

# 5 Hungry philosophers' problem. Modelling with JADE

By Vadim Kimlaychuk, TTU  
[vadim@dcc.ttu.ee](mailto:vadim@dcc.ttu.ee)  
2003-06-08

**Abstract:** The management of shared resources in the systems sensitive to time is tried to be analyzed in multy-agent environment – JADE. As a model for that “5 hungry philosophers” problem (firstly formulated by Deijkstra) [1] was taken to be investigated. This article shows how this model can be represented using agent conception and methods of Extreme Programming [2]. The main goal is to build up the system to be capable to investigate the behaviour of the agents under different conditions. Since the environment is represented also by the agent it is possible to model not only static but also highly dynamic environment. Basically agents are not capable to learn and don't have the memory of passed events, but the system can be extended to do that in the future.

**General terms:** Agent, JADE, Real time

**Keywords:** hungry philosophers, JADE, agents, JAVA, real-time, multi-agent system

## Contents

1	Introduction.....	2
2	Process modelling .....	2
3	Agent approach .....	3
4	Implementation .....	4
4.1	System.....	4
4.2	Agents .....	5
4.2.1	Manager .....	5
4.2.2	Philosopher .....	5
4.2.3	Hospital.....	6
4.2.4	Table .....	6
4.3	Ontology .....	6
4.4	Executive Environment.....	6
5	Obtained Results .....	6
5.1	Time constraints.....	6
5.2	Homogeneous systems.....	7
5.3	Heterogeneous systems .....	8
6	References.....	9
7	About the author .....	10

# 1 Introduction

Classical description of the “5 hungry philosophers” problem assumes that there are five philosophers who work and feel hungry from time to time. When a philosopher feels hungry he goes to the table where there are 5 plates with rice and only one stick between neighbour plates. One of the possible (worst) states of such system is achieved when all 5 philosophers go to the table simultaneously, take one stick each to eat rice and no one can eat (we also assume that to be able to eat rice person needs 2 sticks).

There are many goals which we may achieve solving this problem. How to increase time the philosophers are working? How to avoid the worst situation (when all philosophers die)? How to keep satisfaction of the philosopher at the maximum level? And so on.

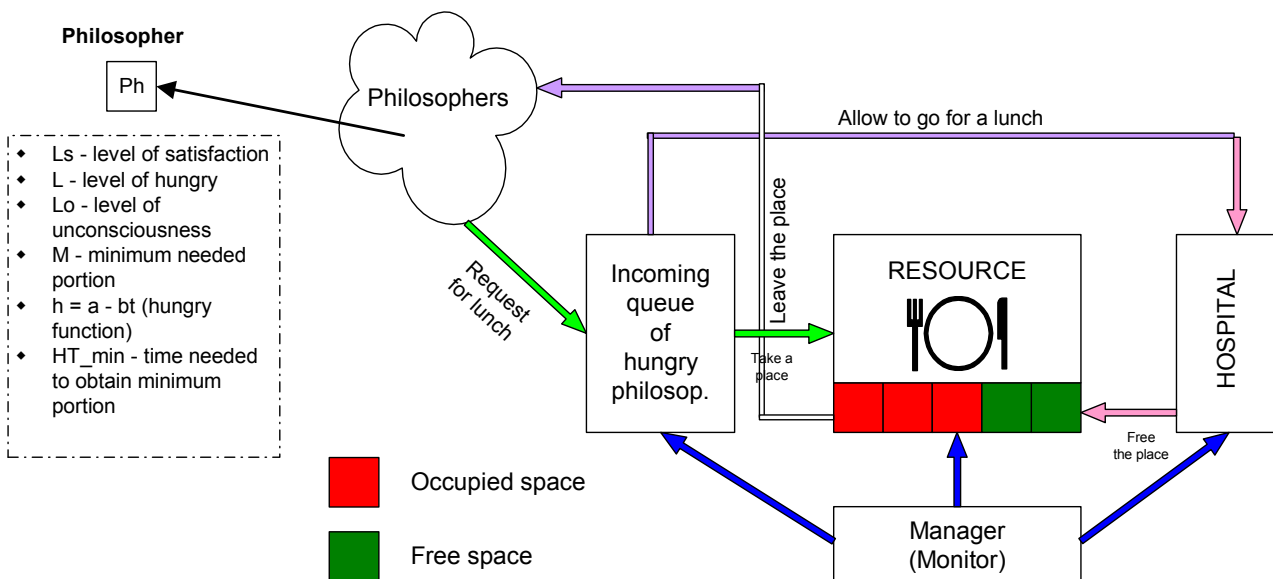
More parameters can be added to the system in order to simulate some real object. For instance, the level of unconsciousness can be added to the philosopher to represent the state when he is unable to go for the lunch by himself and external help (like hospital) is needed. Some examples of more sophisticated formulations of the problem can be found in [3], and [4].

# 2 Process modelling

As there is no need to be strictly followed by classical description, we have slightly modified the task and refused from “sticks”. Instead we have table with maximum 3 simultaneously available free places for 5 philosophers.

The model of system with additional parameters is shown on pic. 1

Functional diagram for the "5 hungry philosophers" problem



Pic N1 – Functional diagram of the system

### 3 Agent approach

All the actions were divided between four types of agents.

Philosopher:

- thinks and works
- keeps account of the consumed food (in a fixed period)
- becomes hungry (according to individual schedule) and applies for the food
- falls ill and applies for hospital, unless he regularly receives sufficient quantity of food in due time

Manager:

- monitors the status of philosophers (and other involved agents)
- attempts to influence the decisions of the table
- defines the functioning goal of the system – e.g. maximizes philosophers working time, or minimizes hospital expenses, or minimizes the consumed food, etc.

Table:

- controls the use of resources (simultaneously 3 of 5 is maximum usage)
- assigns the quantity of food to a customer
- selects customers from the queue according to its on rules

Hospital:

- cures philosophers suffering from being hungry too long
- takes care of the philosophers, gets extra food in a priority queue for them
- if and when a philosopher's health has improved, sends them back to work

Different types of agents carry the role for the different activities in the system. However basically the functions of the Hospital and Table were implemented in the Manager agent.

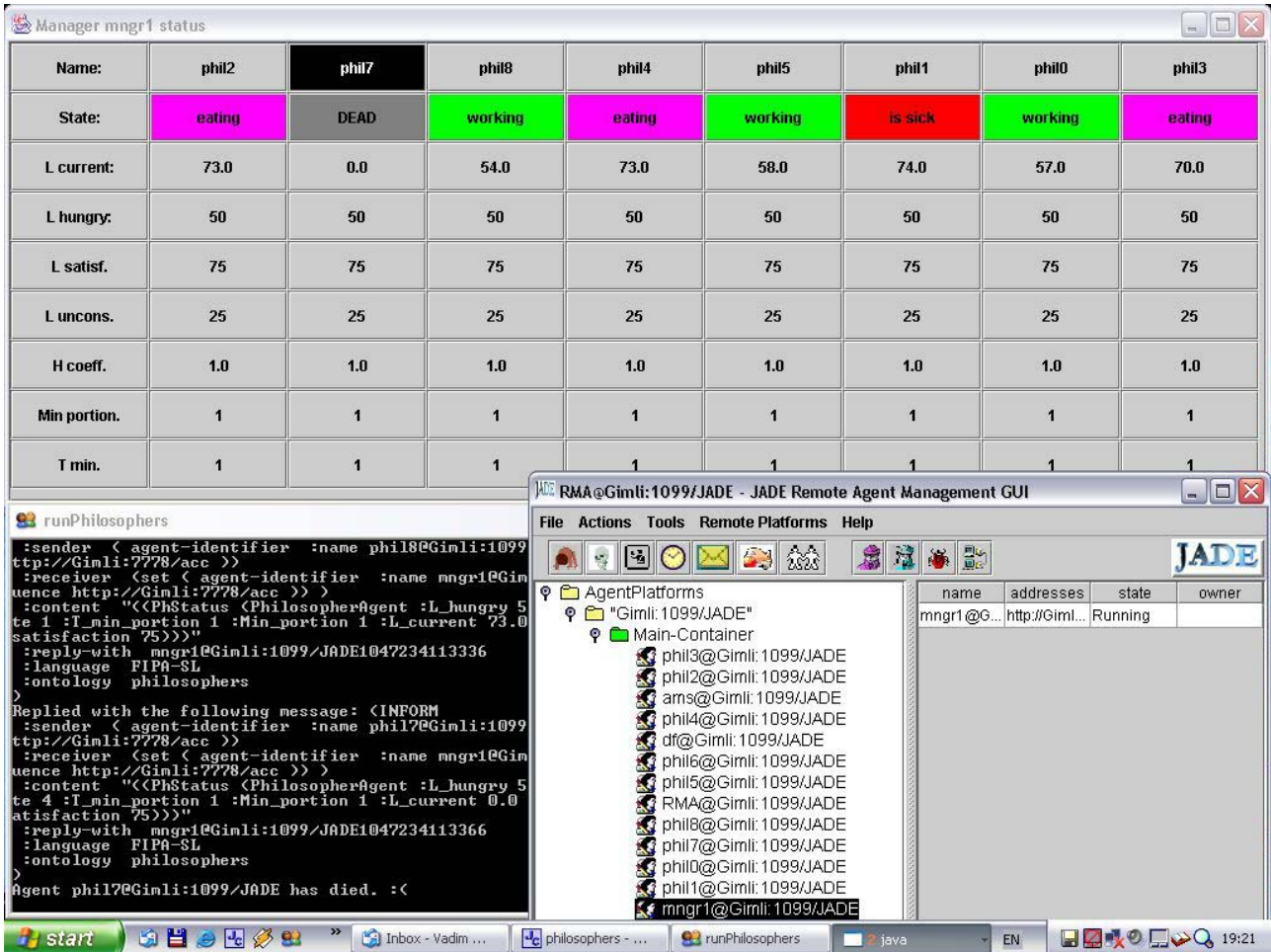
This is done because all the interactions between the Philosopher and other agents are done through the Manager. And for the Philosopher there is no difference who issues the command because all of them are going through manager.

The decision to refuse from 2 types of agents simplifies the process of design and allows obtaining the results about system behaviour more quickly.

## 4 Implementation

### 4.1 System

The source code was written in JAVA [5] using custom developed Ontology [6] which is also a part of the project. JADE environment was used to execute agents and monitor their states. GUI interface of the Managers shows the state of a particular agent with the precision depending from Agent local time period (see section 5.1 for more details). The example is shown on pic.2.



Pic N2 – Screenshot of a working system.

There are 3 windows on the picture:

- Manager status GUI
- JADE console
- JADE Remote Agent Management GUI

While “Manager status GUI” is only informational from JADE console we can collect the output information to the file. JADE RMA GUI is used to control agents, i.e. add/delete, pause/continue, migrate, sniff, and send messages. Parameters to the philosophers are passed through command prompt. If they are not mentioned or their order or number is incorrect agent uses default settings (see section 4.2.2 for details).

## 4.2 Agents

All the agent's functions in this project depend from time linearly though there is no constraints do define the behaviour as a non-linear time function.

### 4.2.1 Manager

Manager agent is really the core of the system. He receives the agent's requests and sends additional messages in order to "learn" the status of each agent in the system. His main characteristics are  $t_{manager}$  (see section 5.1 for method to calculate the maximum value of this parameter) and *Strategy* to implement. Strategy represents the goal for the system. It is the function  $S(a_{i=1..k}^n, n, t)$  which should be maximized (minimized) according to agent parameter(s)  $a_i$  (the number of which can be up to  $k$  – maximum number of parameters), during time  $t$  over  $n$  agents. In our case the function  $S(a_{State=eating}^5, 5, t) \rightarrow \max$ . In other words all our 5 philosophers should be able to work (without dying case) and eat up to their maximum needs.

### 4.2.2 Philosopher

Philosopher is an active element of the system. He "lives" according to his internal time, works and feels sometimes hungry (self-control). There is no measure on the quality of his work since the productivity of philosopher philosophers is of no interest in our current project. It may be done in the future.

Main parameters are represented in table N1.

Table 1

Parameter	Description	Default value
$t_{phil}$	Minimum time interval which philosopher can sense.	1 sec
hunger function	$L_{current} = L_{current} - H_{coefficient} \cdot t_{phil}$	
$L_{current}$	Current level of hungriness (amount of food, energy philosopher has at the time being).	100
$H_{coefficient}$	Hungriness coefficient. Determines how fast the philosopher becomes hungry. $H_{coefficient} > 0$	1
$L_{hungry}$	Limit of hunger. When this level is reached philosopher feels hungry and sends request to eat.	50
$L_{satisfaction}$	Satisfaction level. The minimum level that enables working of a philosopher after eating.	75
$L_{unconsciousness}$	Level after which a philosopher can't take meal by himself. Cure is needed. He submits request to be taken to the hospital	25
$Min_{portion}$	Amount of food which philosopher consumes during $T_{min.portion}$	1
$T_{min.portion}$	Minimum amount of time to get $Min_{portion}$ Measures in philosopher's time units $t_{phil}$	1
<i>State</i>	Current state of the philosopher. Can have one from 4 values: 1-"working", 2-"eating", 3-"sick", 4-"dead"	1 (working)

### 4.2.3 Hospital

Not yet implemented as an agent. His function was divided between Manager and Philosopher by representing “healing action” as a linear function of time:

$$L_{current} = L_{current} + h \cdot t_{phil}$$

where  $h [c^{-1}]$  - healing coefficient.  $h > 0$

### 4.2.4 Table

Not yet implemented as an agent. His function was divided between Manager and Philosopher by representing “eating action” as a linear function of time:

$$L_{current} = L_{current} + e \cdot t_{phil}$$

where  $e [c^{-1}]$  - eating coefficient.  $e > h > 0$

## 4.3 Ontology

The ontology for this project was created and generated using Protégé-2000 [7] development tool with special beangenerator plug-in which creates the structure compatible to JADE.

The ontology consists of 7 predicates and 4 AIDs. They are specially designed for this project only thus can be reused hardly.

## 4.4 Executive Environment

- JADE 2.61 (for the 3.0b small changes and recompilation is needed)
- JRE 1.3.1
- For the processor < PII-300 and RAM<64Mb recompilation with appropriate timing parameters may be needed (see section 5.1)
- OS with GUI (tested on Windows 2000/XP and Linux Mandrake 8.2+KDE)

Since each agent synchronizes its actions with RTC (real-time clock) the whole system is very sensitive to the working environment. Thus the performance on the slowest computer can even be better than on the fastest one depending from which services and programs are running simultaneously. This can be stabilized by using real-time OS (like QNX) together with conceptions of real-time programming in JAVA [8].

## 5 Obtained Results

### 5.1 Time constraints

Before assigning any specific value to the characteristics of a philosopher (like hungriness level, level of satisfaction and so on) the time notion for each agent should be defined because the system is time-sensitive see, for instance, [9]. It should be done in conjunction with the other agents, especially Manager. Manager’s internal time interval value is calculated by formula:

$$t_{\max.manager} < \left( \frac{N_{phil}}{t_{phil}} + \frac{N_{hosp}}{t_{hosp}} + \frac{N_{tbl}}{t_{tbl}} + C \right)^{-1}$$

where  $t_{phil}$  - time interval for the philosopher

$N_{phil}$  - number of philosophers

$t_{hosp}$  - time interval for the hospital

$N_{hosp}$  - number of hospitals

$t_{tbl}$  - time interval for the table

$N_{tbl}$  - number of tables

$C^i [c^{-1}]$  - constant which depends from the executive environment (the speed of the computer, operating system, version of JRE). For fast computers and agents number  $<10$  it can be assigned to “0”.

The above formula is valid in a case we have only 1 manager to control the other agents. If we have at least 2, the task becomes more complicated. We are facing with problems of parallel execution, which demands from us to focus on synchronization between managers.

## 5.2 Homogeneous systems

Homogeneous linear system in our case is the system where all agent instances (all philosophers) have the same linear parameters and start their execution at the same time (from the batch during the system start up)<sup>ii</sup>.

### Theoretical results:

In this type of system when agents start working with the same initial parameters they feel hungry also simultaneously. So, the peak of requests starts at the time  $t_{peak} = \frac{L_{initial} - L_{hungry}}{t_{phil}}$  and if

during  $t_{critical} = \frac{L_{hungry} - L_{unconsciousness}}{t_{phil}}$  agent do not receive invitation to the table he becomes “sick”.

And Hospital agent can restore his ability to work. However there is also a situation when hospital has no free spaces (if we limit the capacity of hospital) to serve the sick philosopher and after

$t_{lifecritical} = \frac{L_{unconsciousness}}{t_{phil}}$  he will die.

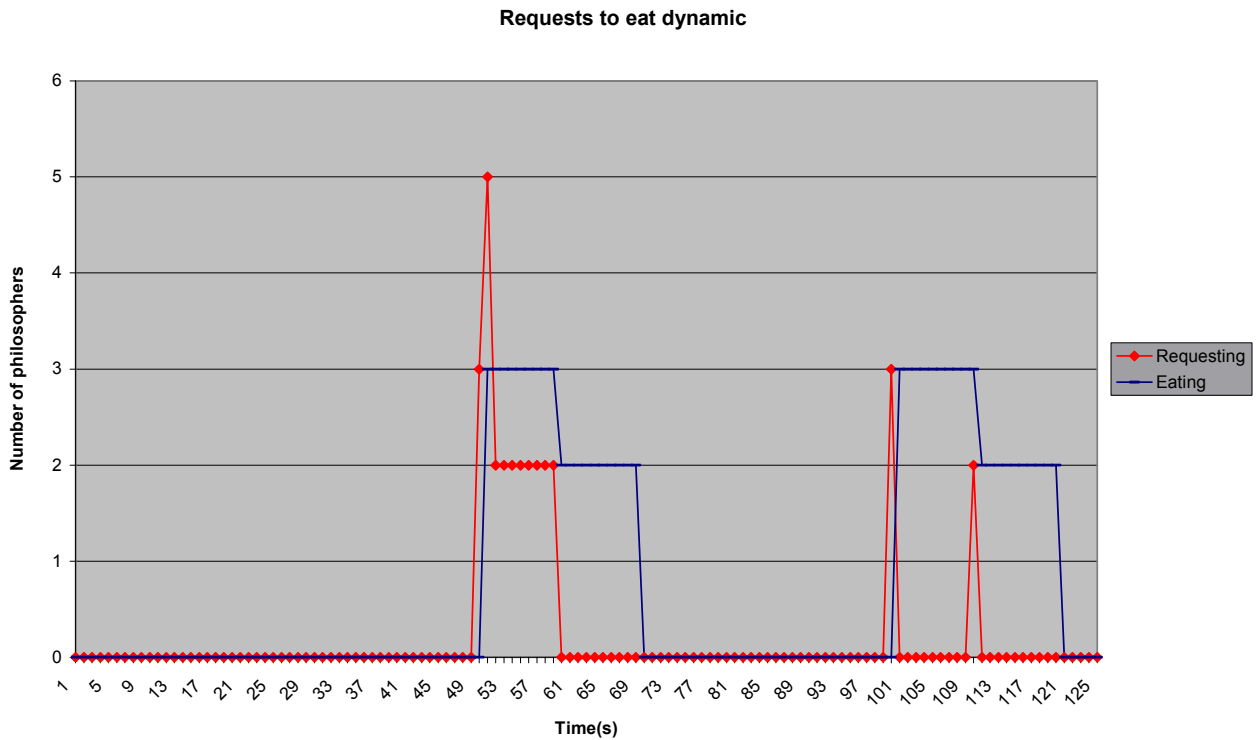
### Practical results:

Practical results with default values are shown on pic. 3.

---

<sup>i</sup> We assume that it is constant to be able calculate Manager maximum time interval easily, but it is certainly not – the execution environment which consists of SW and HW components cannot be constant. Its parameters (like amount of free RAM, swap size, frequency, etc.) change eventually.

<sup>ii</sup> There is no possibility to launch the agents “at the same time”, because JADE launches agents sequentially one after the other. But when  $t_{phil} > t_{batch}$  we can say that agents are executed in the same time. For our task this level of approximation is permissible.



Pic N3 – Requests for eating during the time.

Homogeneous non-linear system in our case is the system where all agent instances (all philosophers) have the same parameters and start their execution at the same time (from the batch during the system start up). At least one parameter of the agent is non-linear.

**Theoretical results:**

**Not yet analysed**

**Practical results:**

**Not yet analysed**

### 5.3 Heterogeneous systems

Heterogeneous linear system in our case is the system where at least 2 agents are different from each other by parameters. All their parameters are linear.

**Theoretical results:**

**Not yet analysed**

**Practical results:**

**Not yet analysed**

Heterogeneous non-linear system in our case is the system where at least 2 agents are different from each other by parameters. At least one parameter of the agent is non-linear.

**Theoretical results:**

**Not yet analysed**

**Practical results:**

**Not yet analysed**

## 6 References

- [1] L.L. Mõtus “Динамика программного обеспечения встроенных систем”, Таллинн, Валгус, 1990
- [2] Holger Knublauch “Extreme Programming of Multi-Agent Systems”, 2002
- [3] K.M. Chandy and J. Misra, Addison-Wesley “Parallel program design . A Foundation”, 1998
- [4] L. Motus and M.G. Rodd, Elsevier/Pergamon “Timing analysis of real-time software”, 1994
- [5] Patrick Naughton, Herbert Schildt “Java™ 2: Complete Reference, Third Edition”, 2001
- [6] Natalya F. Noy, Deborah L. McGuinness ” Ontology Development 101: A Guide to Creating Your First Ontology”, Stanford University, Stanford, CA, 94305
- [7] <http://protege.stanford.edu>
- [8] Greg Bollella “The Real-Time Specification for Java™”
- [9] L. Mõtus “Modeling metric time”

## 7 About the author



Vadim Kimlaychuk received M.S. degrees in Computer Engineering and Information Security from Moscow Institute of Electronic Engineering in 2001. In 2002 started studying Ph.D. in Tallinn Technical University. Area of investigations: intelligent systems (agents). During the first year of education two projects of modelling inter agents communications in JADE are build up.

Current working place is Stoneridge Electronics AB, Sweden. Position: manufacturing verification engineer.

Phone direct: +46 19 673 2594

Fax: +46 19 27 0110

e-mail: [vadim@dcc.ttu.ee](mailto:vadim@dcc.ttu.ee)

[vadim.kimlaychuk@elc.stoneridge.com](mailto:vadim.kimlaychuk@elc.stoneridge.com)