Agent-oriented Foraminifera Habitat Simulation

Maciej Kaziród¹, Wojciech Korczyński¹, Elias Fernández¹,²
Aleksander Byrski¹, Marek Kisiel-Dorohinicki¹, Paweł Topa¹,³
Jarosław Tyszka³, Maciej Komosinski⁴

¹AGH University of Science and Technology, Faculty of Computer Science
Electronics and Telecommunications, Al. Mickiewicza 30, 30-962, Kraków, Poland
²University of Vigo, Faculty of Telecommunications
Campus Universitario, 36-210 Vigo (Pontevedra), Spain
³ING PAN – Institute of Geological Sciences, Polish Academy of Sci., Research Centre in Cracow
ul. Senacka 1, 31-002 Kraków, Poland
⁴Institute of Computing Science, Poznan University of Technology
ul. Piotrowo 2, 60-965 Poznan, Poland

{kazirod.maciej,eliasfernandez.d}@gmail.com
{wojciech.korczynski,olekb,doroh,topa}@agh.edu.pl
ndtyszka@cyf-kr.edu.pl
maciej.komosinski@cs.put.poznan.pl

Abstract
An agent-oriented software solution for simulation of marine unicellular organisms called foraminifera is presented. Their simplified microhabitat interactions are described and implemented to run the model and verify its flexibility. This group of well fossilizable protists has been selected due to its excellent “in fossilio” record that should help to verify our future long-run evolutionary results. The introduced system is built utilizing PyAge platform and based on easily exchangeable components that may be replaced (also in runtime). Selected experiments considering substantial and technological efficiency were conducted and their results are presented and discussed. We show that our artificial life simulation follows realistic rules known from nature. The Lotka-Volterra predator-prey dynamics was simulated that proves the potential for extendability to more complex biological phenomena.

Keywords: Agent-based simulation, Foraminifera, PyAge, microhabitat

1 Introduction

Foraminifera are single-celled eukaryotes that occupy marine benthic and pelagic zones. They have an extraordinary fossil record since the Cambrian (about 540 million years ago). Their cells usually produce organic and/or mineral shells which are easily fossilizable, leaving perfect signals of palaeoenvironmental conditions “frozen” in time and space [3]. During reproduction...
foraminifera leave their shells, which are accumulated on the sea floor, as a part sedimentary record. This makes them an ideal model organism and microfossil often used for palaeoreconstructions and testing general evolutionary hypotheses [7, 16, 21].

Since computer simulations can be helpful in understanding foraminifer evolution, our goal is to construct a model reflecting growth processes of foraminiferal individuals (ontogenesis) within a simplified virtual environment where they could move, eat, grow and reproduce, following rules that resemble their physiology and behavior. The ultimate aim is to test micro- and macroevolutionary rules and interactions controlling the overall system which acts at various spatiotemporal scales necessary for the emergence of complexity of life [12].

Modeling foraminifera started with the pioneering work of Berger (1969) [1]. He proposed the first theoretical morphospace using a simple geometrical model, where circular chambers (in a 2D space) were rotated around a fixed reference point. Berger’s model was able to produce spiral forms only. Similar models were proposed later [3, 6], yet the common assumption in these models was calculation of all geometrical transformations against the fixed reference point. For this reason, these models produced a small variety of stable forms.

In 2002 Topa and Tyszka proposed a new model with the moving reference system [22, 25]. Unlike the previous models, location of a new chamber is calculated with respect to the previous chamber. The new model is able to produce shells with very different growth patterns – spiral, uniserial, biserial or combined [25, 24]. The model is also able to produce forms that have never existed in nature for various reasons [24]. As the model uses up to 6 parameters, the analysis of its morphospace is a challenge. The forms generated by the model can be selected and promoted for reproduction using the exogenous-fitness evolution (as in evolutionary algorithms) and endogenous-fitness (spontaneous) evolution where survival depends on the environment, resources, and individual interactions [13].

The paper is devoted to the presentation of agent-oriented software solution for foraminifera habitat simulation leveraging agent-oriented approach. The notion of agency suits well simulation tasks, especially in the case when many individuals that exhibit certain characteristics are considered. Therefore, autonomy of agents [26] may be efficiently utilized in expressing the behavior of many beings, making easy the process of designing, implementation and execution of defined simulations. It is to note that requirements posed by applications such as foraminiferal habitat, exclude simpler approaches (such as applying Cellular Automata), as the interactions between the environment and the beings simulated and different parameters (such as salinity of the water, currents, insolation) may take real advantage from applying agency notion.

In order to efficiently address requirements of such simulations, versatile software platforms have been proposed [12] (e.g. Swarm [18], Repast [20], MASON [17], Framsticks [14, 13]). However, only a few of them provide support for large-scale simulations (e.g. RepastHPC [5] or distributed and parallel Framsticks architectures [14]). Such simulation platforms might greatly benefit from introduction of a (lightweight) component support.

We discuss a particular system based on the PyAge platform built using Python. This makes the system very easy to modify and further develop with a capability of running in a distributed environment supported by Pyro1. The system was firstly devoted to solving computational problems [11], focusing on efficiency of the software measured from both a technical point of view (capability of simulating of a certain number of foraminifers) and substantial (relevance of the first obtained results to the ones present in the literature).

This paper begins with an introduction and characterization of the habitat of foraminifera included in Sect. 2. Sect. 3 introduces the PyAge platform and describes its architecture and technical characteristics. In Sect. 4, we outline how PyAge platform was applied to the simu-

---

1. https://pypi.python.org/pypi/Pyro4
lation of the foraminifera habitat. The results of our simulation experiments are recapitulated in Sect. 5. Final discussion and conclusions are presented in Sect. 6.

2 Foraminiferal microhabitat

Foraminifera produce shells (tests) covering their soft cytoplasmic bodies. A multichambered shell is developed throughout a whole life of foraminifera by adding successive chambers to the existing shell. In nature, we observe an enormous variety of shell shapes (and chambers), however for many species spheroidal chambers may be a close approximation. Communication between an internal part of shell and the environment is provided by an aperture (a hole) located in each chamber. Foraminifera extend reticulopodia (network pseudopodia) through the aperture (or apertures) in order to gather food, move, and communicate [3, 19].

Foraminifera occupy two habitats, benthic and pelagic. Benthic foraminifera live either on the sea floor around the water/sediment interface or within the top 10 cm of soft, usually fluidal, sediment. Others are attached to all hard surfaces available on the sea floor. Planktonic foraminifera live in open ocean floating in the photic zone of the water column [3].

Foraminifera, like all other organisms, have to dissipate energy to survive, move and reproduce. Most of them feed on dissolved organic molecules, particulate organic matter, bacteria, single-celled algae and small animals such as copepods. In order to simplify feeding strategy, we can assume that the most typical is feeding on algae, which are usually strongly dependent on their availability in time and space. The most common temporal variability is reflected in seasonality associated with temperature and availability of nutrients. In consequence, algae tend to show blooming behaviors. Variability in space is reflected in patchiness observed in various scales dependent on micro- and macroscale environmental dynamics. All these factors have a direct impact on distribution, life history strategies, reproduction modes, and populations dynamics of foraminifera [10, 19].

Foraminifera are able to sense their microenvironments around the cell thanks to extension of large reticulopodial structures. We can assume that if nourishment is available, benthic foraminifera are supposed to stay and feed and iteratively grow by adding chambers following certain portions of food digested. If there is shortage of food, foraminifera can use at least two strategies: (1) wait for food and save energy or (2) move to another, better location. Movement costs energy as well, but might shorten time period for a next food influx.

Foraminiferal life span ranges from a few weeks in some planktonic foraminifera up to a few years in larger benthic foraminifera [9, 10]. A typical life cycle of benthic foraminifera is characterized by complex reproduction modes [9] and life history strategies [19] that are simplified, in our model, to asexual cloning with random mutations. This complexity of strategies will be modelled following realistic genetic and ecological rules in future.

3 PyAge

PyAge is an agent-based computing and simulation platform built with the use of Python technology [11]. Computations may be performed on a single machine or in accordance with the distributed computing model. The discussed platform is composed of loosely coupled components, therefore its functionality may be freely extended by providing custom implementations of the given components. Moreover, these parts can be easily replaced in runtime.

Since PyAge has been created as an agent-based platform, its most important entity is an agent. Agents operate in their basic environment – the workplace. The workplace organizes
agents’ activities by executing their methods in a proper order. The platform is responsible for creation of new agents, controlling their functioning and providing other services.

Fig. 1 presents an overview of the platform architecture. Agents operate in workplaces which are located on separate machines called nodes. Agents are not closely connected with their workplaces – they are able to communicate with other agents and migrate to another workplace, or even to another node.

Figure 1: Illustration of the platform architecture

An agent has been defined as an object with a certain, specified set of methods. Implementation of the main logic related to the computations is contained in a step method. Each agent’s step method is called by its workspace in successive iterations, therefore, agents within one workspace work sequentially.

PyAge is composed of loosely coupled components, thus single modifications do not result in a necessity of changing the implementation of another parts of the platform. Moreover, a component-based structure provides the platform extensibility and flexibility. The former is ensured by the possibility of adding completely new components, while the latter is guaranteed by a reusability of various components providing the same services.

The platform component is a Python object which provides some well-defined services. In order to create user’s own component and use it instead of the currently existing one, one has to implement an object with the same set of methods. There is no necessity of implementing any interfaces or extending any base class, as – according to the Python assumptions – an object is defined by a set of its methods. Such an approach frees one from the responsibility of maintaining the specific hierarchy of types, yet it still allows for using object-oriented techniques such as polymorphism or inheritance.

Users select the proper components just by editing a configuration file, so all the changes in the platform structure and the computation processes are made in the most simple manner.

Python technology, thanks to a duck typing, supports dynamic exchangeability of platform components. It validates code semantics by proving that given object owns a particular set of methods rather than inherits from a specific class or implements some explicit interface [23]. As type checking is done at runtime, programs implemented in Python might function fully dynamically and any element (field, method, class) of the system may be replaced at any moment.
Simulations performed by the PyAge platform are based on an event-driven paradigm. In such a system, operations are modeled by a discrete sequence of events occurring during a system lifetime. This method allows to create the pseudo-parallel systems which are more friendly in development and maintenance. Moreover, event-based simulation systems are often more efficient than systems based on a continuous simulation and do not require complex mechanisms of thread synchronization [4].

With regard to agent-based systems, event-based simulation allows to apply lightweight agents which remarkably increases efficiency of such systems. Many of available simulation platforms and tools (e.g. JADE) use heavy agents implemented as threads which affects negatively their performance and scalability [15].

4 Application of PyAge to foraminiferal microhabitat simulation

Simulation consists in tracking changes in the foraminiferal habitat, representing a small segment of marine floor. The habitat is conveniently modeled as a 2-dimensional or 3-dimensional grid. The complexity of the simulation is characterized as follows:

- Workplaces and agents become a base for the whole simulation, taking role of simulated entities (e.g. benthic foraminifers) and implementing efficient, scalable framework, ready for running in distributed environments.
- Cellular Automata grid may be treated as a legacy from Cellular Automata simulations, however the simulation of foraminiferal physiology and behavior is so complex that implementing simple rules (as in e.g. Conway’s Game of Life [8]) is not enough. Therefore, the grid itself may be treated as a basis for spatial distribution of simulated entities.
- Benthic foraminifers are the main entities that are developing, moving, foraging, reproducing, and at the same time interacting with themselves and with other entities and resources.
- Algae are modeled as food for foraminifers – they have appropriately parameterized growth strategy. Foraminifers within our model forage on algae, and when they lack them in their vicinity, they may move in the direction of the highest algae density gradient.
- Other exemplary characteristics of the environment, such as insolation, salinity or currents (they will be further extended) affect living conditions of foraminifers, their movement, reproduction and foraging strategies, as well as the growth of algae, etc.

In order to implement the above-mentioned layers, the following patterns were introduced into PyAge:

- Simulation takes place in a workplace modeled as a foraminiferal microenvironment, which is built on a grid of a certain fixed size. Foraminifera, modeled as agents, are placed in grid cells alongside with their nourishment, algae. Moreover, a neighborhood of each cell has been defined as a surrounding field within a radius of two cells.
- Although, at this stage of development no evolutionary operations are used, a foraminifer has a genome containing information specific for a given individual, such as energy consumption, its ability to grow and reproduce. During reproduction, genetic information is transferred to the next generation.
- All of the foraminiferal actions are performed in simulation steps. These actions include: moving, consuming algae or reproducing. Before an action, foraminifers may interact with the environment in order to inspect whether a given action is allowed (i.e., if there is a free cell to move to or if there is food available).
- In a simulation step, each foraminifer dissipates a certain defined amount of energy. The amount of consumed energy is directly proportional to the number of its chambers. Foraminifer tries to replenish its energy by eating certain amount of algae. However, in case of food shortage, it moves in a direction determined by a nutrition gradient represented by quantity of algae simulated around a foraminiferal neighborhood. Every move also costs a portion of energy. Foraminifer dies when its stored energy is exhausted and not available around.
- If energy stored by foraminifer exceeds a certain defined level, a new chamber is constructed. If a number of chambers reaches specified limit and the level of energy is high enough, a foraminifera can reproduce – new individuals appear in free neighboring cells. Adult foraminifer deceases after reproduction.
- The platform may record data on “dead” foraminiferal specimens to further test composition and dynamics of virtual “fossil assemblages”.
- The environment also performs actions in every simulation step, e.g. simulating growth of algae. Other ecological processes can also be simulated. For example, the growth of the algae could depend on light intensity, nutrient availability, temperature and/or salinity.

All elements mentioned above have been integrated into the component-based architecture of PyAge. Loose coupling assures that each component can be replaced with an alternative implementation of the same interface – for example, a grid can be changed from 2- to 3-dimensional without altering the implementation of foraminiferal behavior.

In the configuration file, users can specify which components should be used in the simulation and assign values to parameters that define the environment.

5 Experimental results

Foraminiferal habitat can be modeled in 2D and 3D. Figure 2 presents a snapshot from simulations performed for 3D benthic microhabitat, preprocessed and visualized with the software for scientific visualization Amira. Simulations were run on a grid with 50 × 50 × 50 cells for 3000 steps. Foraminifers are visualized as spheres with colors that corresponds to its level of stored energy. The size of spheres represents a volume of cytoplasm filling in all chambers of a virtual shell. Chambers are not visualized. Tiny green globules represent algae. The sediment is neglected to simplify the model. The results can be represented in more impressive way when consecutive steps of simulation are visualized as an animation. Such presentation allows for qualitative verification and analysis of behavioral interactions.

Dynamics of foraminifera and algae population are presented on A and B charts. The difference between them lies in the quantity of the starting population, A: 800 foraminifers, B: 8 foraminifers (initial density of algae population was equal in both cases). In both cases modeled systems present similar behavior — shifted phase of oscillations of foraminifera population and algae. Amplitudes of oscillations are gradually decreasing. The average size of the population stabilizes around the carrying capacity which was similar for both simulations. This results closely resembles the behavior observed for the classical predator-prey models (such as, e.g., Lotka-Volterra model [2]).

To test the scalability of our platform, we have performed several simulations with different grid sizes and measured their execution time. The results, as we comment below, show that the evolution of time in respect to the number of cells on the grid is almost linear. For these tests, we have considered two different scenarios: a simulation with a grid shortly loaded at

http://www.fei.com/software/amira-3d-for-life-sciences/
the beginning (only 8 foraminifera for all the grid simulated) and a grid heavily loaded from
the start(with 0.8 * (GRID_SIZE)^{\frac{1}{2}} of foraminifera). Both scenarios have been applied for
simulations with 2-D and 3-D grids. All tests were repeated 10 times and each simulation were
run through 900 steps. Also, the limit of chambers used was set to 5. The results can be seen
on Figures 3 and 4.

\footnote{For the simulations with a 3-D grid, the second scenario’s initial foraminifera population is 0.8 *
(GRID_SIZE)^{\frac{1}{2}}}$
These tests have been performed on a cluster with x86_64 GNU/Linux as the operative system and the following hardware specifications - Dual-Core AMD Opteron(tm) Processor 1220 (64 bits) and 7871MB of RAM (32 bit) which has a clock of 66MHz.

\[ \text{Initial population of foraminifera} = 8 \times \sqrt{\text{GRID SIZE}} \]

Figure 3: Polynomial fit for the simulations with a 2-D grid

\[ \text{Initial population of foraminifera} = 0.8 \times \sqrt[3]{\text{GRID SIZE}} \]

Figure 4: Polynomial fit for the simulations with a 3-D grid

Fig. 3(a) and 3(b) show the time measured for simulations with a 2-Dimensional grid for different grid size configurations\textsuperscript{4}. From the graphs we can see that the increase in run time when the grid grows in size is almost linear, which is an important result regarding the usability of such a platform. Moreover, the time increases, as expected, when the initial number of foraminifera is higher, in other words, when there are more agents in the system. However, as it can be seen through the comparison of both graphs, the results from both tests are very

\textsuperscript{4}The grid sizes simulated for the tests with a 2-D grid were 10x10, 20x20, 50x50, 80x80, 100x100, 500x500, 800x800, 1000x1000, 1200x1200, 1500x1500, 1800x1800 and 2000x2000.
similar for grid sizes from 10x10x10 to 100x100x100 and only begin to differ noticeably from the simulation with the grid size 500x500x500, which is natural as the number of foraminifers also grows considerably. Nevertheless, this difference is not greater than one order of magnitude, which proves that our system scales well.

Fig. 4(a) and 4(b) show the same tests commented above for a 3-Dimensional grid. There is quite a difference in run time in comparison to the simulations with a 2-D grid. This is, however, perfectly logical as the size of the grid is much bigger in this case, also the movement operations needed in order to organize the agents within this type of grid are more complex. Nonetheless, the graphs still show a linear behavior, which shows that our platform can be useful for simulating 3D environments. Furthermore, there is a little difference between time performance in Fig. 4(b), where the simulation starts with a lower population of foraminifers, and Fig. 4(a). Even so, the memory consumption in Fig. 4(b) is higher.

The simulations have shown that our system scales well for different sizes of the grid. However, the memory consumption also grows when the environment is bigger (bigger grid sizes and more foraminifers). Therefore, the experimental setup should be optimized before running simulations.

6 Conclusions

The main goal of this work was to present the software and its implementation details for a highly flexible agent-based computing and simulation platform applied to the simulation of foraminifera habitat. We could show that artificial life simulation works and even with comparably few assumptions leads to emergence of patterns that are observed in nature.

The presented software built on a lightweight component-oriented platform PyAge turned out to be well-suited for solving the simulation task. The platform can be further parametrized in order to include new simulation aspects, increase the scale, dimensionality, etc.

We plan to further explore capabilities of the constructed platform by enhancing the model and comparing it with state-of-the-art regarding foraminiferal genetics, physiology, behavior and life strategies. We are also working on a similar platform that is being implemented in Erlang, in order to further explore distributed and multi-core computing.

Acknowledgments

The research presented in the paper received support from Polish National Science Center (DEC-2013/09/B/ST10/01734).

References


The grid sizes simulated for the tests with a 3-D grid were 10x10x10, 20x20x20, 30x30x30, 40x40x40, 45x45x45, 50x50x50, 60x60x60, 70x70x70, 80x80x80, 90x90x90 and 100x100x100.


