what to be aware of when negotiating an SLA.
	OCC	opencloudconsortium.org	Has develop benchmarks for Cloud computing. MalStone Benchmark is targeting large data Clouds.
	GICTF	www.gictf.jp	Promotes interfaces through which Cloud services are interworking with each other. It has published: an Intercloud interface specification – the Cloud resource data model and Intercloud protocol – as well as technical requirements for the Intercloud networking, with specific use cases.

Table 4.2: Standard initiatives

Organization	Link	Usefulness for portability
ETSI	www.etsi.org/ technologies- clusters/ technologies/grid- and-cloud-computing	The technical committee for Cloud is looking to interoperable solutions involving IT and Telecom industries, with emphasis on IaaS. It focuses on applications and services which are inter-operable due to the standards as well as on the validation tools that are support such standards.
IEEE	standards.ieee.org/ develop/project/2301. html	The IEEE group is working on standardizing Cloud management (including portability), by using specific interfaces and file formats.
ITU-T	www.itu.int/en/ITU-T/ focusgroups/cloud/ Pages/	A study group which works on standards for renewed networks in conjunction with Clouds.
CSA	cloudsecurityalliance. org/wp-content/ uploads/2011/09/ Domain-6.docx	The working group 'Portability and Interoperability' has elaborated a study on the risks and benefits involved with Cloud portability
NIST	www.nist.gov/itl/ cloud/	Program to develop a set of Cloud computing standards. NIST' special publications are referring to Cloud architectures, Cloud security, Cloud deployment in the context of various strategies of USA federal government
OASIS	www.oasis-open.org/,	Three important technical committees are activating: (1) SAF, referring to capacity, quality of service, and cost; (2) CAMP, Cloud application management for platforms, defining interfaces for self-service provisioning, monitoring, and control of Cloud platforms; (3) TOSCA
OW2	www.ow2.org/view/ Cloud/	Open Source Cloudware initiative (OSCi) intended to develop an OSCi component set with state-of-the-art Cloud stacks or component set; currently it supports CompatibleOne
SNIA	www.snia.org/tech_ activities/work/twgs	Cloud Storage Technical Work Group developing an architecture related to system implementations of Cloud storage technology.
TM Forum	www.tmforum.org/ Digital Services/13907/ home.html	Cloud services initiative stimulating an open marketplace for Cloud services by promoting to large eco-system of enterprise customers, Cloud and technology providers the adoption of standards

for the unique representation of the resources (namely, they are wrappers solutions); for example, jclouds when using Java, libcloud when using Python, SimpleCloud when using PHP, and δ-Cloud when using HTTP.

The jclouds introduces the following abstractions: description of node with metadata such as CPU, RAM, security policy; an abstract representation of a node with parameters such as minCPU, OS type; group of nodes to be managed together; set of command to be executed on nodes; information about the provider.

Most of the above enumerated libraries are language-dependent in their attempt to provide a common access to multiple Clouds. Typically they provide provide a code-based model of the infrastructure and do not offer any mechanism for automatic provisioning and deployment of applications on the Clouds.

Cloud resource management tools and services that are exposed in Table 4.4 are also using adaptors, but they include deployment mechanisms.

Cloudify model for deploying applications includes: a recipe for information like required infrastructure and how it should be used, the cluster of service instances that make up an application tier, the configuration (including provisioning and scaling rules) of an application and the services it is made of, the set of services

Table 4.3: Open-source libraries that support a certain degree of portability

Acronym	Link	Short description
δ-Cloud	deltacloud.apache.org	It is REST-based API written in Ruby necessary to connect to various Cloud providers (Amazon EC2, Eucalytus, SmartCloud, GoGrid, OpenNebula, RackSpace, OpenStack and others)
Dasein Cloud API	dasein-cloud.sourceforge.net/	Java-based library for the access to compute services such as a virtual machine support, volume support, and other features associated with cloud compute as a service
fog	fog.io	Ruby-based library providing a high level interface to various Clouds 'collections' (images, servers). 'Requests' allow to dive in service particular features. 'Mocking' allows simulations by an in-memory resource representation.
jclouds	www.jclouds.org	It is an open source Java library that introduces abstractions aiming the portability of applications. It support more than thirty Cloud providers and software stacks including AWS, GoGrid, vCloud, OpenStack, Azure.
libcloud	libcloud.apache.org	Apache Libcloud is a standard Python library abstracting the differences between multiple Cloud provider APIs
Simple Cloud	www.simplecloud.org	It is a PHP library providing common interfaces for file and document storage services, queue services and infrastructure services

working together, probes used to monitor the status of the system.

OpenNebula provides interfaces to interact with the physical and virtual resources. The interfaces are allowing the use of OCCI, EC2 EBS and Query, a Sunstone GUI, and a particular portal. Low-level APIs are available for Ruby, XML-RPC and Java. A catalog of virtual appliances ready to run in OpenNebula-based environments is also available.

OpenStack offers services for compute, storage and networking. These services include identity, image management, as well as a web interface, and integrate the OpenStack components with each other and with external systems. Moreover, its DevStack enable the run of a OpenStack instance on a laptop or inside one VM.

The mOSAIC open-source PaaS is offering two level of abstractions: from the Cloud provider technologies as the above mentioned technologies, but as well as from the classical synchronous programming style. The applications can follow an event-driven architecture [20]. Programming libraries are available for Java, Python and Erlang. Opposite to OpenNebula web interface and OpenStack dashboard, its web interface focus on the life-cycle of the application components, not on the management of Cloud resources. Adaptors are build for the major Public and Private Cloud technologies. The Personal Testbed Cluster (PTC) allows the development of the applications on a desktop and test on few virtual machines (as much as the desktop allows) and later on the seamless transfer of the applications into Private or Public Clouds.

CompatibleOne is based on a platform and a model for the description, aggregation and integration of distributed resources provisioned by heterogeneous Cloud service providers. Its platform uses the OCCI. CompatibleOne is taking part to OW2 Open Source Cloudware initiative (OSCi).

Commercial solutions like RightScale or Kaavo are able to deploy applications in various Clouds, but not yet migrate the running ones.

In what concerns storage and data portability, three different approaches were proposed: common namespace, service registry and uniform APIs. The most significant initiatives are enumerated bellow in chronological order.

CSAL, introduced by Hill and Humphrey [18], is a storage abstraction layer enabling applications to use scalable storage services provided by Public Clouds and also to be portable across platforms. It provides

Table 4.4: Open-source tool or service that support a certain degree of portability

Acronym	Link	Short description
Aoleus	www.aeolusproject.org	Cloud management software sponsored written in Ruby, open source, and provided by Red Hat. It allows users to select private, public or hybrid Cloud services, using δ-Cloud library
CompatibleOne	compatibleone.org	Its is an open Cloud broker offering an interface allowing for the description in terms of resources of user needs, as well as the provisioning on the most appropriate Cloud environment. It provides also a language for the description and management of Cloud services.
Cloud Foundry	cloudfoundry.org	Open source PaaS on top of VMware-based IaaS.
Cloudify	getcloudify.org	It orchestrates the creation of the infrastructure services requested for an application, from compute resources to networks and block storage devices. It works with OpenStack, monitor the application progress and scale the resources when needed.
ConPaaS	contrail-project.eu/conpaas	Provides a federation layer support for bringing Cloud providers together. It allows Cloud bursting. The Contrail Virtual Execution Platform interfaces with the IaaS layer of Cloud providers and upgrades the supported Cloud providers by adding SLAs features
Docker	github.com/docker/docker	Is an deployment engine relying on the Linux containerization sandboxing method
mOSAIC platform	bitbucket.org/mosaic	An API and a deployable PaaS that allows deployment, configuration management, and control of the life-cycle of applications or services consuming IaaS. Supports more than ten providers
Nimbus	www.nimbusproject.org	Introduces a virtual site layer for dynamically provisioning of distributed resources lying on multiple data centers. It is based on a closed federation model in which resources are shared based on cooperation agreements
OpenNebula	github.com/OpenNebula/one	Open-source tool for data center virtualization. It offers a solution to build and manage virtualized data centers.
OpenStack	www.openstack.org	Is a open-source system to control compute, storage, and networking resources of a data-center through a dashboard.
OPTIMIS	www.optimis-project.eu/Toolkit_v2	Is a platform for Cloud service provisioning. It deals with the lifecycle of the service. Moreover, it addresses the risk and trust management

abstractions for blobs, tables, and queues. A key differentiator for CSAL is that it provides a single namespace for each storage type (by this is able to decouple the application code from explicitly making calls to specific storage services).

Thabet and Boufaida [21] proposed an agent-based architecture to publish and describe the different data services offered by providers. A service registry contains a description of services. An intermediate layer serves to enhance and help manage service invocations. The agents allow searching, collecting and filtering data in order to transfer appropriate and requested data. The adapter agents for converting exchanged data formats from one environment to another.

Rafique et al [22] proposed an abstraction layer that provides a uniform API for three particular PaaS services, including scalable data storage, blob storage, and asynchronous task execution.

Portability at PaaS layer has attract a lot of attention in the last time period. Some significant proposals in chronological order are the followings.

Cuhna et al [23] proposed a definition of a architecture allowing developers to create and expose services through a particular service delivery platform. It enables the portability of service between PaaS through a common API.

Zeginis et al [24] implemented a tool in the frame of Cloud4SOA project that allows the portability of Java applications between multiple PaaS.

Kolb and Wirtz [11] defined a model of current PaaS offerings and identify different portability perspectives. Starting from the model, they derive a standardized profile with a common set of capabilities that can be found among PaaS providers and matched with one another to check application portability based on ecosystem capabilities (project homepage: https://github.com/stefan-kolb/paas-profiles).

4.5. Semantics. The semantics technologies are applied to the interface, data or component levels using often a unified Cloud resource model. Moreover, the semantics are used to annotate applications and services (more precisely, their properties), in order to deploy and execute them efficiently on the selected infrastructure.

Several theoretical studies have been issued expressing various aspects of using semantics for Cloud portability. We mention the most significant ones, in chronological order.

The first proposal for a Cloud ontology was given by Youseff et al [29]. It has prove the benefit of using an ontology as an general method for the conceptual description of information implemented by Cloud resources.

Meximilien et al [30] envisaged a Cloud-agnostic broker for any Cloud client, to not being tied to any Cloud provider. For this purpose a meta-model was created and claimed to be applicable to any Cloud platform. The meta-model serves is an abstract representation of the APIs and Cloud functionalities.

Bernstein and Vij [31] proposed a mediation mechanism capturing the features of a Cloud provider infrastructure, and logically groups and exposes them as standardized units. The mediation mechanism uses a resource catalog approach, based on an ontology of Cloud computing resources and defined using RDF. A 'root' provider act as broker and host this resource catalog.

Ranahabu and Sheth [10] presented a semantics-centric programming paradigm for the development of portable Cloud applications (named MobiCloud). The four types of semantics for a Cloud application are inspired by the four types of semantics for a general service: logic/processes, data, system and non-functional semantics. Annotations are linking different descriptors and models addressing various concerns. To port data, a transformation is activated using the generic data format (this is a example of using the lifting lowering mechanism: the data is first transformed to a common format, i.e. lifted, and then transformed back to the target format, i.e. lowered).

Loutas et al [32] introduced a PaaS semantic interoperability framework that resolve semantic conflicts raised during the migration or deployment of an application. It defines the following dimensions: types of semantics, fundamental PaaS entities, and levels of semantic conflicts.

Cretella and Di Martino [33] proposed an approach to perform automatically the enrichment of API with semantic content and the alignment of software. The approach aimed at analyzing the functionalities and features offered by Cloud service provider, through an automatic analysis of their APIs, and building a semantic representation that is linked to the application and functional domain to which the API refers.

Dukaric and Juric [34] proposed a unified taxonomy of Cloud frameworks. The taxonomy of IaaS consists of several layers: resource abstraction, core service, support (communication layer between core service and resource abstraction), security, management, control, and value-added services. Ten IaaS were mapped to the proposed taxonomy and then compared using the taxonomy.

Di Martino [35] is underlining that metadata added to specific service APIs through annotations pointing to generic operational models could enable the portability and interoperability among the heterogeneous Cloud environments.

Another portability solution that is discussed in [35] is that of the Cloud patterns usage. The patterns are describing common behaviour in Cloud environments and they support the idea of an application redesign, reengineering or redeployment. Multiple pattern catalogs are currently available, like the ones from `cloudpatterns.org`.

Table 4.5 enumerates several concept implementations.

Table 4.5: Implementations of semantic concept for ensuring portability

Acronym	Reference	Short description
Cloud4SOA	[25]	Multi-cloud PaaS management that uses semantic-based adapters to bridge multiple PaaS. It is based on an approach to semantically interconnect heterogeneous PaaS offerings of different Cloud providers sharing the same technology. Cloud4SOA solution comprises of a set of interlinked collaborating software components and models to provide developers and platform providers with a number of core capabilities: matchmaking, management, monitoring and migration of applications.
CPIM	[26]	A Java library exposing a vendor independent API that provides an abstract intermediation layer for the most important middleware services typically offered by PaaS (e.g. NoSQL services, message queues and memcache). The library supports the portability of applications across Java platforms for AppEngine and Azure, reconciles the difference in the semantics of the queue services offered the two PaaS, and implements two different services called task queue, adopting the AppEngine semantics and message queue adopting the Azure semantics.
mOSAIC ontology& semantic engine	[27]; [28]	A Cloud ontology was developed in OWL. Its main part contains the definition of general concepts as well as the relationships with the Cloud services (like resource, VM). The APIs and Services sub-ontologies describe the model, the profile and the grounding for the APIs and services. The Requirements sub-ontology is modelling service level agreements. The Semantic Engine is used for Cloud applications development: it offers a semantic discovery of API components and functionalities, at the design time, as well as of resources, at deployment time. It supports the user in selecting the list of needed resources to be acquired from the Cloud providers.
UCI	`code.google.com/p/unifiedcloud/`	Unified Cloud Interface is a programmatic point of contact referring to the infrastructure stack through a unified interface. It uses the resource description framework (RDF) to describe a Cloud data model.

4.6. Model-driven engineering. Model driven engineering (MDE) is based on two core ideas [5]: abstraction and automation. Abstraction enables decoupling application development from targeting specific platforms. Automation refers, among others, to the ability to change the level of abstraction automatically, using model transformations (e.g. domain specific languages, shortly DSLs). The DSLs allows to the domain experts to create programs having little knowledge in programming.

The model-driven approach, often summarized as model once, generate anywhere, is particularly relevant when it comes to provisioning and deployment of applications and services across multiple Clouds, as well as migrating the source code and data from one service provider to another. Through high level models and automatic transformations, the application developers can focus to a conceptual and platform-independent level.

Unfortunately, there is no consensus about the set of languages, models, transformations or software processes to be used for a successful development of Cloud applications using MDE (according Da Silva and Lucredio [36]). Current efforts are focused mostly on the identification of abstractions allowing the specification and creation of systems independently of the underlying platform (as reflected also in the previous section), and the automatic code generation for specific Cloud services.

One main concern for Cloud portability is the variety of data representations in Clouds. MDE is trying to respond to this challenges using abstract data models. Logical data models are presented for example by Samba [37], while Shirazi et al [38] provided a solution through design patterns to enable the portability between

graph databases and column family databases.

Di Martino et al [39] investigated a methodology to automatize the recognition of similar elements in Cloud related design patterns, leading to an automatic mapping among representations of application that follow these patterns.

Ranabahu et al [40] presented an abstraction-driven approach to address the application portability issues and focus on the application development process. The approach is using a specification in form DSL script, to generate platform specific but functionally equivalent executable applications. The approach is based on a formal definition of a Cloud application, based on the assumptions that each semantic aspect can be expressed using a DSL, and the abstract syntax model of each of these DSLs can be represented using a graph. The transformation from the domain model to a Cloud-supported implementation model depends on the details of the domain meta-model. Proof-of-concept tests were done with the MobiCloud platform proposed by the same authors.

A proof-of-concept platform, named CodeCloud, introduced by Caballer et al [41], supports the execution of scientific applications covering different programming models on Cloud infrastructures. The platform automatically deploy the applications on required virtual infrastructure and perform the execution of the jobs according to the specified programming model.

Model-driven development combined with model-driven risk analysis for Cloud computing application cycle is currently studied in MODAClouds project (overview papers: [42], [43] and [44]). Its aim is to enable application developers to specify the models of Cloud services in which they are interested and to enrich these models with quality parameters. Moreover, to close the current gap between the design and run-time stages of an application, performing quality predictions, as well as run-time monitoring and optimizations, provide valuable information to the design time environment.

Three levels of abstraction are considered by MODAClouds: CCIM, the Cloud enabled Computation Independent Model to describe an application and its data; CPIM, the Cloud-Provider Independent Model to describe Cloud concerns related to the application in a Cloud agnostic way; and CSPM, the Provider Specific Model to describe the Cloud concerns needed to deploy and provision the application on a specific Cloud. CCIM describes Cloud applications at the service level. Hence, for a given application it contains the description of the services that compose it, the public interfaces of each service, the business processes that describe their orchestration and the domain model of the data exchanged by them through their public interfaces. Modeling concepts and technologies for the provisioning, deployment and adaptation of applications in various Clouds (CPIM/CPSM) can be done using the uniform interfaces (mentioned in the previous section) provided by various libraries for application development, deployment and control at run-time.

To illustrate the differences between the three abstraction layers, the data models are considered here. The CCIM Data Model describes the main data structures associated to the software to be. It is expressed in terms of a typical Entity Relationship diagram and enriched by a meta-model that specify functional and non-functional data properties. The CPIM Data Model describes the data model in terms of models being typically offered by Cloud providers, without referring to a particular one. Since the majority of available data storage software provides services that include distributed file system, NoSQL solutions, and blobs, these three models are considered at the CPIM level that are suitable to represent these different cases. The models are conceived for graph data, hierarchical data and flat data. The CSPM Data Model describes data structures implemented in the Cloud providers. The following families are considered: relational data, hierarchical data, flat data, key-value data, column-based data.

The models@runtime paradigm proposes to leverage models during the execution of adaptive software systems to monitor and control the way they adapt [45] . This approach enables the continuous evolution of the system with no strict boundaries between design-time and runtime activities (supporting therefore the DevOps concept). The approach is adopted for Clouds by CloudML, promoted by MODAClouds, ARTIST [46] and PaaSage [47], a follow up of PIM4Cloud from REMICS [48]. CloudML is domain-specific modeling language enhanced with a run-time environment. It facilitates the specification at design-time of provisioning, deployment, and adaptation concerns, as well as their enactment at run-time [49].

5. Barriers, Challenges and Open Issues. Reflecting on the above described solutions, we can discover several advantages on using one or another solution, as well as their limitations that are generating a need for

Table 5.1: Advantages and limitations of various approaches for Cloud portability

Approach	Advantages	Limitations
Standard	– Result of a collective agreement – Extract the key actions and characteristics – Should be implementable	– Not widely adopted – From the provider point of view, hinders diversity – The number of emerging and overlapping standard makes the problem to grow
Open library	– Offers an abstraction layer that is simple to use – Available for major languages – Similarity with major Cloud provider APIs – Decouple the application code from the underlying Cloud service – Adaptors available for major Cloud services	– Usually refers to the common denominator of the Cloud services – Language dependent – Adaptors needs to be build for emerging new services – The connected service programming style usually maintained – Require Cloud computing knowledge as deployment is usually not supported – Introduces an overhead compared with the direct connection to the Cloud service
Open service	– Offer unique entry point for application deployment and Cloud resource management – Application deployment can be done by non-Cloud specialists – Part of them are offering also APIs for programming applications – Usually it offers support for multiple programming languages – Monitoring tools are generate alarms needed to trigger a re-deployment	– The diversity of deployment services raise also another dimension for the portability – Manual intervention at deployment phase is usually still needed – Life migration is still not possible – Re-deployments are not automated – Rely upon adaptors that need to be build for new services or updated when a service version appears
Semantics	– Offers an abstraction layer that can support various customers – Offers viable mechanisms for common understanding of service terminology and actions – Allow the annotation of services with quality marks by externals from the provider team	– Not widely adopted – The variety of taxonomies and ontologies makes the portability problem to grow – The overhead of semantic processing is not negligible
Model-driven	– Enhance the abstraction layers with an automation process – Allow a feedback from operational modules to the design modules	– Available tools are not yet generating code for various Clouds – The models that are used potentially omit special features of the services

improvements in the near future.

Table 5.1 is pointing therefore to some advantages and limitations of the main existing solutions for Cloud portability that were presented in the previous section.

In what follows we discuss in more details what is missing and what is possible to be done in a near future to achieve Cloud portability.

5.1. Standards. Standardization is not an appealing solution for some Cloud service providers. The providers are interested to differentiate themselves by unique offers.

OVF, OCCI and CDMI have partially failed to be adopted on large scale by the providers as are currently implemented only in few Cloud management tools. Unless there is a well-accepted and widely used standard, it is a questionable solution.

As the emerging standards are not widely adopted, a common understanding on a certain action or feature of a Cloud service has not yet been reached. Consequently, the third party supporting for example an uniform interface needs to understand each particular service interface in order to plug it in their service or software. This fact is not compliant with the requirement of a seamless joining procedure for a new service. If the conditions that the interface requires to comply with is constantly increasing market, the compliance with them is considered a moving target, that is difficult to reach.

New standards are emerging nowadays, like TOSCA or CAMP, and several collaborative groups are working to elaborate other proposals. However there are several gaps in the collections of available standards, like proposals for Cloud metrics and real-time monitoring, interfaces for security(-as-a-)services, accountability associated with transparency and responsibility.

A unified policy of the contractual terms was not yet established at national or international levels, while several proposals are on the way. One of the most disputed topic is the privacy and data protection compliance, for which no general accepted proposal is currently available.

There is a clear need to define standardized, machine-readable and self-descriptive representations of the basic characteristics of the resources (like API, size, or platform), the advanced characteristics of the resources (like quality attribute values or pricing), the negotiation processes and protocols, as well as billing processes and protocols [17].

System developers need to leverage standards to support the architecture of a system. However, such standards should not drive such an architecture [17]. The software architecture for Cloud-based systems in which standards-reliant components should be implemented as components that are separated from the rest of the system (in order to minimize the impact on the evolution of the standards).

Moreover, the standardization should focus now on the basic use cases of user authentication and data migration. Workload migration and workload management can be the subject of dynamic use cases in which location, negotiation, and provisioning of Cloud resources occur at run-time.

Cloud service portability has not been examined until now with regard to management and operational tasks. These tasks are an increasing and significant cost factor. Moreover, standard for defining composite applications and their management are now emerging.

The review made by Jamshidi et al [50] identified the need for a standardized migration framework. The lack of attention to crosscutting concerns and migration execution is also observed. The study also showed a lack of tool support to automate and facilitate Cloud migration tasks.

5.2. Design. The diversity of the APIs is natural, as each providers intends to offer something new or unique compared with other offers, in order to attract customers. The portability issue is therefore an issue for the management and governance levels where automation should be achieved as much as possible.

Limitations of abstraction layers include maintenance in response to changes made by a Cloud provider, and the limited coverage of provider functionalities.

The mix-in of services from different providers can be a strong argument in using an unique entry level instead a direct connection to only one provider, despite the overhead that is associated with the new layer. Therefore the management and service automation levels are the hot-spots of the development activities in the last three years.

An experimented developer can see a lack of flexibility in using a uniform interface or a model driven approach. The primary concern is the limitation of the abstractions to the set of smallest common subset of features. Following such approaches, a fully portable application can only be made at the level of the least capable service. This fact becomes a serious limitation when the applications need to be exploit the capabilities of the targeted Cloud.

While several libraries offering are offering diverse uniform interfaces, there is no wide acceptance of one or another proposal.

All tools that are offering unique entry points do not come with a structured approach, and the provided methods are at a technical level, thus, the application developer will typically be left hacking at code level rather than engineering the application following a structured tool supported methodology.

Once the application is deployed and adapted to a certain Cloud, in order to move it in another Cloud, an inspection of the source code is needed to identify the specific API calls or to build a model or representation of the code. Tools that can do that are only in early stage of prototyping and not yet available for large scale usage.

One strategy to avoid the limitation of a uniform interface with the smallest set it feature is to use a DSL to generate the code, and then to customize the generated code [10]. The customization should be loosely attached to the code to avoid overriding by subsequent updates.

Silva et al [4] made a comprehensive survey of Cloud lock-in solutions. They claim that there is no solution dealing with the impact of changes in the business process due to the use of one specific product, or working on issues related to the cross-organizational business in a portability scenario. Furthermore they claim that, from a research point of view, a challenge is to conduct more rigorous studies (e.g. only part of the studies of studies included a form of evaluation). Another challenge is increasing the use of empirical evidence to support the studies.

A roadmap for the Cloud software engineering was proposed by Da Silva and Lucredio [36]. The key-points of their roadmap are: solutions for avoidance of data lock-in; decision making about the migration towards the Cloud; legacy software migration; a re-engineering process for the Cloud migration; mechanisms to facilitate the hybrid Cloud; implementation of Modelling as a Service; Cloud service composition; case studies; open source platforms.

5.3. Run-time. At the infrastructure level, several partial solutions exists to migrate virtual machines, virtual storage or services. Despite the presence of standard OVF format, the virtual machines and their associated data are not yet ready for real-time migration. For example, Amazon is one of the few Cloud providers who are allowing to export virtual machines; however, their related resources (e.g. network, storage) cannot be exported too. In general, virtual machine images cannot be transferred from one hypervisor to another. The virtual machine images can be converted today with tools like qemu-image, but this requires to stop the virtual machine and to apply the adaptation off-line.

Most IaaS Clouds expose limited capabilities to control how a service behaves at run-time, beyond basic low-level scalability rules for VMs (once it has been deployed). Different Cloud services rely on specific rule engines to help enforcing the rules governing the service during its whole life-cycle. This fact reduces the portability chances of any given set of rules. The usage of a standard procedure for specifying the governance rules is needed to ensure the rule portability. According Moran et al [51], the Rule Interchange Format (RIF) is likely a candidate as specification language.

Portability degree needs to be detected by experiments. There are only few reports that are exposing such experiments. For example, Gonidis et al [5]: reported the results of an experiment carried out on cross-platform application development and deployment with four PaaS (OpenShift, App Engine, Heroku and Elastic Beanstalk).

5.4. Research agenda for Cloud portability. Following the comments that were exposed in the previous section, Table 5.2 proposes a research agenda for Cloud portability, with topics split on short and long term (next two years, respective a half decade).

6. Conclusions. In this paper, we conducted a survey on portability of the applications that are consuming Cloud services. We provided a taxonomy of Cloud portability, the latest solutions, reviewed existing works and identified the missing pieces needed to solve the problem. In the end, we also pointed out potentially future research directions and useful design guidelines for Cloud portability.

The focus of the paper was on the methodological and technical tools supporting the portability. However the porting a particular application requires also a decision methodology and an approach for the analysis of the destination Cloud environment, that were not discussed in this paper.

We hope this work provides a well-established framework that helps researchers to find existing proposals easily and to develop well founded future works.

Table 5.2: Research agenda for achieving Cloud portability

Approach	Short term	Long term
Standards	– Enhance the number of standard implementations	– Unified policy of the service level agreements
	– Establish standards for metrics, monitoring, accounting, security	– Establish standards for workload and data migration
	– Establish standards for machine-readable representations of services, quality, negotiations, processes	– Reference architecture for basic components of software consuming Cloud services
Design	– Support for decision making for Cloud migration	– Define re-engineering process for Cloud
	– Introduce Modelling-as-a-Service	– Mechanisms for code inspections and rewriting
	– Mechanism for service compositions	– Follow a structural approach in the design of the supporting tools
	– Build use cases and benchmarks for Cloud portability	– Ensure the portability of elasticity rule engines
	– Define the portability degree	– Combine automation with customization
Run-time	– Adopt open-source platforms	– Mechanisms for real-time migration
	– Increase the use of empirical evidence of portability	– Tools for the full service cycle, including Cloud governance
	– Automate re-deployments	– Open-source platforms ensuring automated portability or encompassing various approaches

Acknowledgment. The work of the first author is supported partially by AMICAS –Romanian grant PN-II-ID-PCE-2011-3-0260). The model-driven engineering part of the paper is a result of MODAClouds – European Commission grant FP7-ICT-2011-8-318484.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.future.2008.12.001

[2] D. Bradshaw, G. Folco, G. Cattaneo, and M. Kolding, "Quantitative estimates of the demand for cloud computing in Europe and the likely barriers to up-take," EDC, Tech. Rep. SMART 2011/0045, February 2012. [Online]. Available: http://cordis.europa.eu/fp7/ict/ssai/docs/study45-d2-interim-report.pdf

[3] T. Binz, G. Breiter, F. Leyman, and T. Spatzier, "Portable cloud services using TOSCA," *Internet Computing, IEEE*, vol. 16, no. 3, pp. 80–85, May 2012. [Online]. Available: http://dx.doi.org/10.1109/MIC.2012.43

[4] G. Silva, L. Rose, and R. Calinescu, "A systematic review of cloud lock-in solutions," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 2, Dec 2013, pp. 363–368. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.130

[5] F. Gonidis, A. J. H. Simons, I. Paraskakis, and D. Kourtesis, "Cloud application portability: An initial view," in *Proceedings of the 6th Balkan Conference in Informatics*, ser. BCI '13. New York, NY, USA: ACM, 2013, pp. 275–282. [Online]. Available: http://doi.acm.org/10.1145/2490257.2490290

[6] D. Petcu, G. Macariu, S. Panica, and C. Crăciun, "Portable cloud applications-from theory to practice," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1417–1430, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2012.01.009

[7] D. Petcu, "Portability and interoperability between clouds: Challenges and case study," in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, W. Abramowicz, I. Llorente, M. Surridge, A. Zisman, and J. Vayssire, Eds. Springer Berlin Heidelberg, 2011, vol. 6994, pp. 62–74. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24755-2_6

[8] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST Special Publication 800-145, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[9] J. Lenhard and G. Wirtz, "Measuring the portability of executable service-oriented processes," in *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International*, Sept 2013, pp. 117–126. [Online]. Available: http://dx.doi.org/10.1109/EDOC.2013.21

[10] A. Ranabahu and A. Sheth, "Semantics centric solutions for application and data portability in cloud computing," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2010, pp. 234–241. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2010.48

[11] S. Kolb and G. Wirtz, "Towards application portability in platform as a service," in *Proceedings of the 8th IEEE International Symposium on Service-Oriented System Engineering (SOSE), Oxford, United Kingdom*. IEEE Computer Society, 2014, pp. 218–229.

[12] K. Oberle and M. Fisher, "ETSI cloud – initial standardization requirements for cloud services," in *Economics of Grids, Clouds, Systems, and Services*, ser. Lecture Notes in Computer Science, J. Altmann and O. Rana, Eds. Springer Berlin Heidelberg, 2010, vol. 6296, pp. 105–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15681-6_8

[13] M. Jonnalagedda, M. C. Jaeger, U. Hohenstein, and G. Kaefer, "Application portability for public and private clouds," in *Proceedings of the 1st International Conference on Cloud Computing and Services Science*. SciTePress, 2011, pp. 484–493. [Online]. Available: http://dx.doi.org/10.5220/0003394104840493

[14] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 7:1–7:47, May 2014. [Online]. Available: http://doi.acm.org/10.1145/2593512

[15] G. Silva, L. Rose, and R. Calinescu, "Towards a model-driven solution to the vendor lock-in problem in cloud computing," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, Dec 2013, pp. 711–716. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.131

[16] N. Loutas, E. Kamateri, F. Bosi, and K. Tarabanis, "Cloud computing interoperability: The state of play," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2011, pp. 752–757. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2011.116

[17] G. Lewis, "Role of standards in cloud-computing interoperability," in *2013 46th Hawaii International Conference on System Sciences (HICSS)*, Jan 2013, pp. 1652–1661. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2013.470

[18] Z. Hill and M. Humphrey, "CSAL: A cloud storage abstraction layer to enable portable cloud applications," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2010, pp. 504–511. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2010.88

[19] C. Krintz, "The AppScale cloud platform: Enabling portable, scalable web application deployment," *IEEE Internet Computing*, vol. 17, no. 2, pp. 72–75, March 2013. [Online]. Available: http://dx.doi.org/10.1109/MIC.2013.38

[20] D. Petcu, B. Martino, S. Venticinque, M. Rak, T. Mahr, G. Lopez4, F. Brito, R. Cossu, M. Stopar, S. Sperka, and V. Stankovski, "Experiences in building a mOSAIC of clouds," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 12, 2013. [Online]. Available: http://www.journalofcloudcomputing.com/content/2/1/12

[21] M. Thabet and M. Boufaida, "An agent-based architecture and a two-phase protocol for the data portability in clouds," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2013, pp. 785–790. [Online]. Available: http://dx.doi.org/10.1109/WAINA.2013.129

[22] A. Rafique, S. Walraven, B. Lagaisse, T. Desair, and W. Joosen, "Towards portability and interoperability support in middleware for hybrid clouds," in *Proceedings of the 2014 IEEE INFOCOM Workshop on Cross-Cloud Systems*. IEEE Computer Society, 2014, pp. 7–12.

[23] D. Cunha, P. Neves, and P. Sousa, "Interoperability and portability of cloud service enablers in a PaaS environment," in *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*. SciTePress, 2012, pp. 432–437. [Online]. Available: http://dx.doi.org/10.5220/0003959204320437

[24] D. Zeginis, F. D'Andria, S. Bocconi, J. G. Cruz, O. C. Martin, P. Gouvas, G. Ledakis, and K. A. Tarabanis, "A user-centric multi-PaaS application management solution for hybrid multi-cloud scenarios," *Scalable Computing: Practice and Experience*, vol. 14, no. 1, 2013. [Online]. Available: http://www.scpe.org/index.php/scpe/article/view/824

[25] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. DAndria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, and K. Tarabanis, "Cloud4soa: A semantic-interoperability paas solution for multi-cloud platform management and portability," in *Service-Oriented and Cloud Computing*, ser. Lecture Notes in Computer Science, K.-K. Lau, W. Lamersdorf, and E. Pimentel, Eds. Springer Berlin Heidelberg, 2013, vol. 8135, pp. 64–78. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40651-5_6

[26] F. Giove, D. Longoni, M. S. Yancheshmeh, D. Ardagna, and E. Di Nitto, "An approach for the development of portable applications on PaaS clouds," in *Proceedings of the 3rd International Conference on Cloud Computing and Services Science*. SciTePress, 2013, pp. 591–601. [Online]. Available: http://dx.doi.org/10.5220/0004511605910601

[27] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mOSAIC ontology for cloud resources annotation," in *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2011, pp. 973–980.

[28] G. Cretella and B. Di Martino, "Towards a semantic engine for cloud applications development," in *2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, July 2012, pp. 198–203. [Online]. Available: http://dx.doi.org/10.1109/CISIS.2012.159

[29] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov 2008, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/GCE.2008.4738443

[30] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson, "Toward cloud-agnostic middlewares," in

Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, ser. OOPSLA '09. New York, NY, USA: ACM, 2009, pp. 619–626. [Online]. Available: http://dx.doi.org/10.1145/1639950.1639957

[31] D. Bernstein and D. Vij, "Using semantic web ontology for Intercloud directories and exchanges," in *2010 International Conference on Internet Computing*, H. R. Arabnia, V. A. Clincy, J. Lu, A. Marsh, and A. M. G. Solo, Eds. CSREA Press, July 2010, pp. 18–24.

[32] N. Loutas, E. Kamateri, and K. Tarabanis, "A semantic interoperability framework for cloud platform as a service," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2011, pp. 280–287. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2011.45

[33] G. Cretella and B. Di Martino, "Towards automatic analysis of cloud vendors APIs for supporting cloud application portability," in *2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, July 2012, pp. 61–67. [Online]. Available: http://dx.doi.org/10.1109/CISIS.2012.162

[34] R. Dukaric and M. B. Juric, "Towards a unified taxonomy and architecture of cloud frameworks," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1196 – 1210, 2013, special section: Hybrid Cloud Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001793

[35] B. Di Martino, "Applications portability and services interoperability among multiple clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 74 – 77, 2014. [Online]. Available: http://dx.doi.org/10.1109/MCC.2014.1

[36] E. Da Silva and D. Lucredio, "Software engineering for the cloud: A research roadmap," in *2012 26th Brazilian Symposium on Software Engineering (SBES)*, Sept 2012, pp. 71–80. [Online]. Available: http://dx.doi.org/10.1109/SBES.2012.12

[37] A. Samba, "Logical data models for cloud computing architectures," *IT Professional*, vol. 14, no. 1, pp. 19–26, Jan 2012. [Online]. Available: http://dx.doi.org/10.1109/MITP.2011.113

[38] M. Shirazi, H. C. Kuan, and H. Dolatabadi, "Design patterns to enable data portability between clouds' databases," in *2012 12th International Conference on Computational Science and Its Applications (ICCSA)*, June 2012, pp. 117–120. [Online]. Available: http://dx.doi.org/10.1109/ICCSA.2012.29

[39] B. Di Martino, G. Cretella, and A. Esposito, "Semantic and agnostic representation of cloud patterns for cloud interoperability and portability," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 2, Dec 2013, pp. 182–187. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.123

[40] A. Ranabahu, E. Maximilien, A. Sheth, and K. Thirunarayan, "Application portability in cloud computing: An abstraction driven perspective," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2013. [Online]. Available: http://dx.doi.org/10.1109/TSC.2013.25

[41] M. Caballer, C. de Alfonso, G. Molta, E. Romero, I. Blanquer, and A. Garcia, "CodeCloud: A platform to enable execution of programming models on the clouds," *Journal of Systems and Software*, no. 0, pp. –, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2014.02.005

[42] D. Ardagna, E. Di Nitto, G. Casale, D. Petcu, P. Mohagheghi, S. Mosser, P. Matthews, A. Gericke, C. Ballagny, F. D'Andria, C. Nechifor, and C. Sheridan, "ModaClouds: A model-driven approach for the design and execution of applications on multiple clouds," in *2012 ICSE Workshop on Modeling in Software Engineering (MISE)*, June 2012, pp. 50–56. [Online]. Available: http://dx.doi.org/10.1109/MISE.2012.6226014

[43] E. Di Nitto, M. A. Almeida da Silva, D. Ardagna, G. Casale, C. D. Craciun, N. Ferry, V. Muntes, and A. Solberg, "Supporting the development and operation of multi-cloud applications: The ModaClouds approach," in *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Sept 2013, pp. 417–423. [Online]. Available: http://dx.doi.org/10.1109/SYNASC.2013.61

[44] D. Petcu, E. Di Nitto, D. Ardagna, A. Solberg, and G. Casale, "Towards multi-clouds engineering," in *2014 IEEE INFOCOM, Workshop on Cross-Cloud Systems (CrossCloud)*, April 2014, p. in print.

[45] G. Blair, N. Bencomo, and R. France, "Models@ run.time," *Computer*, vol. 42, no. 10, pp. 22–27, Oct 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.326

[46] A. Menychtas, C. Santzaridou, G. Kousiouris, T. Varvarigou, L. Orue-Echevarria, J. Alonso, J. Gorronogoitia, H. Bruneliere, O. Strauss, T. Senkova, B. Pellens, and P. Stuer, "ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud," in *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Sept 2013, pp. 424–431. [Online]. Available: http://dx.doi.org/10.1109/SYNASC.2013.62

[47] N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg, "Managing multi-cloud systems with CloudMF," in *Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies (NordiCloud)*, 2013, pp. 38-45. [Online]. Available: http://dx.doi.org/10.1145/2513534.2513542

[48] A. Sadovykh, C. Hein, B. Morin, P. Mohagheghi, and A.J. Berre, "REMICS – REuse and Migration of Legacy Applications to Interoperable Cloud Services," in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, W. Abramowicz, I. Llorente, M. Surridge, A. Zisman, and J. Vayssire, Eds. Springer Berlin Heidelberg, 2011, vol. 6994, pp. 315–316. [Online]. Available: Available: http://dx.doi.org/10.1007/978-3-642-24755-2_32

[49] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems," in *2013 IEEE 6th International Conference on Cloud Computing (CLOUD)*, L. O'Conner, Ed. IEEE Computer Society, 2013, pp. 887–894. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2013.133

[50] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 142–157, July 2013. [Online]. Available: http://dx.doi.org/10.1109/TCC.2013.10

[51] D. Moran, L. Vaquero, and F. Galan, "Elastically ruling the cloud: Specifying application's behavior in federated clouds," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, July 2011, pp. 89–96. [Online]. Available:

http://dx.doi.org/10.1109/CLOUD.2011.53

Edited by: Marcin Paprzycki
Received: Aug 21, 2014
Accepted: Sept 21, 2014

DOI 10.12694/scpe.v15i3.1020
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 3, pp. 273–285. http://www.scpe.org

FAIRNESS OF TASK SCHEDULING IN HIGH PERFORMANCE COMPUTING ENVIRONMENTS

ART SEDIGHI[†‡] YUEFAN DENG[§] AND PENG ZHANG[¶]

Abstract. We claim that the current scheduling systems for high performance computing environments are unable to fairly distribute resources among the users, and as such, are unable to maximize the overall user satisfaction. We demonstrate that a user can game the system to cause a temporal starvation to the other users of the system, even though all users will eventually finish their job in the shared-computing environment. Undesired and unfair delays in the current fair-shared schedulers impede these schedulers from wide deployments to users with diverse usage profiles.

Key words: High performance computing, task scheduling, shared infrastructure, parallel computing, grid computing.

AMS subject classifications. 68M20

1. Introduction. Shared-computing environments where more than one user requests access to the underling system have to deal with many unknown aspects, such as how to deal with selfish behaviors of participants in real-time, which may result in low utilization of resources. There has been research in this area as it pertains to network routing [1], but very little has been done to consider selfish behavior in high performance computing (HPC) environments. With network routing, selfish behavior is more likely to backfire as many congestion control mechanisms simply drop packets and cause the originator to retry [2]. In shared-computing environments where task schedulers frequently use what is known as fair-share mechanisms [3] to schedule resources, users could game the system to monopolize resources and essentially "win the game" as in Game Theory. As such, game theory semantics describes our computing model nicely: users being the players of a game; strategies map directly with how a given task is submitted for completion and is scheduled; scheduler being the game mediator; and the amount of computing resources available to a user based on the current state of the infrastructure maps to the utility perceived by that user. A win is to gain a larger share of available resources. All users finish their jobs eventually, but the delays caused by competition are undesirable system events [4].

It is conceivable to treat a scheduler as the mediator to a zero-sum game. We assume the total number of resources available to a scheduler and as well as the number of users to be constant. As such, the number of resources that one user is assigned directly reduces the available resource pool for other users. Based on a user's task submission strategy, a user realizes a utility that is derived from the number of resources it is assigned by the scheduler. To address this classic problem, we adopt the concept of Nash equilibrium in game theory. Nash equilibrium represents the socially optimum solution, where no player is inclined to unilaterally change its strategy and increase its utility [5]. Strong Nash equilibrium further states that no subsets of players have the incentive either [6]. Based on this, if a user has the willingness to change his strategy and, as a result, change his perceived utility, we can deduce that we are not at an equilibrium point. Finding the solution to Nash equilibrium is not the goal of this paper but revealing the shortcomings of the fair share scheduler in shared environment is. Finding the Nash equilibrium point or the strong equilibrium point has been shown to be at least NP [7, 8, 9].

2. Background. HPC clusters are used for interactive workloads where immediate response for a workload is desired, and such workloads are broken into smaller tasks to be executed in parallel. A task is the smallest unit of instructions (atomic unit) that needs to be executed on a node. A job is an assembly of such tasks. A job cannot be marked complete until all tasks are complete. This is different than job-based systems, where a single job represents the entire workload and its completion means the completion of that workload. In a

[†]Corresponding author
[‡]Department of Industrial Engineering, Texas Tech University, Lubbock, TX 79409 (art.sedighi@ttu.edu)
[§]Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, NY 11794-3600 and National Supercomputer Center in Jinan, Shandong 250101, China (yuefan.deng@stonybrook.edu)
[¶]Department of Biomedical Engineering, State University of New York at Stony Brook, NY 11794-8151 (peng.zhang@stonybrook.edu)

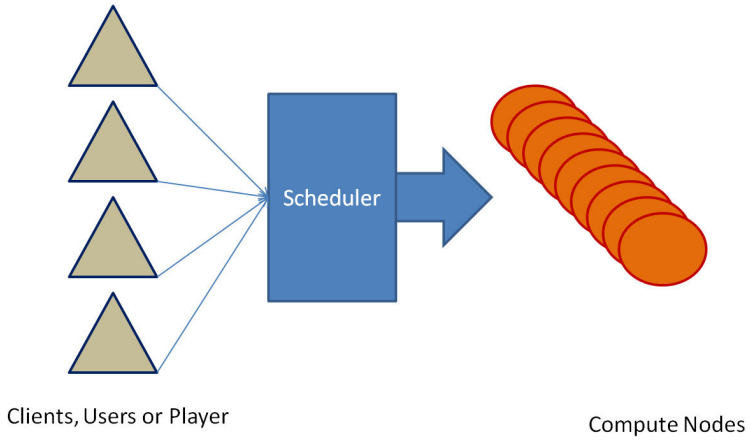

Fig. 2.1: The components of a shared-computing environment

task-based system, tasks are submitted over a period of time, and the method of submission has a significant effect on the overall completion time.

Interactive users continue to be attached to cluster until all tasks have been completed. Interactive workloads that run in the HPC environment are mostly single instruction multiple data (SIMD), where the same process (instruction) is executed on multiple data sets simultaneously. Depending on the size of the data set, and other factors [5], an imbalance could take place where not all the tasks complete at the same time. We will further show that this imbalance is further exacerbated by the scheduling mechanism.

HPC clusters achieve their desired high performance through the use of an integrated hardware, software and network [10, 11, 12, 13, 14]. A typical cluster contains 500 to 1000 physical nodes, but larger ones are possible [15, 16, 17, 18]. Considering a system with three participants in the aforementioned HPC cluster environment (Figure 2.1), we introduce the following terms:

1. Users: a client in need of computing resources such as a researcher in a biotech company, or a trader for a financial firm. The user interfaces with the HPC environment through a set of API's that facilitate communications with the environment. A rational user desires to utilize as many resources as possible without any regard to other users. Such greedy behavior may yield optimal solutions for one's own tasks, however, as many users share the same resource pool, creates temporary denial of access or starvation to other users.

2. Computing nodes (or resources): the servers that process the computing tasks. A node is the smallest processing unit and can be assigned to a single user. In game theory terminology, the computing nodes are the prize or outcome of the game. Each iteration of the game ends with the assignment of computing resources to each player, and as such the perceived utility of that user. The larger the portion of assigned computing nodes to a user, the higher the user's perceived utility. Since there are many tasks, and each task is short in duration, the utility of each user can change frequently during the game.

3. Scheduler (resource manager): the mediator that matches tasks with the available computing nodes optimal utilization of the HPC clusters.

The cardinality of a deployed infrastructure is ad-hoc in that there may be one or many of each of these components. HPC environments involve many users, most of whom are unaware of others' presence. Their decisions are based purely on selfish reasons. The goal of a given user is, and always should be, to game the scheduler in order to receive maximum resources to complete its tasks the quickest.

Scheduling systems used in HPC environments are based on queuing methodology where incoming queues are used to store tasks until they are scheduled for execution [19]. We are not introducing a new scheduling algorithm, but rather focusing on the inherent problem that is common to the queuing-based scheduling systems that use a fair share algorithm to distribute tasks.

Schedulers such as LSF [20], Sun Grid Engine (SGE) [21], PBS [22], Condor [23], Maui and Moab [24],

GridWay [25], Unicore [26] and GRMS [27] represent the de facto standard solutions that implement a First Come First Served (FCFS) policy for de-queuing tasks and scheduling for execution [19]. Even though the aforementioned systems implement the basic FCFS algorithm, these offerings also have the capability to prioritize based on the task characteristics [28, 29] such as Largest Job First, Smallest Job First (SJF), or Earliest Deadline First (EDF) [30], where a different priority is assigned to a given task according to the policy that the algorithm aims to maximize.

Most schedulers use a variation of the fair share algorithm [3] as the means of matching resource supply with user demand. For example, in a queuing system that uses the SJF policy, all the tasks of a given size (or runtime) is picked from the queue, and a fair share algorithm is applied to distribute these tasks across the available resources. In a queuing system that applies the basic FCFS queuing policy, the oldest tasks are taken off the queue and scheduled using a fair share algorithm. Fair share does so by dividing the resources through a ratio analysis of "the user who 'deserves' more, gets more". The amount that one deserves for a given time slot ($r_{c_k}(t)$) depends on the total number of tasks (T) pending to be completed.

$$r_{c_k}(t) = \frac{T_{c_k}}{\sum_{i=0}^{n} T_{c_i}} * R \quad (c_0 < c_k < c_n) \tag{2.1}$$

Subject to:

$$R = \sum_{i=0}^{n} r_{c_i}(t) \tag{2.2}$$

To simplify the ratio calculations without major drawbacks, we assumed that all tasks require the same computation, and all resources are homogeneous. Furthermore, we represent utilization in a binary format of "assigned" vs. "not unassigned". A given resource can go unassigned, i.e., idle, for a given scheduling cycle.

Let us consider the scheduler for two scheduling cycles (t_1 and t_2) where the scheduler takes a snapshot of the number of tasks of a given user waiting in a queue for a resource to be completed. Note that the resulting resource allocation depends only on the current state without knowledge of prior allocation. In order to drain the queues faster, the larger queue is assigned a larger portion of the available resources as demonstrated in Table 2.1. For example, for two users (c_1 and c_2) with c_1 having a queue size of 10 and c_2 a queue size of 90, user c_1 is assigned 10% of the resources while c_2 gets 90%. If $R = 6$, then we can demonstrate that c_1 is temporary starved.

Table 2.1: Fair-shared scheduling with temporal starvation of c_1

	t_1	t_2
c_1	$r_{c_1}(t_1) = \frac{10}{100} * 6 = 0.60 \sim 0$	$r_{c_1}(t_2) = \frac{10}{95} * 6 = 0.63 \sim 0$
c_2	$r_{c_2}(t_1) = \frac{90}{100} * 6 = 5.40 \sim 5$	$r_{c_2}(t_2) = \frac{85}{95} * 6 = 5.36 \sim 5$

As presented, the decision is not made based on any historical data. A snapshot of the current status is used to decide the next steps with the following assumptions:

- Running queue of pending tasks is an undesirable system event;
- The heavy users get more resources as they desire more;
- Temporal scheduling decisions are a good indication of the overall optimization problem.

The assumptions help drain the queue as fast as possible and, to achieve this, the heavy user is given an implicit higher priority. This static scheduling is considered fair-share in that each user will get its "fair share" of the system resources based on demand. Some scheduling policies use a policy-based mechanism, and solve the problem by introducing constraints in order to further aid the decision-making process.

We further assume:

1. There are many users, each with one or more jobs:

A typical environment has many users, each of which having its own collection of jobs that needs to be completed.

2. Each job is composed of many tasks:
In order to take advantage of the parallelism offered by the infrastructure, tasks must run in parallel, and when all tasks are completed, the overall job is completed. Essentially, tasks are atomic units to be scheduled by the scheduler. The rate of incoming tasks (e.g., 4 incoming tasks/sec) directly affects the pending tasks in the scheduling queue. A user has the ability to control the rate it submits tasks into the queue.

3. Users are unaware of others' behavior and strategies:
Although the scheduler decision system is static and fair, users are unaware of other users' behaviors. As such, each user will strive to get the most of what is available, and will act rationally and thus selfishly. We will further assume that tasks belonging to different jobs can compete with each other as well.

4. Limited resources where the total resources available is finite:
Usually, it takes much longer to acquire new hardware than the runtime of a given task. An argument is made about the ability to "spin-up" virtual environment, but virtual environments are still bound by the underlying constraints and availability of the hardware platform. Furthermore, the total number of resources is only a fraction of the total tasks pending to be executed. In our case, we assume we only have 500 resources, i.e., up to 500 tasks can run in parallel at any given time.

5. Selfishness and satisfaction:
Each user acts rationally in its desired completion of its job before others'. A user is acting rationally, if it is acting selfishly.

3. Rationale. An HPC cluster acts as a repeated zero-sum game with multiple users. The strategy that each user chooses in order to submit tasks directly affects the allocation of resources. In fair-share, the strategy chosen can starve some users and reduce the overall quality of the scheduler. In this type of environment one user's change in strategy directly affects the outcome of resource assignment. We show that shared environments with fair-shared scheduling algorithms never reach a Nash's equilibrium point [31] and that while some users are satisfied, maximum satisfaction is not perceived by all.

Our discussion revolves around a repeated game as with each new task, a new iteration of a game is played. We consider one task T_i to be a single iteration of a game denoted by J. The total time for a job t_J is then the total of the runtimes of all the tasks t_{T_i}.

$$J = \{T_1, T_2, T_3, \ldots, T_n\} \tag{3.1}$$

$$t_J = \sum_{i=0}^{n} t_{T_i} \tag{3.2}$$

More specifically:

$$\text{Runtime for a single iteration } (t_{T_i}) \ll \text{Total runtime for the entire game } (t_J) \tag{3.3}$$

As the total number of resources does not change, one user's submission strategy affects the number of resources that it is assigned. This in turn will affect all the other users. Nash equilibrium has been proven to be the social optimum point where no player can do better by unilaterally changing his strategy [5]; then if a user can do better by changing the strategy it submits tasks to the scheduler, the scheduler does not schedule in a fair manner. We will show that the scheduler can be gamed and "tricked" into starving a competing user, and that it is unable to cope with rapid changes in the submission profile.

4. Previous Work. Fair share algorithm was introduced in [3] and it was originally designed to allocate resources in time-sharing systems where the number of resources is limited and static. The research in this area [4, 6, 32, 33, 34] has been focused around job scheduling as opposed to task scheduling which represent a small part of an overall long-running job. Job fairness is measured based on actual start time as opposed to "could have" start times [35].

Paper [36] gives a good overview of the HPC schedulers available, with the majority of the schedulers under discussion being parallel task schedulers.

For the purposes of this research, we follow the scheduling taxonomy outlined in [37]. As such, we focused on the dynamic scheduling methodology [38], where jobs can and do come online dynamically. In such a

scenario, the goal of the scheduler then becomes maximizing resource utilization as opposed to lowering the overall runtime of the job [39]. We will show that this results in temporarily starving certain users, and as mentioned in the previous section, instills an imbalance.

Paper [40] argues that by increasing the number of resources in a given environment, the load imbalance would be reduced. Paper [41] uses priorities to alter scheduling decisions to reduce imbalance in the environment. We assume that all priorities are the same and do not change throughout the entire run.

Paper [4] considers restricted scheduling where a job must be assigned to a specific machine. We argue that a scheduler is optimal if it minimizes the social function: max total load, job latency, and weighted sum latency. In order to create a socially optimal solution, Nash equilibrium is considered as a basis of stability and the optimal social function. Paper [33] focuses on scheduling of jobs on identical machines. It also uses Nash equilibrium (NE) as the basis of fairness, but it assumes that jobs cannot be interlaced, whereas in our model, a machine could be executing task from various users over the course of time.

Paper [35] compares the scheduling methodologies and fairness of a supercomputing cluster versus a uniprocessor system. Since jobs are not further decomposable and cannot be preempted the cluster's fair-share algorithm cannot be fair, and not necessarily "is not" fair. Paper [32], in addition to considering batch job scheduling environment, uses the notion of social justice [42] as the measure of fairness. A schedule is fair in such a scenario if no job is affected by a later-arriving job. No research has been done for the topic of fairness as it pertains to task scheduling in shared environments with potentially conflicting interests.

Paper [25, 43] defines a congestion game to be a scenario where the payoff of a given player depends on the resource it is assigned, and how busy that resource may be based on how many other players are also on that resource. That paper, focusing on having identical machines and same-size jobs, argues that a scheduler may schedule a number of tasks on a given machine, if the number of pending tasks is greater than the total number of resources. As the number of tasks scheduled on a given machine increases – they are all competing for the same resource – the utility perceived by a given user could change based on how congested a resource becomes. This model does not work well with shared infrastructures and more specifically the model that each job is composed of a number of tasks (assumption 2). There could be other processes running on that machine, but the scheduler is strictly responsible for assigning one task to a resource.

In unrelated machines where the infrastructure is heterogeneous, [34] argues that congestion does not take place since the load of a given task is different on different machines. This model does not work in a shared infrastructure model as a job is completed if and only if all of its tasks have been completed. Based on this, unrelated machines cannot be treated any different than related machines. A shared resource infrastructure breaks down to its components of individual resources that need to work together to complete a set of tasks for a user. In short, there is an intrinsic commonality between related and unrelated resources in a shared environment. This commonality is the fact that each resource can contribute to the overall completion of a job.

There has also been research done around truthful algorithms as it pertains to scheduling tasks [44, 45]. In shared environments where there is little monetary incentive to be truthful about requirements, these algorithms do not work. In task-based systems, it is also very difficult to calculate the exact resource requirement as the tasks are each short in duration.

5. Methods. We modeled eight different cases, with each case representing a typical scenario on a system being managed by a fair-share scheduler. Each case compares two loads or submission strategies chosen by two competing users. We consider the following submission strategies in this paper:

- S_1: Normal-load submission strategy. This is the base load where all other strategies are compared against each other. For this strategy we assume that the user is submitting tasks at a constant rate throughout the simulation.
- S_2: Low-load submission strategy. For this strategy we assume that the user is submitting tasks at a constant rate but at some fraction of the normal strategy throughout the simulation.
- S_3: Sine-load submission strategy. For this strategy we assume that the user is submitting tasks in a start-stop fashion that resembles a sine wave
- S_4: Cosine-load submission strategy. For this strategy we assume that the user is submitting tasks in a start-stop fashion that resembles a cosine wave (time shifted sine wave)
- S_5: Burst-load submission strategy. For this strategy we assume that the user is submitting all tasks in a

very short amount of time and waits for results.

• S_6: Exponential-load submission strategy. For this strategy we assume that the user is submitting tasks at an increasing rate over time.

• S_7: Early-riser-load submission strategy. For this strategy we assume that the user is submitting all tasks in a very short amount of time before other users enter the system.

• S_8: Delayed-load submission strategy. For this strategy we assume that the user starts submitting tasks after all other users have entered the system.

The most basic load is denoted as the normal load submitted by c_1 where 1000 tasks are submitted by the user per time interval t, and that load does not change during simulation. Sine, cosine and step function workloads are aimed at demonstrating a start-stop workload where the user submits a certain number of tasks; waits and then submits another batch of tasks. Such loads are more indicative of real-world applications as a user may need to access an external data source (like a database for example) to gather the task data to submit. The exponential load will demonstrate the long-term effect of a shared infrastructure.

Case 1: Burst Workload where one user's workload is steady and low, and the other user comes in bursts. For purposes of keeping track, we have designated c_1 and c_5 to represent these two users.

Table 5.1: Task Submission Strategy for Test Case 1

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_1	S_1	$L_1(x,t) = 1000 * L_0(x,t)$
c_5	S_5	$L_5(x,t) = 10 * L_1(x,t) * U(t-a+b)$ $U(t-a+b) = \begin{cases} 1 & a < t < b \\ 0 & \text{otherwise} \end{cases}$

Where:

$$L_0(x,t) = 1$$

Case 2: Complementary workload where the timing of two users' workload complement each other. In this scenario, one user is idle while the other user is submitting tasks. For purposes of keeping track, we have designated c_3 and c_4 to represent these two users.

Table 5.2: Task Submission Strategy for Test Case 2

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_3	S_3	$L_3(x,t) = \max\{L_1(x,t)\sin(\pi t/2), 0\}$
c_4	S_4	$L_4(x,t) = \max\{L_1(x,t)\cos(\pi t/2), 0\}$

This type of workload is best suited for a shared environment as one user's busy period complements another user's idle time.

Case 2a: Complementary workloads with unbalanced profiles where the timing of two users' workload complement each other, but one user's workload is higher than the other's.

Case 3: Conflicting workloads where one user is steady but the second user is increasing its workload over time.

This case could conceivably be applied to a HPC environment where with the passage of time, more users join the environment with c_3 representing a single user and c_7 representing the load from the other users that

Table 5.3: Task Submission Strategy for Test Case 2a

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_3	S_3	$L_3(x,t) = \max\{L_1(x,t)\sin(\pi t/2), 0\}$
c_4	S_4	$L_4(x,t) = \max\{2 * L_1(x,t)\cos(\pi t/2), 0\}$

Table 5.4: Task Submission Strategy for Test Case 3

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_2	S_2	$L_2(x,t) = \frac{L_1(x,t)}{10}$
c_6	S_6	$L_6(x,0) = 1000 * L_0(x,t)$ $L_6(x,t) = L_6(x,t-1)^{1.01} \quad t \geq 1$

continually onboard the environment. The exponential load can be thought of as a way of demonstrating the rest of the users yet to come.

Case 4: Steady workloads where the two users have a steady stream of workloads that enter the system.

Table 5.5: Task Submission Strategy for Test Case 4

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_1	S_1	$L_1(x,t) = 1000$
c_2	S_2	$L_2(x,t) = \frac{L_1(x,t)}{10}$

Case 5: Early-riser workloads where one user enters the system early in order to claim resources.

Table 5.6: Task Submission Strategy for Test Case 5

Users	**Load Strategy**	**Load in** $\frac{\textbf{tasks}}{\textbf{time slice } (t)}$
c_7	S_7	$L_7(x,t) = 10 * L_1(x,t) * U(t-a)$ $U(t-a) = \begin{cases} 0 & a \leq t \\ 1 & 0 \leq t \leq a \end{cases}$
c_8	S_8	$L_8(x,t) = L_1(x,t) * U(t-b)$ $U(t-b) = \begin{cases} 1 & b \leq t,\ a \leq b \\ 0 & \text{otherwise} \end{cases}$

The total number of pending tasks waiting to be scheduled $Q_i(x,t)$ depends on the number of tasks that were scheduled and completed in the previous time slot.

$$Q_{c_i}(x,t) = L_{c_i}(x,t-1) + L_{c_i}(x,t) - r_{c_i}(t-1) \tag{5.1}$$

We further assume in our model that a given task T_k can be completed during the given time slot t, and that all the tasks are completed successfully. Introducing failure in our model would simply increase $Q_i(x,t)$,

and we are modeling that through known loads presented in this section.

6. Results. We will demonstrate the short-term effects of choosing a different submission strategy, followed by the long term effects. We will further elaborate on the scenario with limited number of tasks versus infinite tasks submitted by each user. Our simulations, unless otherwise expressed, ran under the following conditions:

Assume that $J(c_i) = \{T_1(c_i), T_2(c_i), \ldots, T_n(c_i)\}$ is a job set of user c_i. Function $T_j(c_i)$ means a job of user c_i with execution time T_j. $\|J(c_i)\|$ is the number of elements in the set J, that is, the number of tasks of user c_i. Based on these, we can see:

- Maximum number of tasks for both users was the same and capped to $T_{\max} = 100{,}000$

$$\|J(c_i)\| \leq T_{\max} \quad \forall i$$

It means for any user c_i, the number of tasks allowed to submit is no more than $T_{\max}$.

- All the tasks are the same length, and can be executed in a time slot

$$\forall_T \quad t_{T_i} = a$$
$$T_j(c_i) \equiv T_J \leq T_a \quad \forall j$$

$T_j(c_i)$ is referred to as the execution time of job T_j of user c_i and T_J is a constant, that is, the execution time of each and every job in the set $J(c_i)$. Thus, it means all the jobs in set $J(c_i)$ have a constant execution time, i.e., all the tasks in the set are of the same length. T_a is the length of a single time slot. Therefore, each and every task can be executed within one time slot.

- Simulation runs for 100 time slots

$$t = \{1 \ldots 100\}$$

Assume that $T_a(n)$ means the n-th time slot and thus here $n \in [1, 100]$.

- Scheduling takes place at the beginning of each time slot and its duration is negligible
- Total number of available nodes is constant throughout the simulation and is set to 500.

Case 1: For the first simulation, we modeled a scenario where one user is submitting tasks at a steady rate and at a time later, a second user submits the same number of tasks but in a much shorter time frame. Figure 6.1 depicts the input task profile for the two users. As can be expected, all the available nodes are dedicated to User 1 to start. As soon as User 5 starts submitting, the resources start shifting to User 5 (Figure 6.2), and it stays that way until the pending task size of the two users start to match up (Figure 6.3).

Case 2: This simulation models the case where the two users have complementary workloads. c_3 and c_4 are the two users tested, and one user's task submission takes places after the other user has completed submitting its tasks to be executed. The number of tasks submitted by the two users is exactly the same for each time slot for the initial case (Figure 6.4), and it clearly demonstrates how the sharing of resources can take place with pure complementary workloads. The resource distribution stays consistent throughout the simulation (Figure 6.5).

Case 2a: We extended this simulation with one of the users (c_4) having 2× the tasks per time slot. At the beginning of the simulation, as expected, the resource allocation was fairly dynamic in the way resources were shifted from one user to the next. As time went on, and as the pending task queue trumpeted the incoming task rate, the resource allocation became more static in that c_4 became the more dominant (Figure 6.6) receiver of resources.

Even with complementary task profiles, if rate of incoming task rate is not equal, the pending task queue inequality will cause one user to dominate the resource pool.

Case 3: This case deals with an environment where one user's workload is a very small fraction of the overall workload submitted by other users, (Figure 6.7). This is true as more users are on-boarded to the environment. Case 3 deals with this scenario, and demonstrates the resource allocation for c_2 as the number of tasks increase actually begins to decline (Figure 6.8).

Case 4 is a trivial case where the two users are sending tasks at a steady rate, but one user's task submission is lower consistently. The resource allocation does not change throughout as expected (Figure 6.9).

Case 5: this scenario represents a user entering the system earlier than the other user. Both users are submitting the same number of tasks to the system, but c_7 does so in a much shorter timeframe (1/10 of the

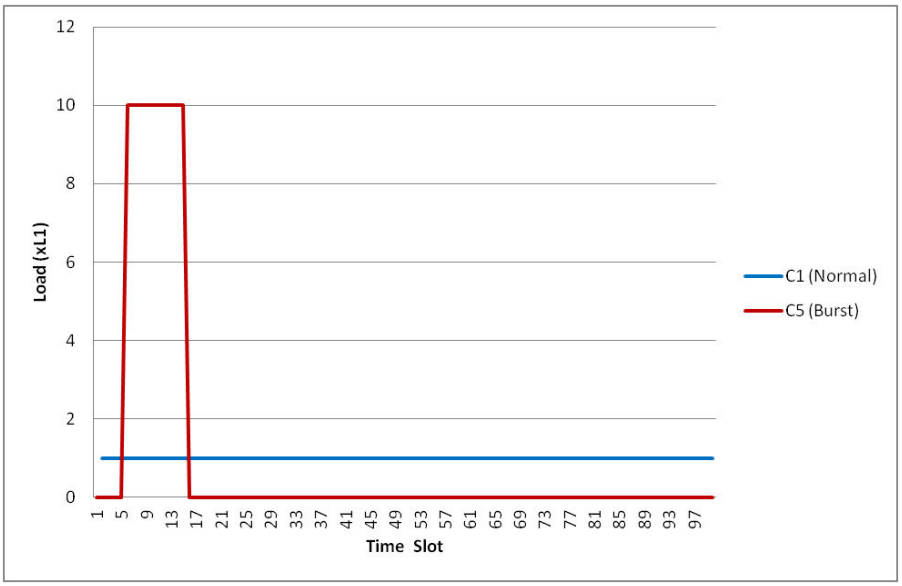

Fig. 6.1: Input task profile for Case 1 (Load is the multiplier of L_1)

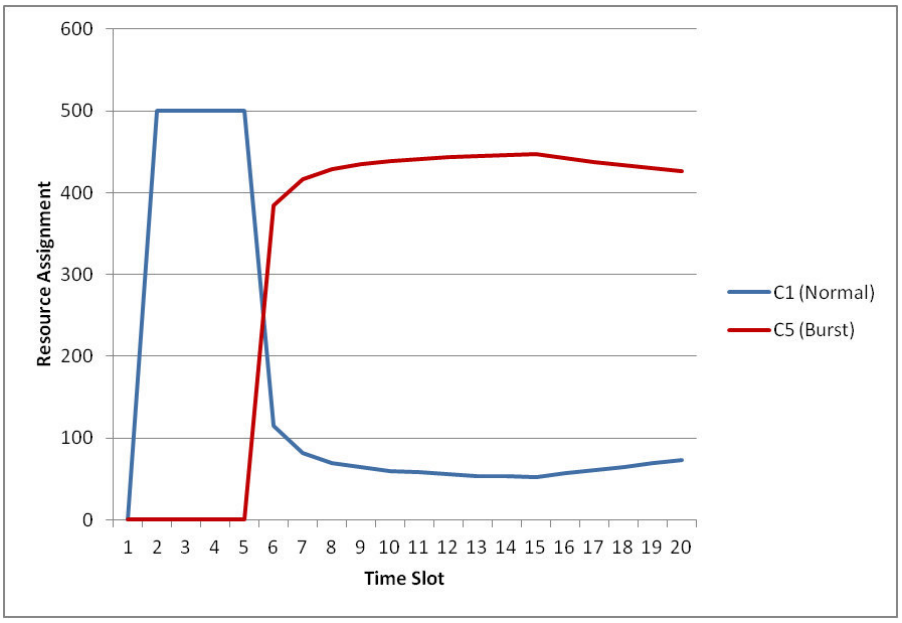

Fig. 6.2: Resources shift to User 5 and starve User 1 ($t = 1 \ldots 20$)

total submission time) (Figure 6.10). The resources are dedicated to c_7 until c_8 builds up a queue of equivalent size (Figure 6.11).

7. Analysis. Fair share does well with like workloads. As such, it intrinsically normalizes the workloads to make them alike. In doing so, the pending task queue of users with few pending tasks tend to suffer. This situation creates a false-sense of fairness and allows power players to monopolize the system.

Case 1 showed a typical simulation where steady workload is derailed by a user capable of overloading the

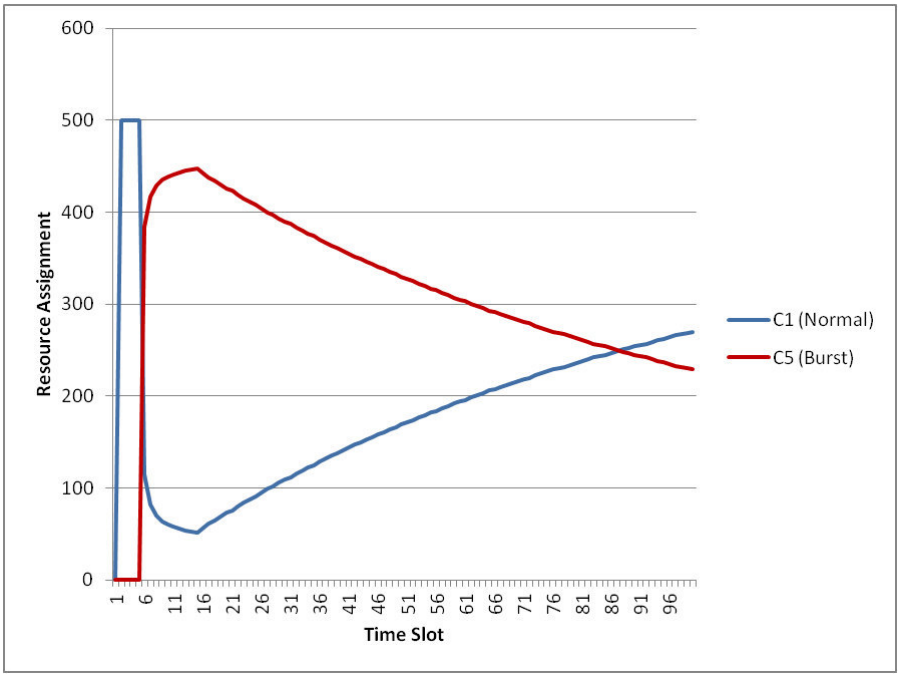

Fig. 6.3: Resource Allocation for Case 1

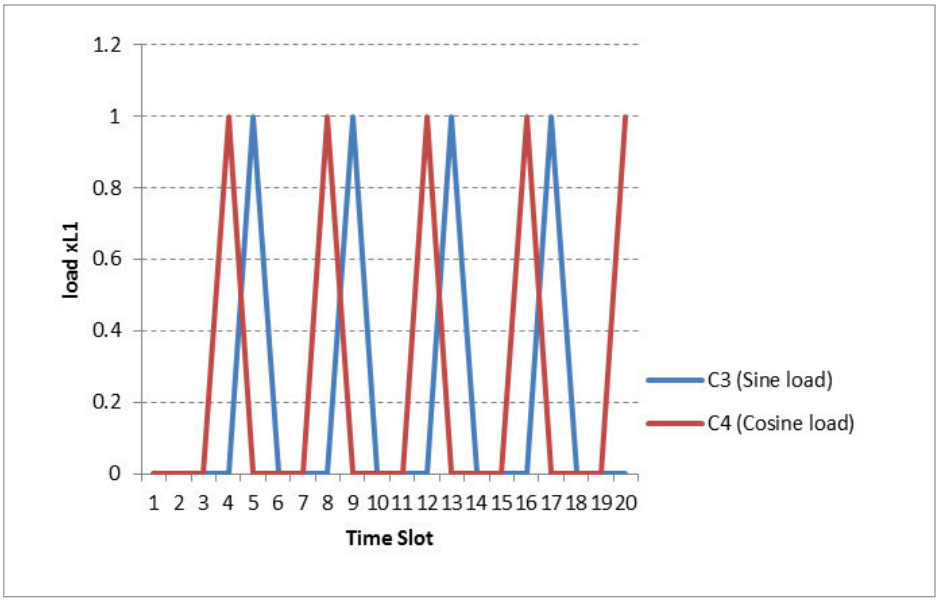

Fig. 6.4: Input task profile for Case 2 (Load is the multiplier of L_1)

system. User c_5 has the ability to burst tasks into the system in a short amount of time, and is thus capable of gaming the scheduler to assign the majority of the resources to it until its queue is drained. Case 2 shows complementary work profiles that qualify for a perfect sharing of resources. On the other hand, if the job profiles are not exactly the same as in case 2a, the long-term effect will be one of the users being starved.

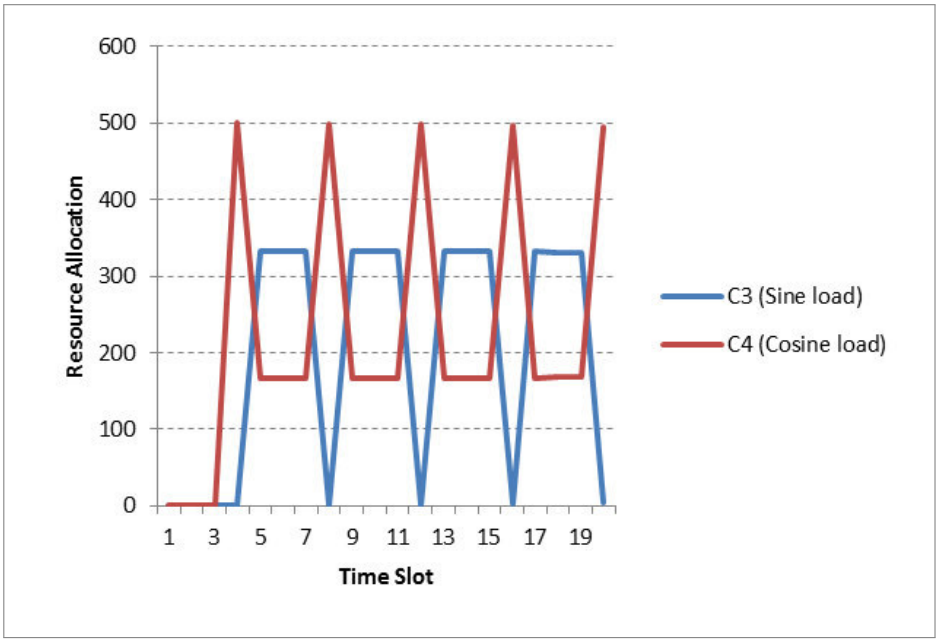

Fig. 6.5: Resource Allocation for Case 2

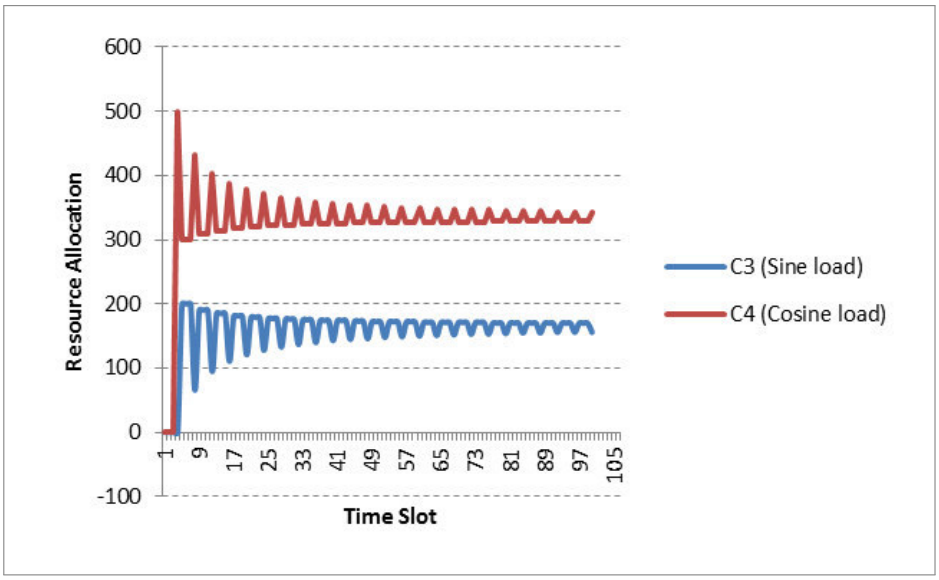

Fig. 6.6: Resource Allocation for Case 2a

Case 3 could represent a new user entering an already resource-lacking system. When there are many users waiting in the queue to be processed, a new user has a very tough time getting new resources. In heavily shared environments, cases 2a and 3 could become devastating to a new incoming user. Case 4 is where a fair-share scheduler is actually fair; steady workloads that span long periods of time.

Case 5 shows a late comer to a system that is already experiencing large queues. It shows that until the queues are drained, a newcomer will not get adequate resources.

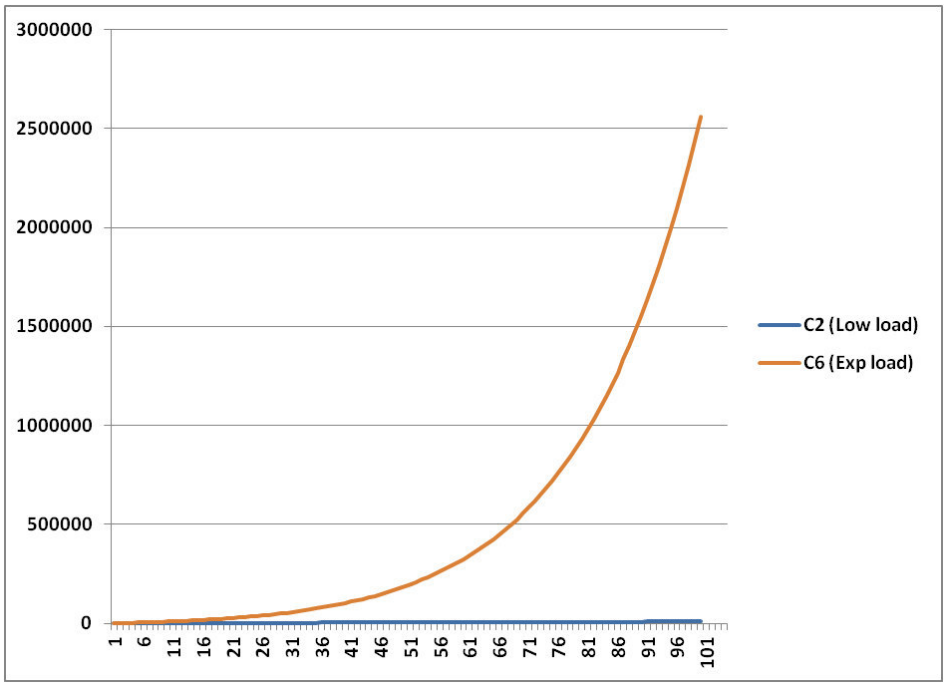

Fig. 6.7: Workload Comparison for Case 3

Fair-share algorithms are used widely as the most efficient way of scheduling tasks in HPC environments. We demonstrated in this paper that fair-share algorithms fail to do so in cases where the workloads vary in any significant way.

8. Conclusions and Future work. We revealed the shortcomings of a typical fair-share scheduling system for HPC environments. We demonstrated that fair-share schedulers normalize the queue depth first before being fair to all users. A fair-share scheduler requires that all the workloads be the same, all have the same start-time, all have the same end time, and all submit tasks to the system at the same rate. Any deviation from this will result in the scheduler having to compensate for the inequality by assigning more resources to the user[s] in a manner to create normalized workload across all users. This approach works for steady workloads where the request profile does not change. In such a scenario, fair-share schedulers show fair resource division across the users. For a complementary workload with conflicting profiles, fair-share schedulers give a false-promise of sharing, but are unable to deal with varying workload in the long run. In scenarios where workloads are simply different, the scheduler assigns resources to the user capable of loading the system at a higher rate relative to other users. This causes temporary starvation of users as shown in cases 1, 3 and 5.

Identifying such problems is the key contribution of the current work while solving them is our future research.

REFERENCES

[1] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis, *The structure and complexity of Nash equilibria for a selfish routing game*, in Automata, Languages and Programming, P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, and M. Hennessy, eds., Lecture Notes in Comput. Sci., Vol. 2380, Springer, Berlin, 2002, pp. 123–134.
[2] W. R. Stevens, M. Allman, and V. Paxson, TCP *congestion control*, Consultant (1999).
[3] J. Kay and P. Lauder, *A fair share scheduler*, Commun. ACM, 31 (1988), pp. 44–55.
[4] D. Ferraioli and C. Ventre, *On the price of anarchy of restricted job scheduling games*, in ICTCS, 2009, pp. 113–116.
[5] J. F. Nash, *Equilibrium points in n-person games*, Proc. Nat. Acad. Sci. U.S.A., 36 (1950), pp. 48–49.

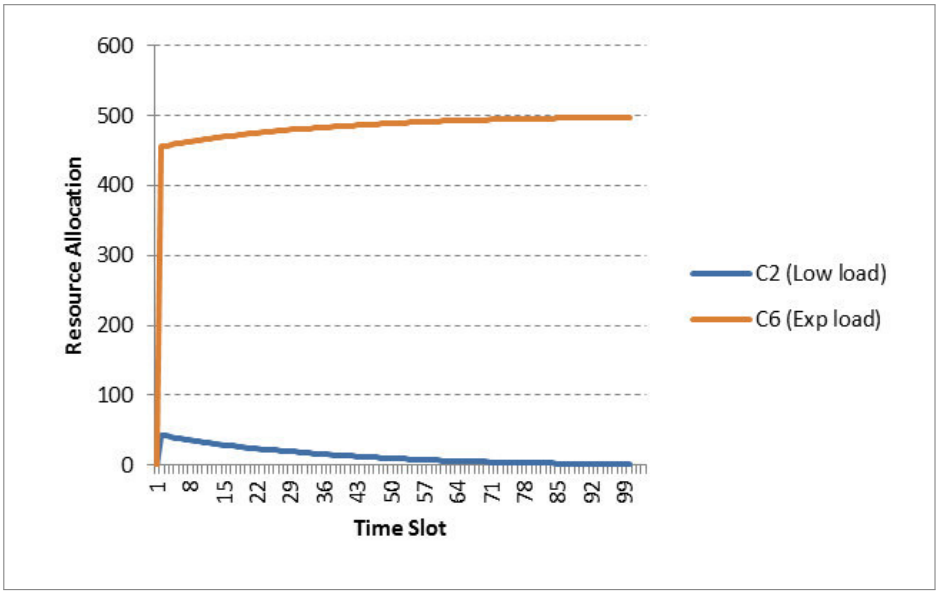

Fig. 6.8: Resource Allocation for Case 3

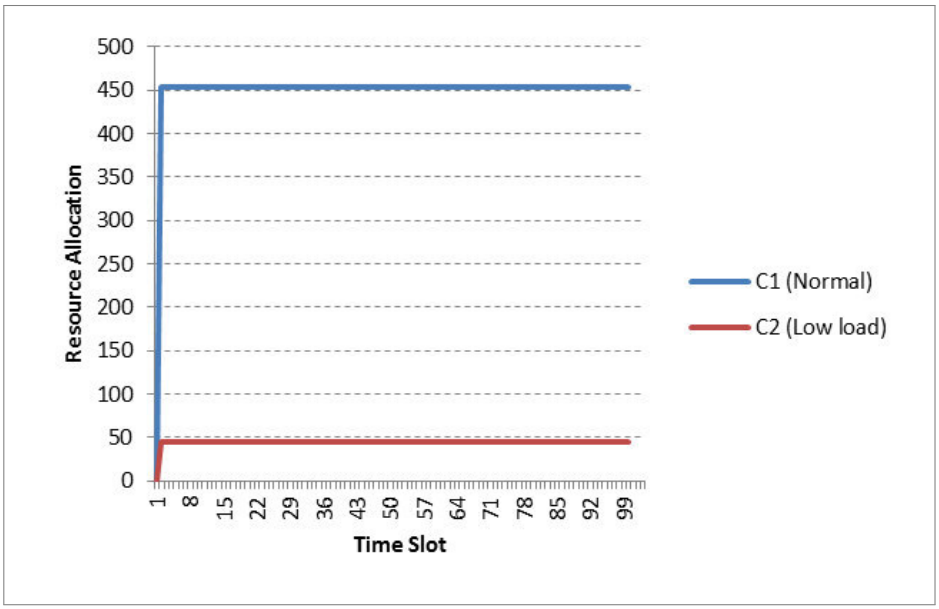

Fig. 6.9: Resource Allocation for Case 4

[6] A. Fiat, H. Kaplan, M. Levy, and S. Olonetsky, *Strong price of anarchy for machine load balancing*, in Automata, Languages and Programming, L. Arge, C. Cachin, T. Jurdziński, and A. Tarlecki, eds., Lecture Notes in Comput. Sci., Vol. 4596, Springer, Berlin, 2007, pp. 583–594.

[7] L. Epstein, M. Feldman, T. Tamir, Ł. Witkowski, and M. Witkowski, *Approximate strong equilibria in job scheduling games with two uniformly related machines*, Discrete Appl. Math., 161 (2013), pp. 1843–1858.

[8] M. Feldman and T. Tamir, *Approximate strong equilibrium in job scheduling games*, J. Artificial Intelligence Res., 36 (2009), pp. 387–414.

[9] C. Daskalakis, *Nash equilibria*: *Complexity, symmetries, and approximation*, Computer Science Review, 3 (2009), pp. 87–

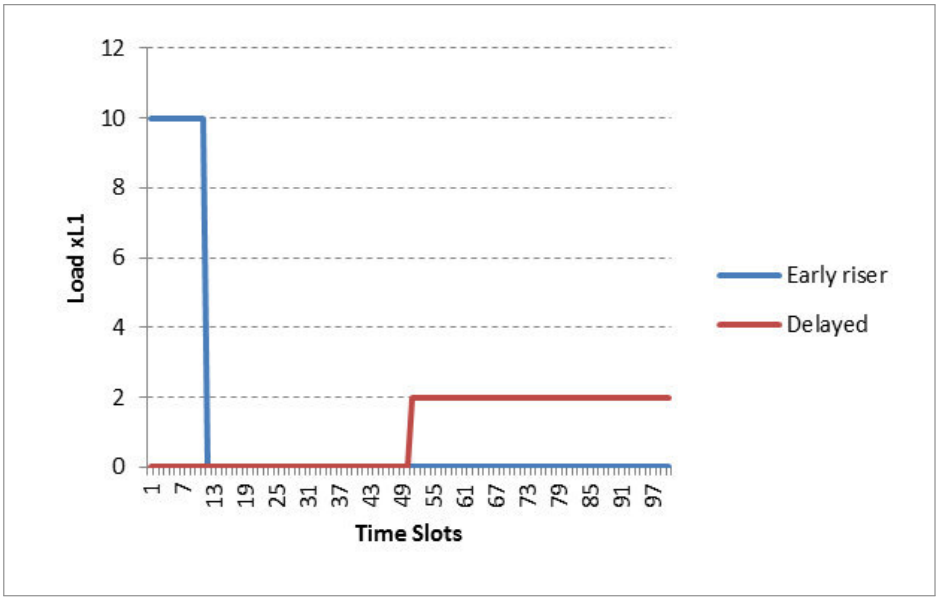

Fig. 6.10: Load Profile for Case 5

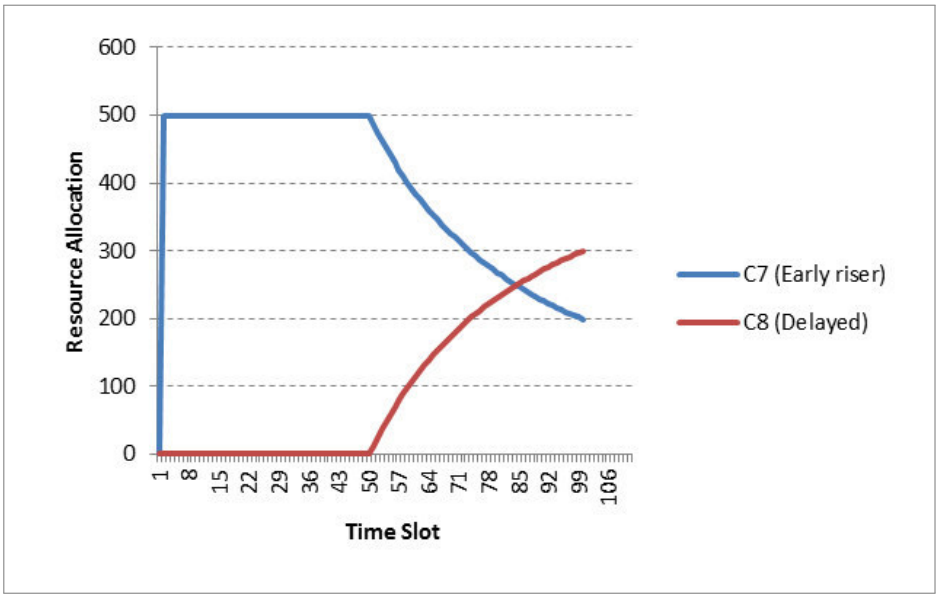

Fig. 6.11: Resource Allocation for Case 5

100.

[10] F. Pinel, J. E. Pecero, S. U. Khan, and P. Bouvry, *Energy-efficient scheduling on milliclusters with performance constraints*, in Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications, IEEE Computer Society, Washington, DC, 2011, pp. 44–49.

[11] L. Wang, S. U. Khan, D. Chen, J. Koodziej, R. Ranjan, C.-z. Xu, and A. Zomaya, *Energy-aware parallel task scheduling in a cluster*, Future Generation Computer Systems, 29 (2013), pp. 1661–1670.

[12] P. Zhang, Y. Gao, J. Fierson, and Y. Deng, *Eigenanalysis-based task mapping on parallel computers with cellular networks*, Math. Comp., 83 (2014), pp. 1727–1756.

[13] P. Zhang and Y. Deng, *Design and analysis of pipelined broadcast algorithms for the all-port interlaced bypass torus networks*, IEEE Trans. Parallel Distrib. Systems, 23 (2012), pp. 2245–2253.
[14] P. Zhang and Y. Deng, *An analysis of the topological properties of the interlaced bypass torus (*iBT*) networks*, Appl. Math. Lett., 25 (2012), pp. 2147–2155.
[15] Y. Deng, P. Zhang, C. Marques, R. Powell, and L. Zhang, *Analysis of Linpack and power efficiencies of the world's* TOP500 *supercomputers*, Parallel Comput., 39 (2013), pp. 271–279.
[16] J. J. Dongarra, H. W. Meuer, and E. Strohmaier, TOP500 *supercomputer sites*, Supercomputer, 13 (1997), pp. 89–120.
[17] P. Zhang, R. Powell, and Y. Deng, *Interlacing bypass rings to torus networks for more efficient networks*, IEEE Trans. Parallel Distrib. Systems, 22 (2011), pp. 287–295.
[18] R. Feng, P. Zhang, and Y. Deng, *Network design considerations for exascale supercomputers*, in Proceedings of the 24th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2012), 2012, pp. 86–93.
[19] D. Klusácek, *Scheduling in grid environment*, Ph.D. thesis, Masaryk University, Brno, Czech Republic, 2008.
[20] M. Q. Xu, *Effective metacomputing using* LSF *multicluster*, in Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, 2001, p. 100.
[21] W. Gentzsch, *Sun grid engine*: *Towards creating a compute power grid*, in Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, 2001, pp. 35–36.
[22] H. Feng, V. Misra, and D. Rubenstein, PBS: *a unified priority-based scheduler*, in ACM SIGMETRICS Performance Evaluation Review, 35 (2007), pp. 203–214.
[23] F. Berman, G. Fox, and A. J. Hey, *Grid Computing*: *Making the Global Infrastructure a Reality, Vol. 2*, John Wiley and Sons, New York, 2003.
[24] Adaptive Computing, *Moab* HPC *Suite - Basic Edition* 8.0.0, September 2014.
[25] E. Huedo, R. S. Montero, and I. M. Llorente, *The GridWay framework for adaptive scheduling and execution on grids*, Scalable Computing: Practice and Experience, 6 (2001).
[26] M. Romberg, *Unicore*: *Beyond web-based job-submission*, in Proceedings of the 42nd Cray User Group Conference, 2000, pp. 22–26.
[27] J. Huang, Y. Wang, N. Vaidyanathan, and F. Cao, GRMS: *A global resource management system for distributed* QoS *and criticality support*, in Proceedings of the 1997 International Conference on Multimedia Computing and Systems, IEEE Computer Society, Washington, DC, 1997, pp. 424–432.
[28] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, *Size-based scheduling to improve web performance*, ACM Transactions on Computer Systems, 21 (2003), pp. 207-233.
[29] K. Aida, *Effect of job size characteristics on job scheduling performance*, in Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph, eds., Lecture Notes in Comput. Sci., Vol. 1911, Springer, Berlin, 2000, pp. 1–17.
[30] S. Baruah, S. Funk, and J. Goossens, *Robustness results concerning* EDF *scheduling upon uniform multiprocessors*, IEEE Trans. Comput., 52 (2003), pp. 1185–1195.
[31] J. Nash, *Non-cooperative games*, Ann. of Math. (2), 54 (1951), pp. 286–295.
[32] V. J. Leung, G. Sabin, and P. Sadayappan, *Parallel job scheduling policies to improve fairness*: *a case study*, in Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, IEEE Computer Society, Washington, DC, 2010, pp. 346–353.
[33] G. Christodoulou, L. Gourves, and F. Pascual, *Scheduling selfish tasks*: *About the performance of truthful algorithms*, in Computing and Combinatorics, G. Lin, ed., Lecture Notes in Comput. Sci., Vol. 4598, Springer, Berlin, 2007, pp. 187–197.
[34] N. Andelman, M. Feldman, and Y. Mansour, *Strong price of anarchy*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 2007, pp. 189–198.
[35] S. D. Kleban and S. H. Clearwater, *Fair share on high performance computing systems*: *What does fair really mean?*, in Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, 2003, pp. 146–153.
[36] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, and N. Ghani, *A survey on resource allocation in high performance distributed computing systems*, Parallel Comput., 39 (2013), pp. 709–736.
[37] T. L. Casavant and J. G. Kuhl, *A taxonomy of scheduling in general-purpose distributed computing systems*, IEEE Trans. Software Engrg., 14 (1988), pp. 141–154.
[38] H. El-Rewini, T. G. Lewis, and H. H. Ali, *Task Scheduling in Parallel and Distributed Systems*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1994.
[39] H. A. James, *Scheduling in metacomputing systems*, Ph.D. thesis, Department of Computer Science, University of Adelaide, Australia, 1999.
[40] C. Boneti, F. J. Cazorla, R. Gioiosa, A. Buyuktosunoglu, C.-Y. Cher, and M. Valero, *Software-controlled priority characterization of* power5 *processor*, ACM SIGARCH Computer Architecture News, 36 (2008), pp. 415–426.
[41] C. Boneti, R. Gioiosa, F. J. Cazorla, and M. Valero, *A dynamic scheduler for balancing* HPC *applications*, in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE Press, Piscataway, NJ, 2008, p. 41:1–41:12.
[42] R. C. Larson, OR *forum-perspectives on queues*: *social justice and the psychology of queueing*, Oper. Res., 35 (1987), pp. 895–905.
[43] R. W. Rosenthal, *A class of games possessing pure-strategy Nash equilibria*, Internat. J. Game Theory, 2 (1973), pp. 65–67.
[44] E. Angel, E. Bampis, and N. Thibault, *Randomized truthful algorithms for scheduling selfish tasks on parallel machines*, in LATIN 2010: Theoretical Informatics, A. López-Ortiz, ed., Lecture Notes in Comput. Sci., Vol. 6034, Springer, Berlin, 2010, pp. 38–48.
[45] E. Angel, E. Bampis, and F. Pascual, *Truthful algorithms for scheduling selfish tasks on parallel machines*, Theoret.

Comput. Sci., 369 (2006), pp. 157–168.

Edited by: Dana Petcu
Received: July 29, 2014
Accepted: Sept 21, 2014

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2_ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.