British Journal of Applied Science & Technology
4(12): 1841-1857, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

Analysis of Mobile Agent Optimization Patterns

Faiz Al Shrouf[1*], Aiman Turani[2], Ayman Abu Baker[2] and Ahmad Al Omri[3]

[1]Amman Arab University, Faculty of Computer Science and Informatics, Amman, Jordan.
[2]Applied Science University, Faculty of Engineering, Amman, Jordan.
[3]Applied Science University, Faculty of Information Technology, Amman, Jordan.

Authors' contributions

This work was carried out in collaboration between all authors. Author FAS designed the study, performed the statistical analysis, wrote the protocol and wrote the first draft of the manuscript and managed literature searches. Authors AT, AAB and AAO managed the analyses of the study and literature searches. All authors read and approved the final manuscript.

Original Research Article

Received 24th September 2013
Accepted 3rd February 2014
Published 3rd March 2014

ABSTRACT

Aims: This study extends classification of mobile agent design patterns to involve optimization patterns. We propose two optimization design patterns for mobile agents namely: V-Agent Optimization Pattern and P-Agent Optimization Pattern. The purpose of this paper is to report agents' performance based on mathematical computing model and to support reusability of designs in mobile computing area.
Study Design: This study was developed in collaboration between researchers from computer science department of Amman Arab University, department of computer engineering, department of Network of Applied Science University.
Place and Duration of Study: Samples were implemented in the Department of Computing and Mathematics (Computer Science) and Department of Information System of Amman Arab University for Higher Studies between October 2012 and July 2013.
Methodology: Sample of four master mobile agents that create three slave mobile agents. Master agents are working on set of clients using Aglet alpha release 2.0.5, Tahiti working server and Java Execution Environment (JEE) platform. Slave agents are created by master agents that receive multiple messages from master agents. Master agents send 4000 messages. Consequently, slave agents record message response time in milliseconds. Finally, optimized computed time is computed using the two optimizers: V-Agent Optimizer Design Pattern and P-Agent Optimizer Design Pattern.

**Corresponding author: E-mail: faiz_alshrouf@aau.edu.jo;*

Results: Different sample sizes of two data sets are analyzed: seven master mobile agents vs. four slave mobile agents (number of master agents is larger than slave mobile agents**,** five master mobile agents vs. five slave mobile agents (number of master mobile agents equal to number of slave mobile agents), and four master mobile agents vs. seven slave mobile agents, (number of master mobile agents is less than number of slave mobile agents). Results are based on two main factors: first, disparity of mobile agents in the first data set is computed with variance σ^2 is less than 6 and the second data set, variance is larger than 6 between master mobile agents and slave mobile agents, second, number of messages was still fixed in the two data sets of 4000 messages.
Conclusion: finally, it was reported that the optimized computed time for both data sets using P-Agent Optimizer Pattern is less than V-Agent Optimizer Pattern in all cases of different master mobile agents vs. slave mobile agents. It was concluded in this study that P-Agent Optimizer Design Pattern is more efficient, reusable and scalable than V-Agent Optimizer Design Pattern.

Keywords: Mobile agents; agent design patterns; optimization patterns; V-agent optimizer pattern; P-agent optimizer pattern.

1. INTRODUCTION

Mobile agents are autonomous active objects or software entities, migrating between different network locations, executing tasks locally and continuing their execution at the point where they stopped before migration [1,2]. Mobile agents can also have features like intelligence, pro-activeness, and responsiveness.

Mobile agents can migrate across the network connections representing users in different tasks. They facilitate user's tasks in distributed systems and improve applications such as Internet, Mobile Data Computing, E-Commerce, and many mobile computing areas.

The use of design patterns is an approach to improve the development process of applications [3,4] and to identify the elements of good and reusable designs for mobile agent applications. Several design patterns were developed for mobile agent systems such as representative patterns [1], behavioral patterns [5], architectural patterns [6,7], communication patterns [8], coordination patterns [9] and data migration patterns [10].

The focus of this paper is to develop a new set of agent design patterns for mobile agent message collaboration between set of master agents and slave agents. These set of patterns belongs to a new development class called optimization patterns. In this class, we have proposed two design patterns, V-agent optimizer and P-agent optimizer pattern. Our patterns have an object-oriented flavor; that is we describe them using object and class notations and implemented them in object-oriented languages such as Java.

This paper is structured as follows: in Section 2 presents an overview of patterns; Section 3 describes agent design patterns; Section 4 describes a new performance model in agent – oriented development engineering; Section 5 proposes a mathematical model based on the performance model. Section 6 and section 7 report the new mobile agent optimization patterns; Section 8 presents a discussion and analysis while Section 9 concludes the paper.

2. OVERVIEW OF PATTERNS

Experienced software developers and architects have addressed software problems by creating a body of literature that documents the following types of reusable pattern structures:

- Design patterns: provide a scheme for refining the elements of software system and the relationships between them [11] and describe a common structure of communicating elements that solves a general design problem within a particular context.
- Architectural patterns: express the fundamental overall structural organization of software systems and provide set of predefined subsystems, specify their responsibilities and include guidelines for organizing the relationships between them.
- Pattern language: define a vocabulary for talking about software development problems [12,13] and provide a process for the orderly resolution of these problems.

However, design patterns enhance reuse by capturing and reusing static and dynamic structure and collaboration of components in software designs. They are particularly useful for documenting recurring software architectures, which are abstractions of software components that experienced developers apply to resolve common design and implementation problems. Design patterns also raise the level of discourse in project design and programming activities, which helps improve team productivity and software quality.

Much of research work in agent community [14] has primarily focused on discovering and documenting patterns, to reap the benefits of deploying these proven design solutions, patterns should be used as a first class modeling blocks in designing agent applications.

3. AGENT DESIGN PATTERNS

Design patterns have been very successful in object-oriented system development [15]. As agents provide better abstraction, they allow easier identification of reusable parts in Multi-Agent Systems (MAS) development [16].

Several proposals on design patterns for mobile agent systems have been found in literature, for example [1] discovered useful patterns for mobile agent systems and divided into three classes: Traveling patterns, such as Itinerary, Forwarding and Ticket, patterns are concerned with mobility management of an agent for one or more destinations, Task patterns such as Master-Slave and Plan patterns, are concerned how these tasks are delegated to one or more agents. Interaction patterns, such as Meeting, Locker, Messenger, and Organized group patterns, are concerned with locating agents and facilitating their interactions.

To improve understanding and usage of agent oriented design patterns, it is necessary to classify patterns [17-19] attempts have been made to classify agent patterns and to integrate these patterns in agent-oriented methodologies and development frameworks.

Our work establishes useful patterns for newly developed class called optimization patterns for mobile agent systems. In this paper, we proposed two design patterns called V-agent optimizer pattern and P-agent optimizer pattern. A comparison has been made between these patterns to force like efficient mobility and message coordination between set of master mobile agents and slave mobile agents in distributed and open Multi-Agent System (MAS).

4. PERFORMANCE MODEL

The main benefit of mobile agents in the development of distributed systems is to reduce the network load compared to other paradigms. This is done by moving the mobile agent code to data instead of moving data to the code as it is done by client-server architecture.

Several studies were used in the development of mobile agents performance based on mathematical analysis; for example Petri net models [20] are performance models that produce design patterns in Aglet mobile agent libraries, which are described to support automated software engineering of mobile agent systems. The master-slave design pattern is described for modeling concurrent behavior and verified with Aglet implementation.

In our system master mobile agents are working on clients create slave mobile agents which dispatch to set of servers that perform required task. Master mobile agents have message passing performance model called Message Broadcast Model (MBM) [21], which is described in Fig. 1.

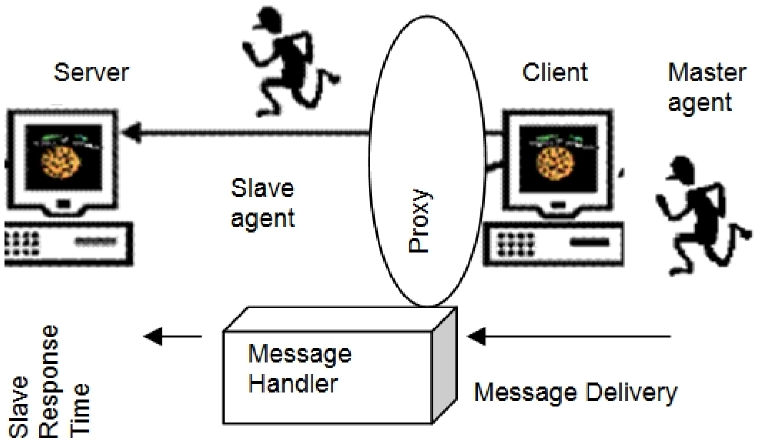


Fig. 1. Message Broadcast Model (MBM)

5. MATHEMATICAL MODEL

Message communication model is the base concept between master mobile agents and slave mobile agents. Message communication model has been described by [22], the principle of this model is used to identify the mathematical model in such each master mobile agent works on client has to deliver set of simple messages to slave mobile agents work on server side, and each slave mobile agent carries out messages based on its response time, for example, a master mobile agent deliver a specific set of equal messages (Message Delivery from master mobile agent1, MD_{m1}) to all slave mobile agents, the first slave mobile agent carries out each message in time given in millisecond (t_{m1s1}), Note that each slave mobile agent carries out a simple message delivered from a master mobile agent according to synchronous communication time. Each slave mobile agent can determine total messages response from all master mobile agents (Message Response from slave Mobile Agent1 is

MR_{s1}). Based on MBM, we proposed a mathematical model called Message Delivery Response Matrix (MDRM), which involves the following components:

5.1 Master Agents

Master agents are created on set of clients, we use Java Execution Environment (JEE) platform consisting of Tahiti server and Java runtime of Aglet development Kit. After the user instantiates interface, it creates slave agents, reads the database for the list of servers, and checks the availability of servers it sends the slave agents to those servers.

5.2 Slave Agents

Slave agents migrate and functioning on servers. They respond to master mobile agents messages, at remote site slave agents mobile receives multiple messages for given tasks and sends results to master agents by embedding it into message handler. Each message registered with the handler which is responsible to record its response time received by the slave agent, and the corresponding slave agent, which sends the message and forward results to the master agent.

5.3 Message Response (MR)

Message response represents total messages that each slave responds in broadcast messaging queue in the message handler.

5.4 Message Delivery (MD)

Message delivery represents total messages that each master agent should deliver in broadcast messaging queue in the message handler.

5.5 Slave Response Time

Slave response time is the estimated time of slave agents carry out a task from a master agent. This time is calculated directly when the slave instantiate the message from the message manager class of the Aglet API. It is necessary here to mention that all messages are carried out in synchronous fashion and the delay time during processing is not encountered. In this paper, this is considered the cornerstone of designing optimization patterns in MAS. Slave agents response time can easily calculated using synchronization scheme.

Based on these assumptions, we consider set of clients C_k, which involves set of master agents M_k, where C_k={C_{1m1},C_{2m2},………, $_{Ckmk}$}, furthermore, consider a set of servers S_n are distributed across network involves a set of created slave agents, S_n ={S_{1s1}, S_{2s2}, …….,S_{nsn}}. Suppose MD_{mi} is the total number of messages to be delivered from the master agent (m_i) to slave agents carrying out tasks on server side, and suppose messages MR_{sj} is the total number of response messages that a slave agent (s_j) responds from all master agents in the message handler in the broadcast queue.

Furthermore, we propose that total messages that to be delivered by all master agents are equal to total response messages in broadcast messaging queue by slave agents as in equation (1), in other words:

$$\sum_{i=1}^{k} MD_{mi} = \sum_{j=1}^{n} MR_{sj} \tag{1}$$

Now, suppose that t_{misj} is the estimated response time of a slave agent (s_j) performing a task on the behalf of a master agent (m_i) and λ_{misj} are number of messages delivered by a master agent (m_i) to a slave agent (s_j). Based on these assumptions, we proposed the Message Delivery/Response Matrix (MDRM). MDRM involves: master agents, slave agents, message delivered by master agents, message response by slave agents, and response time of each slave agent. MDRM is shown in Table 1.

Table 1. Message delivery response matrix (MDRM)

	S_{1s1}	S_{2s2}	. . .	S_{nsn}	MD
C_{1m1}	t_{m1s1} λ_{m1s1}	t_{m1s2} λ_{m1s2}	. . .	t_{m1sn} λ_{m1sn}	MD_{m1}
C_{2m2}	t_{m2s1} λ_{m2s1}	t_{m2s2} λ_{m2s2}	. . .	t_{m2sn} λ_{m2sn}	MD_{m2}
. . .	. . .	. . .	. . .	. . .	. . .
C_{kmk}	t_{mks1} λ_{mks1}	t_{mks2} λ_{mks2}	. . .	t_{mksn} λ_{mksn}	MD_{mk}
MR	MR_{s1}	MR_{s2}	. . .	MR_{sn}	

Assumptions stated above conclude that MDRM follows a linear programming model leading to set of rules:

- Determine the objective function.
- Determine model constraints.
- Determine and validate model non negative values.

The objective function is given by equation (2), constraints given by equation (3) and (4) respectively, and non negativity condition is given by equation (4) as follows:

$$\min(t) = \sum_{i=1}^{k}\sum_{j=1}^{n} \lambda_{misj} t_{ij} \tag{2}$$

$$\sum_{j=1}^{n} \lambda_{misj} = MD_{mi} \quad i = 1,2,.....,k \tag{3}$$

$$\sum_{i=1}^{k} \lambda_{misj} = MR_{sj} \quad j = 1,2,.......,n \tag{4}$$

$$\lambda_{misj} \geq 0 \quad For\ all\ i\ and\ j$$

6. INFORMATION GENERATION

Both design patterns requires set of functional requirements for running, in this example we developed four master mobile agents, three slave mobile agents, and two stationary developed agents called V-agent and P-agent. All master agents require Tahiti server running on port 4434 which create slave agents running on port 6000; while both V-agent and P-agent process computed slave time response and total time response on different port server 7000. Total number of Messages (4000) is generated arbitrary by master agents. Contents of messages involve: displaying a word, search for a name in database, opening a server port number, closing a file, correcting a mistake, connecting to network, etc. Slave agent response time of each slave agent from a given master agent is processed by V-agent and P-agent. Thus both design patterns based on V-agent and P-agent focus on different approaches discussed in the proceedings sections.

7. V-AGENT OPTIMIZER DESIGN PATTERN

Mobile Agent Optimization Patterns are derived from MDRM where both two design patterns are extracted. MDRM supports a flexible mathematical generic model for representing, optimizing, and evaluating mobile agent optimization pattern. We are starting to build the first optimization pattern called V-Agent Optimizer design pattern. The name of the pattern is given according to a stationary master agent, we call it V-agent, and we follow the methodology of analysis given in [1]. This methodology uses the Unified Modeling Language (UML).

7.1 Approach

V-agent approach as given by [21] for mobile agent performance optimization utilizes Vogel's Approximation Method, which gives an initial solution. V-agent follows the following steps:

- Find Penalty Time (PT) between two successive time values in each row and column.
- Choose the larges PT among rows and columns.
- Fill the cell with master message delivery and slave message response.
- Repeat the process to fill all cells.
- Calculate the initial solution using equation (2)

7.2 Intent

V-agent optimizer design pattern defines a scheme whereby a stationary V-master agent distributes messages to slave agents, calculate the penalty time response from slave agents and calculates the optimum performance time in milliseconds (ms) for all scheme representing master mobile agents and slave mobile agents.

7.3 Applicability

This pattern is used in the following cases:

- When master mobile agents require slave agents to carry out tasks in synchronous fashion based on slave response time.

- When slave agents needs to collaborate and coordinate messages.
- When the user needs to differentiate between methods for optimization techniques and study the behavior of mobile agents in MAS.

7.4 Forces

The following are the domain forces that V-agent optimizer design pattern needs to consider:

- Openness: This design has to consider the openness of distributed application in multi-agent systems where master mobile agents coordinate and delegate messages to slave agents based on slave agents response time for carrying out their tasks.
- Mobility: Efficient mobility of agents is a key in this design.

7.5 Motivation

Several key factors are considered as objectives given below:

- Communication: Is the base mechanism for agent's messages collaboration in multi-agent systems. However, current mobile agent systems focus on content of messages, delivery and response time to carry out tasks.
- Performance: Is a main issue in mobile agent systems. The need is to discover several mathematical approaches that explore performance optimization issues.
- Reliability: Is another issue that measure agent's capabilities and their achievement in distributed systems applications. The design is reliable and implemented in network environment.

7.6 Participants

Agents in this design are participating in the pattern by means of object flavor analysis and design. These are described as follows:

- Master mobile agents that create slave mobile agents and specify number of messages to be delivered to slave agents.
- Slave agents that specify two main objectives; the first objective is each slave agent should specify message response time and the second objective specifies the total number of response messages.
- Concrete slave which initializes the process for each slave running and created by each master agent.
- V-agent the main agent in the design that implements two main methods: Doiteration (); which calculates Penalty Time (PT) between two successive slave response times and calculatePerformanceTime(), which computes the total initial time in millisecond. This time represents the optimum time calculated by V-agent using this design.

A class diagram representing the structural relationship of participants in v-agent optimizer is given in Fig. 2.

7.7 Collaborations

Collaboration among agents is shown in agent interaction diagram Fig. 3. This diagram depicts sequences of messages used by agents for sending and receiving agent's behavior.

- Master agent creates slave agents and V-agent.
- Slave agents dispatch to remote host, carry out tasks and receive messages from their master agents.
- V-agent resides on the local host.
- Each slave agent sets out its response time and number of response messages.
- Each master agent sets out messages to be delivered to each slave agent.
- V-agent calculates Penalty Time based on each slave message response time.
- V-agent distributes messages to slave agents based on the least Penalty Time.
- V-agent finalizes the process and computes the optimum performance time.

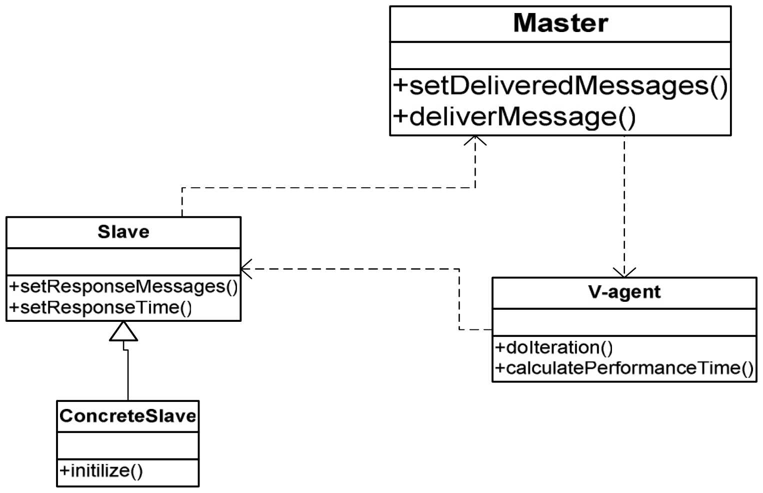


Fig. 2. V-agent design pattern class diagram

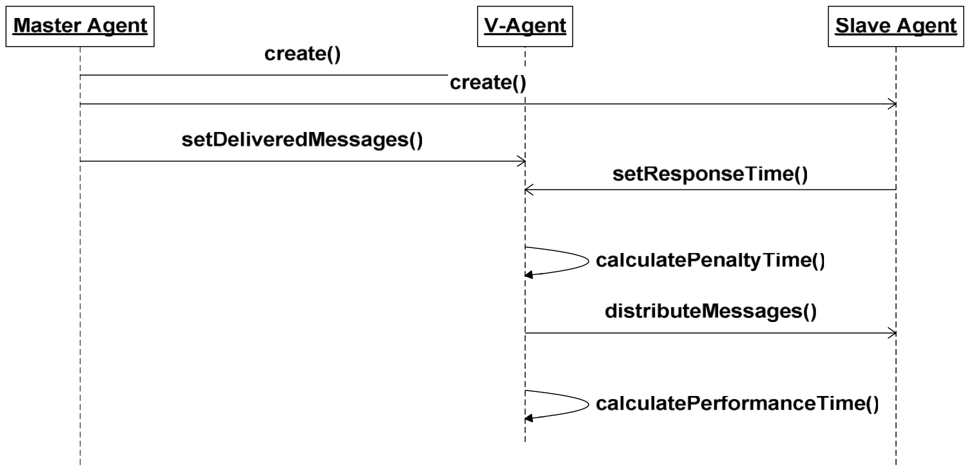


Fig. 3. Collaborations in V-agent optimizer design pattern

7.8 Consequences

V-agent optimizer design pattern provides the following consequences:

- The pattern decouples agents and reuses code amongst agent classes, thus making mobile agent lightweight. This ensures the efficient mobility in case of network latency and network bandwidth and traffic.
- The pattern provides a communication technique amongst agents. This is achieved through synchronous communication messages between different agents based on a response time fashion.
- The pattern enables mathematical approaches to explore agent capabilities which may be specific to agent interaction.

7.9 Implementation

Suppose we define set of four master agents {m1, m2, m3, m4} with set of total message to deliver 4000 messages as {1000, 1000, 1000, 1000} to set of three slave agents {s1, s2, s3} with total message response {1000, 1200, 1800} respectively. The allocated slave message response time of one message in milliseconds is given in Table 2. Note that number of delivered messages by master agents and number of response messages by slave agents is allocated by V-agent, while response message time is allocated by each slave agent in the message handler.

Table 2. V-Agent implementation scenario

	S_1	S_2	S_3	MD
m_1	10 λ_{m1s1}	9 λ_{m1s2}	8 λ_{m1s3}	1000
m_2	8 λ_{m2s1}	6 λ_{m2s2}	11 λ_{m2s3}	1000
m_3	10 λ_{m3s1}	3 λ_{m3s2}	5 λ_{m3s3}	1000
m_4	7 λ_{m4s1}	9 λ_{m4s2}	8 λ_{m4s3}	1000
MR	1000	1200	1800	4000

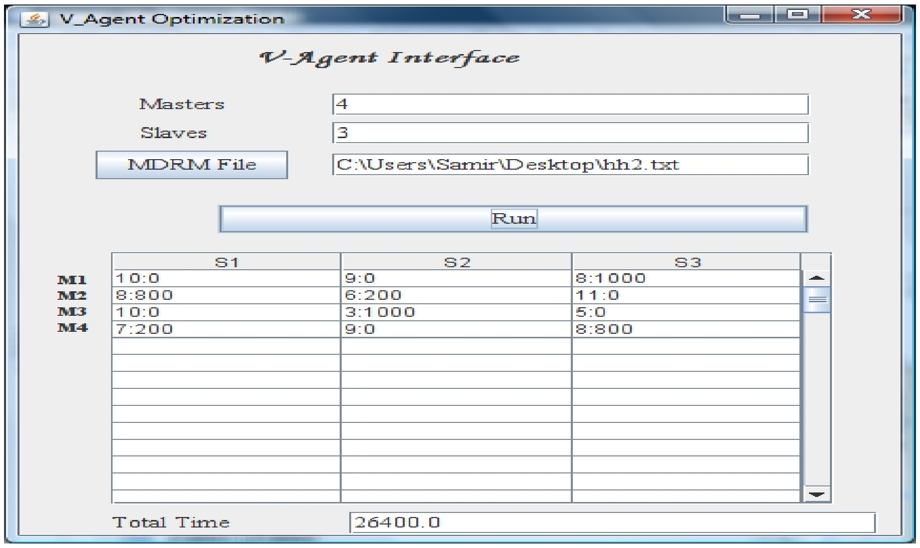
V-agents starts processing by assigning number of messages by calculating values of {λ_{m1s1}, λ_{m2s2}, etc, ……., } amongst rows and columns leading to choose the largest PT (the difference between two successive response time in rows and columns), if more than one PT values are equal, then V-agent choose any PT. Therefore, the first calculated PT is 3 and the second column is selected; hence, V-agent assigns messages between the third master agent and the second slave agent (m3, s2). V-agent allocates 1000 messages from m3 to s2. Now, s2 has 200 messages are waiting in the message handler. Therefore, V-agent completes the first process by assigning 1000 messages from the third master agent to the second slave agent.

V-agent computes the second PT amongst rows and columns. Now, it picks the second column with PT value (3). V-agent chooses the larges time in that column 2, and it optimizes the total messages between (m2, s2). Therefore, V-agent assigns the remaining messages to s2 and has 800 remaining messages in the message handler. V-agent iterates the remaining PT values using the same way until all master messages deliver to all slave agents. Table 3, concludes the final results by V-agent.

Table 3. Performance optimization analysis using V-agent

	S_1		S_2		S_3		MD
m_1		10		9	1000	8	1000
m_2	800	8	200	6		11	1000
m_3		10	1000	3		5	1000
m_4	200	7		9	800	8	1000
MR	1000		1200		1800		4000

Finally, V-agent computes the performance optimum response time using according to equation (1) t= 26,400 milliseconds (ms). The performance optimization screenshot is given in Fig. 4.


V_Agent Optimization

V-Agent Interface

Masters: 4

Slaves: 3

MDRM File: C:\Users\Samir\Desktop\hh2.txt

Run

	S1	S2	S3
M1	10:0	9:0	8:1000
M2	8:800	6:200	11:0
M3	10:0	3:1000	5:0
M4	7:200	9:0	8:800

Total Time: 26400.0

Fig. 4. V-agent design pattern performance optimization

8. P-AGENT OPTIMIZER DESIGN PATTERN

This pattern shares V-agent optimizer pattern in intent, applicability, forces and motivation. In this section we demonstrate the P-agent pattern approach and implementation scenario.

8.1 Approach and Collaborations

This pattern introduces an agent called P-agent, which uses an approach called closed path. P-agent starts the process as given in Table 3, the final distribution of messages given by V-agent. P-agent follows the following steps:

- Compute number of occupied cells in message distribution pattern given by V-agent. We define occupied cells as messages covered from master agents to corresponding slave agents in the final V-agent optimization design pattern of Table 3. Non occupied cells are cells which are not covered in the pattern of Table 3. From master agents to their corresponding slave agents.
- Apply the formula: Occupied-cells=number of master agents in the design plus slave agents minus 1.
- P-agent processes a closed path starting from non occupied cell, gives a positive sign at slave message response time in that cell, follows closed path to the next occupied cell, and gives a negative sign at the occupied cell filled with slave agent message response time.
- Repeat the process up or down to the next occupied slave message response time with positive and negative signs successively. All occupied cells representing slave agent message response time are tracked with successive and negative signs.
- Follow a closed path to return to the initial starting non occupied cell of the non occupied cell of the slave agent message response time.
- Calculate the P-agent value (p-value) by adding slave message response time successively representing positive and negative signs in the closed path.
- P-agent chooses the largest negative p-value to start iterative process and specifies number of messages to be delivered from the corresponding master agent to slave agent.
- P-agent iterates above steps to find the next p-value. The process is repeated until all (p-values) computed by P-agent are positive.

Collaborations between different classes are given in Fig. 5, which represents four main classes: Master Agent, Slave Agent, V-Agent, and P-agent. P-agent has five functions in design pattern: utilizes V-agent results, find closed path, compute (p-values) representing positive and negative slave message response time, distribute master agents messages to their slave agents based on (p-values), and finally, computes the performance optimum time.

8.2 Implementation

In this section, we present an implementation scenario for P-agent optimizer design pattern. P-agent design optimizer pattern utilizes results of final optimization of V-agent design pattern given in Table 3. This table shows the distribution of final number of delivered messages from master agents to their corresponding slave agents. Number of messages is marked as occupied cells given as ($m_1 \to s_3$), ($m_2 \to s_1$), ($m_2 \to s_2$), ($m_3 \to s_2$), ($m_4 \to s_1$), ($m_4 \to s_3$).

P-agent tracks all non occupied cells from master agents to slave agents in the final table of performance optimization of V-agent. The tracking process checks that occupied cells in the final optimization table should be equal to number of master agents plus slave agents minus 1, and computes p-values corresponding to each non occupied cell and follows a closed path starting from initial non occupied cell and adding successive positive and negative signs representing slave messages response time. For example, P-agent tracks the closed path and computes the p-value, which is +3 (+10-8+8-7) for the non occupied cell ($m_1 \rightarrow s_1$) as shown in Table 4.

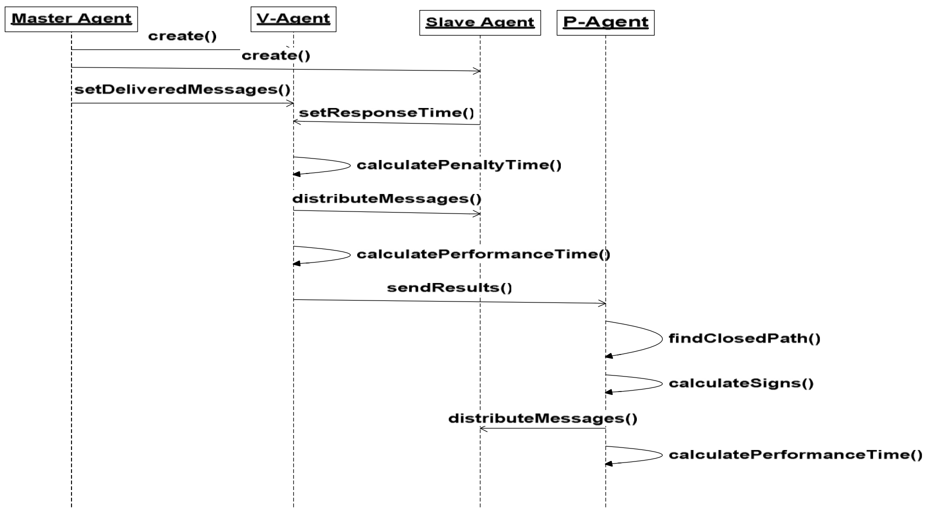


Fig. 5. Collaborations in P-agent optimizer design pattern

Table 4. P-Agent Closed Path Tracking

	S_1	S_2	S_3	MD
m_1	10 +	9	8 1000 -	1000
m_2	8 800	6 200	11	1000
m_3	10	3 1000	5	1000
m_4	7 200 -	9	8 800 +	1000
MR	1000	1200	1800	4000

The remaining non occupied cells are computed as follows:

- ($m_2 \rightarrow s_3$) +11-8+7-8; p-value is +2
- ($m_3 \rightarrow s_1$) +10-3+6-8; p-value is +5
- ($m_3 \rightarrow s_3$) +5-8+7-8+6-3; p-value is -1

- ($m_4 \rightarrow s_2$) +9-6+8-7; p-value is +4

Now, P-agent chooses the largest p-value with negative sign (-1) and ignores all positive p-values and starts the distribution of messages from ($m_3 \rightarrow s_3$). In case of equally p-values, P-agent chooses any picked one. In this case, P-agent assigns 1000 messages from ($m_3 \rightarrow s_3$).

Distribution of messages from master agents through P-agent is an iterative process and P-agent repeats the process to find another p-value in the next process. The final performance optimization scheme is given in Table 5.

Table 5. Performance Optimization Analysis Using P-Agent

	S_1		S_2		S_3		MD
m_1		10	200	9	800	8	1000
m_2		8	1000	6		11	1000
m_3		10		3	1000	5	1000
m_4	1000	7		9		8	1000
MR	1000		1200		1800		4000

P-agent iterates the final performance optimization table given above and terminates the process either all computed p-values are non negatives or number of occupied cells is not equal to master agents plus slave agents minus 1. In this implementation scenario P-agent terminates the process since number of occupied cells is equals 5. The final optimum computed time given by P-agent using equation (1) is t=26,200 milliseconds (ms)

9. RESULTS AND DISCUSSION

To study the behavior of both design patterns, we have analyzed two different data sets. We use the same environment representing (JEE) platform, Aglet Development Kit, and Tahiti server to create master and slave mobile agents. The analysis process based mainly on different factors including number of master agents, number of slave agents, slave response times and their disparity, number of delivery messages of master agents, and number of response messages of slave agents. We also recommend three cases for our study: first; number master agents is larger than number of slave agents, second; both master agents and slave agents are equal, and third; number of master agents is less than number of slave agents. We compute the variance σ^2 for both data sets. The analysis reports that P-agent optimizer design pattern gives better results than V-agent optimizer design pattern for both data sets, and the performance optimization computed time is reduced in case of P-agent optimizer design pattern.

The first data set given in Table 6 concludes that P-agent optimizer design pattern performance optimization is quite small than V-agent design pattern performance optimization, when (the variance σ^2 between slave agents are less than 6), while the second data set as given in Table 7, P-agent performance optimization is much better than V-agent performance optimization, when (the variance σ^2 between master agents are larger than 6).

Table 6. Performance analysis of first data set (Time. Milliseconds)

Master x Slave	σ^2	**P-Agent t=ms**	**V-Agent t=ms**
7m x 4s	4.452	86,400	87,800
5 m x 5s	4.240	49,200	50,800
4m x 7s	2.658	73,500	75,500

Table 7. Performance analysis of second data set (Time. Milliseconds)

Master x Slave	σ^2	**P-Agent t=ms**	**V-Agent t=ms**
7m x 4s	6.158	59,600	66,300
5 m x 5s	6.681	37,300	45,100
4m x 7s	6.811	58,900	65,500

10. CONCLUSIONS AND FUTURE WORK

The use of design patterns has generally increased due to advantages they can bring to applications development, like reuse and better understanding of applications. These advantages can be also obtained when developing mobile-agent based applications by using design patterns. In this sense, several mobile agent patterns have already been proposed. These patterns present solutions that can be reused, avoiding loss of time and effort to investigate problems that have been solved.

In this work, we have proposed performance optimization design patterns for mobile agent systems and we call them V-agent optimizer design pattern and P-agent optimizer design pattern. In general, coordination mechanism is a key and it is very useful for considering these patterns. Also, the functionality of these patterns can be added to explore agent collaboration in multi agent systems. Another advantage of using these patterns is to study the behavior of agents and their features, agent interaction which is supported by mathematical approaches and models that support mobility performance optimization analysis.

A drawback that is related to use of optimization design patterns is that; it is specific to the Aglet development platform and it is insufficient to support intelligence for P-agent and V-agent. These patterns should be implemented in another platform like JADE, which combines migration and intelligence disciplines.

We hope this paper gives the outlines of considering new design patterns and will motivate others to continue and discover more patterns that make it easier for designers of distributed applications to gain and learn the use of agent technology in mobile computing environment.

ACKNOWLEDGEMENTS

We owe thanks to Amman Arab University for Higher Studies (Faculty of Computer Sciences and Informatics) and Applied Science University (Faculty of Computer Engineering and Faculty of Information Technology for supporting this research.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Aridor Y, Lange D. Agent design patterns: Elements of agent application design. Proceedings of 2nd International Conference on Autonomous Agents, ACM press. USA; 1998.
2. Lange D, Oshima M. Programming and deploying java mobile agents with aglets. Addison-Wesley Reading MA; 1998.
3. Ammar H, Yacoub S. Pattern-oriented analysis and design. Person Education Inc.; 2004.
4. Kendall E, Krishna P, Pathak C, Suresh C. Patterns of intelligent and mobile agents. Proceedings of the 2nd International Conference on Autonomous Agents, ACM press New York, USA; 1998.
5. Ojha A, Padhan S, Patra M. Designing Mobile Agents using Helper Pattern. Research India Laboratories; 2007.
6. Schmidt D, Buchman F. Patterns, Frameworks and Middleware: Their synergistic relationships. Proceedings of 25th IEEE International Conference on Software Engineering (ICSE'03); 2003.
7. Kendall E. Role Modeling for Agent System Analysis, Design and Implementation (ASA/MA) ACM press; 1999.
8. Miera N, Silva I. A Set of Agent Patterns for more Expressive Approach. INESC-ID&IST. RUA Alves. 2000;12(9):44-50.
9. Tolksdrof R. Coordination Patterns of Mobile Information Agents. Proceedings of Cooperative Information Agents II, 2nd International Workshop, CIA. Springer; 1998.
10. Osunad S, Atanda F. Analysis of two mobile agent data migration patterns. Journal of Mobile Communication. 2008;2(2):64-72.
11. Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley Reading, MA; 1995.
12. Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl I, Angel S. A pattern language. Oxford University Press, New York, USA; 1977.
13. Weiss M. A Pattern Language for Motivating the use of Agents. Giogini P, et al.: AOIS, LNAI. 2004;142-157.
14. Radizah M, Safaai D, Ammar H. Pattern oriented design for multi agent system: a conceptual model. Journal of Object Technology. 2007;6(4):55-75.
15. Alur D, Crupi J, Malks D. Core J2EE Design patterns: best practices and design strategies. Person Education Inc.; 2006.
16. Tahara Y, Ohsuga A, Honiden S. Agent system development method based on agent patterns. Proceedings of the 21st International Conference on Software Engineering. IEEE Computer Society Press; 1999.
17. Al Shrouf F, Turani A. Agent Business Systems Engineering Development Approach. Journal of European Scientific Research. 2009;29(4):549-556.

18. Malyankar R. A pattern template for intelligent agent systems. Proceedings of the workshop on agent based decision support for managing the internet enabled supply chain. Seattle; 1999.
19. Weiss M. A pattern language for motivating the use of agents. Proceedings of the 6th International Conference on Agent-Oriented Information Systems. Springer; 2004.
20. Rana O, Walker D. A Performance Model for Task and Interaction Patterns in Mobile Agent Systems. Proceedings of IEEE International Conference on Performance Computing and Communication; 2000.
21. Al Shrouf F, Eshtay M, Abuhumidan K. Performance Optimization for Mobile Agent Message Broadcast Model using V-Agent. International Journal of Computer Science and Network Security, IJCSNS. 2008;8(8):285-290.
22. Al Shrouf F, Abusaimeh H, Al Shqeerat K, Al Omari M. Evaluating time performance optimization analysis for mobile agent message communication using assignment computing agent. Journal of Applied Sciences. 2012;12(12):1290-1296.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=445&id=5&aid=3876