

MINIMIZING THE COMPLEXITY EFFECTS TO MAS ARCHITECTURES DESIGN BASED ON FG4COMPLEXITY APPROACH

Howayda Abdallah Ali Elmarzaki and Tawfig M. Abdelaziz
Department of Software Engineering, Benghazi University, Benghazi, Libya

ABSTRACT

The efficiency of multi agent system design mainly depends on the quality of a theoretical architecture of such systems. Therefore, quality issues should be considered at an early stage in the software development. Large systems such as multi agents systems (MAS) require many communications and interactions to accomplish their tasks, and this leads to complexity of architecture design (AD) which have crucial influence on architecture design quality. This work attempts to introduce approach works on increase the architecture design quality of MAS by minimizing the effect of complexity.

KEYWORDS

Multi agent system (MAS), a general architectures, Quality attributes, Recommendations systems (RS).

1. INTRODUCTION

MAS belong to Artificial Intelligence field, the study addressing the approaches of construction of complex systems using a large number of agents, which alter their behavior in order to accommodate with a particular problem [1], [2]. An intelligent agent can be reactive and proactive, [3] due it responses to the actions and alteration which appears in the working environment, can tack the initiative to establish the goals and interacts with other agents [4], [1], [5]. Most literatures indicate that the complexity arises clearly in architecture design of multi agent systems that assigned many and different tasks [6], [7], [8]. The research work introduces an approach to increase the AD quality of MAS by reducing the effect of complexity. The solution mainly presents a set of guidelines including the influential factors on the complexity of AD. These factors are extracted from several sides of AD. Several factors and guidelines are presented to decrease the complexity in architectures of multi agent systems. Each FG is established based on developer's previous practice or experimental methods. The FG is extracted from concepts which related to software architecture and they are presented as symbols used in application phase. For example, depending on FGM1 the hierarchical decomposition approach can be applied on books recommendations system to determine the main components in visual manner to increase the understandability. The modularity has a major role in reducing the complexity in software design since the interaction among agents to accomplish their tasks can lead to system complexity. Thus, this approach increases the architecture design quality of MAS

by minimizing the effect of complexity. The reduction of complexity from AD, eventually reinforces the reusability concept.

2. PROPOSED SOLUTION APPROACH

- The proposed solution is to achieve the desired goals of this research work. It mainly presents a set of guidelines including the influential factors on the complexity of architecture design. These factors are extracted from several sides of AD which should be taken into consideration at the early stages of developing the architecture.
- The sides represent concepts (Abstraction, Modularity and Modeling) which be able applying in both analyses and design phases. Figure1 illustrating the approached concepts in FG4Complexity approach.

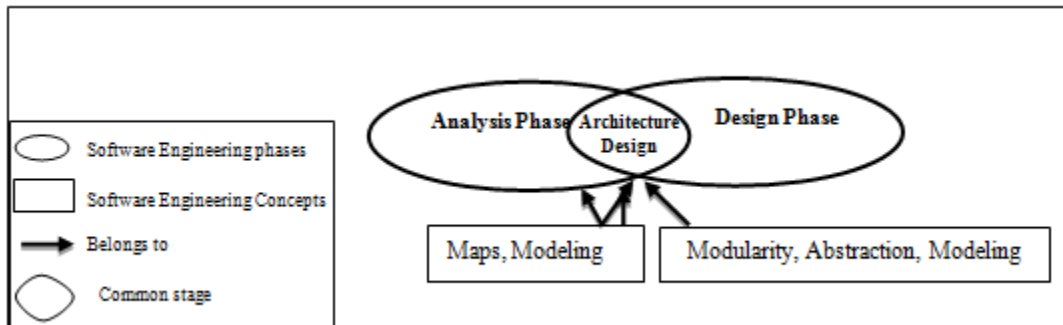


Figure 2: The concepts of analyzing and design which were addressed in FG4 Complexity approach.

- To label the proposed solution approach we suggested that "FG4Complexity". Thereby, "F" liter means Factors, "G" liter means Guidelines, and the "number 4" means for. The next figure shows the proposed approach mechanism.
- The work will be applied via some models used in methodologies related to agents systems such as HLIM[9], MASD [10].

2.1. FACTORS AND GUIDELINES (FG)

In this section several factors and guidelines are presented to decrease the complexity in architectures of multi agent systems. Each FG is established based on developer's previous practice or experimental methods. The FG is extracted from concepts which related to software architecture and they are presented as symbols used in application phase. For example, the FG is related to modeling concept and represented by FGMOD symbol. The FG is related to abstraction concept and represented by FGA symbol and the FG is also related to modularity concept and represented by FGM symbol. Also, each FG should be numbered for example, FGA4 means the factor and guideline number4 in abstraction concept section, FGMOD2 means the factor and guideline number2 in modeling concept section as illustrated in the table below.

Instances	Symbols Interpretation	symbols	Architecture Concept
FGA1....i where I is Integer number	Factors and Guideline of Abstraction	FGA	Abstraction
FGM1....i where I is Integer number	Factors and Guideline of Modularity	FGM	Modularity
FGMOD1....i where I is Integer number	Factors and Guideline of Modeling	FGMOD	Modeling

Table1: The symbols interpretation of architecture concepts

Factors and Guidelines for Abstraction (FGA)

FGA1. Developers should use Simplifying Abstraction type if they want to decrease the dynamic complexity type. [11]

The Clarification:

There are two types of abstraction. The first type is called Simplifying Abstraction (the transition from the middle level to the top level of abstraction), and the second one is generalizing abstraction (the transition from the lowest level to middle level of abstraction). Simplifying Abstraction is the type of abstraction that is used when we want to reduce dynamic complexity and generalizing abstraction is used if we have several components that have many similarities and only differ in some aspects. In fact, this type is very useful if we need to reuse the design. The first type of abstraction is more abstract than the second one. Although, the developers always make a generalizing abstraction before they use Simplifying Abstraction. By this, the parameters and their types are identified before bringing them together to a more abstract design.

There is simple example of Class [12] or software module of library system to clarify the alteration to Simplifying Abstraction as follows.

Suppose we have GUI modules of agent system describe many dialogs for example:

- A is the root dialog which includes a chosen item from the library.
- A1, A2 are both GUI dialogs windows.
- Ag is the window title (String) and linked to the root dialog.
- P is the (parameter) which consists of variable T (Title Name).
- t1, t2 are different titles, for example t1 is "Choose the Book" and t2 is "Choose the Magazine".

By using simplifying abstraction we should abstract the modules of agent system from detailed concept in figure 4.5 part (A) to make it more comprehensible. This means we should apply the following steps.

- Transition from the middle level to the top level of abstraction.
- Low level will be ignored.

- Removing each parameter in middle level (in fig. 3, Part B we should remove the parameter (P) completely by abstracting from Ag to A. This makes the usage of A simpler and less complex than the usage of Ag).
- Adding appropriate name of abstraction to describe what have been removed in fig (3, Part B) we using (Choose item) as appropriate name.

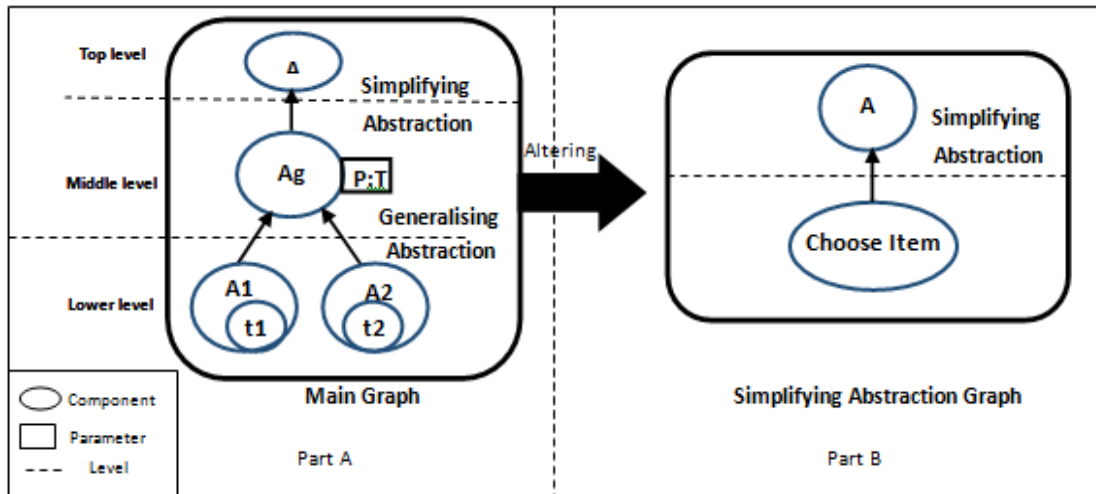


Figure 3: The altering to simplifying abstraction

FGA2. Choosing the appropriate level of abstraction. [13]

The Clarification:

Taking the appropriate level of abstraction is a very important task for developers to increase understanding; thus, decreasing the complexity by using the abstraction levels. In this work, the architecture design will be described based on two levels of abstraction: high level (specification) and detailed level (realization). The figure 4 explains the high level and the detailed level. The first level specifies the main components and its relationships; while, the second level realizes more details than the first one.

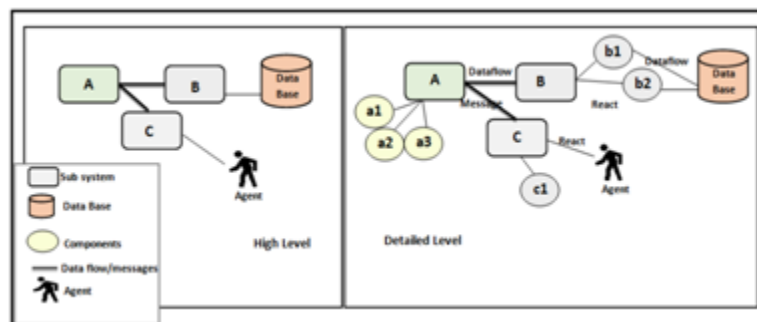


Figure 4: The high and detail levels of abstraction.

FGA3. Avoid to adopting the concept of (gold plating). [14]

The Clarification:

Gold plating is the act of giving the customer more than what he originally asked for. This addition of system functions is reflected on the abstraction task of software system that is undesirable. It is usually performed to make the client happy and pleased; although, it makes the architecture design more have complex components.

Factors and Guidelines for Modularity (FGM)

FGM1. Using Hierarchical Decomposition Approach (HDA) which considers a major method of handling complexity in conventional software analysis and design.[6], [15], [16]

The Clarification:

HDA involves the top-down design which starts by defining the top level components. This design contains the main components. After this, sub components are defined in the lower-level. This decomposition in each level is effective for controlling complexity (if it enforces information hiding) by demanding lower level components as explained in the next example [6].

Example: The example illustrates how using HDA to design particular software of digital clock as the figure 5 shows.

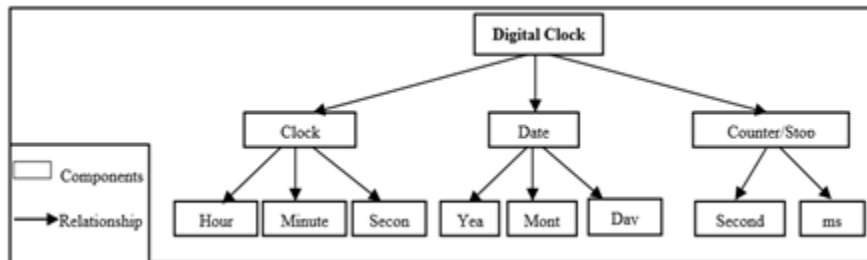


Figure 5: Illustrating the Hierarchical Decomposition Approach

FGM2. It is useful to establish the software modularity based on roles or measurements such as Cohesion Communication Measurement (CCM). [17]

The Clarification:

It is crucial to realize that the complexity of any system stems from a large number of system components and interaction required between these components. This is brought out clearly in large and complex system as MAS. If this is the case, then, the modularity rules needs to be taken into consideration the crucial issue of a complex system designs. This complex design is comprised of multiple agents and interactions. In this sense, the modularity concept could be decomposed in components and again the components into sub-components and so on, till some basic entities are obtained. The measurment of communication cohesion introduces approximate ratio to internal interactions on external interactions for each agent. After applying CCM, the

observed results if $CCM \geq 0.91$ of the Agent, then it will be targeted for further decomposition. Hence, FGM2 is based on measurement principle during AD phase. According to this measurement decomposition produces independent results. Figure 6 illustrates CCM mechanism, and table 2, demonstrates more decomposition.

CCM Mechanisem	
$CCM (A_i) = \frac{R_{internal}}{R_{internal} + R_{External}}$	
Abbreviation	Illustration
CCM	Communicative Cohesion Metrics defined in terms of the ratio of internal relationships (interactions) to the total number of relationships.
(Ai)	(A)Agent, where i=1 to N. Example: Agent1, Agent2 and so on.
R internal	Internal interaction
R external	External interaction

Table 2: The abbreviations of CCM metric Approach (HDA)

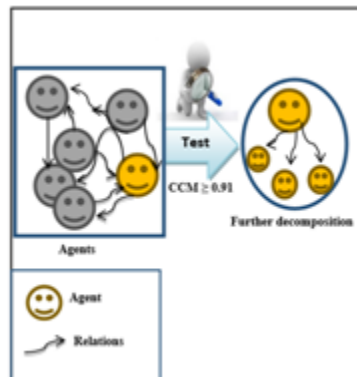


Figure 6: The agent targeted to further decomposition (HDA)

Factors and Guidelines for Modeling (FGMOD)

FGMOD1. Using Use Case Maps (UCM) to clarify the most relevant, interesting, and critical tasks of MAS system. [18]

The Clarification:

UCM act as a bridge between requirements analysis and design phases. It provide a behavior structure for evaluating architecture decisions at a high level of design. In this context, these maps can become applicable on AD at the same stage (After requirements analyses and before design).

It can also be used to emphasize the tasks (Responsibilities) of MAS along paths among components and clarify the interaction. There are many notations using in UCM. The following example illustrates the usage of UCM method through focus on some notations such as: Task, component, path of scenario, (start and end points of scenario), and the interactions among components.

Example

The example describes a simple UCM where a user (Nancy) attempts to make a phone call with another user (Jack) through a network of agents. Each user has an agent responsible for managing subscribed telephony features. Nancy first sends a connection request (req) to the network through her agent. This request causes the called agent to verify (vrfy) whether the called outcome is idle or busy (conditions are italicized). If he is, then there will be some status update (upd) and a ring signal will be activated on Jack’s side (ring). Else, a message stating that Jack is not available will be prepared (mj) and sent back to Nancy (msg). A scenario starts with a pre-condition (filled circle labeled req) and ends with one or more resulting events and/or post-conditions (bars), in our situation ring or msg.

The responsibilities (vrfy, upd, mj) have been activated along the way. In this example, the responsibilities are allocated to abstract components (boxes Nancy, AgentA, Jack and AgentB), which could be realized as objects, processes, agents, databases, even roles, actors, or persons.

The structure of a UCM can be formed in different ways (views). For example, one may start by identifying the responsibilities (Figure 7 (a)). They can then be allocated to scenarios (Figure 7 (b)) or to components (Figure 7 (c)). Eventually, the views are merged to form a finishing map (Figure 7 (d)).

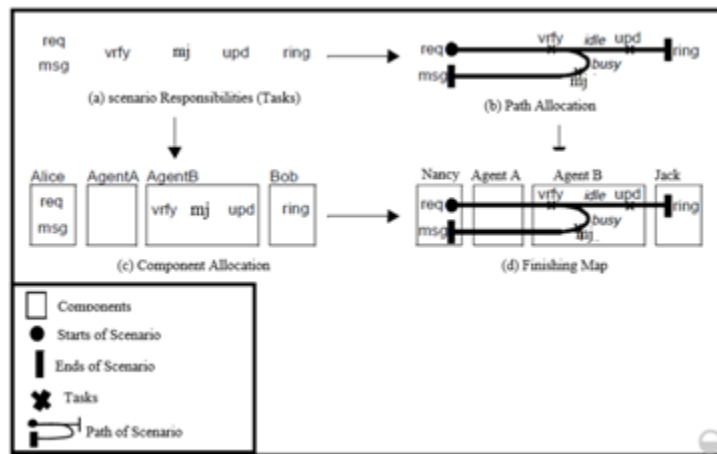


Figure 7: The Use Case Map construction (HDA)

FGMOD2. Using simple notations is very important to enhance understandability and decrease complexities in AD such as arrows, components, domains...etc. [19]

The Clarification:

According to some available literatures, there are a lot of various notations used to describe the AD of software systems. Some of these notations are simple and intuitive while others need to be

understood. To model the software architecture, we need to capture some aspects such as components, interactions, and context then model them. In the context of avoiding the complexities that arise from misunderstanding we suggest some simple notations are proposed and used to describe the architecture as shown in table 3.











Description	Notations
Bold arrows to represent the messages among agents through the interactions.	
Normal arrow to represent the dataflow	
Dotty arrows to represent the messages which are exchanged from extra system such as the black board system.	
Doubly directions arrows represent the dataflow if it is the same exchanged between two components.	
Dotty rectangles to represent the domains.	
Distinguish component to represent Agent.	
Distinguish component to represent list.	
Distinguish component to represent many lists.	
Distinguish component to represent data base storage.	
Distinguish component to represent data base resources.	

Table 3: The proposed notations

3. CASE STUDY APPLICATION STEPS AND DISPLAY THE RESULTS

The case study is a "books recommendations system" based on MAS to help users select books. The system can switch to three recommendation approaches Content-based filtering approach (CBF) [20], [21] Collaborative Filtering approach (CF) [22], [23] and knowledge based approach (KBA). [24], [25] The agents within the system can exchange the messages among each other via one of agent communication languages. In this case study, the messages exchanged will be via Knowledge Query and Manipulation Language (KQML). The work will be applied via some models used in methodologies related to agents systems such as HLIM[9], MASD [10]

3.1. AGENTS AND THEIR TASKS

A brief summary of agents and their tasks in the next table:

Agents	Roles (Tasks)
Profiling agent	• Gathering the user's preferences, gathering the relevance feedback, and building and updating the active user profile
NDA	Gathering the user current needs
Filtering agent	• Producing the recommendations, removing the books that are not currently offered from the recommendation list, and transferring the recommendation to the GUI
Retrieval agent	• Retrieving the books that are currently offered from the books database and storing the available books in the recommender system database
Translation agent	• Producing books translation service for users

Table 4: The agents and their tasks

3.2. CONCEPTUAL OVERVIEW OF BOOKS RECOMMENDATIONS SYSTEM ARCHITECTURE DESIGN.

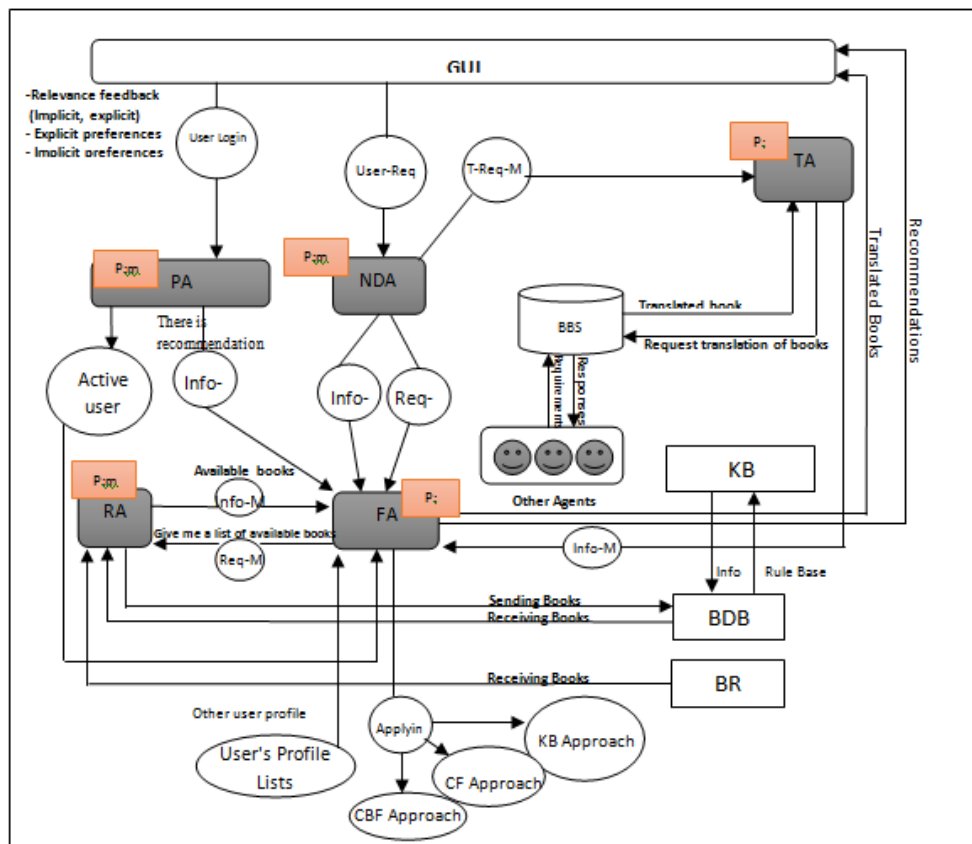


Figure8: Conceptual overview of books recommendations system

3.3. THE FG4 COMPLEXITY APPROACH APPLICATION STRATEGY

As we have earlier pointed out that all the previous FG will be within 4 steps to correspond to the current case study as the next figure shows:

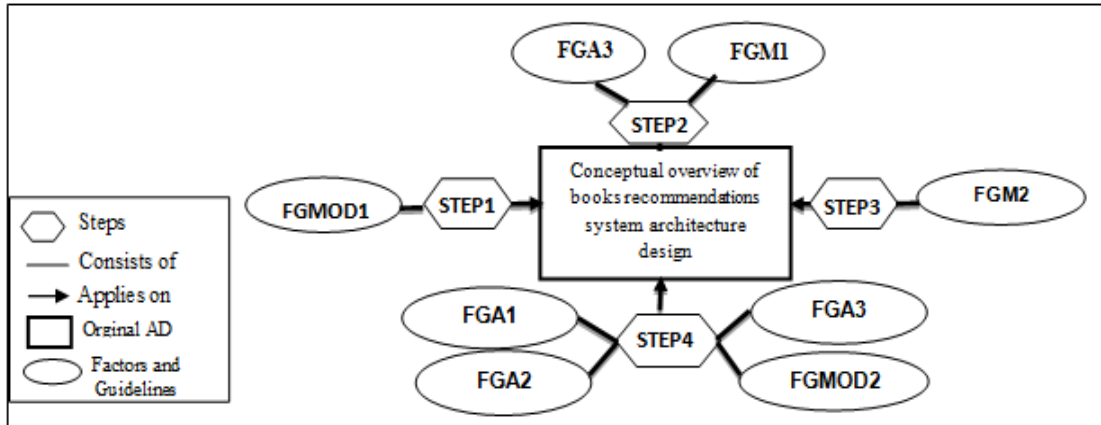


Figure9: Illustrating of the applied steps on AD

Step1. Initially, this step is based on applying UCM represented in FGMOD1 of FG4Complexity approach which used in between analysis and design phases. These maps give high view of system specifically the responsibilities (Tasks) and interactions in a simple way, reinforce system understanding and overcome some situations of complexity such as intercommunication among agents. The following figure illustrate example to use the use case maps in analysing agents, tasks, scenarios and the most significant interactions among agents in books recommendations system. [26], [27]

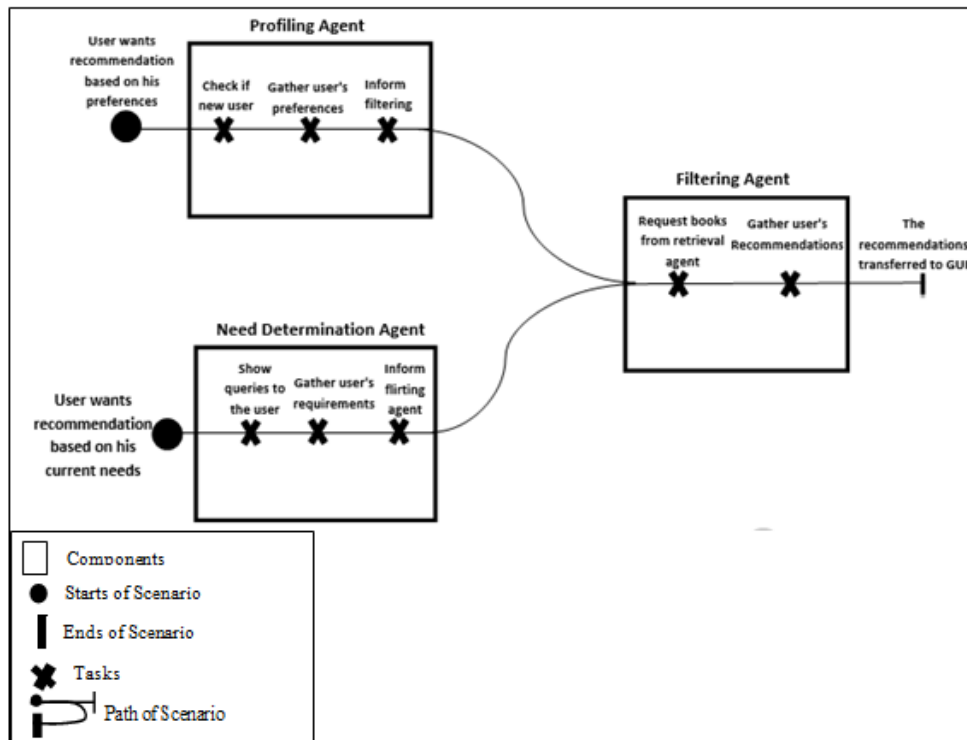


Figure 10: The UCM of translating book mechanism.

Step2. If the system requirement specifications (SRS) [28] of a system do not have a translation function; then, this function is considered as Gold Plating concept; therefore, we should apply the FGA3 which avoid the part of gold plating represented in translation agent (TA) and all components connected from AD as illustrated in the figure below.

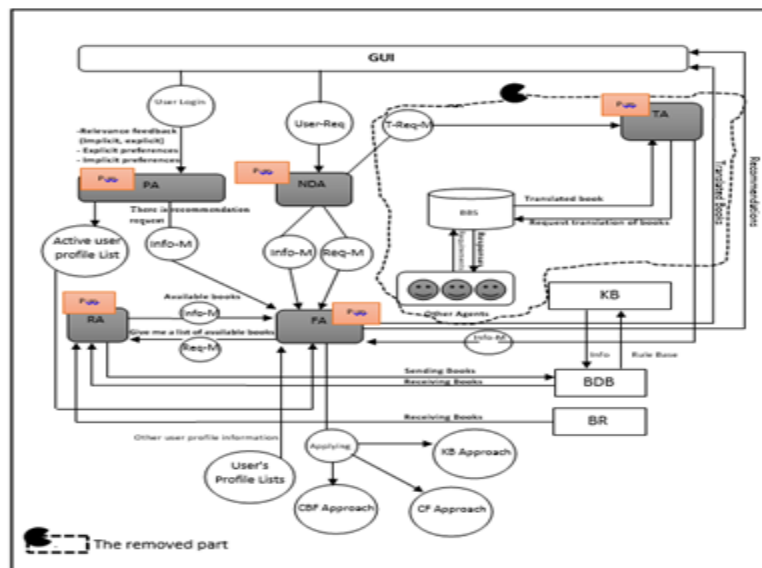


Figure 11: Omitting the part representing the gold plating

Depending on FGM1 the hierarchical decomposition approach (HDA) could be applied on books recommendation system to demonstrate the main components in visual manner to increase the understandability. Next table shows the main components and their connected components in books recommendations system.

Main Components	Connected component(1)	Connected component(2)	Connected component(3)
Retrieval Agent	Book Data Base	Filtering Agent	Book Resource
Filtering Agent	Knowledge Base	GUI	Retrieval Agent
Profiling Agent	GUI	-	-
Need determination Agent	GUI	-	-
Book Data Base	Retrieval Agent	-	-
Book Resource	Retrieval Agent	-	-
Knowledge Base	Filtering Agent	-	-
GUI	Profiling Agent	NDA	Filtering Agent

Table 5: The main components and their connected components in books recommendations system

Next figure demonstrates the majeure components in case study by applying HDA.

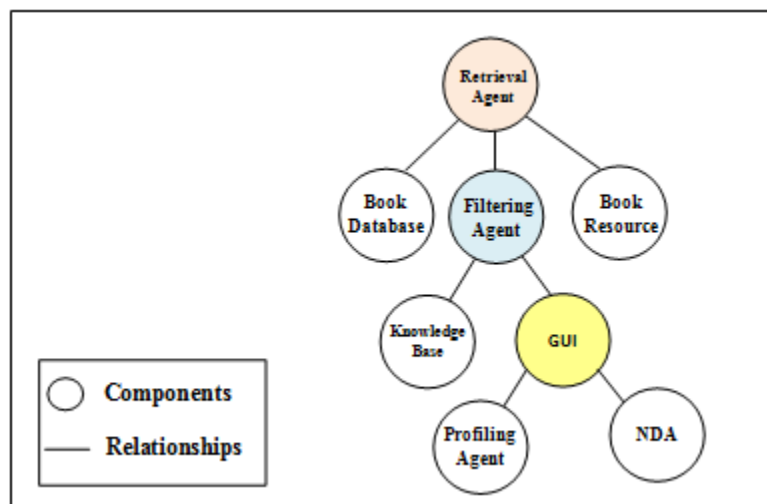


Figure 12: Conceptual system after applying HDA

Step3. As we have pointed out, the modularity has a major role in decreasing the complexity in software design since the interaction among agents to accomplish their tasks can lead to system

complexity. This step totally relies on cohesion measurement principle which uses the Communication Cohesion Measurement (CCM). This measurement works as a testing tool. This enables us to discover which agent needs more decompositions. In this research work, we have four agents described in the case study: filtering agent, profiling agent, need determination agent, and retrieval agent in respect that the translation agent has been omitted in the last step. The formulation of communication cohesive measurement is .The next illustration shows how.

Based on the architecture design of book recommendation system, the filtering agent has 4 internal relationships and 2 external relationships, profiling agent has just one internal relationship and 4 external relationships, need determination agent has one internal relationship and 2 external relationships and retrieval agent has 4 internal relationships and 3 external relationships as shown in the following:

$$CCM(A_i) = \frac{R_{\text{internal}}}{R_{\text{internal}} + R_{\text{External}}}$$

Profiling agent		Filtering agent		NDA		Retrieval agent	
R internal	1	R internal	7	R internal	1	R internal	3
R external	2	R external	4	R external	3	R external	2
CCM(PA)	1/3	CCM(FA)	7/11	CCM(NDA)	1/4	CCM(RA)	3/5
Assessment		Assessment		Assessment		Assessment	
CCM(PA) = 0.3		CCM(FA) = 0.6		CCM(NDA) = 0.3		CCM(RA) = 0.6	

Table 6: The calculating by using CCM technique

So, the results are: CCM (FA) < 0.91, CCM (NDA) <0.91, CCM (RA) <0.91, and CCM (PA) <0.91. It is worth noticing that all results less than 0.91 by this, they do not need more decomposition.

Step4. Applying a group of FG on the architecture design. This group consist of FGA1, FGA2, FGA3 and FGMOD2 which influence the architecture directly and the changes can clearly be observed. Next figures show the architectural design after applied **FG4Complexity** approach.

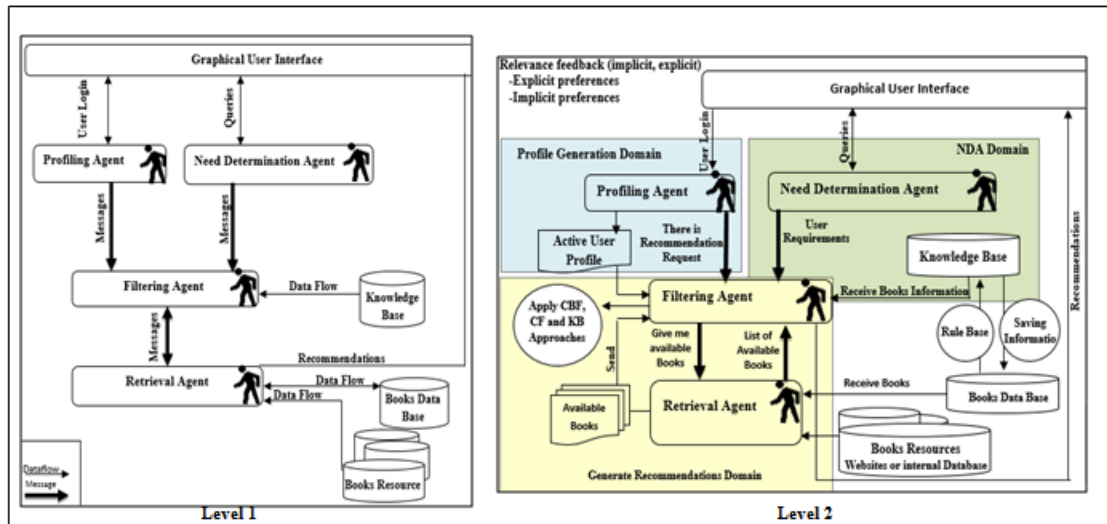


Figure 13: the architectural design after applied **FG4Complexity** approach

4. CONCLUSION

The Research work approached the complexity of architectures design (AD) in systems based on multi agents (MAS) by a proposed solution method represented in a set of guidelines. These guidelines were introduced by extracting the factors affecting the complexity from three major sides of AD represented in abstraction, modularity and modeling thus, the approach labeled as "FG4complexity". It discussed the decrease of coupling which usually occurs during the interactions among agents and supporting the understandability of MAS architectures. The FG4complexity approach is useful for large systems such as recommendation systems that are based on MAS to avoid the complexity problems found in the most existing architectures. Thus, it enhances the quality standards, the reduction of complexity from (AD), and eventually reinforces the reusability concept.

FUTURE WORK

For future work, other aspects of architecture design will be addressed to attempt to make the proposed approach more effective. Those aspects may be are represented in the style, design patterns, documentation and so on. ALSO, we hope to apply the FG4complexity approach on other larger and more complex systems.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank ALLAH, without ALLAH this work would never have been finished. I would like to express my sincere thanks to my supervisor Dr. Twfig Eltwel for his invaluable guidance and advice. I would like to thank my beloved husband Fathi El faitouri for his unlimited and faithful support as well as his patience and unconditional love. Also, The last but not least, I am profoundly grateful to my kind friend Asya Sohaim for her fruitful collaboration and advice. Finally, I thank everyone who encouraged me

REFERENCES

- [1] I. a. S. Markic, Maja and Maras, Josip, "Intelligent Multi Agent Systems for Decision Support in Insurance Industry," in Information and Communication Technology, Electronics and Microelectronics (MIPRO), ed: IEEE, 2014, pp. 1118--1123.
- [2] M. Oprea, "Applications of multi-agent systems," in Information Technology, ed: Springer, 2004, pp. 239-270.
- [3] Z. S. Ahmed Taki, "Formal Specification of Multi-Agent System Architecture," presented at the International Conference on Advanced Aspects of Software Engineering, ICAASE,, 2014.
- [4] K. O. Chin, K. S. Gan, R. Alfred, P. Anthony, and D. Lukose, "Agent Architecture: An Overviews," Transactions on science and technology, vol. 1, pp. 18-35, 2014.
- [5] S. A. D. Mark F. Wood, "An Overview of the Multiagent Systems Engineering Methodology," Springer, vol. 1957, 2001.
- [6] B. H. Far, "Software Agents: Quality, Complexity and Uncertainty Issues," IEEE, 2002.
- [7] B. R. Sinha, P. P. Dey, M. Amin, and H. Badkoobehi, "Software complexity measurement using multiple criteria," Journal of Computing Sciences in Colleges, vol. 28, pp. 155-162, 2013.
- [8] D. N. M. Ghazal Keshavarz, Dr. Mirmohsen Pedram "Metric for Early Measurement of Software Complexity," International Journal on Computer Science and Engineering (IJCSE) vol. 3, 2011.
- [9] M. Elammari and W. Lalonde, "An agent-oriented methodology: High-level and intermediate models," in Proc. of the 1st Int. Workshop. on Agent-Oriented Information Systems, 1999, pp. 1-16.
- [10] M. E. T. Abdelaziz1, R. Unland3, C. Branki4, "MASD: Multi-Agent Systems Development Methodology," Multiagent and Grid Systems Journal,, 2010.
- [11] S. a. D. Wagner, Florian, "Abstractness, Specificity, and Complexity in Software Design," ACM, pp. 35--42, 2011.
- [12] A. A. A. a. I. Bouchrika, "From UML 2.0 Interaction Fragments to PROMELA using a Graph Transformation Approach," The International Arab Conference on Information Technology (ACIT'2013), 2013.
- [13] F. Tsui, A. Gharaat, S. Duggins, and E. Jung, "Measuring Levels of Abstraction in Software Development," in SEKE, 2011, pp. 466-469.
- [14] B. J. Kirandeep Kaur, Rekha Rani, "Analysis of Gold Plating: A Software Development Risk," International Journal of Computer Science and Communication Engineering, vol. 2, 2013.
- [15] F. Medeiro, B. Pérez-Verdú, and A. Rodríguez-Vázquez, Top-down design of high-performance sigma-delta modulators vol. 480: Springer Science & Business Media, 2013.
- [16] H. a. v. V. De Bruin, Hans, "Quality-driven software architecture composition," Journal of Systems and Software, Elsevier, vol. 66, pp. 269--284, 2003.
- [17] S. Misra, "An approach for the empirical validation of software complexity measures," Acta Polytechnica Hungarica, vol. 8, pp. 141-160, 2011.

- [18] A. Lawgali, "TRACEABILITY OF UNIFIED MODELING LANGUAGE DIAGRAMS FROM USE CASE MAPS," International Journal of Software Engineering & Applications (IJSEA), 2017.
- [19] A. Zalewski, "Modelling and evaluation of software architectures," Prace Naukowe Politechniki Warszawskiej. Elektronika, 2013.
- [20] B. L. M. Montaner, and J. De La, "A Taxonomy of Recommender Agents on the Internet," Artificial Intelligence Review, vol. 19, 2003.
- [21] H. Castillo, "Hybrid Content-Based Collaborative-Filtering Music Recommendations," Department of Computer Science, Information System Engineering (ISE), Netherlands, 2007.
- [22] J. Itmazi, "Flexible Learning Management System To Support Learning In The Traditional And Open Universities," PhD Thesis , university of Granada, 2005.
- [23] J. Obando, "Methodology to obtain the user's Human Values Scale from Smart User Models," PhD Thesis, Department of Electronics, Computer Science and Automatic Control, University of Girona, 2008.
- [24] R. Burke, "Hybrid Recommender Systems:Survey and Experiments," User Modeling and User-Adapted Interaction, vol. 12, 2002.
- [25] T. T. a. R. Cohen, "Hybrid Recommender Systems for Electronic Commerce," in the 17th National Conference on Artificial Intelligence AAAI, 2000.
- [26] R. N. A. A. E. frerjani, "Towards A General Architecture for Building Intelligent, Flexible, and Adaptable Recommender System Based on MAS Technology," post graduation, compluter science, benghazi IEEE journal, 2010.
- [27] E. M. Saleh, "Architecture for Design Pattern Selection based on Multi-Agent System," post graduation, benghazi university, 2014.
- [28] P. Thitisathienkul and N. Prompoon, "Quality assessment method for software requirements specifications based on document characteristics and its structure," in Trustworthy Systems and Their Applications (TSA), 2015 Second International Conference on, 2015, pp. 51-60.

AUTHORS

M.SC. HOWAYDA ABDALLAH ALI ELMARZAKI Faculty of information technology, Benghazi University, Libya. Software Engineering Department Born in 03, October 1980 and Married with 1 child.



Dr. Tawfig Mohammed Abdelaziz Faculty of information technology, Benghazi university, libya. Software Engineering Department Born in 03, October 1965 and Married with 6 children.

