5-2023

A Multi-Agent Architecture for An Intelligent Web-Based Educational System

Taddese Erba
St. Cloud State University

A Multi-Agent Architecture for An Intelligent Web-Based Educational System

By

Taddese Erba

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

In Computer Science

May, 2023

Abstract

An intelligent educational system must constitute an adaptive system built on multi-agent system architecture. The multi-agent architecture component provides self-organization, self-direction, and other control functionalities that are crucially important for an educational system. On the other hand, the adaptiveness of the system is necessary to provide customization, diversification, and interactional functionalities. Therefore, an educational system architecture that integrates multi-agent functionality [50] with adaptiveness can offer the learner the required independent learning experience.

An educational system architecture is a complex structure with an intricate hierarchal organization where the functional components of the system undergo sophisticated and unpredictable internal interactions to perform its function. Hence, the system architecture must constitute adaptive and autonomous agents differentiated according to their functions, called multi-agent systems (MASs).

The research paper proposes an adaptive hierarchal multi-agent educational system (AHMAES) [51] as an alternative to the traditional education delivery method. The document explains the various architectural characteristics of an adaptive multi-agent educational system and critically analyzes the system's factors for software quality attributes.

Acknowledgment

My special heartfelt thanks go to my academic advisor, Professor Ramnath Sarnath. Thank you for not giving up on me quickly. A warm thank you go to my committee members, professors Andrew A Anda, Maninder Singh, and Instructional Technology Specialist Mr. Greg Jorgensen, who served as committee members on the Starred paper. All other individuals in the Computer Science & Information Technology department who tirelessly assisted me in giving me the golden opportunity to work on this research topic, "*A Multi-Agent Architecture For An Intelligent Web-Based Educational System."*

I also want to express my special gratitude to my sisters Aster Daba Erba, Zewuditu Daba Erba, Adanech Daba Erba, and Gilenesh Tadesse Gemechu, who encouraged me and stood by my side throughout my academic career.

Table of Contents

List of Figures

Figure Page

Figure Page

Figure Page

List of Tables

Major Sections of the Research Paper

As stated in the introduction section of this paper, the body of the research paper contains nine chapters. Each chapter will explore a distinct aspect of the proposed architecture. The presentation of the chapters follows the logical order of information flow. The first Chapter is the introduction where the background of the paper establishes the background of the research followed by the thesis statement. Next, the section introduces the proposed architecture's distinctive characteristics that serve as a foundation for the adaptive multi-agent educational system. Unit 2 offers a detailed discussion of multi-agent architecture. Finally, the subsequent chapters will cover the remaining part of the section, as outlined in the introduction section of the research paper.

Chapter 1: Introduction

1.1. Background

Modern education's form and contents have undergone an immense evolutionary transformation. In an online article, Paterson states that education evolved from an arbitrary model where no curriculum was necessary to a standard contemporary form with a centralized and highly structured mode [1], [96]. Furthermore, scholars in education agree that the progressive change in the shape and content of modern education made its delivery much more complex than ever [2]. The noticeable progressive change in the structure of contemporary education and the ever-increasing complexity of its contents suggests that the delivery methods and the tools used to convey instruction need transformation and modernization [4].

However, transforming education from the traditional and centralized instructor-led curriculum to an adaptive and personalized computer-based curriculum [43] has never received sufficient attention. Moreover, the transformation and technologization of education remained static, and the larger educational spectrum was left alone captive to the traditional educational system [3], [39]. The absence of a detailed design and architectural framework of an adaptive educational system that explains the system's full functionalities is the primary reason our educational system remains static, untransformed, and archaic.

Therefore, it is necessary to present an alternative educational system that can meet current educational trends, including Nano learning [36], [38], adaptive learning using artificial intelligence, augmented learning, and personalized education [42], [43]. This paper presents an adaptive multi-agent educational system architecture [51] [53] with the following functionalities: intelligence, customizability, self-organization, interaction, dynamic and continuous improvement, and content filtering [4]. These functionalities can solve the insidious problem that has prevailed in contemporary education for over a century and accommodate the current academic demands. In other word, an excellent educational system is a standalone and self-sufficient system that can effectively deliver education and monitor the learner's learning progress by providing the necessary guidance to produce comprehensive learning outcomes [8].

The research paper is organized into four sections. The *introduction section* presents an overview of the research paper, the historical backgrounds that led the author to choose this research topic, and the likely outcome of the research. Section two, the *thesis statement section***,** presents the questions the research paper anticipates answering. Section three offers the *research method,* and section four is the *main body* of the research paper.

The main body of the research paper contains nine chapters. Chapter one is about the requirement specifications. This chapter thoroughly discusses the software architecture characteristics such as adaptability, interactivity, autonomy, decentralization, continuous and dynamic improvement, and content filtering. Chapter two presents multi-agent architecture, its organization, architectural attributes, and characteristics of agents. Chapter three will decompose the individual software characteristics described in chapter one to document the details. Chapter four discusses software quality attributes used to assess the feasibility of the software. Chapter five presents examples of multi-agent architecture to provide research background information. Chapter six commits to comparatively analyzing the software architecture regarding the software quality attributes. Chapter seven compares the architectures presented as examples (chapter five) to the proposed architecture (chapter two) based on software organization and functionality. Chapter eight is the conclusion, and chapter nine presents an annotated bibliography.

1.2. Thesis statement

An adaptive multi-agent educational system is a complex system that constitutes two or more software agents. A software agent is a functional component of the software that can accomplish a task instead of a human agent or another software agent. Software agents can act and react to changes in their environment. They can organize themselves to achieve a common goal. Such an organization of multiple agents creates a Multi-Agent System (MAS)[15]. I will discuss the common properties of agents in chapter two of this paper.

A multi-agent system is essential for education because it supports independent student learning [50], [51]. In addition, a software system that promotes independent learning must incorporate architectural components with effective measurement and evaluation metrics, advanced learning functionalities, and predictive proficiency. It is difficult, if not impossible, to achieve the functionality mentioned above without a multi-agent system. Furthermore, the multi-agent system provides a secure, self-sufficient, and highly interactive learning platform to engage the learner and produce the desired learning outcome [13], [50], [51].

A multi-agent educational system can improve the learner's learning experiences by enhancing student engagement. The software system promotes student engagement by availing consistent and constant learning support that is impossible in traditional education. For example, the multi-agent system can differentiate instructions to the student's needs, apply repeated adaptation to course presentations to meet the student's expectations, and promote continuous student engagement through uninterrupted guidance. Chapter one will elaborate more on the importance of multi-agent systems and how we can achieve them by incorporating specific software characteristics.

The research paper aims to identify and critically analyze the *software architecture* required for a distributed adaptive multi-agent educational system**.** The main objectives of the research paper are:

1. Explore and discuss the characteristics of the system architecture in detail. In chapter one, I will discuss seven aspects of system architecture, of which the first five are crucial to educational system architecture.
2. Perform a comparative evaluation of the system architecture by critically analyzing the characteristics of the architecture for software quality attributes. I will re-iterate the software characteristics and explain how they meet specific quality attributes in chapter six.

In general, the research paper aims to answer the following pivotal questions related to the architecture of a quality educational software system. The first question: *What essential characteristics should an adaptive multi-agent educational system architecture necessarily incorporate, and why?* The second question: *Which software quality attributes does the smart educational architecture guarantee to satisfy, and how?*

1.3. Research Method

I systematically collected scientific studies published in journals, books, articles, digital resources, and other publications and organized them into valuable information. In addition, I carefully looked at observable facts that confirmed the information gathered from different sources. When I collected data, I carefully considered my sources' authenticity and cross-checked facts from various sources.

The information presented in this research paper is mainly from digital resources such as the Association for Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), educational websites, academic websites such as google scholars, research gate, and ScienceDirect websites, and many other relevant and reliable digital sources. I also carefully looked at many available educational technologies in the market, tried them, and studied the futures they provide to identify their pros and cons.

Moreover, I carefully examined more than 90 works of literature on adaptive multi-agent educational systems to help me decide the system's architecture more accurately and closely to what is proposed by professional computer scientists. Next, I carefully compared the architectural features offered by scholars to organize them into multi-level *Multi-Agent Systems (MASs)*. Finally, I determined the characteristics of each MAS and their intercommunication.

The paper is informative and provides a comprehensive insight into an adaptive multi-agent educational system. However, while this research paper can serve as the starting point for further research work and independent studies, the contents presented in this research are limited. Furthermore, the material does not intend to produce the implementation or evaluation of the software system for reasons I will explain in an upcoming section.

Chapter 2: Requirement Specification

A software development process begins with the gathering and analysis of requirement specifications. The requirement specification depends on various factors, such as the target users, the areas of use, the data type needed, and the output (s) of the software. The functional and non-functional components that serve as the building blocks of the software system emerge from requirement specifications. This chapter discusses the characteristics of operational units in an adaptive multi-agent educational system.

2.1. Adaptability

According to Subramanian, adaptability refers to the extent to which a software system adjusts to environmental changes [13]. In an educational software system, adaptability implies the ability of educational software to analyze the student's performance in real-time and modify instructional materials and presentations based on the student's *performance data*. The above statement indicates that an adaptive system [47] uses real-time instructional content delivery that adapts to learners' needs and preferences, accommodating learning differences. The following diagram exemplifies the learning content adaptability based on students' performance data.

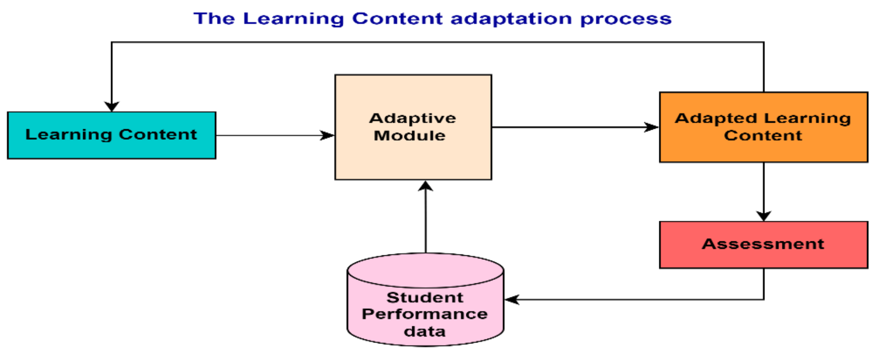

Fig. 1. Content adaptation process.

As shown in fig. 1 above, an adaptive educational system takes the crude learning content and passes it to its adaptive module to apply content modification. The adaptive module considers past student performance on the same or similar content to determine the content adjustment. Next, the component will produce or select an assessment that closely aligns with the content and administer it to the student to determine if further adaptation is required or if the student is ready to move to the next set of instructions. The adaptation process will repeat itself if the student does not demonstrate proficiency in the content area.

The above content adaptation process closely aligns with Wilichowski's statement that articulates how adaptive systems can modify the presentation of the materials in response to students' performance and adjust the learning path [47], [60]. Wilichowski further stated that in addition to facilitating instructional differentiation through presentation materials modification, adaptive systems promote self-led learning where students take charge of their education.

The preceding discussions and Wilichowski’s analysis indicate that a learning system must possess an adaptive architecture to support an enhanced learning environment that gives the student complete control over the speed and quantity of learning. Furthermore, studies show that students' engagement level increases with the power they gain over their learning pace and level [60].

Moreover, adaptive components are essential because they allow the system to capture and analyze the student’s learned behaviors, use the data to customize instructional content, and selectively present the content based on the student's progress. Fig. 1 above clearly depicts the preceding behavior of the adaptive component. The adaptive system captures the students’ performance data, stores it, and references it to determine the progress of the student and the mode of instructional adaptation specific to the student. The adaptive component use agents to achieve its functionalities. Chapter three discusses the details of the adaptive component agents and their functions.

In addition to content customization, an adaptive component is crucial in a learning system because it can accurately diagnose learned characteristics to prescribe an optimal learning experience. To optimize the learning experience, an adaptive system will consider incoming knowledge, skill and ability, demographics and sociocultural variables, and affective variables to accommodate the learning differences.

For example, consider an ESL student who struggles with grasping mathematical operation concepts due to her poor English. In this case, the adaptive system will modify the instruction and present elaborative demonstrations of the idea instead of progressively assigning exercises without considering the student's poor performance. The system can then check if the student recognizes the concept by asking related questions in the knowledge domain. Once the system ensures that the student understands the idea, it begins data collection and analysis to determine their level of understanding.

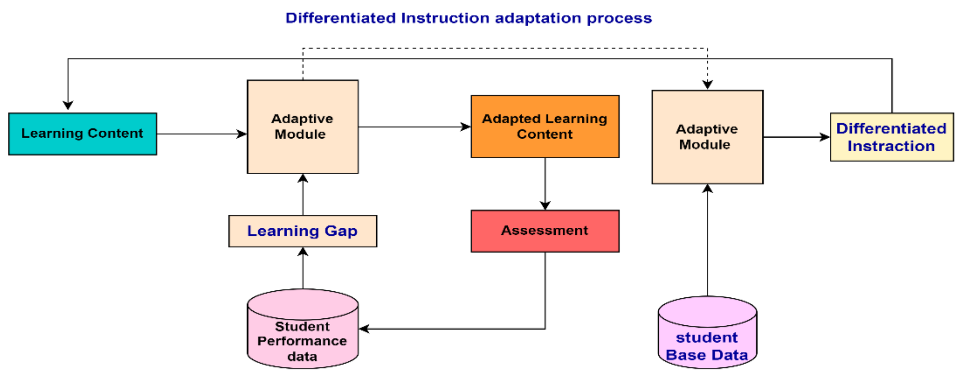

Fig. 2. Differentiated Instruction.

In some cases, content adaptation may require content differentiation to meet the learner's learning needs. In classroom-based traditional education, differentiated instruction is “an approach whereby teachers adjust their curriculum and instruction to maximize the learning of all students” in their class [93]. Content differentiation is required because a typical classroom encompasses diverse students from different cultural and socioeconomic backgrounds.

Fig. 2 above demonstrates the content differentiation process as part of the content adaptation process. The module will begin with the content adaptation process, as depicted in fig. 1 above. The module will ignore the content differentiation process if the content adaptation process is successful. However, suppose the content adaptation process is unsuccessful. In that case, in addition to the student performance data, the module will look at the student’s base data that hold additional information about the student to determine the content differentiation level and type that fits the student’s learning needs.

An adaptive educational system requires access to an extensive repository of learning resources to implement content adaptation and differentiation. First, the mandatory learning assets should align with the curriculum requirements created by domain experts. Hence, the creation of learning resources requires the broad involvement of educators. Another alternative to creating resources from scratch is to adapt them from other educational resource databases. Whether resources are made from scratch or adapted from existing academic databases, the resources should align with the curriculum requirements and involve all educators at all levels.

Concerning resource deposition, Khosravi suggests three strategies for creating learning resource repositories [61]. The first strategy is *crowdsourcing*, in which students develop high-quality learning resources through their creative and evaluative judgments. The second strategy is *moderated crowdsourcing*, where "knowledgeable and time-poor academics" oversee student content creation. The last strategy uses AI algorithms where contents are dynamically created and adapted to suit students' abilities and preferences.

There are various reasons why educational software requires adaptivity. Pugliese listed more than ten advantages of adaptive learning systems [27]. Among the reasons mentioned in the article by Pugliese, the potential of solving the perennial problem in public education and regulating course content level difficulty that remained the lasting scorch of traditional education in the municipal education sector are prime. Pugliese prescribed adaptive systems as an effective treatment for the scorch that continued to itch the conventional education system. I highly recommend reading the article by Pugliese, freely available on the internet.

Likewise, Wilichowski argued that an educational system incorporating adaptive functionality is cost-effective [60]. Accordingly, once it is well-developed and fully functional, adaptive educational systems can reduce the cost of education. For instance, replenishing online materials is much lower than the cost of producing, distributing, and selling physical textbooks. This means that maintaining an online educational resource reduces the cost of running schools for academic institutions and the expenses of the students.

2.2. Interactivity

Interactivity is a loosely defined term that can mean different things in different frameworks [62]. In the traditional classroom, interactivity implies collaboration between students and their teacher to achieve an academic goal. Collaborations can be in the form of give-and-take activities that promote students' engagement and understanding to elicit communication [48] skills at appropriate functional levels. According to various sources, interactivity refers to a continuous, dynamic, and reciprocal communication process between users and computer software and the degree to which the interaction happens [20].

Kuang identifies two broad categories of interactivity [62]. The first group relates to "the nominal or dichotomous feature of interactivity" that has to do with having or not having interactive features. However, interactive features can mean anything that can appeal to users' senses and is too broad to determine. The second category approaches interactivity from its "functional characteristics," such as "email links, access to extra information links, feedback forms, chat rooms, audio and video downloads," etc.

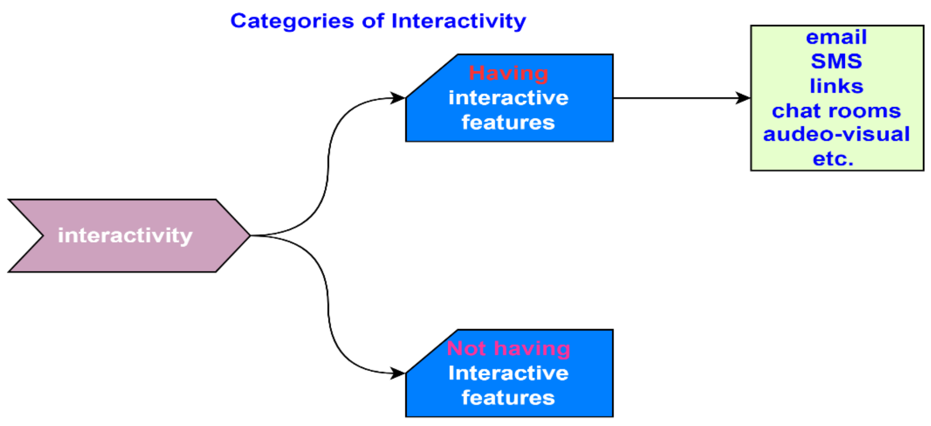
Categories of Interactivity
interactivity
Having interactive features
email
SMS
links
chat rooms
audeo-visual
etc.
Not having Interactive features

Fig. 3. Categories of Interactivity.

Regardless of its loose definitions, broad connotations, and mode of presence, interactivity is imperative for an intelligent educational system because it determines students' cognitive engagement and motivation. According to research by the University of California, Berkeley, interactive components as part of the learning platform encouraged students to engage in the learning process and retain a higher proportion of the learning content [18]. Therefore, it is necessary to incorporate interactivity in an educational architecture that supports an intelligent learning environment with which the learner interacts continuously.

Interactivity, as a component of an educational system, is vital because it supports the simulation of dynamic classroom events. Although it slightly differs from teacher to teacher, a typical classroom is a highly dynamic environment that goes through multiple states during a classroom session. The initial classroom state is the introduction, in which the teacher welcomes the students and introduces the topic. Next, reminders and announcements follow where the teacher reminds students about upcoming due dates or other assigned tasks. The third state is the revision of past lesson(s), where the teacher highlights the main points from the last class session(s). The final state is cyclic, which goes through various stages where the teacher delivers content.

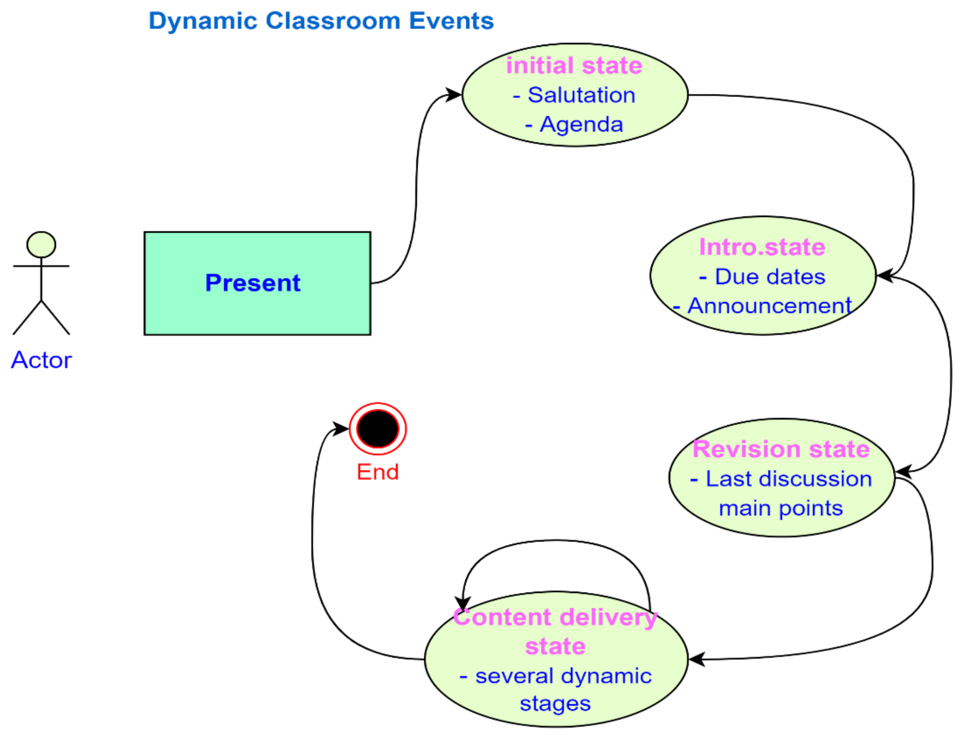

Fig. 4. Simulation of dynamic classroom events.

Simulation of dynamic events is necessary to encourage cognitive engagement. Various sources show that dynamic events such as demonstration, simulation, and elaboration through examples encourage students to stay focused and assist them in comprehending complex ideas more accurately and quickly [17], [18]. For instance, consider a physics class where the teacher discusses planetary motion. The student can quickly learn the relationship between the period of motion and the inverse square radius relationship if the complex motion of the object is simulated to the student. Furthermore, in addition to the demonstration, the simulation of the step-by-step computation will uncover the relationship between the area swiped by the object, the inverse square radius, and the period of motion, which will further enlighten the student's understanding.

Second, interactivity provides extra practice opportunities for emerging skills. An emerging skill is a new skill that the student recently acquired, such as new mathematics or reading skill that requires reinforcement and reshaping until the student understands it conceptually and internalizes it. An emerging talent belongs to the critical thinking development category. Hence, it should be nurtured with appropriate intervention and continuous follow-ups until such skills fully develop independently.

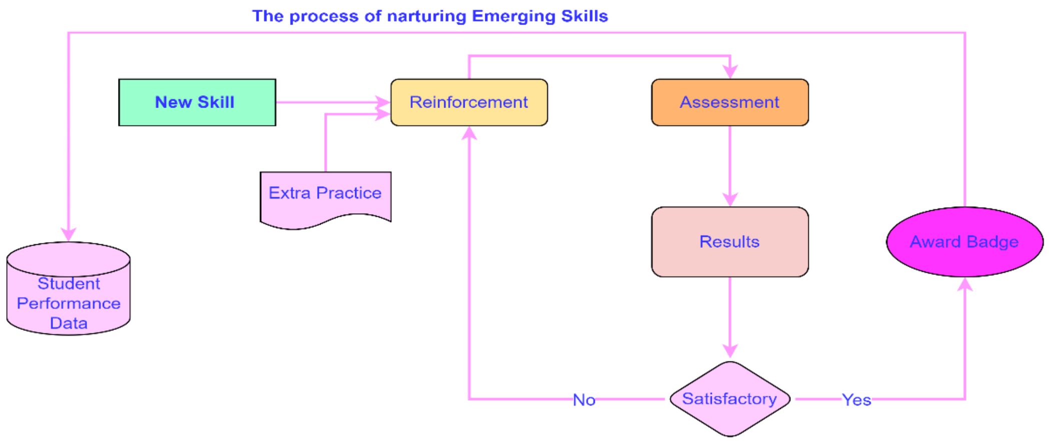

Fig. 5. Reinforcement learning process.

The nurturing of emerging skills is a complex process that starts with identifying the new talent. As indicated in chapter three, the Emerging Skill Agent (ESA) identifies the new skill by keeping track of all skill sets the student acquired previously. A skill not in the previous skill sets is then categorized as a new skill. The newly acquired skill is then passed to the reinforcement module, providing relevant extra practice opportunities followed by appropriate assessments. If the assessment results indicate mastery of the new skill, the new skill becomes part of the student’s skill set, and the performance data is updated. Otherwise, the process will continue until the student demonstrates mastery of the newly acquired skill.

An educational system realizes emerging skill reinforcement processes through its interactive module. The module identifies emerging skills from the students' skill sets and cultivates them through reinforcement learning. For example, suppose a student acquires a new skill of identifying word structure in a sentence. In that case, the system should recognize the unique talent and provide the student with extra practice opportunities by selectively and progressively delivering distinct types of sentence structures until the student demonstrates proficiency in the new skill. The system displays such behavior only if its architecture exhibits reinforcement learning, as shown in fig. 5. At the same time, the system can encourage the student to continue learning by rewarding achievement badges for each level the student completes successfully.

The example above suggests that a standalone intelligent educational system needs mechanisms to track emerging skills and enforce continuous and progressive learning until the skills are fully internalized. The system demonstrates such a mechanism through its agents in its interactive modules (see 3.4). Furthermore, reinforcement leads to a well-nurtured, strengthened, and internalized skill that can last longer than loosely acquired knowledge. Conversely, losing a craft or ability results in a knowledge gap that can significantly affect the students' academic performance in the long run.

The system’s ability to implement a reinforcement learning mechanism relies on the Emerging Skills Tracker Agent (ESTA) in the interactive MAS (Chapter 3). The agent will keep a record of the skills achieved by the learner. Suppose it identifies a talent that is not part of its record. In that case, it will categorize it as a new skill and begins gauging the student's achievement on the unique skill set according to a skill matrix provided by the school or the district education bureau [39].

Third, an interactive component supports the use of alternative multimedia options. Multimedia options are dynamic teaching resources that aid or model the teaching in a classroom and constitutes: animations, audio-visuals, simulations, PowerPoints, experimentation, and demonstration that a teacher can employ to help him make effective content delivery. A teacher uses different delivery options in a traditional classroom to create an effective instructional delivery. Similarly, an effective instructional delivery system must have alternatives to make instruction delivery robust and effective; alternative multimedia options can play a part here.

An instructional delivery system requires alternative multimedia options to meet the diverse learning preferences of the learner. For instance, a student may prefer watching movies related to the content, while another may prefer reading material. In this case, the interactive component must present both options to meet the needs of the students.

The alternative multimedia options select the materials adaptively from the pool of resources and present them according to the student's learning preferences, the relevance of the material, and course completion requirements. For example, reading may be more relevant to audio dictation when the content involves visual aids such as pictures or graphs. In this case, presenting the content in the reading format is more relevant than presenting it in audio format. On the other hand, if the content involves reading (for example, reading comprehension assignment), then reading is required as part of the course completion requirements, and it is the only option.

In addition, the system suggests different multimedia options based on their significance. For instance, consider a student who prefers reading texts to watching a video or a simulation and usually skips over these contents. Consider also this student is working on the planetary motion topic. Suppose the student's assessment performance on this topic is unsatisfactory upon consecutive attempts. In that case, the student is likely struggling with grasping the concept, and the simulation of the idea is highly relevant; the system should suggest the same to the student.

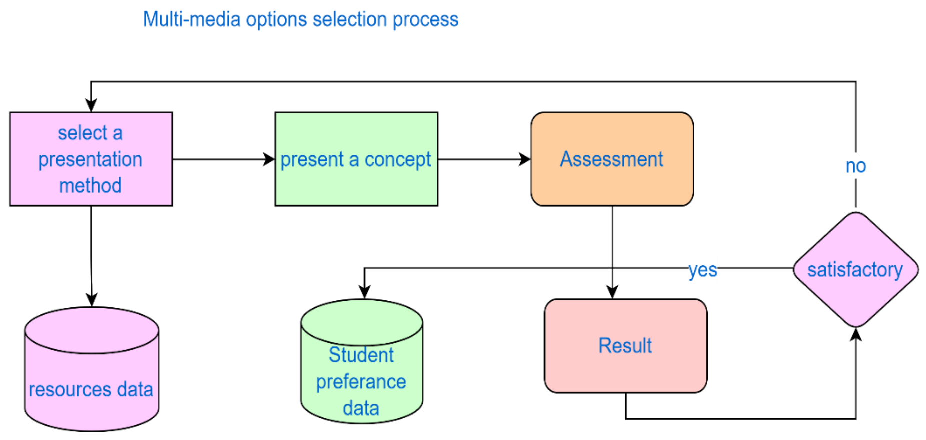

Fig. 6. Multimedia selection process.

Fig. 6 breaks down the complex multimedia selection process into a simplified diagram. The module will start with selecting a presentation method that it finds relevant to the content and the student preferences derived from the available data. Next, the module makes the presentation and initiates data collection on participation, the duration the student spends on the course, and past assessment performance to determine the student's level of engagement. Finally, the module administers an assessment. If the assessment results are satisfactory, the student preference data is updated, and the module moves on to the next set of instructions. Otherwise, the process repeats with a different multimedia option until the student demonstrates proficiency in the current content.

The system incorporates the multimedia selection process, as depicted in fig. 6 above, through its special agent known as the Alternative Multimedia Option Agent (AMOA) (see 3.4.3). The AMOA is a part of the interactive module (MAS) that specializes in the multimedia selection process. Its job is to make the content delivery process effective, allowing the learner to grasp and retain the concept quickly.

Furthermore, the availability of diverse instructional materials will maximize the number of motivated and engaged students in the learning process. Some students are proficient readers. Enough reading materials can motivate the skilled reader to engage in learning, maximizing the learning outcome. Other students are slow readers. The slow reader students benefit from other media options such as pictures and audio-visual materials.

Finally, an interactive system provides a framework for socialization and social interaction**.** Socialization and social interaction interfaces are applications such as chat services, messaging services, video conferencing services, breakroom services, grouping services, forums, etc., that connect students and serve as the medium of interaction between small groups of students or students' communities [25],[38].

An interactive educational system must avail socialization and social interaction services either through interfacing with socialization and communication service providing clients such as Facebook, What's up, Twitter, etc., or by incorporating components that can effectively achieve the required services. One critical requirement of socialization and social interaction applications is their privacy and security. In either case, using these services should not compromise the privacy and security requirements of the system.

Socialization and interaction between students and teachers are necessary because they create a platform for uninterrupted learning. Suppose a group of students is working on a school project. Suppose they are using an online learning platform to complete the assignment. In this case, the physically isolated students can continue working together, exchange ideas, ask questions, and receive fast responses if the system provides an interaction mechanism. The availability of such a mechanism promotes engagement and makes learning an uninterrupted process.

Social interactions can either be among students or between students and system agent(s). For instance, the agents can categorize students and put them in the same group based on their learning preferences and specialization. According to Ahamed, Students with similar choices and academic discipline can easily clique and socialize with no or little barriers and work towards a common goal [16]. In his article, Ahamed indicated that content-driven group collaboration effectively creates eco-connectivity among groups with similar preferences and specializations.

2.3. Autonomy

According to the article by Borchert published by the National Institute of Standards and Technology (NIST), the term autonomous system (AS) usually refers to a group of networks with a collection of a large number of devices and software systems that performs complex tasks [94]. In the context of this definition, the best example of an autonomous system would be the internet. Although it is not the only autonomous system, the internet is undoubtedly an independent system comprising a group of networks and a collection of many devices and software.

In the book, *Human-Machine Shared Contexts*, to define autonomous systems, Lawless et al. quoted Watson and Scheidt, "… systems that can change their behaviors in response to unanticipated events during operation" [63]. Angelov defines autonomous systems as "systems with a certain level of evolving intelligence" [64]. The above definitions imply that autonomous systems are active systems that evolve to respond to environmental dynamics.

An adaptive and intelligent multi-agent learning system must be autonomous to independently perform the complex and dynamic tasks that a classroom teacher performs daily. As discussed in section 1.2 above, the classroom teacher performs complex, dynamic, and multi-stage duties that the adaptive intelligent multi-agent system must simulate in the teacher's absence. However, the simulation of the actual teacher’s classroom duties is unachievable unless the system is autonomous (self-sufficient).

Unfortunately, this was not the case when Bork predicted that a computer-based educational system would become the most predominant form of instructional delivery [17]. Yet, today, with highly sophisticated computing machines and marvelous machine learning algorithms, we witness that Bork's prediction is a reality. Autonomous systems are the current trend in technological advancement, and scholars are exploring their application areas, such as self-driving cars, autonomous mobile robots, automatic tailor machines, autopilots, etc.

The autonomous module (MAS) starts with some knowledge about a new user and establishes interaction based on what it knows about the user. For example, the user’s base profile data can serve this purpose. As the user (learner) continues to interact with the system, the system will initiate data collection about the nature of the interaction, analysis the data, and adjust itself dynamically based on the analysis result. This process will continue until the system has "enough" data or the user changes the interaction pattern.

For example, suppose the student is reading a section of the course. Suppose the student navigates away from the course several times to look up the definitions of terms used in the presentation. In that case, the student is likely struggling with describing terms in this content area. The module can determine the need to present definitions and the material altogether.

Evolving intelligent systems achieve autonomy through learning changes in their environment and adapting to learned changes. For instance, the system can autonomously simulate the complex and dynamic learning process using machine learning paradigms. However, the system requires the following elements to achieve the intended characteristics.

There are various techniques to enhance system autonomy (degree of independence). The first method is to build modules that display *self-contained* system properties. A self-contained system (SCS) is a software system containing small and isolated application units that operate independently to perform a specific function. The components can work independently and maintain asynchronous communication whenever possible over RESTfull HTTP or lightweight messaging [29].

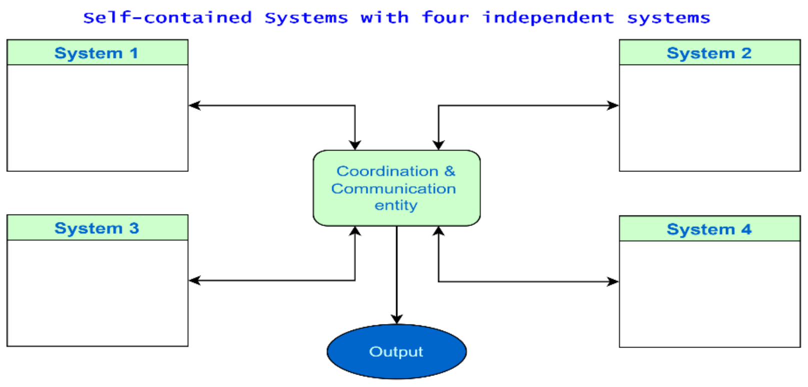
Self-contained Systems with four independent systems
System 1
System 2
Coordination & Communication entity
System 3
System 4
Output

Fig. 7. Self-contained systems interaction.

Fig. 7 shows a self-contained system that comprises four smaller isolated units that undertakes specific operations independently. The units interact through joint coordination and communication to achieve a single output. In a learning platform, the individual units represent the system modules (MASs) or the system agents. Chapter two discusses the properties of agents, and one property is their independence.

One of the advantages of self-contained systems is that they can achieve greater autonomy by minimizing component mutuality. Minimized component mutuality is desirable because if components are highly reliant, errors that originated in one element can propagate to other parts, degrading system availability and reliability and eventually leading to unpredictable system behavior that makes system learning difficult [65]. However, if the system is self-contained, error propagation is minimal, and the system can learn and evolve quickly [65]; not the case for monolithic systems, where the individual components are strongly coupled.

Another advantage of self-contained systems is that components have greater neutrality, maximizing system availability. The elements have their UI, specific business logic, and a separate database [65]. If an element experiences system failure, the other pieces continue to function without suffering from the collapse of the failed component. Hence, minimized failure (maximized availability) is an excellent quality that enhances system learning.

Another effective way to enhance system autonomy is by incorporating *intelligent software systems (ISS)* [63]. Troyer describes software intelligence as "the ability to perceive, process, remember, learn, and determine the course of action …". ISS can anticipate the future outcome of an action and recommend the best procedure for a particular process. More importantly, an ISS can make intelligible decisions that lead to the best results.

ISS can achieve autonomy by enhancing the system's decision-making ability, a characteristic required for a system's independence. As stated above, ISS can anticipate future outcomes from current patterns using a large dataset that machine learning (ML) algorithms require for the system's dynamic evolvability. The large dataset in an ML algorithm and the system's anticipation power can significantly improve the system's autonomy.

Furthermore, a system that learns from previous mistakes and remembers the learned behavior can improve until it becomes autonomous [64]. For example, suppose the system component is assigned to perform task X. Also, consider the initial performance of the system is *k*, *k* less than 100%. Then, if this system can learn from the errors it makes in predicting an outcome at a rate of *r* over a time t, its prediction proficiency is given by:

$$Learning\ (X) = \ k\left(1 + \frac{r}{100}\right)^{t}$$

$$Considering\ all\ other\ factors\ are\ constant.$$

The third method to introduce system autonomy is by incorporating *predictive analysis* behaviors**.** Predictive analysis is a method that uses data, statistical algorithms, predictive modeling, and machine learning principles to make predictions [65]. Furthermore, predictive analysis techniques use data mining to acquire the required information, applying statistical algorithms to approximate future outcomes. This technique is widely used in forecasting marketing trends, customer service, voice-to-text conversion, investment portfolio development, etc. [65].

Predictive analysis behavior supports autonomy by increasing the accuracy of a prediction. Predictive analysis performs complex statistical calculations to enhance prediction accuracy. As the system's prediction accuracy improves, it guarantees its evolvability, eventually leading to independence. For example, consider content filtering and presentation. Initially, the system will rely on the base user data to adapt and present contents. Next, the system collects additional user behavioral data and performs a predictive analysis. Finally, the module uses the evaluation result in consecutive prediction attempts to improve content filtering and presentation, leading to better prediction accuracy and, hence, system independence.

Furthermore, the system's predictive analysis behavior implies the module's ability to independently analyze each learning behavior to determine the following learning goals based on the student's preferences and progress. The system will use determinant factors such as the student's past performance, the course completion rate of the past learning objectives, course duration, and working time to make a predictive decision.

For instance, suppose the student is proficient in learning goal A and struggles with goal B. In this case, the system will check all the appropriate instructional delivery methods for learning goal B, and the student spent enough time on the learning materials, although they couldn't make much progress. Therefore, if learning goals A and B belong to a common domain, the system will proceed to the next learning goal in the same field since the student's overall performance in the learning domain is acceptable. On the other hand, suppose learning goal B is not in the domain of goal A. In that case, the system will measure the student's performance in the field B belongs to and, based on the result, will decide whether the student should continue with that domain or redirect to the domain in which they performed well.

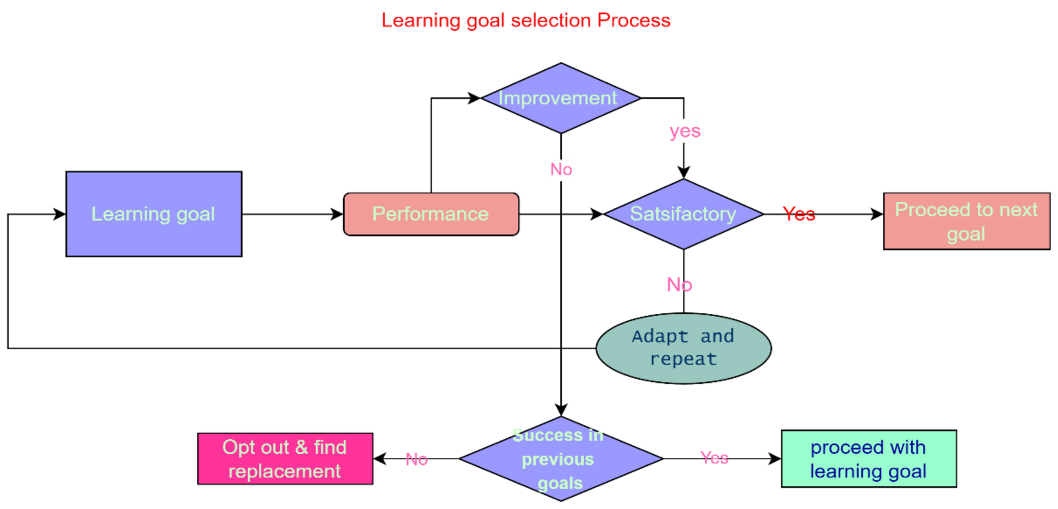
Learning goal selection Process
Improvement
yes
No
Learning goal
Performance
Satsifactory
Yes
Proceed to next goal
No
Adapt and repeat
Opt out & find replacement
No
Success in previous goals
Yes
proceed with learning goal

Fig. 8. Learning goal selection process.

Fig. 8 depicts the decision-making process for learning goal selection. The autonomous module begins with a learning goal in the student's grade level and learning domain. Next, the module looks at the student's performance data obtained from assessment results to determine if the assessment results are satisfactory or need improvement. If the results are good, the module promotes the student to the next learning goal in the domain. However, if the results are unsatisfactory, the module will notify the adaptivity module to initiate content modification. The adaptation will be followed by another performance check. If the performance result yields improvement, the system will repeat the process. Suppose the performance data shows no improvement upon consecutive course adaptation attempts. In that case, the module will check if there is a success in a previously related goal and advance the student to the next learning goal. If no hit is observed in the previous plans, the system opts out of the target and finds a replacement.

Predictive analysis results will help the student anticipate the expected outcomes of a learning process and work towards achieving the intended goal. In addition, since predictive analysis is efficient in making accurate predictions and provides timely feedback, the student can use results obtained from the evaluation to work towards what they anticipate achieving. Furthermore, predictive analysis will enhance the system's ability to continuously collect the students' performance data and store it permanently to make inferences for future predictive analysis needs. Data collection will include the type of question (subjective or objective), its difficulty level, learning goal, repeated or first response (number of repetitions), response accuracy level, current overall (average) score, and other relevant information such as the student's progress, proficiency, learning rate, and retention rate etc.

2.4. Decentralization

Decentralization is a recent development trend used in building complex and change-tolerant software [67]. Although the meaning of its concepts remains the same, politics and economics use the term decentralized more extensively than software system architecture. In the broad spectrum, decentralization stands for withdrawal from being centralized [66]. According to an article by Suryanarayana hosted at the Institute for Software Research (ISR), decentralized architecture is a software entity with "multiple (disagreeing) agencies that have different and possibly conflicting goals" [67].

A decentralized educational software system can provide services to multiple agencies (schools in our case) without limitations or restrictions. Decentralization allows different schools to apply unparalleled educational policies without needing to change or modify the software while they use the same database and network. The system will not restrict the service (as in the case of a centralized software system); instead, it will provide a platform for the agencies to establish educational services that meet their institutional goals. The agencies will assume complete control and are responsible for what they offer and how they offer essential services on the educational platform.

One conventional approach to decentralization is blockchain technologies [68, 69]. According to Hayes, "a blockchain is a distributed database that is shared among the nodes of a computer network." A blockchain collects and stores data in groups of blocks with limited storage capacity and chains the data group together when filled in what is known as the blockchain. A blockchain records and distributes digital information without permitting changes to the dispatched data over the network.

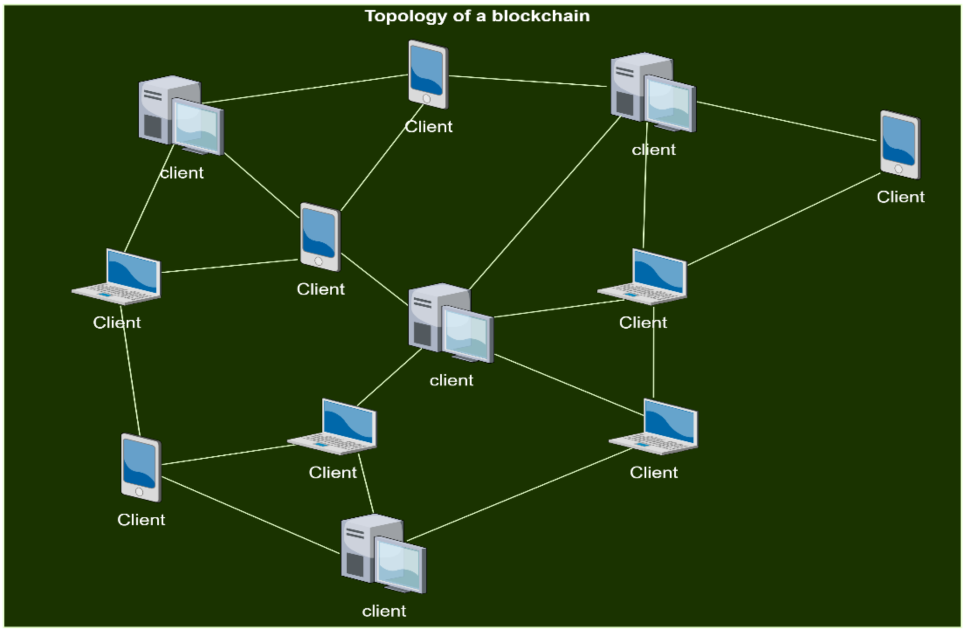

Fig. 9. An example of blockchain topology.

A well-known reason for requiring a decentralized structure for an intelligent educational system is that it necessitates accessibility, enhanced computational speed, and fast access. Placing hardware and software resources close to their access location can ease accessibility, enhance computational speed, and minimize the overall access time by reducing propagation time. In addition to improving computational speed and minimizing the required access time, decentralizing resources is a modern-day practice that increases software applications' reliability, scalability, and security [68].

Another reason to incorporate decentralization in an adaptive multi-agent educational system is that it necessitates distributed architectural framework to support and run the complex hierarchal Learning Management System (LMS). One of the drawbacks of the existing LMS [58] is their centralized nature, leading to a single point of failure. Moreover, a centralized system tends to work the same way, and if we want to use it differently, we have to modify or implement the future on the central server.

In addition to the reasons mentioned above, there are various learning platform-specific reasons for an LMS to support decentralized architecture. First, a learning platform must have a sequencing capability that requires a decentralized structure. An adaptive multi-agent educational system must support sequential learning for all subject areas, contents within each subject, and learning goals described within the ranges [57], [58]. The system must present the learning materials sequentially and dynamically based on a set of predefined learning goals, the assessment results of the learner, and the learner's learning preferences. Distinctive sequencing of the learning materials is vital because each learner's learning experience and selection differs. Hence, the system should measure the learned skill sets, compare them to standards, make the necessary adjustments to the learner's achievements, and decide on the following skill target for each learner. Since the individual users (learners) own data in a decentralized approach, the delivering system can achieve sequencing of learning contents in an isolated fashion.

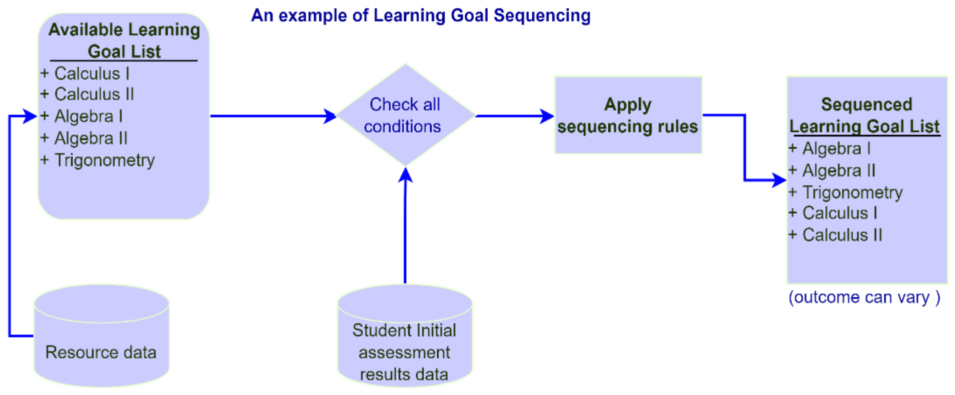

Fig. 10. Learning Goal Sequencing process.

Fig. 10 shows the learning goal sequencing process as one content sequencing category. First, the module selects the learning contents from a poll of learning resources based on the student's learning need, learning goal, and grade level. Next, the module infers the student's assessment results to check if the student met the pre-requirement for a course. If the pre-requirement is met, it applies sequencing rules and puts the course for which the requirement is satisfied at the top. Finally, the module assumes that the student will complete the top course successfully and start to place other classes in the order of their logical hierarchy.

Second, a learning platform must support a progressive learning environment to promote liberal and independent learning. Liberal learning is a new educational movement that stresses learning by doing. Education research affirms that liberal learning significantly increases learning retention [26] and encourages engagement [26]. A decentralized educational system supports liberal learning by providing learners with engaging, differentiated, and practical learning experiences. In addition, a decentralized system optimizes resource distribution with better performance and data consistency that can provide a playground for progressive learning.

Decentralization realizes liberal learning by making learning private. Independent learning is reasonable because it neither restricts the learners' potential nor necessitates the learner a completion time frame. Traditional education requires students to stick to a certain amount of time (for example, a term) to complete a course. However, students' learning speeds in a given class differ significantly. Some students are fast learners, and the quick learner could finish the class in a shorter time. Other students are slow learners, and the slow learners could take additional time to materialize and understand the course contents. Unlike traditional education, liberal learning benefits both the quick learner and the slow learner by allowing them to adjust the speed at which they want to learn.

Hence, decentralized systems support self-paced learning where learners assume complete control over their learning speeds. The system establishes a personalized learning environment [43], [59] independent of external factors. In a learning setting free from external influence, the learning process happens at the learner's pace. In an individualized and self-paced learning environment, the learner will possess an independent learning environment where the learner assumes complete control and responsibility.

Lastly, decentralization supports specialization tracking & redirecting. An educational system must track the progress of the learner, the areas of failure, and the areas of experts. The system collects data by monitoring the learner's progress, analyzing the data, and determining the areas of deficiency and success. After enough data is collected and analyzed, the system should tell the learner's specialization areas and redirect them to the resources that can further nurture the specializations until they internalize them.

Specialization tracking and redirecting is possible only if the learning platform supports decentralization where student data are distinct and isolated. Although data isolation is achievable in centralized and distributed systems, centralized data access is not as efficient as decentralized data access. In complex and multi-tiered systems like the one proposed in this paper, where data is frequently accessed and manipulated, isolated data dramatically increases system performance and throughput.

2.5. Continuous and dynamic improvement

One of the critical features of an adaptive multi-agent educational system is its ability to collect and store data, analyze the stored data, and predict the best course of action to dynamically apply continuous improvement to the learning experience based on the analysis results. When applied to the educational system and the educational delivery method, the complex term, continuous and dynamic improvement**,** has dual and varying connotations.

In the context of a system, the term continuous and dynamic improvement refers to the progressive evolvability of the system with improved quality attributes. For example, a progressively evolving system can make performance and other software quality attribute adjustments by learning from previous mistakes, as discussed under autonomy (section 1.4 above). Continuous and dynamic improvement is not the same as what is known as continuous integration (CI) or continuous development (CD). While CI/CD is a phenomenon involving the modification of software applications under a controlled and developer-led situation, continuous and dynamic improvement (CDI) is a spontaneous process involving no developer participation. Hence, CDI is mainly concerned with improving quality attributes with no additional component integration. CI/CD targets both component change and quality attribute improvements.

The term continuous and dynamic improvement also implies progressive enhancement and adaptation of methodology and content throughout the life of a knowledge domain. A progressive adaptation of methods and content is significant in a learning platform because of the dynamic nature of the learning process. Suppose one develops a learning platform with the current best teaching practices and contents with no room for dynamic improvement. The tool functions well in the present society, and learners are happy with it. However, since the social structure is nonstatic, ten years from now, social needs may change, and the learning needs with it making the existing teaching methods and contents obsolete. The old (legacy) system may require extensive modifications to continue using it. However, if the system can dynamically and progressively improve by learning the new teaching methods and contents, it does not require extensive modification to continue to use it.

Dynamic and continuous improvement is essential because a learning system must continually and progressively enhance content delivery, sequencing, diversification, evaluation methods, progressive differentiation of zone of proximal development, etc. The module (MAS) achieves the mentioned enhancements by keeping track of the student's engagement and participation in terms of the length of time, the number of materials covered for each content, and the presentation methodology used to infer learning reference. Data manipulation follows the data collection to determine the required changes. Based on the results, the system integrates the changes and continues with the data collection and manipulation process. Moreover, the system will continuously improve content and content presentation methods based on the automated feedback it generates from the student's engagement data.

The module depends on two crucial data sources for continuous and dynamic improvement. The first dataset is related to the system's accuracy in decision-making. The system should collect, store, and analyze data used in decision-making and the decision results to act on and make self-diagnostic progress. Collection and manipulation of the decision-making dataset will enhance the system's progressive evolvability, as discussed in the previous section of the chapter. The second dataset is the student's learning data used to determine the learning content and the required presentation method for effective delivery. It is self-evident that the first dataset might depend on the quality of the student learning data.

Steady and dynamic learning experience improvement will always begin using a base learner profile. Learners will provide personal information related to demographics and educational background to serve as a starting point. The base student data (base information) must be good enough to assign the student to an academic level that may move up or down depending on the initial assessment results of the student.

The next step is to administer a series of assessments to determine the approximate performance level of the student. Once an approximate performance level is selected, the system should initiate data collection, analyze the data, and continuously apply a dynamic improvement to the student's learning experience based on the analysis results.

As discussed above, there are various reasons to include continuous and dynamic improvement characteristics in an educational system. Furthermore, software characteristics such as adaptability and autonomy heavily depend on the data collected by the system. Therefore, the progressive improvement of the system will gradually lead to better adaptability and enhanced independence. Moreover, continuous and dynamic improvement can improve the student data quality over time and achieve better feedback on the student's academic performance.

2.6. Content filtering

According to Loshin, content filtering is a software or hardware screening process to restrict access to objectionable application contents. In simple and layman's terms, content filtering implies classifying content into good and bad categories and blocking the bad from passing and reaching a viewable state. Consequently, content filtering is standard in government sectors, educational institutions, companies, and parents.

There are various types of content filtering. Web filtering is the most common web content-blocking form. Email filtering is the process of identifying malicious and harmful email content and categorizing them as spam. Executable filtering identifies malicious software and stops it from running in an operating system. Finally, DNS filtering is the process of blocking network access from potentially harmful sources.

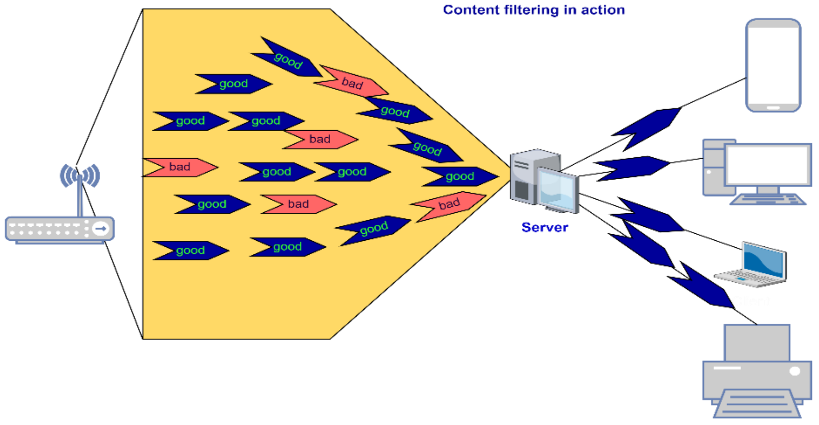

Fig. 11. Content filtering process.

Using network applications such as firewalls or domain name systems (DNS) servers is a relatively simple method of filtering content. Network administrators can configure servers and firewalls to block unwanted content and stop them from getting to the user. Fig. 11 shows how the server identifies and blocks harmful content while allowing good content to pass. Nowadays, most filtering software uses machine learning classifiers to effectively identify and block inappropriate content.

One of the reasons why content filtering should be part of a learning platform is that government policies require it. The Children's Internet Protection Act (CIPA) requires all schools to implement an internet safety policy that includes content filtering to protect children and minors from inappropriate internet content [21]. Inappropriate internet content includes "pornographic nature or sexually explicit, inciting crime, violence, or intolerance: topics deemed inappropriate for juveniles by the code of Virginia" [22]. Therefore, regardless of its application area, content filtering is undeniably an excellent practice to incorporate into the learning system to make the learning environment safe and free of worries.

Another crucial reason for incorporating content filtering features in an educational platform is *scaffolding*. *The Glossary of Education Reform* defines scaffolding as "a variety of instructional techniques used to move students progressively toward more robust understanding and, ultimately, greater independence in the learning process" [23]. The module must apply content filtering to determine the content's appropriateness to a student based on the *student's zone of proximal development (ZPD)* [95] and their performance in the subject area. The module identifies the student’s ZPD by measuring the gap between what the student can do and what they are supposed to do, given the age and class level of the student. Based on the observed student's ZPD, the MAS must apply the base standards to determine the lower and upper classes to filter contents.

The module achieves scaffolding by determining the student's knowledge level of a learning target. Next, the component establishes the learning goal(s). Then, select the presentation method and the accompanying assessments. Finally, the module continuously monitors the student's progress as it gradually fades support away.

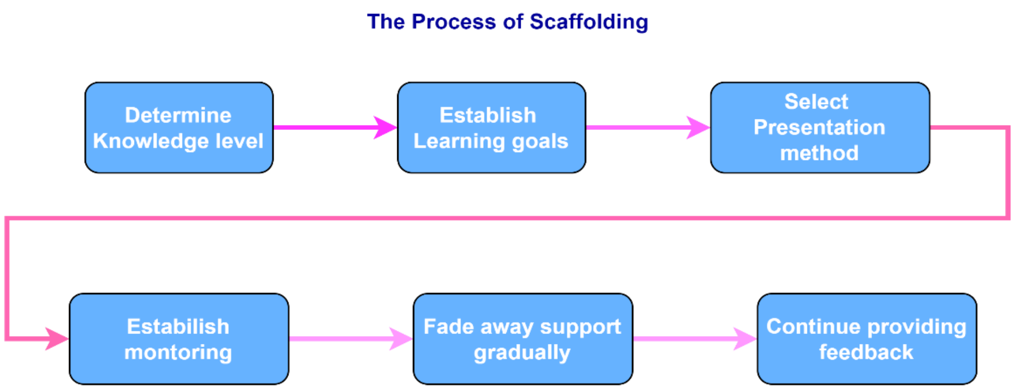
The Process of Scaffolding
Determine Knowledge level
Establish Learning goals
Select Presentation method
Estabilish montoring
Fade away support gradually
Continue providing feedback

Fig. 12. The process of scaffolding.

But what about exceptional students who perform way above the upper-level standards? The system should refer the student to a designated expert agent to make the necessary assessment and move the student to the next level. The agent should collect all available data about the student, make a rigorous analysis of the data, determine the accuracy of the investigation, and decide based on the obtained data results. If the decision favors the referral, the student will move to the next available level. If the decision is against the referral, the agent will continue collecting more data and periodically performing data analysis to monitor future changes.

2.7. Concurrency

If there is an essential term in computer science that has become so common today is the word concurrency. Concurrency is everywhere! We open and run multiple websites simultaneously (maybe we are processing a bank transaction online while downloading a valuable e-book). We open and run various applications on our smart devices (texting a friend while we are recording a video) etc.

Concurrency implies performing multiple computations at the same time. There are three key sources of concurrent execution.

- Network-connected computers running the same application at the same time.
- Multiple applications running on a single computer.
- Multiple processors on the same computer run various applications at the same time.

A multifaceted educational system can achieve concurrency by using one or a combination of the sources of concurrency mechanisms mentioned above. For example, the system runs on multiple networks in a distributed environment, making it a candidate for network-connected computers running the same application simultaneously. Furthermore, numerous applications, video streaming, file viewing, completion of an assessment, etc., can happen concurrently on the same computer. Finally, all computational devices we use today have multiple processors that run various programs simultaneously.

Concurrency is an indispensable component of a complex and large-scale adaptive multi-agent educational system because a large and complex system with many potential subscribers must be systematic, productive, meticulous, and satisfactory. In short, the system must be dependable. Dependability, the main factor determining the system's overall productivity, constitutes availability, reliability, maintainability, durability, and security [24]. Moreover, the system must provide good services within an acceptable response time, i.e., the system is satisfactory (meets the minimum user expectation) in terms of functionality and performance. A system that supports multitasking and parallel computation can demonstrate remarkable dependability by decreasing response time and increasing performance.

Concurrency is the system's ability to demonstrate dependability by overlapping program modules' execution. Although synchronization poses a challenge, concurrency can improve a system's resource sharing. Systems that use resource sharing can minimize the resources needed and guarantee efficient use of available resources. In addition, program execution can take place on different processor cores for a multiprocessor system, significantly reducing response time.

An adaptive multi-agent educational system that supports concurrency can improve students' psychological experience by fostering the students' perception of using the technology. According to an empirical study conducted in Vietnam by Maheshwari, perceived usefulness (PU and perceived ease of use (PEU) [25] will affect the user's acceptance of educational technology. The student's perception that the educational system's commendable performance will develop confidence, fostering a positive psychological experience.

Furthermore, an adaptive multi-agent educational system that supports concurrency minimizes distraction and encourages students' engagement. Conversely, a slow and unresponsive system will discourage students by promoting disturbance or intermittence. For example, a study conducted on teaching behavior, academic learning time, and student achievement by Fisher et al., describes, "The proportion of allocated time that students are engaged is positively associated with learning" [27]. On the other hand, if the students disengage due to system multifunction or slow response, that time is negatively associated with learning.

Due to time and resource limitations, this study highlighted characteristics of an adaptive educational system such as mobility, distribution, security, privacy, and openness. Furthermore, the reader should explore the behaviors of agent-based systems such as social organization, cooperation, coordination, control, and communication [48]. Agents can also display local views, proactivity, reactivity, and social skills that promote student engagement in the learning process.

Chapter 3: Multi-agent Architecture

This chapter explores multi-agent architecture paradigms crucial for the educational system architecture requirements specifications described in chapter 1. The organization of the chapter follows the logical order of the concepts it discusses. The chapter starts by defining multi-agent and exploring the meaning of multi-agent architecture [15], [45]. Next, the chapter covers the different properties of agents. The following subsection will present and discuss the organization of multi-agent architecture. After that, the architectural attributes of multi-agent architecture will be addressed. Multi-agent architecture and the requirement specifications will then follow. Finally, the chapter presents multi-tiered multi-agent systems and their applicability.

3.1. Multi-Agent System Architecture

According to the article by Palau et al., a multi-agent system is a system in which the structure and topology of its agents determine its architecture. Another definition by Wikipedia states, "A multi-agent (MAS or self-organizing system) is a computerized system composed of multiple interacting intelligent agents" [45], [54], [69]. A multi-agent system architecture consists of two entities: the software agents and their environment. Software agents are complex programming entities that can act instead of a user. They are persistent, goal-oriented, and environment-specific. A setting of a software agent, also known as the environment, is an external condition that supports the actions of the software agent.

Multi-agent systems comprise various agents that can coordinate and cooperate to achieve a specific common goal. Palau identifies four major multi-agent architecture categories in the article, multi-agent system architecture for collaborative prognostics: centralized, hierarchical, heterarchical, and distributed [70]. The piece states that the architecture of a multi-agent system determines the structure and topology of the constituent agents.

A centralized multi-agent system encompasses the simplest, easy-to-understand, and most intuitive architecture. In centralized systems, one or more client (or a peripheral server) nodes connect to a central server through a connection protocol to send and receive communication messages. The server that gets the request message from each node, the primary server, should always be up and running, waiting for a connection request from the clients (peripheral nodes).

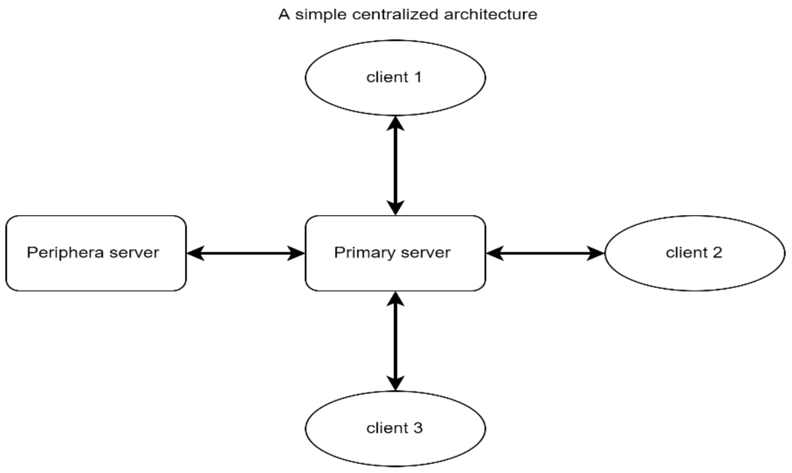

Fig. 13. A simplified visualization of centralized architecture.

Centralized architectures are characterized by a single global clock on which the entire system depends, leading to clock synchronization problems. Furthermore, the server is the sole central unit to handle all the communication and coordination tasks. Hence, the failure of the primary server leads to the collapse of the whole system. In addition, the primary server can overwhelmingly become busy if the peripheral nodes send/receive messages simultaneously, leading to performance issues.

Hierarchical architecture is a software architecture composed of software modules or subsystems arranged into a tiered structure. Hierarchical approaches are common in developing network system protocols and operating systems. The arrangement generally comprises the lower, middle, and upper layers. The lower-level subsystem provides services to the adjacent upper-level systems. The middle layer offers domain-dependent functionalities, whereas the upper-level service is an interactive interface.

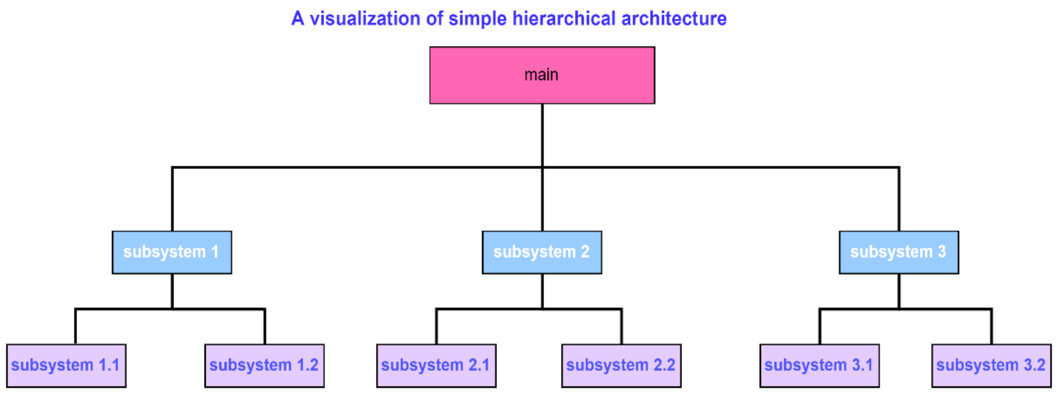

Fig. 14. A hierarchical Architecture

One of the advantages of hierarchical architecture is its simplicity in decomposing the system into its components. The subsystems at each level can be visualized as a standalone system that can be changed independently of the other subsystems at the same hierarchical level. Furthermore, it is easy to implement and use in object-oriented design. Many object-oriented software designs use this pattern due to its simplicity and easiness.

Decentralized systems constitute an architecture in which every node makes an independent decision that determines the overall system behavior. Unlike centralized architectures, there are no protocol-dependent entities in a decentralized system. Furthermore, decentralized architectures do not have a global clock, so the system will not experience clock synchronization problems. Multiple central units listen to node connections, eliminating a single-point failure problem in centralized architectures.

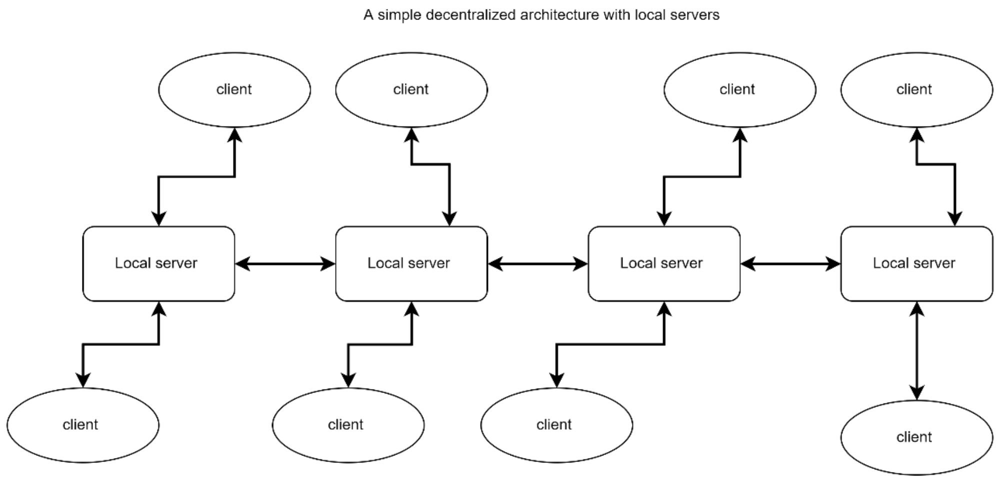

Fig. 15. Decentralized Systems Architecture.

Although decentralized architectures are generally considered better than centralized systems, they are not free of limitations. For example, the coordination problem that can lead to difficulty in achieving collective tasks is one of the limitations of decentralized architectures. Moreover, it is impossible to regulate or control the behavior of a node.

Distributed architecture is an architecture in which isolated components can communicate only through communication networks. However, the members are on the same hierarchical level and can make independent decisions from the higher-level agents. Moreover, the members are connected and maintain peer-to-peer communication. Distributed architecture agents collaborate to accomplish a task. Collaboration can happen via exchanging messages or other mechanisms, such as running a portion of the software system on a different machine.

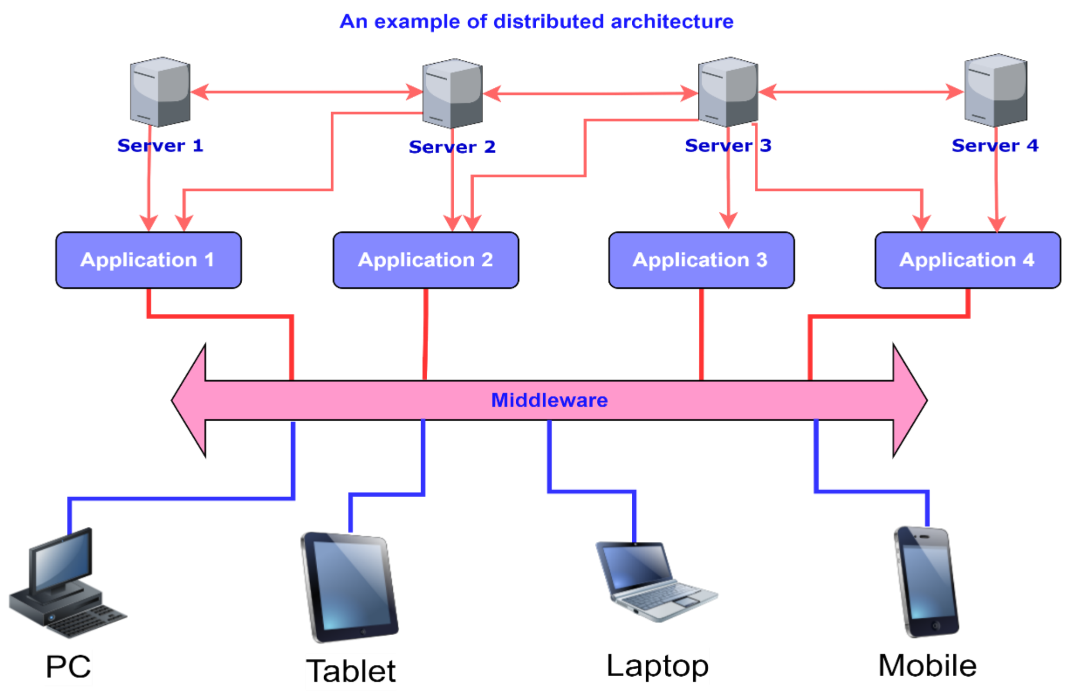

Fig. 16. Distributed Architecture.

Distributed systems have various advantages. First, information processing is fast since it is not confined to a single machine (resources are located across the network on multiple devices). For example, the PC can access application 1 by connecting to Server 1 or 2. Similarly, the laptop can access applications 2, 3, or 4 by linking to Server 3, etc. Second, distributed systems are flexible in using applications from different vendors. Finally, distributed architecture's openness and platform independence make it the best fit for complex systems that run seamlessly on the web.

Heterarchical architecture is similar to hierarchical architecture, with peer-to-peer communication capability between the digital twins [70]. The higher-level platform decides which digital twins communicate with each other using its bundling algorithm.

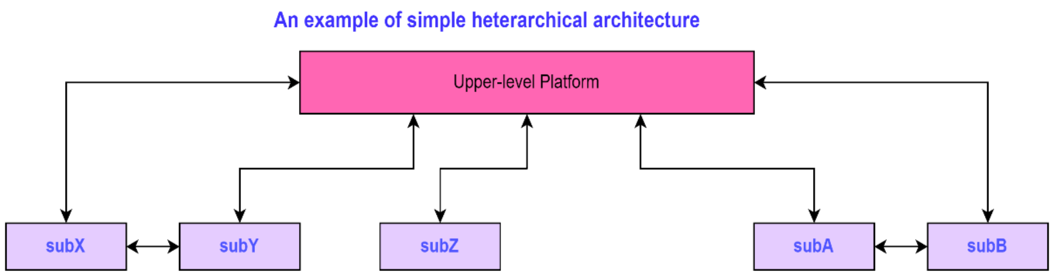

Fig. 17. Heterarchical architecture.

One of the advantages of heterarchical architecture is the availability of horizontal interaction between subsystems (digital twins) that encourages communication efficiency. For example, suppose subsystem subX has a token needed by subsystem subY, and there is a communication channel between the two subsystems. In that case, subX can pass the token directly to subY without sending it back to the upper-level platform. For this reason, this architecture is widely used in industry-level communication control.

3.2. Properties of Agents

Agents are software entities that can perceive, act, and interact with their environment or other agents [40], [71]. A software agent can perform various tasks continuously and autonomously on behalf of a human agent or another software agent. Software agents are either autonomous (act independently) or collaborative (work with other agents). Moreover, an intelligent software agent [54] can recognize the behavioral patterns of a human agent, learn the pattern, and repeat the action independently.

A software agent can be classified based on the "tasks it accomplishes, its control structure, the range and effectiveness of its actions, the range of sensitivity of its senses, or how much internal state they possess" [71]. Hence, there are many different categories of agents. Considering their tasks, one can find at least four types of agents. The first category, intelligent agents, displays artificial intelligence, such as learning, reasoning, and decision-making. The second category of agents, called autonomous agents, can act differently based on the nature of the task they handle or the target objective. Distributed agents, the third category, run on physically different devices in an isolated environment. Finally, multi-agent systems consist of distributed agents that work together collaboratively to achieve a goal that a single agent cannot accomplish.

All agents display one or more collaboration, learning, or autonomy properties. We can classify agents into four broad classes based on their exhibit properties. In addition, these properties extend to other unique properties. I will incorporate the extended properties as we discuss the agents' properties. Furthermore, I will keep the discussion on the properties of agents as straightforward as possible while its descriptiveness stays appropriate.

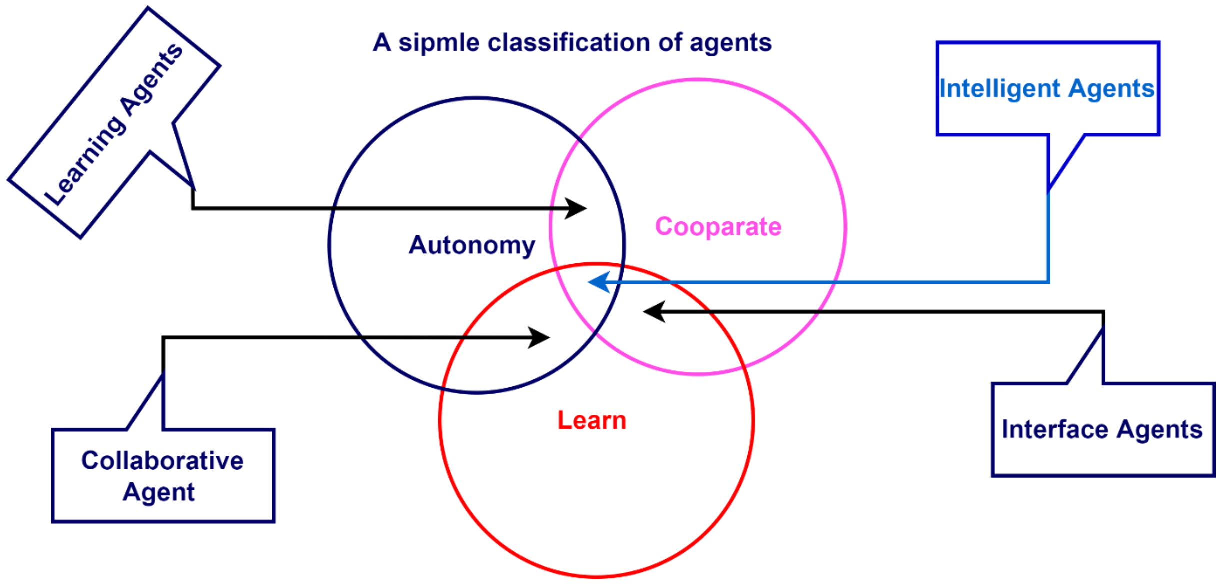

Fig. 18. Agents Classification.

Collaborative agents come into play when the task they perform is beyond the capability of any of the agents in the group. Hence, a collaborative agent is a collection of two or more agents working cooperatively to solve a problem. Collaboration is a common characteristic of distributed multi-agent systems. Agents in a distributed multi-agent system work together by establishing and maintaining an effective communication link.

The goal of *collaborative agents* is to establish a set of agents (multi-agents) that work together to accomplish a complex task. In a collaborative multi-agent setting, agents may perform independent tasks whose collective effect achieves a difficult task. In a cooperative environment, agents can have a specialization (specialization agent), can migrate (mobile agent, sometimes called information agents [71]), or be stationary (static agent).

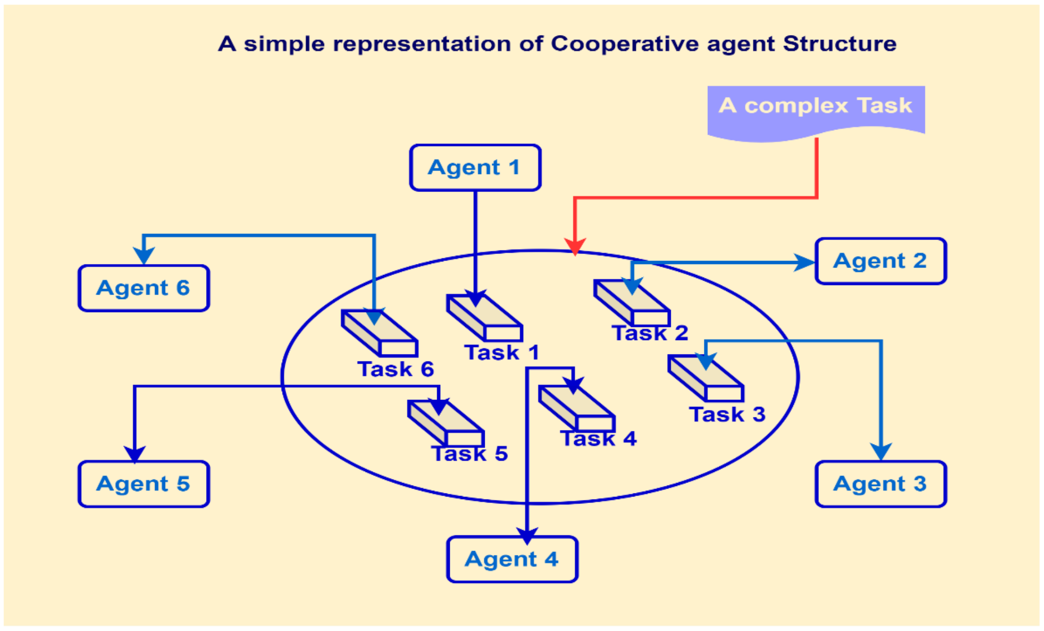

Fig. 19. Cooperative Agent Structure

A *learning agent* is a software entity that possesses learning capabilities. Such an agent can acquire "knowledge" from experience and recall it whenever such knowledge is invoked. Learning agents can start with little or no expertise and become aware of their environment over time.

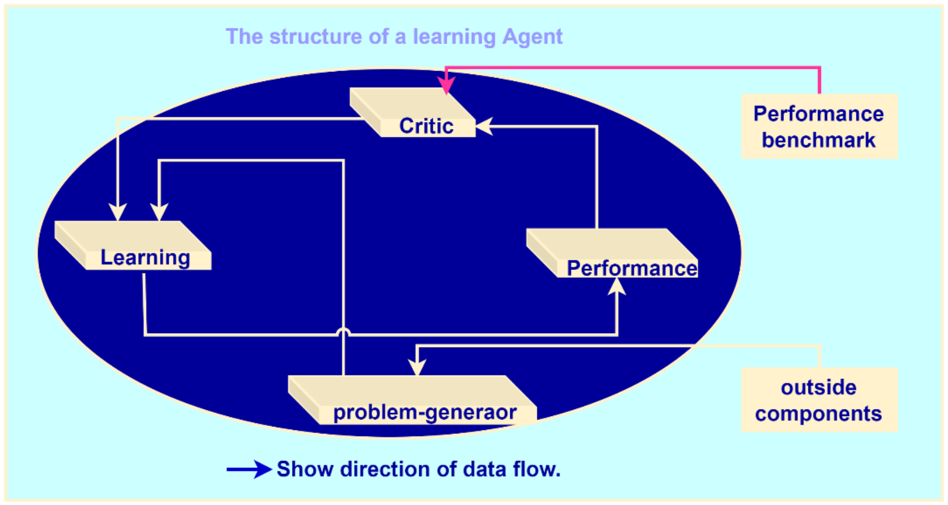

Fig. 20. A Learning Agent's Internal structure.

Learning agents have four major components that engage in the learning process: the critic, the learning, the performance, and the problem-generator. The critical element evaluates the agent's performance against a performance benchmark set, while the learning elements improve the agent's performance over time by taking inputs from the critical element. The performance element makes decisions and takes actions to improve the agent's performance. Finally, the problem-generator component gets information from other parts to suggest actions resulting in a better experience.

An *interface agent* is an agent that performs pieces of a task instead of a human agent or another software agent. They assist their user in accomplishing a task or learning how to use an application. Such an agent can track the actions of their users and perform a better way to achieve the chore. For example, the Microsoft word processor interface helps the user by consistently providing spelling and grammar suggestions to the user.

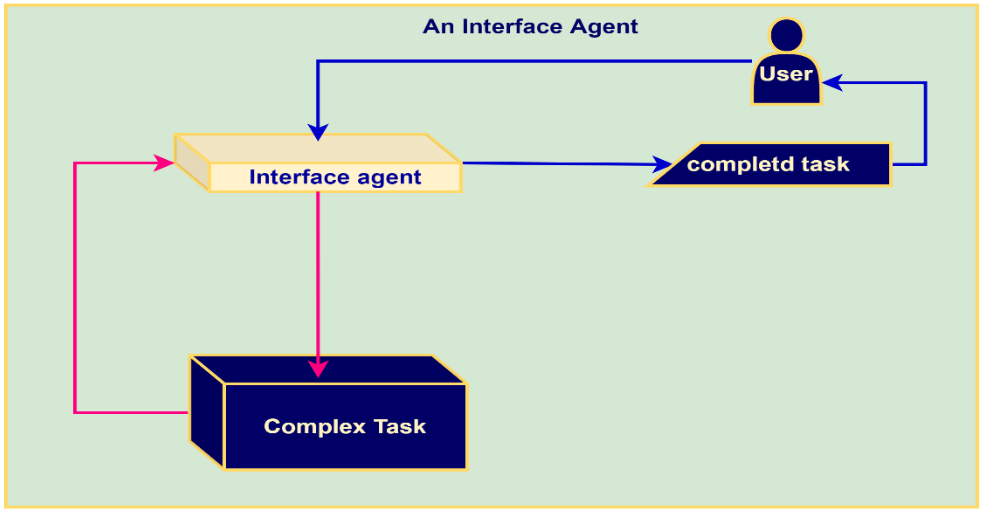

Fig. 21. An interface agent in play.

According to Mahmoud, interface agents provide collaborative assistance to their users in four ways. First, they can observe their users perform and imitate tasks [71]. In other words, the agents can learn an efficient way of doing a job through observation and imitation. Second, software agents can progressively improve at performing a task by receiving user feedback. Positive feedback will encourage the agent to repeat a pattern under similar circumstances, while negative feedback discourages repeating the pattern. Third, software agents can provide better assistance by receiving explicit user instructions. Finally, agents can achieve better experience and performance by appealing to other agents for help.

An *intelligent agent (IA)* is a software and hardware entity that can perceive its environment and act autonomously. Essential characteristics of intelligent agents are their ability to learn, adapt, evolve, and improve. In addition, intelligent agents can react proactively to changes in their environment. For example, a thermostat can sense its room's temperature difference and adjust the room temperature according to the temperature changes in the room environment.

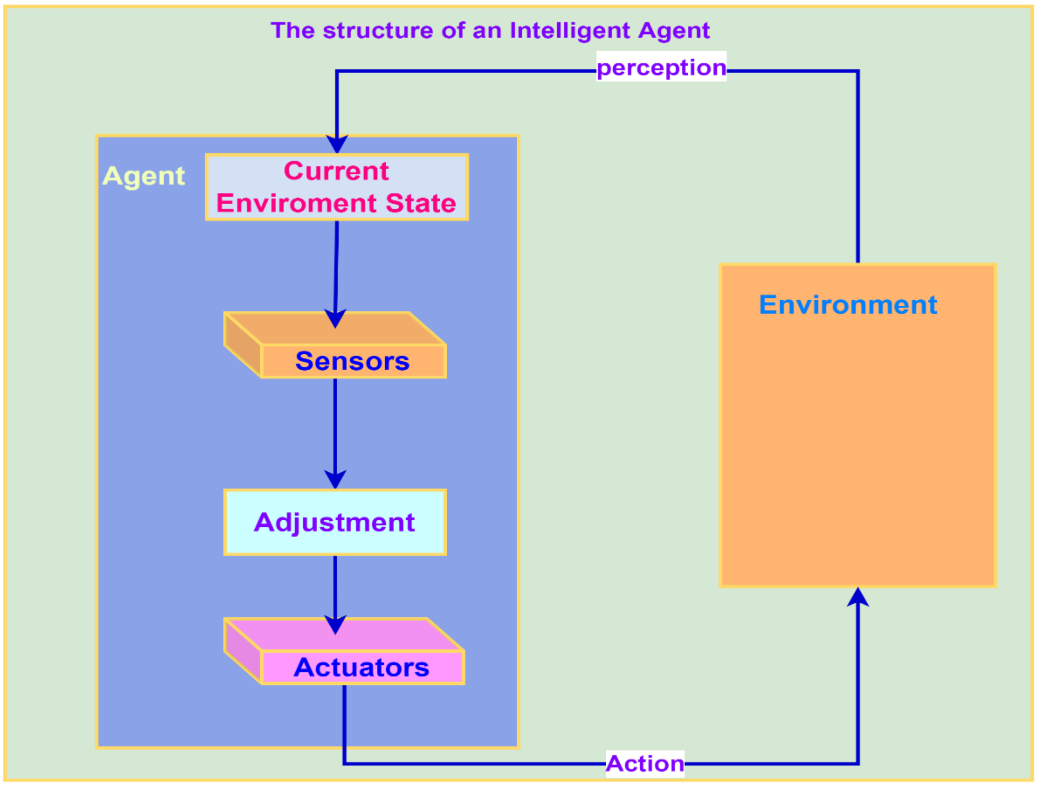

Fig. 22. Organization of a simple reflexive IA.

3.3. Organization of Multi-Agent Architecture

An article by Abbas, *Organization of Multi-Agent Systems: An Overview*, states that MASs are the promise of future software engineering as they appropriately handle adaptive, complex, and evolving software systems [72]. Multi-agent architectures are the natural way to handle the representation of complex systems that act and interact flexibly and autonomously. An architectural model consisting of MASs is a common trend in software with some degree of artificial intelligence due to their flexibility, autonomy, interactivity, etc., as discussed in this chapter's previous subsection.

A multi-agent architecture (MAS) is a collection of autonomous agents located in a dynamic environment with the unique ability to handle the changes in their ecosystem [72]. According to Abbas, there are two major perspectives of MAS: agent-centered (ACMAS) and organization-centered MAS (OCMAS).

An ACMAS focuses on individual agents that take part in the overall architecture. This viewpoint centers on the local behaviors of the agents and proximal interactions, disregarding their global effects. Interactions that lead to the globally required functions emerge due to the impact of bottom-up relations.

In the AC approach, the individual agents are the building blocks of the system's organization. The respective agents exist as the only visible parts of the system's structure building the structure from the bottom-up with no organizational features where constraints are explicitly dependent on the internal behaviors of each constituent. In an ACMAS, each agent is independent and collectively in a shared environment.

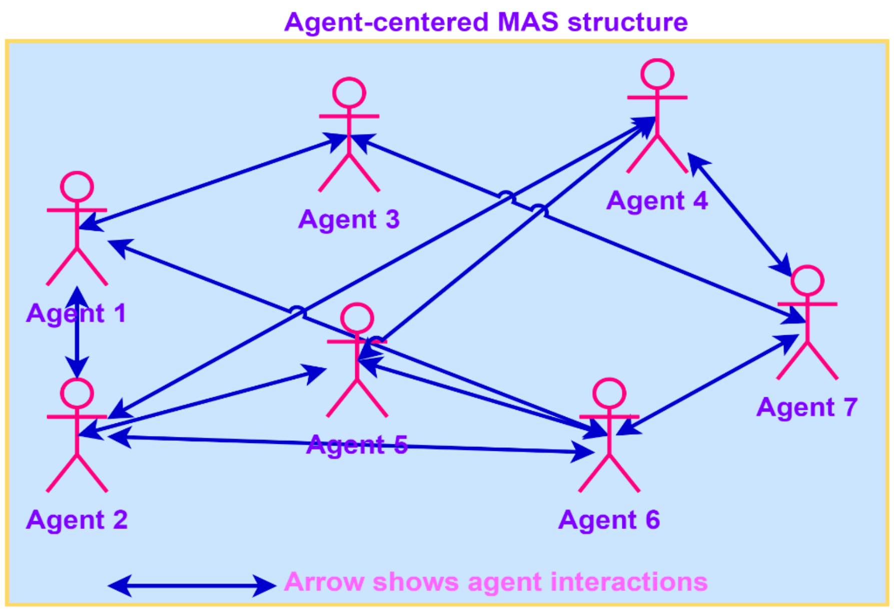

Fig. 23. ACMAS architecture.

An ACMAS is an unpredictable and uncertain architecture "because the whole is greater than the sum of its parts" [72]. Abbas arguably states that the approach can result in undesirable software behaviors and shouldn't be used in developing complex multi-agent systems [72]. Such a design can also affect the system's performance and other quality attributes. Furthermore, Abbas argues that leaving the system's organization accountability to the individual agents in addition to their functional responsibilities creates a dual responsibility which leads to implementation complications and unpredictable behaviors,

An OCMAS (Organizational-centered MAS) is a multi-agent system based on the agents' association. The organization of agents neither describes performed tasks nor enforces rules on the cognitive aspects of the agents. In an OCMAS, the system's structure gets extensive attention, while the individual agent's role is limited to local behaviors only [72]. OCMAS follows the top-bottom organization principle; hence, agents' interaction and coordination follow specific norms specified in the organization abstraction.

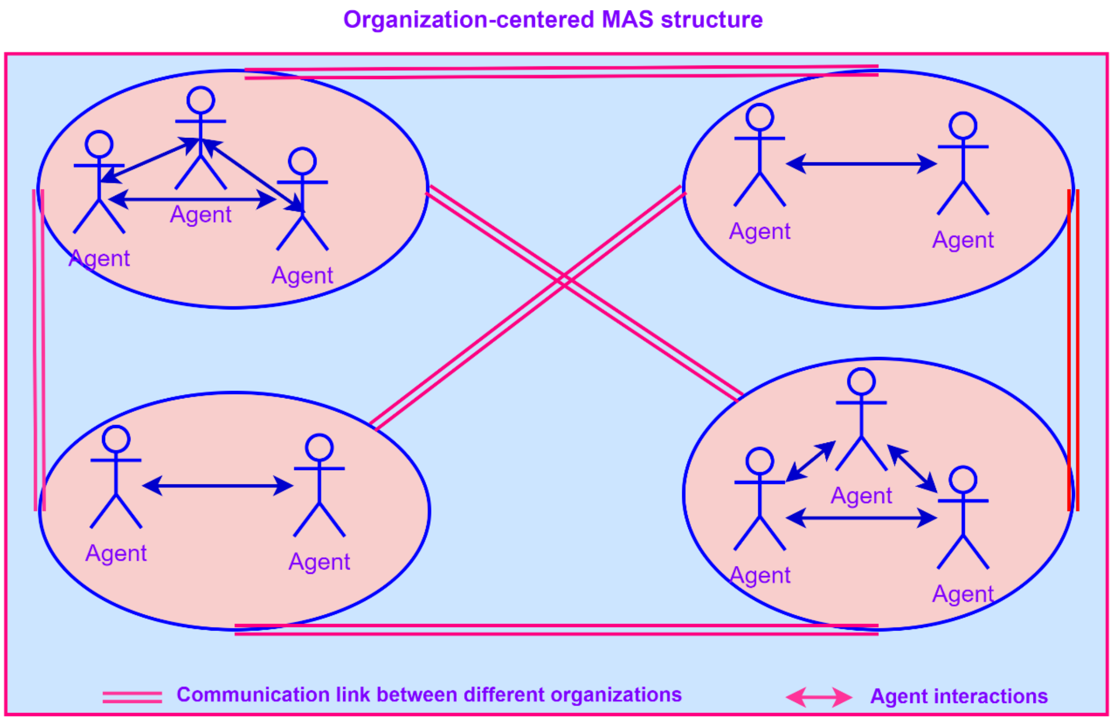
Organization-centered MAS structure
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Agent
Communication link between different organizations
Agent interactions

Fig. 24. OCMAS Architecture.

OCMAS architecture plays a crucial role in simplifying the design of complex software. In addition, the agent's local organization reduces the system's complexity and enhances efficiency. Abbas states that grouping agents that can perform a specific role can provide a means to handle system drawbacks such as complexity, uncertainty, and system dynamism. Therefore, as the software system complexity increases and the agent-based architecture is highly likely, OCMAS becomes the central choice for designing complex systems.

3.4. Architectural Attributes of Multi-Agent Architectures

A multi-agent architecture comprises several entities such as agents, environments, interactions, organizations, and institutions. Hence, MASs display complex attributes related to the different objects, such as self-organization and self-direction, communication, autonomy, openness, infrastructure services, and system robustness. Next, I will present a comprehensive discussion on each MAS property.

3.4.1. Self-organization

The property of agents to spontaneously arrive at hierarchical or other types of order that arise with continuous interaction is known as *Self-organization*. The self-organization process does not need the presence of an external agent to drive it. Instead, it happens between two or more agents that mutually consent to interact and gradually spread the interaction to other agents interested in it. For example, consider agents A and B, who want to achieve task X and consent to interact to accomplish the task. Also, consider agent C and agent D, which will reach task Y from task X. Although all the agents are independent, they have a mutual interest in achieving a goal. Hence, the four agents can organize themselves so that their organization will help them achieve the goal faster and easier.

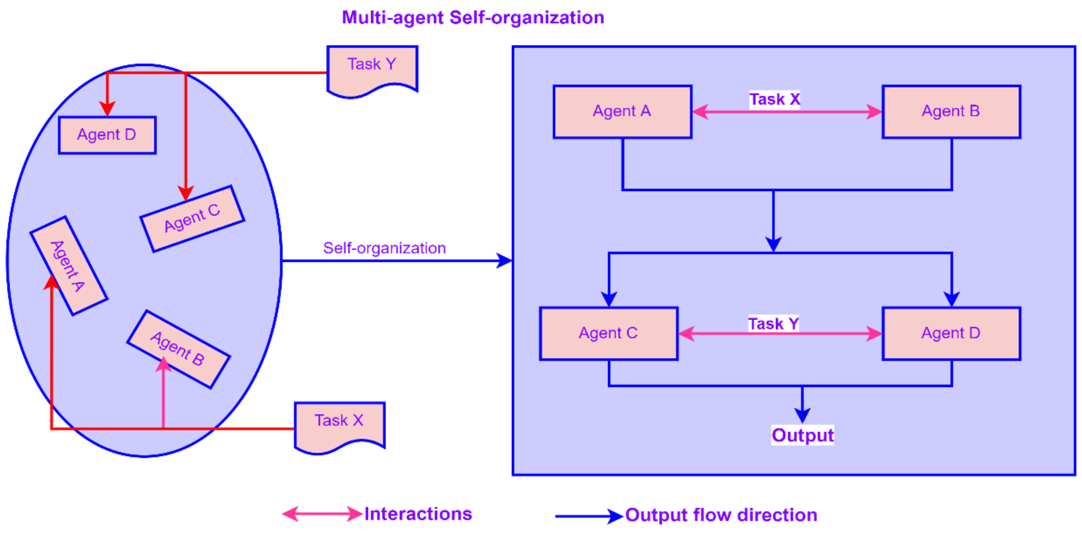

Fig. 25. Multi-agent Self-organization.

MAS system Self-organization is not binding. If an agent does not want to participate or is busy performing a different duty, the agent is free to leave the association, and the system should still function as before. However, if the agent is the only agent in the subsystem that can achieve the task, although it is still free to move away from performing the duty, it can only postpone it instead of abandoning the job.

Self-organization in multi-agent systems can lead to improved performance and high productivity. A group of agents working on a similar task accomplishes the job faster and with greater output. In addition, such agents can learn quickly and become proficient in their duties, leading to expected improvement.

3.4.2. Communication

According to Kim et al., communication is a common phenomenon in MASs [73]. MAS communication can occur as message passing or willful cooperation between two or more agents in the system. Agents in a MAS use a specially designed communication protocol based on the agent architecture, organization, or the tasks they perform cooperatively [74]. Furthermore, Shehory identifies three primary communication attributes that are prevalent in MASs.

When the communication between agents in a MAS depends on client/server protocol, the communication loses its symmetry; only one entity (the server) controls the interaction, and the other entity (the client) only responds upon request. Asymmetric communication among agents is undesirable as it deprives autonomy [75]. Therefore, software engineers must consider implementing *symmetric communication* between agents, such as peer-to-peer communication, although such communication might increase protocol complexity and slow transmission.

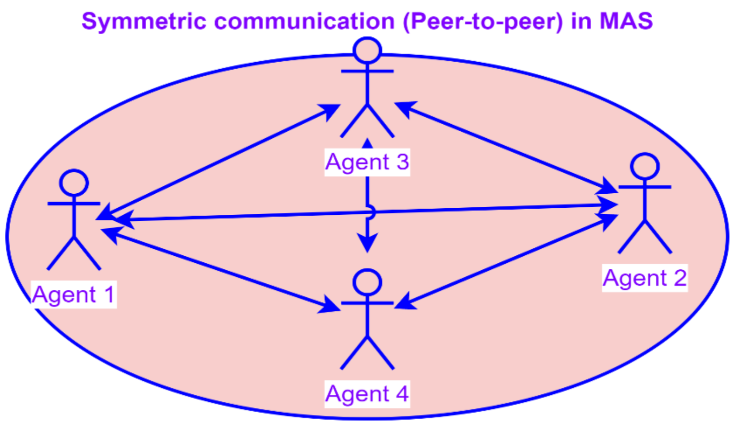

Fig. 26. Symmetric Communication in MAS.

Another type of communication among agents is message passing. If the MAS consists open system, messages should be directed to a recipient (single addressee) or multiple recipients (multicast). However, if the MAS consists of a closed system, broadcasting messages is the most appropriate method to use for communication by message passing.

Agents can also communicate by establishing connections among themselves. A connection-oriented protocol is desirable when agents accomplish a dependent task concurrently.

In such a case, agents must maintain communication and coordination as they perform the ask, and connection-oriented communication is advantageous.

3.4.3. Autonomy

We discussed autonomy several times in this paper. Hence, it is objectionable to waste much time discussing this topic. However, one crucial aspect that must be mentioned is the place of autonomy in finding the best solution to a problem.

3.4.4. Openness

Systems **dynamic openness** refers to the system's ability to allow agents to move freely in and out of the system [75]. Dynamic openness makes the system more flexible, available, and resourceful. Such dynamism is crucial for MASs deployed in environments with high uncertainty. However, in an environment that displays dynamic openness, a robust agent location mechanism is vital to notify the agents about any current planning.

According to Shehory **static openness** refers to the situation in which agents in an environment are relatively fixed (located constantly in the same domain); all agents either maintain the list of all the other agents or are notified that a new agent is added to the environment. Static openness eliminates the need for agent locating mechanism, although it reduces system flexibility, robustness, and dynamicity.

Off-line openness is the most restrictive type of openness. It allows the addition of new agents only off-line, i.e., the system stays off-line while adding the new agent and need to restart after the process is completed. Off-line openness restricts the system's flexibility, availability, and dynamicity. Hence, such a MAS must be implemented if the problem it solves is well-defined and predictable.

3.4.5. Robustness

A desirable property of MASs is their robustness. MASs robustness emerges from their distribution of execution, which increases the system's overall performance and tolerance to failure. In addition, MASs are highly tolerant of malfunctions due to the existence of replicas, an agent with the same or similar capabilities. If an agent is unavailable due to multifunction, one of the replicas can overtake the task and accomplish it. Hence, failures are hardly noticed in MASs, leading to high availability.

3.5. Multi-Tiered Multi-Agent Systems

A multi-tiered multi-agent Architecture (MTMAA) is a structure with multiple MASs organized so that agents organize themselves into different tiers to perform a particular task. At the same time, MASs maintain communication and coordination between the tiered organization. In the article by Shehory, this type of organization is called federated MAS. Federated organization is an emerging technology for empowering applications operating on extensive data [75] maintained by the MASs.

Shehory describes federated agent organizations as ensuring interoperability and coordination through what are known as facilitators. Facilitators are interfaces through which agents communicate or interact. Hence, agents will connect with their local facilitators to talk to each other, and the facilitators will maintain a link among themselves. In a federated organization, agents concede to abandoning some of their autonomy to their local facilitator; however, agents do not address their message to a particular agent as this is the task of the facilitators.

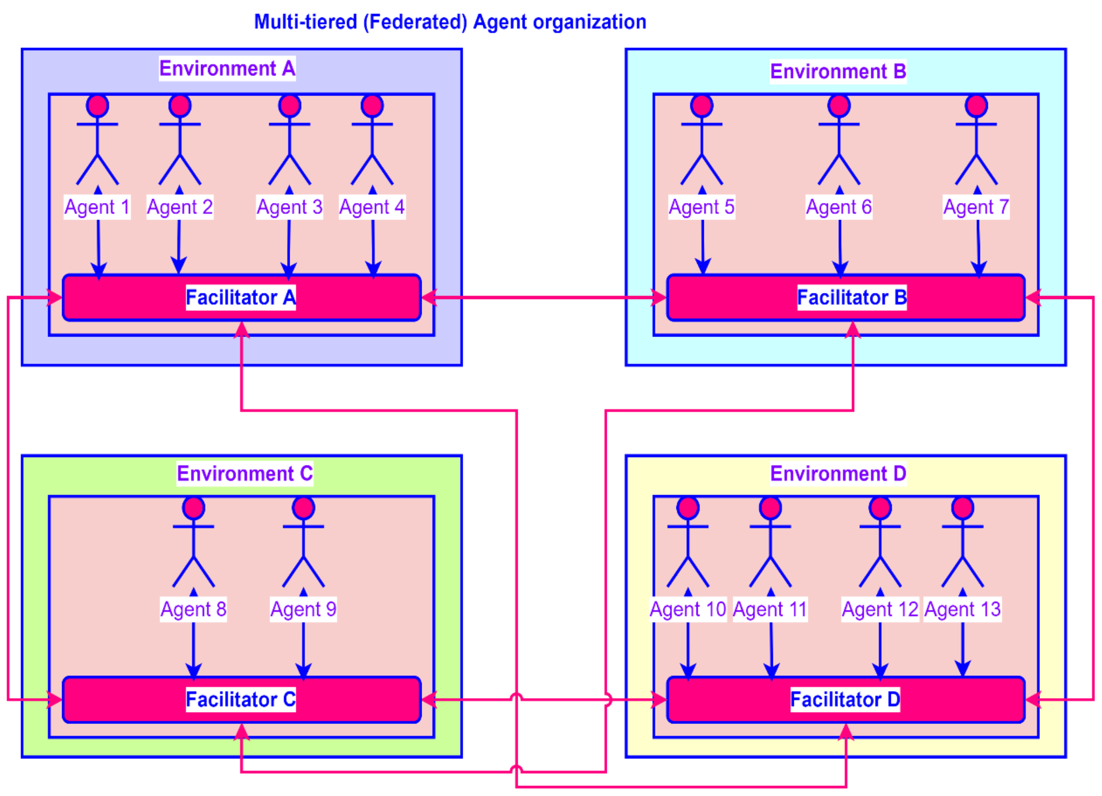

Fig. 27. Federated MAS organization.

Although agents can communicate only through their local facilitators in a federated organization (fig. 27), they still preserve the freedom to dynamically connect or disconnect from their facilitators. However, agents in a federated organization must concede to send or receive services only through their facilitators; the facilitators must handle a service request agents post to it, whether the message is directed to an agent within the MAS or an agent in another MAS.

Facilitators do not simply act as a transmission medium, passing messages to and from agents. They can also perform complex operations involving message decomposition, bundling, translation, or other sophisticated functions. Hence, they can be viewed as intermediary agents. Shehory suggests the middle agent view of facilitators, as suggested by many researchers.

3.6. Applying Multi-Agent Architectures to Requirements Specifications

We have discussed multi-agent architecture, what agents are, their internal structure, their various properties, the different categories of multi-agents, their organization, and interactions; it is time to apply what we know to the proposed educational system. Hence, this section will present the various architectural characteristics of the proposed educational system (chapter 1) and apply multi-agent architecture to them.

3.6.1. The Architectural Model of a MAS in Educational System

The proposed architecture utilizes the Multi-Tiered Multi-Agent approach (MAMTA), where a MAS represents each architectural property. Moreover, the MASs have the corresponding User Interface (UI) and a facilitator coordinating communication with other MASs.

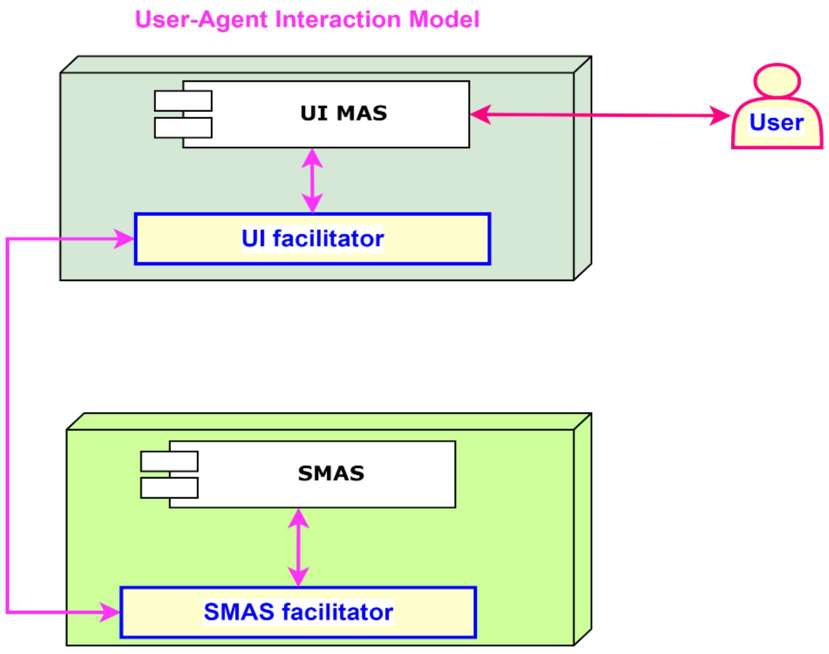

Fig. 28. A general representation of the proposed architecture.

The UI MAS contains all the agents that handle the different interactions of the user with the system. User interactions include requesting transcripts, viewing course information, taking an assessment, checking progress reports, etc. The UI MAS will accept the user requests and direct them to its facilitator, where the requests are organized, translated, or decomposed to be directed to the appropriate SMAS facilitator.

Fig. 28 shows a typical interaction between a user and a MAS in an adaptive multi-agent educational system. First, the user will interact with the system through the user interface. The UI MAS is an environmental element (top-level interface) that will coordinate the interaction between the user, the environment, and the system (agents). Next, the UI MAS will pass the message to the interaction control interface, UI Facilitator (UIF), to coordinate and determine the specialized target agent and send it to the specialist. Finally, the specialist will determine the appropriate action and take action and return the action result to its facilitator. Fig. 29 shows the general structure of the proposed multi-agent architecture. As indicated in the architecture, the Specialization MAS (SMAS) represents the MASs that characterize the educational system, such as Adaptive MAS(AMAS), Interactive MAS (IMAS), Autonomy MAS (AuMAS), Decentralization MAS (DMAS), Continuous and Dynamic Improvement MAS (CDIMAS), and Content Filtering MAS(CFMAS). The following subsection offers a more detailed description of each MAS, *Documenting the architecture*.

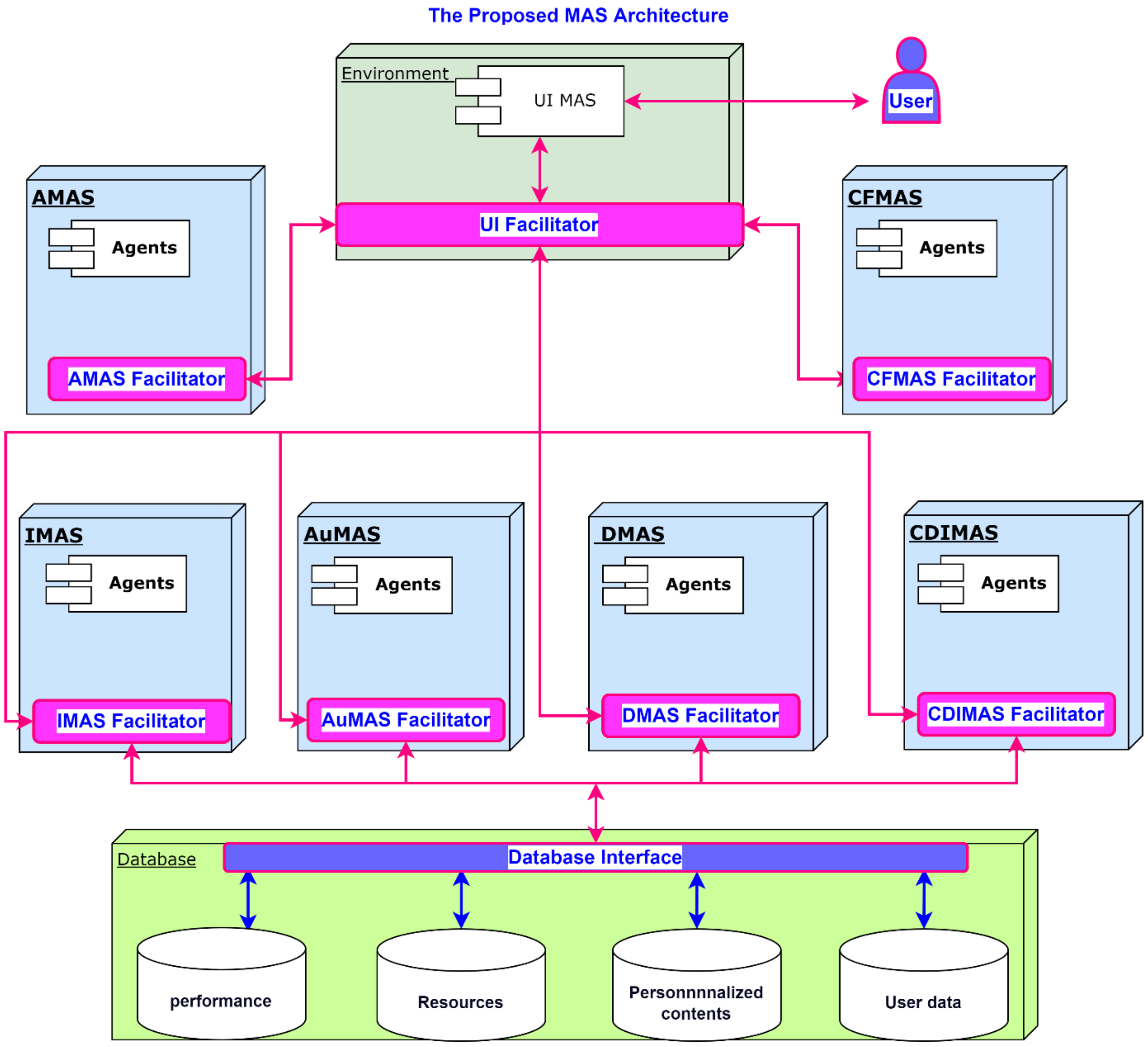

Fig. 29. Proposed MAS organization.

Chapter 4: Documenting the Architecture

The term documenting architecture should sound familiar to a developer, an architect, or an architecture student. However, the purpose behind architectural documentation and the intrinsic information it should carry with itself might not be apparent even to its producer. Bass states that quality architectural documentation is better than answering a hundred questions about architecture. Hence, an architectural document is a detailed and self-explanatory document aimed at all stakeholders, irrespective of their understanding of the underlying principles.

In his book, *Software architecture in Practice, 4th ed.*, Base mentioned four fundamental uses of documentation [52].

1. *"Architectural documentation serves as a means of education."* Therefore, the document should be clear enough to be used by different users with different levels of understanding. For example, a new employee should find it helpful to understand how to accomplish a development task related to the architecture.
2. *"Architectural documentation serves as the primary vehicle for communication among stakeholders."* This is to say that the architectural documentation should contain enough information for everyone who may be interested in it. In addition, the document should be clear enough and well-detailed so that it is self-explanatory to everyone, including the architect.
3. *"Architectural documentation serves as the basis for system analysis and construction."* Base described the above sentence: "Architecture tells implementers which modules to implement and how those modules are wired together." Design and implementation emerge

from architectural documentation. A good architecture document is a map that shows the path and destination of a software development journey.

4. *"Architectural documentation serves as the basis for forensics when an incident occurs."* For example, in case of an incident such as a system failure, it is necessary to investigate the root cause of the problem to fix it correctly. Therefore, architectural documentation is an indispensable first-hand source for incident investigation since it shows the detailed control flow, component descriptions, and system protocols.

Architectural documentation presents different views of architecture. "A view is a representation of a set of system elements and relations among them" [52]. Furthermore, Bass states that documenting architecture involves documenting the relevant views and adding information that applies to more than one view. The subsequent sections of this chapter will present the proposed architecture's relevant view and add information to the architecture.

4.1. The User Interface MAS (UIMAS)

The user interface is a point of interaction and communication (information exchange) between the system and the user. The interaction or communication involves a human with a computer or a computer with another computer. The user interface environment can be cooperative, meaning the user will play its part and wait for the computer to respond, or the computer will wait for data from the user to perform a task. The user and the computer may compete for time in a competitive environment such as a game environment. Fig. 30 shows the typical User Interface MAS's overall organization and constituent components (UIMAS).

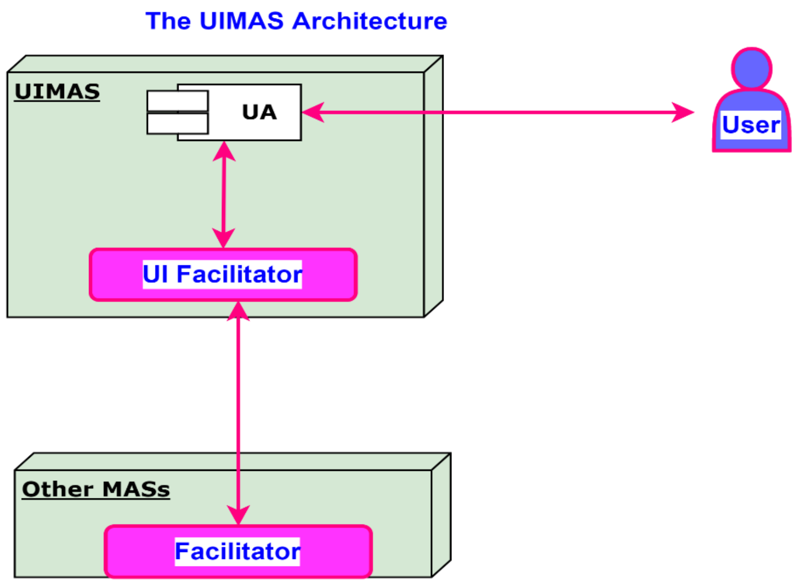

Fig. 30. UIMAS organization.

The UIMAS consists of the User, the User Agent (UA), and the User Facilitator (UIF).

4.1.1. The User

The user represents the human entity that interacts with the system. Users' interactions involve the sending and receiving of information. For example, the users send data such as a request and receive the processed response. Users must type in a text, click on a link or button, or request a connection to trigger communication.

Users' interactions can be semiautomated or static. In a semiautomated interaction, the user will partially trigger the communication (for example, the initial trigger), and the system knows the chain of commands it needs to execute. However, in a static interaction, the user must continuously interact with the system to keep it running.

4.1.2. The User Agent (UA)

The user agent is the only agent in the UIMAS organization. The user agent provides an interface that helps users interact with the system. There are different types of UIs, such as Graphic User Interface (GUI), Command Line Interface (CLI), and Menu-driven Interface (MDI). In addition, some modern electronic devices can provide more Interfaces, such as Touch User Interface (TUI), Voice User Interface (VUI), and other types of interfaces.

The UA is a module that receives communication from the human user and passes them to the UI facilitator. The UA organizes the messages sent by the user, determines the validity of the interaction, and makes the necessary changes to the communications to assist the user with their decisions. Moreover, the UA can anticipate the likely output, learn communication patterns, or display other properties of agents.

4.1.3. User Interface Facilitator (UIF)

The UIF is an intelligent facilitator that can take the UA's inputs and pass them to the appropriate MAS facilitator. Facilitators can perform complex tasks described in the next section. All MASs must have one facilitator that can handle agent-to-agent communication. Since all facilitators have the same functionality, I will discuss their functions in one place under the subtopic *The Facilitator Interface*.

4.2. The Facilitator Interface (FI)

In the real-world context, a facilitator is someone or something that facilitates something. For example, for a group of persons working together or conducting a meeting, the role of the facilitator is to plan, guide, and manage the group event. In addition, facilitators are also medium of information exchange. For instance, if a participant wants to ask a quest, ask a question, present a view, or object to an idea, they do it through the facilitator.

Similarly, a software facilitator provides a medium for agents' interaction. A facilitator is a software program that coordinates information exchange among agents. Facilitators do not simply transmit agents' messages; they coordinate agents' communication, translate messages as necessary, and decompose or bundle messages. Furthermore, facilitators provide a layer of reliable message passing (acts as a middleware for agent-to-agent communication) and initialize and monitor the execution of its agents.

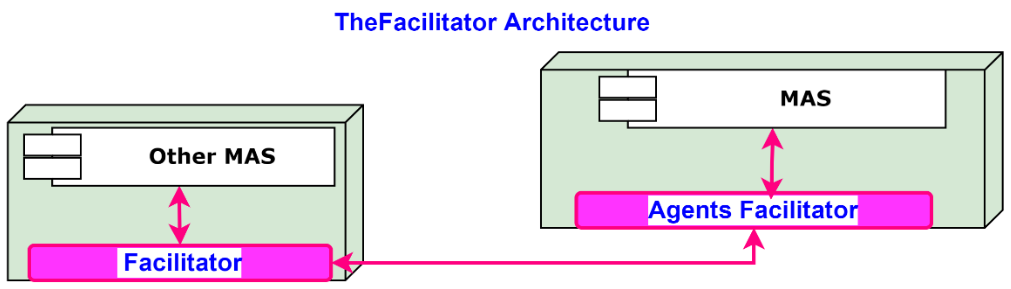

Fig. 31. Facilitator.

Facilitators can also communicate and coordinate with each other in addition to their support for agent-to-agent communication. One of the advantages of facilitators is that agents do not need to know each other to communicate. For example, an agent posts its message to its local facilitator. The local facilitator will process the message and forward it to another local or remote facilitator, where the agent is federated. The receiving facilitator identifies the agent that can handle it and deliver it to the agent.

4.3. The Adaptative MAS Architecture (AMAS)

The adaptive MAS hosts a collection of agents that accomplish the tasks required to realize software adaptability, as specified in chapter one. However, the number and type of agents involved in the adaptation process are unclear or difficult to tell as it partially depends on the level and type of adaptation one wants to achieve. Hence, deciding the number and type of agents required to achieve adaptation is challenging.

Therefore, the paper will present several agents the author thinks can make potentially suitable candidates. Then, the reader can decide which are good candidates and which are not during implementation. Furthermore, relocating agents to different MASs should be considered if that can simplify the implementation task.

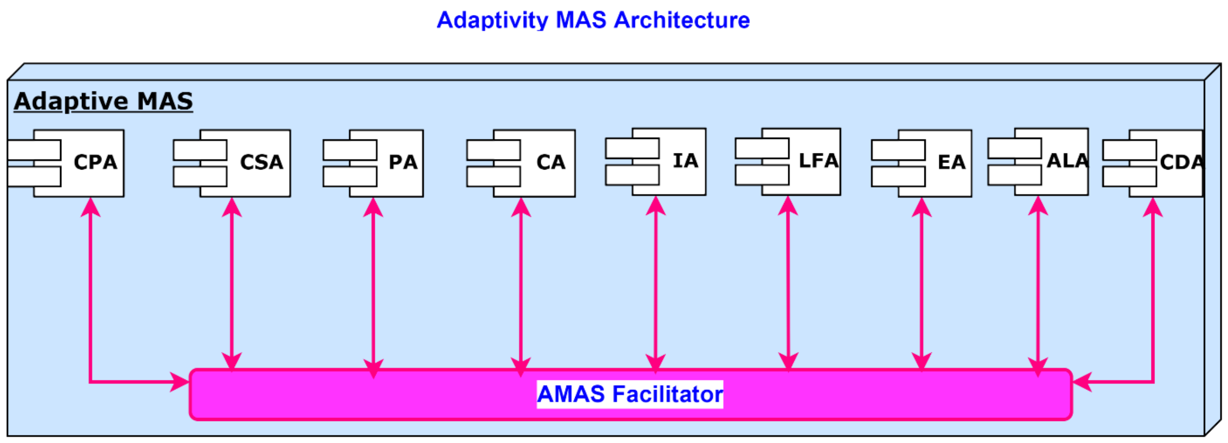

Fig. 32. AMAS Organization.

The AMAS constitutes nine agents required to make the system adaptive and AMAS facilitator. The following is the list of all the nine agents and their short descriptions.

4.3.1. Course Presentation Agent (CPA)

The course presentation agent (CPA) performs all the course presentation and modulation duties. This agent acts in place of a teacher; hence, a virtual teacher. It presents the contents progressively as determined by the Content Sequencing Agent (CSA) and Content Selection (Assortment) Agent (CAA). The CPA should vary its presentation mode based on the user preference according to the suggestion it receives from the Cognitive Agent (CA). A difficult task of the CPA is to drive real-time interaction with the user that requires some level of cognition (which might be handled by the cognitive agent (CA) if it does not create an agent dependency problem).

4.3.2. Content Assortment (selection) Agent (CAA)

The Content Selection Agent (CAA) will apply content sequencing rules and ML paradigms to dynamically and intelligently customize the learning contents. As discussed above, the CAA works in alliance with the CPA. Therefore, the CAA gets the contents from the content sequencing agent (CSA) and, if content customization (differentiation) is required, consults with the personalization (differentiation) agent (DA) to fit them to learners' needs.

The state education board or the district office defines the educational policy used in content sequencing rules. Finally, the agent selects the contents dynamically by applying machine learning algorithms.

4.3.3. Performance Agent (PA)

The Performance Agent (PA) will track the learner's performance and update the progress report in the database. This agent’s task is not as complex as the two previous agents. The agent uses performance measurement matrices, formal rules, and activities the user must complete. For example, an assessment is an activity that the user must complete. A reading assignment can be another activity that the user needs to meet.

4.3.4. Cognitive Agent (CA)

The Cognitive Agent is one of the most complex agents in adaptive MAS. Due to its complexity, it is not yet clear how to fully implement it; many researchers in artificial intelligence are still working toward achieving the full realization of CAs [Kadam]. However, it is possible to implement rule-based cognitive agents that use ML paradigms, such as reinforcement learning with limited capabilities.

Cognitive agents (CAs) use ML algorism to interact with the user and provide suggestions, tips, and tricks to assist the learning process. For example, suppose a learner is working on a mathematical problem and cannot complete the solution. In this case, the CA can interfere and assist the learner by providing tips about the part of the solution that went wrong. Furthermore, suppose more assistance is necessary to solve the problem. In that case, the CA can bring up a problem directly related to the section of the previous solution that the learner is struggling with and demonstrate a solution to help the student solve the original problem.

4.3.5. Preference (Inclination) Agent (IA)

The preference (Inclination) agent (IA) uses performance parameters and the user past learning experience to adjust content presentation mechanisms. The IA agent assists the CPA in deciding on the mode of presentation choice. The IA agent looks at the performance data of the learner on a particular topic and the user's past learning preferences [55] to decide the mode of presentation for the current subject and announces the choice so that the CPA can use it.

4.3.6. Learning Feedback Agent (LFA)

The LFA performs a unique and straightforward task. This agent will provide consistent and continuous feedback to the learner on the learning outcomes and assessment results. In addition, the LFA updates the students' progress status, sends progress reports and suggestions to the student, and updates the student's advisor about the student's progress. The LFA agent and the evaluation agent (EA) will access the students' performance data from the PA.

4.3.7. Evaluation Agent (EA)

The Evaluation Agent (EA) will select and administer assessments based on the learning goals and determine the learner's learning achievement. The EA will perform the duty of an assessment office that administers examinations. In addition, the agent will constantly update the assessment data that LFA needs.

4.3.8. Advanced Learner Agent (ALA)

The Advanced Learner Agent (ALA) evaluates the learner's learning level. The agent uses the student's performance data and a unique performance matrix [36] [37] established by the school to select the candidate. For example, suppose the student's performance fits the advanced-level learner (ALL) category. In that case, the agent will administer a special assessment to check the student's knowledge level and advise the CAA to make the needed adjustment. The CAA will then refer the student to the content differentiation agent (CDA) to produce differentiated content that meets the student's learning needs.

4.3.9. Personalization (Content Differentiation) Agent (CDA)

The Content Differentiation Agent (CDA) differentiates the content to meet the learner's unique needs. For example, suppose a student is a slow learner. In that case, the CDA agent will decide the contents, content sequence, and corresponding assessments to meet the student's learning needs based on the student's performance data and preferences. On the other hand, if the student is identified as an advanced learner, the agent will select advanced topics in the learning domain and relevant assessments to meet the student's learning needs.

4.4. The Interactive MAS (IMAS)Architecture

The interactive MAS (IMAS) consists of various agents working to make the system interactive. The primary purpose of IMAS is to equip the system to meet behavioral measurements that many adaptive learning systems lack. For example, according to an online article by Pugliese, "Adaptive Learning Systems: Surviving the Storm," adaptive systems cannot engage learners in group activities because they lack communication analysis of online communities.

As discussed in chapter one, a system's interactivity involves several elements. The first and most apparent interactive aspect of a system is the look and feel of the system. The system's appearance is one of the elements that engage the users. A user is more likely to stay and interact with a presentable system. A user-friendly aspect is another interactive element of a system. For example, a system with some level of intelligence that simplifies the user's task (IntelliSense) is more likely to engage its users. Other elements include the system's response time, methods to display results, and essential feedback.

However, the above factors are only the aesthetic components of interactivity. The learning system environment needs functional interactivity that involves the psychological and cognitive aspects of the user. Therefore, this section aims to present and discuss the higher-level interactive elements of the system known as interactivity's functional attributes. The Interactive MAS architecture constitutes the following agents.

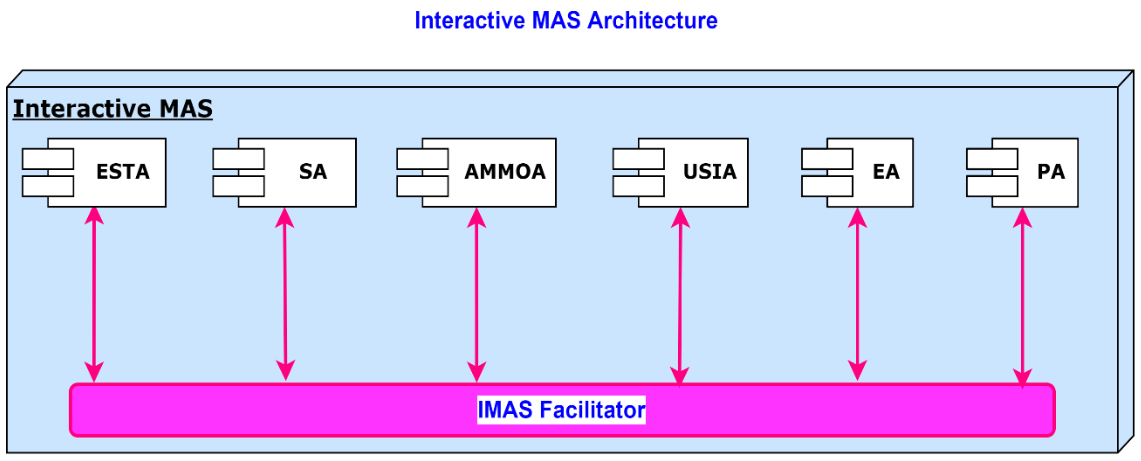

Fig. 33. IMAS Organization.

4.4.1. Emerging Skill Tracker Agent (ESTA)

The Emerging Skill Tracker Agent (ESTA) finds new learning skills and nurtures them until the skill is internalized by the user (student). For example, suppose skill S is among the skills in an educational domain D that the student has never learned (new talent). As the student progresses in learning domain D, he begins to learn the new skill S. Once the student starts learning the skill S, it is no more a new skill but an emerging skill. The ESTA identifies the emerging skill from the student's progress data and sets a plan to cultivate it.

There are a lot of ways to nurture emerging skills. One way is to direct the student to the resources that help develop talent. Another effective method to cultivate expertise is to provide tutoring services on emerging skill sets. In any case, the agent should support the student with exceptional assistance until the student demonstrates mastery of the basic concepts at which the CPA can do the remaining part of the teaching.

4.4.2. Socialization Agent (SA)

The Socialization Agent (SA) provides services such as connecting the user with other users in the same learning category. This agent provides the platform needed for socialization services. The agent will use the students' demographic data, such as age, grade level, learning preference, [55] socio-cultural background, and other socialization parameters, to group students into the same category. Once the agent identifies the group, it provides friend suggestion services, sends them to the individual students and the public profiles of each student in the category, and leaves the choice to the student.

Furthermore, the socialization agent should interact with the student naturally. The SA uses AI to talk to students with natural (human-like) communication. Although there has been a significant achievement so far, it is not yet fully known how to implement ML algorithms that can successfully achieve the theory of mind.

4.4.3. Alternative Multimedia Option Agent (AMMOA)

The Alternate Multimedia Option Agent (AMMOA) is another less sophisticated agent with relatively simple tasks. The primary function of the AMMOA is to provide the user with alternative content presentation methods according to the user's preference. First, the agent will use the learning preference agent's (IA) data to infer the students’ learning choices. Then, the agent suggests learning preferences for the student that the CPA agent uses for content delivery.

4.4.4. User-System Interaction Agent (USIA)

The User-system Interaction Agent (USIA) tracks how the user interacts with the system to use the data in preference decisions and presentation selection. The EA uses the data collected by the USIA for decision-making.

4.4.5. Engagement Agent (EA)

The EA tracks the engagement pattern of the user on a presentation method, course content, etc., and evokes the presentation agent to change or adapt content. For example, the agent will use the engagement matrix [38] to determine the student's engagement level with a specific presentation mode and collect the corresponding data to make learning preference decisions.

4.4.6. Pedagogical Agent (PA)

The Pedagogical Agent (PA) guides the user through the content making the presentation dynamic. The agent applies specific rules or policies to decide the scope and sequence of the knowledge domain. The agent also considers the students’ demographic data and other parameters while making content and sequence decisions [20].

4.5. Autonomy MAS (AuMAS)

The system’s autonomy MAS has more to do with its states than its functions. The agents in AuMAS oversee the overall system health improvements, keep system logs, and provide the system with preservability and durability. If there is a change in the system's state, one or more of the AuMASs will take action to restore the system to its healthy condition.

The AuMASs guarantee a system independent of its environment and human intervention. The system must sense its environment and keep track of its healthy state. In addition, it must perceive and understand unrelated data sources and automatically restore to its immediate healthy state. System restoration should be preceded by planning not to corrupt data or reverse completed processes. Hence, autonomous MASs need the following agents to carry out their intended duties successfully.

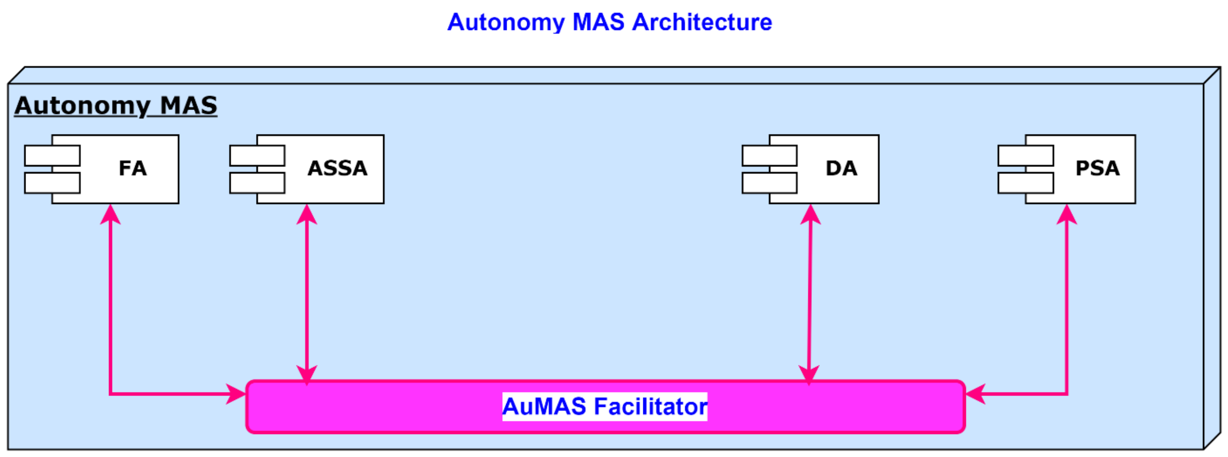

Fig. 34. AuMAS organization.

4.5.1. Diagnostic Agent (DA)

The Diagnostic Agent (DA) has system error detection and correction modules. If the system encounters an error or state change, the DA agent will detect the error, identify its cause, record it, and take corrective action. The DA agent gets a state change or system error report from AuMASF, which comes from the FA. After the DA obtains the change that needs attention, it will run a system diagnosis to identify the cause of the problem. Finally, the DA will request the previous healthy system state from AuMASF reported by ASSA and restore the system accordingly.

4.5.2. Feedback Agent (FA)

The Feedback Agent (FA) will provide continuous feedback on the overall system state change. The feedback obtained from FA will trigger the diagnostic agent (DA) to run error detection, identify and log the cause of the error, and take corrective action to restore the system to its previous healthy state.

Hence, if there is a change in the system's state, the FA will send the change to the AuMAS facilitator. Finally, the facilitator will decide what system adjustments are required and let the concerned agent(s) know to take corrective action.

4.5.3. Active System State Agent (ASSA)

The Active System State Agent (ASSA) will keep the current healthy system state log used for system repair in case of system failure. The task of the ASSA is to keep track of the healthy system state before the FA reports a change. If the FA senses a difference in the system state environment and notifies AuMASF, the ASSA will need to register the system state just before the shift in the system's healthy state.

4.5.4. Process State Agent (PSA)

The Process State Agent (PSA) keeps the collection of active processes and their state. In case of system failure or a system multifunction and system restoration, the PSA report is needed to identify processes that run to completion or need to restart. The PAS is hence, needed for systems consistency and dependability. PSA will maintain completed states so the user will not lose all completed processes.

4.6. Decentralization MAS (DMAS)

The Decentralization MAS (DMAS) is responsible for localizing control and decision to each system node according to the need of the independent educational institutions. Because educational institutions are separate entities, they must be able to use the system autonomously. Moreover, since each student's learning pace and preference are distinct, the system must create a separate learning space (access node) for the individual students. Therefore, decentralization plays an indispensable role at organizational and individual student levels.

One of the benefits of decentralization is enhanced system security. For example, resources stored at a local node will stay local unless the owner grants access permission. Moreover, it eases policy implementation at the organization level since the local node behaves as if segregated from the rest of the nodes.

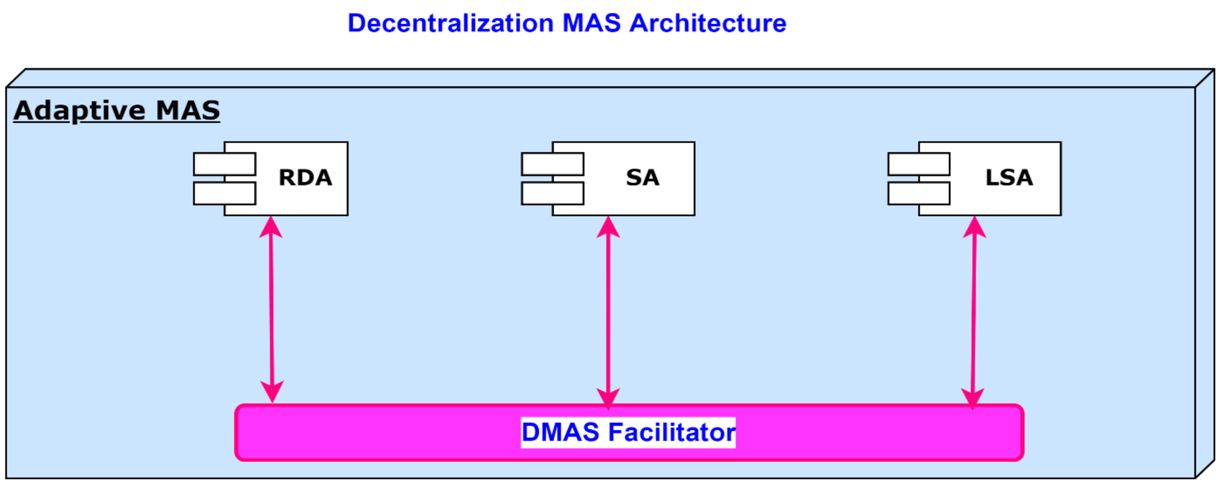

Fig. 35. DMAS organization.

The Decentralization MAS (DMAS) consists of the following agents:

4.6.1. Resource Dispatching Agent (RDA)

The Resource Dispatching Agent (RDA) uses local institution policies to relocate resources between nodes. For example, suppose institution X contains subnodes A, B, and C under its network domain. If the institution allows shared access to resources among the subnodes, the RDA can move shared resources among those subnodes.

The resource dispatching agent (RDA) minimizes the required storage space by keeping the resource at its current location while providing access.

4.6.2. Sequencing Agent (SA)

The Sequencing Agent (SA) is responsible for maintaining the sequence of learning goals defined by the educational policy. Different institutions may have distinct educational policies that slightly affect the chronological order of learning. Hence, the SA should be able to accommodate variations in sequencing by following the guidelines set in the policy document.

4.6.3. Learning Space Agent (LSA)

The Learning Space Agent (LSA) creates a separate learning space for each learner. The LSA must segregate a learner (user) from all other learners because each student's learning style, speed, and field of study are unique. When a new student creates a personal learning profile, the LSA will initialize the student’s learning space by generating all the MASs, including its MASS. Once the LSA creates the learning space, students can access the course registration system and all related learning materials.

4.7. Continuous and Dynamic Improvement MAS (CDIMAS)

Continuous and Dynamic Improvement (CDI) is a process where a system or its subpart attempts to indefinitely and dynamically improve over time. Hence, the CDI MAS implements a method to enhance a given task over time. CDI MAS aims to achieve a quality target that can scale out incrementally. For example, if target 1, target 2, and target 3 are the proposed quality targets, the system attempts to reach target 1 before targets 2 and 3. Once it hits target 1, the system will aim to hit target 2. Finally, the system seeks to achieve target 3, which may extend to target 4 as needed.

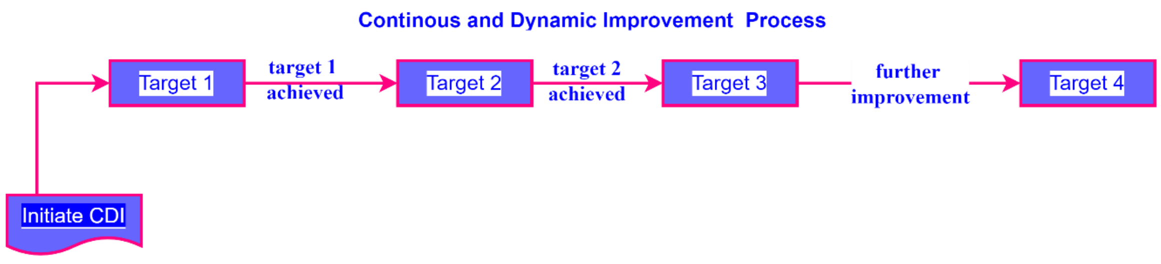

Fig. 36. Continuous and Dynamic Improvement Process.

CDIMAS must carry out the following four steps to implement the above progressive process.

1. *Plan* – The MAS must identify the necessary changes at the planning stage. For instance, the MAS should discover that target 1 is the required change at its initial CDI. Alternatively, if target 1 was already satisfied, MAS must discover target 2 as the necessary change.
2. *Implement* – carry out the change on a small scale. The MAS must initiate changes on a small portion of the identified data and enforce the sought change. Instead of

changing all the data simultaneously, which can lead to data loss or corruption in case the implemented change fails, the MAS must change a small portion of the data and wait for test results.

3. *Check* – use the new data and analyze the change results to determine if the changes meet the expected results. The new results can also be compared to the existing results to see any improvement.
4. *Act* – if the results show success, implement the change incrementally until all the data is current and reflects the target.

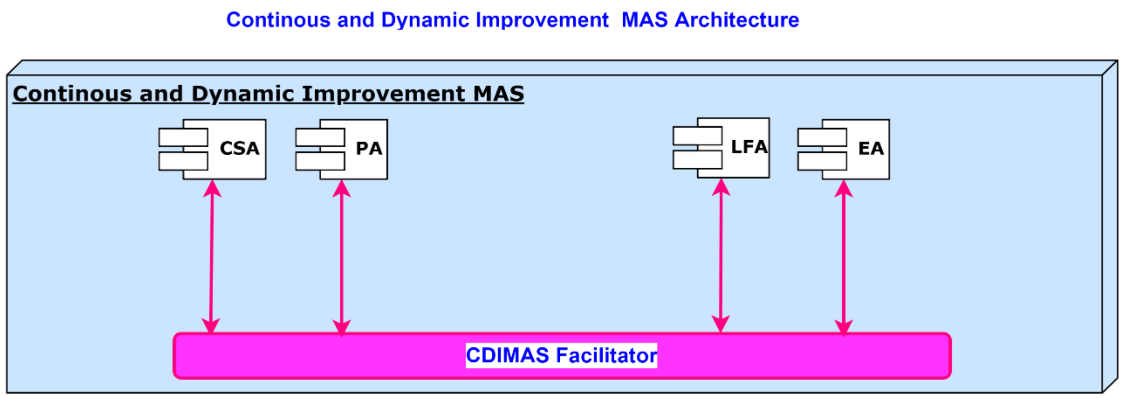
Continous and Dynamic Improvement MAS Architecture
Continous and Dynamic Improvement MAS
CSA
PA
LFA
EA
CDIMAS Facilitator

Fig. 37. CDIMAS organization.

The agents of Continuous and Dynamic Improvement MAS are:

4.7.1. Level Assessment Agents (LAA)

The Level Assessment Agent (LAA) assesses the current achievement of the learner to determine the next set of learning goals. The LAA must administer a series of assessments to determine the overall achievement. The agent will compare the student's assessment results against an achievement matrix (scale) [38] determined by an authoritative body. The comparison helps the agent identify areas of experts and weaknesses that will be used to decide on the next set of learning goals.

4.7.2. Data Collection Agent (DCA)

The Data Collection Agent (DCA) collects level assessment data and stores it in the database. The data analysis and prediction agent require this data.

4.7.3. Initial Assessment Agent (IAA)

The Initial assessment agent is the first agent that meets the learner after the learner creates their profile. The IAA will administer initial assessments to determine incoming knowledge, skill, and educational level. The intaking institution or other authorized body determines the contents and nature of the evaluation.

Furthermore, setting up the IAA agent to perform initial assessments for every new learning goal is possible. This will help to identify existing knowledge and adjust the learning content accordingly.

4.7.4. Data Analysis and Prediction Agent (DAPA)

The Data Analysis and Prediction Agent (DAPA) uses engagement and assessment data to make predictions that the presentation agent requires. DAPA aims to gauge student engagement and a given presentation method. If the level of engagement is not adequate according to a standard set in the engagement matrix, the presentation agent will adapt the presentation mechanism to get a better concentration.

4.8. Content Filtering MAS(CFMAS)

Although content filtering can mean several things, as we saw in chapter one, we refer to separating content into different categories according to the student's learning needs. For example, some students are slow learners, a few are fast learners, and others are average learners. Hence, content filtering helps us meet all learners' needs by adjusting the type and depth of the content to the student's learning level.

Content filtering can also be used to indicate the students learning direction. For instance, suppose a student's learning level is relatively higher for subject area A than subject area B. This indicator is also consistent over a considerable length of time. In that case, one can safely say the student's learning direction is A, and the system should provide relevant guidance to redirect the student toward learning area A.

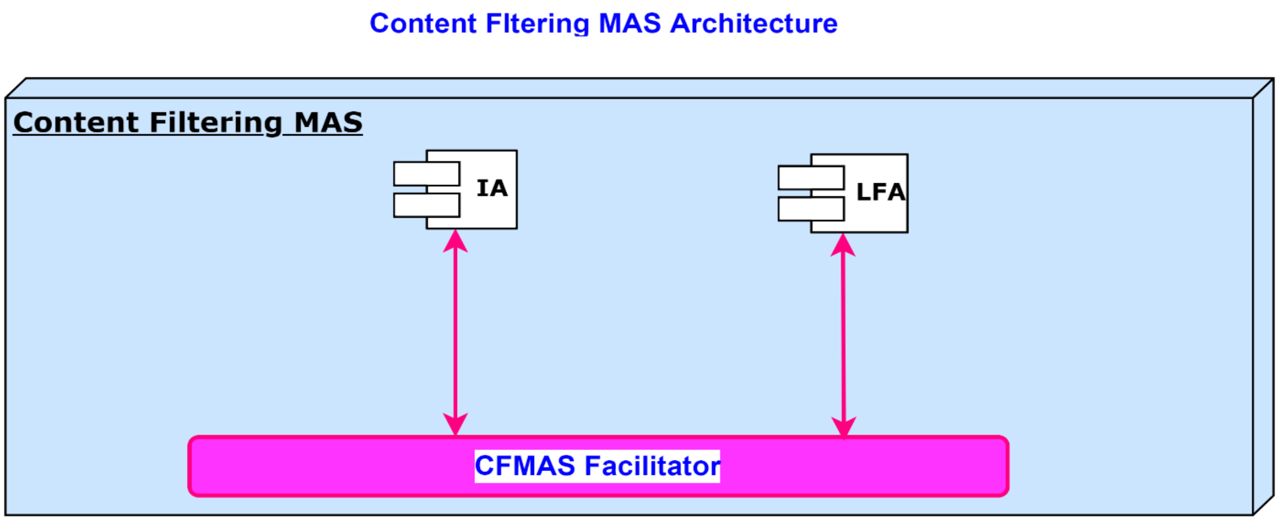

Fig. 38. CFMAS Organization.

The agents involved in Content Filtering MAS are:

4.8.1. Expert Agent (EA)

The Expert Agent (EA) monitors the learner’s performance for exceptional learning ability and communicates the result to the performance and progress agents. The purpose of EA is to identify students with outstanding learning abilities and guide them by providing learning resources to help them further exceed their educational careers.

4.8.2. Scaffolding Agent (SA)

The scaffolding process for learning goals involves the Scaffolding Agent (SA). The SA will unfold content into manageable pieces so that learning can happen progressively and in small increments. For example, the SA represents a multistep problem as a series of manageable problems so that the student can synthesize the pieces into a profound concept.

4.9. The Database Interface

The database interface is a significant part of the proposed architecture. The database interface shows the required databases to make the system fully functional. Although the database interface does not constitute agents, it is an active part of the system that should exist with the structure to make it operational. This section will discuss the different components of the database system.

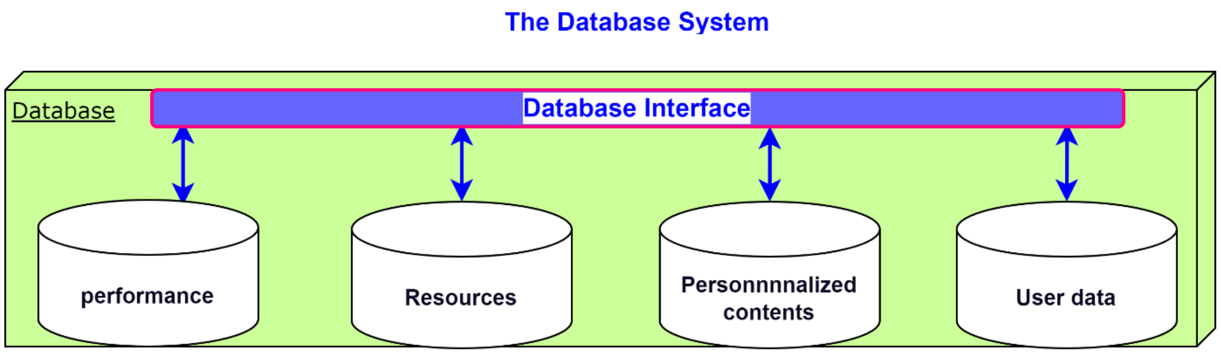

Fig. 39. The Database System Organization.

The database system contains the following elements.

4.9.1. Performance Database

The performance database stores the student's performance data. The performance data is necessary to determine the student's performance and improvement over time. In addition, as discussed in the preceding sections, various agents need this database to perform their duties.

4.9.2. Resources Database

As the name suggests, this database stores learning resources required by the student's current learning level. So, for example, when one of the agents determines the student's learning level, it notifies the database interface to pull all the resources required for the grade level and semester.

4.9.3. Personalized Contents Database

The personalized contents database contains customized content for each field of study according to the student’s learning preferences. One of the Adaptive agents discussed at the beginning of this section will populate this database.

4.9.4. User Database (Profile Database)

The user database stores the initial profile data of the user. Then, agents use the profile data to determine the demographic requirements of the user (student).

Chapter 5 – Quality attributes of software architecture

The adaptive multi-agent educational system properties are foundational for designing the systems software and the architectural decision-making process. After selectively describing the system properties (chapter one), the next task compares the system against a set of software architecture quality standards. This section will produce the foundation to assess adaptive multi-agent system architecture [45] against collection quality attributes.

To evaluate software system architecture, software system designers and quality controllers use design structure matrices (DSM), sometimes called dependency and structure modeling [14]. MacCormack states that the design structure matrix can either follow the core-peripheral design pattern or the hierarchal design pattern [10]. However, Clements says that the evaluation of a complex system should follow the general software architecture evaluation practices [12]. In his book, *Software architecture in practice, 4th ed.*, Bass vividly presented the standards of software architecture evaluation that software system designers and architects use to evaluate the feasibility of software architectures [52].

5.1. Software Quality attributes

The software development process is a tedious task that is prone to errors. An error at any stage of the development process can easily propagate without notice to the product deployment state and cause performance and other issues that can create customer dissatisfaction. To minimize the risk of architectural errors, software architects use a set of quality attributes to gauge the architecture.

Software quality attributes are features that characterize the expected properties of a software application. A qualitative or quantitative measurement of the quality attributes with high yield scores implies that the software is guaranteed operational. This section discovers the software quality attributes, such as performance, reliability, availability, security, portability, etc., found in many software evaluation textbooks and related articles [34].

5.1.1. Performance

Performance is the number of transactions per unit of time. In other words, it is the amount of time it takes the system to complete a user transaction. Software performance implies how well a software system, or its component meets its requirements for the timeline. The two most common software performance factors are response time and throughput. Response time is the time it takes from initiating an event to receiving the response about the completion of the event. Hence, response time contains three parts:

1. The *Forward Time* – is the time the signal requires to travel from its origin to where it triggers a process.
2. The *Processing Time* – is the time it takes to complete the triggered event.
3. The *Delivery time* – is the time required to send back the completed process.

The following diagram shows the decomposition of the response time of a system.

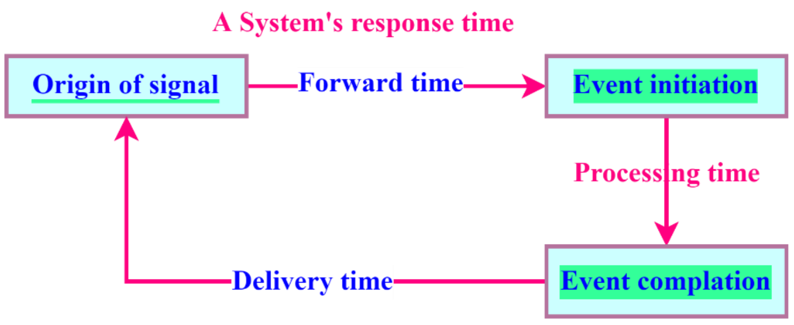

Fig. 40. Decomposition of a System's Response Time.

Hence, the total Response Time (RT), also known as responsiveness, is

$$Forward\ time\ (FT) + Processing\ Time\ (PT) + Delivery\ Time\ (DT)$$

$$==> RT = FT + PT + DT$$

Throughput refers to the number of requests processed per unit time. Like response time, it is an essential factor of performance. The system's throughput depends on the Response Time (RT) of the system and some non-software factors such as hardware components, network performance, and the capacity of input and output buffers.

The performance of a system is significant because it affects its essential characteristics, such as scalability, modifiability, and availability [52], [12]. For example, a high-performance system is scalable – i.e., increasing the work capacity of the system is easy. Furthermore, high-performance systems are modifiable – i.e., the system can incorporate more modules or functionalities; and the system is available – i.e., the system is running continuously from a user perspective [34].

5.1.2. Usability

Usability refers to the ease of using the software system. A user-friendly system is a software that can accomplish the desired task in a supportive manner. Usability is an essential characteristic of a software system because it reflects the user’s perception of quality. In the book *Software Architecture in Practice,* Bass states five areas of usability directly related to the friendliness of the system.

- *Learning system features* – if the user encounter difficulty using the system, the system must provide an easy-to-understand help feature that can guide the user through the learning process.

- *Using the system efficiently* should provide several options for performing a task. For example, the user may wish to suspend a task, complete another task and resume the previous job later.
- *Minimizing the impact of user errors* – the system should provide the option to cancel or abort an action the user started by mistake. For example, if the user issues a command and later identifies it as an error while the process is in progress, the system should provide the user with a means to abort it.
- *Adapting the system to user needs* – the system's ability to customize itself to fit its current needs. For example, the system can open to the last page where the user left.
- *Increasing confidence and satisfaction* – the system's transparency on actions it performs. For instance, the system displays the activities it is performing to the user.

Although usability implies a wide range of systems' ability to assist the user in completing a task, in terms of the origin of the usability event, it can be categorized as user initiative, system initiative, or mixed initiative. The user triggers a user initiative event, whereas the system starts a system initiative event. Finally, mixed initiative refers to an event triggered by both the user and the system.

In general, the usability of the system considers human behaviors such as intolerance (getting emotional very quickly), predisposed to errors (carelessness), or laziness (not interested in putting enough interest in doing something). Therefore, the system's usability is the sole property of the system that users refer to as "cool"; hence, it is the crucial component of user experience (UX) design.

Usability includes system reliability and accuracy. Reliability, a customer-oriented view of software quality, measures the probability of failure-free operation of a software system. A software system is reliable not only for its failure-free operation but also for its accuracy. Accuracy

is the system's efficiency in producing correct output. Users can easily be discouraged by a system that produces erroneous results.

5.1.3. Availability

In his book, Clements defines availability as "the proportion of time the system is up and running" [16, 29]. If the system fails at time t, how long after the system is up and running again? Suppose it takes Δt time until the system is back to normal. Availability is given by:

$$\boldsymbol{Availability = \frac{t}{t\ +\ \Delta t}}$$

As *Δt* increases, availability will decrease and vice versa. Hence, to increase availability, we need to minimize *Δt.*

It is challenging and sometimes impossible to predict and mitigate system failure as it can arise from any source and happen unexpectedly. Atchison recommends five tips for improving software system availability [30]. Among the five tips recommended by Atchison, his recommendation to build a system with failure in mind and regularly monitor availability are highly compelling.

A software architect should always consider the possibility of failure when building software—considering the chance of failure will help a software architect plan with contingency in mind that minimizes the severity of the malfunction. For instance, a software architect who plans to monitor availability will likely discover server health-related issues, configuration change issues, and application performance issues early before they incur severe system malfunction and cost.

The availability of software systems depends on reliability and recoverability. Reliability means that a software system is consistently functional with acceptable accuracy [34]. Ideally, the system must run continuously in a defined, stable environment. Moreover, the system must yield consistent accuracy for every consecutive run. Recoverability refers to the property of the software to restore to normal after unanticipated failure. The measure of recoverability, known as *Mean Time to Recovery (MTTR)*, is the time it takes the system from when system failure was first noticed until the system is back and running.

According to Bass, availability is also the system's ability to mask or repair faults, so they don't become failures [52]. This definition refers to the characterization of the system to prevent failure for loss is the deviation of the system from the average that is externally noticeable. Furthermore, Bass states that availability is closely related to security and performance. The failure of a system signifies unavailability – denial of services; an unavailable system has poor performance.

5.1.4. Security

Software security is an idea in software development to protect software against malicious attacks and hacker risks. Software security must provide integrity, safety, authenticity, reliability, and availability. The most common software security threats are language-based flaws, pitfalls, and software bugs such as buffer overflow and inconsistent error handling [31].

As noted above, building secure software does not happen naturally. It requires an exhaustive and repeated software test and code review to catch and remove bugs, find and correct flaws early, investigate security vulnerabilities and fix them. A developer can deal with security problems more effectively in the early software development life cycle [31]. The result is secure, reliable, and available software with minimum development cost.

In designing software architecture, software security is the software's ability to identify and resist unauthorized access to the software system through denial of services [12], [52], passwords, and multi-factor authentication services. A secured system should protect against unauthorized access while access is available to authorized users. Identifying users through passwords and multi-factor authentication can guard the system against attacks and data corruption.

Bass states that guarding the system is characterized by three simple but significant security measures [52].

I. *Confidentiality* – protecting data and services against unauthorized access. For instance, hackers cannot access the information stored in the system at all levels.

II. *Integrity* – protecting data from unauthorized manipulation. For example, no one can access and change the personal information stored in a system except the authorized user.

III. *Availability* – identifying security risks does not make the system unavailable to the authorized user. For example, if the system was made inaccessible due to suspected unauthorized use, it must be available as soon as trust is restored.

An idea closely related to security is privacy. The *Oxford Languages Dictionary* defines privacy as “the state or condition of being free from being observed or disturbed by other people.” An online article on the *University of California, San Diego* website described privacy as “ … the right to be let alone, free from interference or intrusion.”

Privacy is about limiting access to information on a user and who shouldn’t get access to the information stored about the user. The statement “who shouldn't get access” invokes the concern “who should get access.” The National Institute of Standards and Technology (NIST) defines private information as information that can distinguish or trace an individual's identity or linkable information to an individual.

Who should gain access to private information lawfully is tricky and much more complicated than one can imagine. First, there are collecting agency regulations and government agency policies regarding privacy. Second, there is no centralized location where users can keep their information, posing a vulnerability problem. Finally, users store inconsistent information across platforms and applications, challenging access control.

5.1.5. Modifiability

Modifiability refers to a software's ability to efficiently and cost-effectively accept and incorporate changes to the system's component or components whenever changes are deemed necessary. Change can be a key software upgrade, version update, server platform change, or database migration. An architecture should accommodate reasonable changes to it or a part of the software as far as the essential architectural components can still fit the change without disrupting the basic structure of the software.

The decomposition of the program functionality into standalone, interchangeable modules and connecting each piece via an interface to ensure interoperability is the backbone of software development that guarantees the modifiability of the software. Modular software is free of cohesion and coupling. According to Bachmann, software architecture with minimal cohesion and coupling can be modified easily and cost-effectively [32].

Software systems demand changes for multiple reasons. For example, adding new features to the existing system, fixing defects detected after deployment, upgrading security, improving performance, etc., necessitates unavoidable changes [52]. Bass identified our primary consideration regarding software modifiability.

1. *Alterable component* – which part of the software system can change? A system has several pieces; the environment, the platform, the hardware, the middleware, the network, the operating system, etc., are susceptible to change. Therefore, the Modifiability of the software system should take all possible system elements as potential candidates for alteration.
2. *Affinity to change* – which components are highly likely to change, and which ones are less likely to change? Since it is impossible to plan for all potential changes, the architect must focus on highly possible adjustments.
3. *Change initiator and change time* – who will make the change, and how long after deployment is a change likely? Users can initiate insignificant changes such as changing the screen saver, color, appearance, or other user contexts. User-initiated changes are dynamic and unpredictable but are less likely to affect the system's architecture. However, developer-initiated changes are highly dependent on the modifiability of the

system architecture because it is very likely to affect the system's architecture. Developer-initiated changes can happen during compilation, build, configuration setup, or execution.

4. *Cost of the change* – what is the cost of the change? Bass indicated modifiability involves two types of expenses: the cost of introducing the mechanisms and the cost of making modifications using the means. Establishing the mechanisms refers to the fee incurred due to employing a change agent that can trigger changes. The cost incurred from making modifications is the cost of editing or updating the source code. In general:

$$Total\ cost = cost\ to\ introduce\ the\ mechanisms + cost\ to\ make\ the\ changes$$

Modifiability of the software system involves blends such as scalability, variability, portability, and location independence. Although older architectural textbooks and articles classify the above lists as standalone quality attributes, Bass categorizes them as blends of modifiability with specific flavors [33].

Scalability means getting more to or from something. Scalability can mean expanding by adding more resources. For example, the software may require additional servers or complementary components with added functionality. Scalability can also mean enhancing performance or throughput. For example, change the hardware component (upgrading memory) or the software performance parameters to make it efficient.

Portability, an indispensable property in many large-scale distributed web applications, refers to the usability of the software system on multiple platforms with different software environments. Minimizing software platform dependencies ensures portability. For instance, using platform-independent development languages such as java will enhance portability. Portability is closely related to deployability [33].

Variability is a term used to indicate the software's capability to accommodate variations. Bass described variability as “the ability of a system and its supporting artifacts, such as

code, requirements, test plans, and documentation, to support the production of a set of variants that differ in a preplanned fashion.” [52] for example, a software system with free and pro versions has product variability. The pro version of the same product usually provides more functionality than the free version.

Location independence is when distributed software components interact without knowing each other’s locations [52]. This means that the parts can be deployed anywhere, and clients that need to interact with these components must discover them dynamically. Alternatively, new software features can make themselves discoverable once they are deployed.

5.1.6. Testability

Testing is one of the extensive and budget-intensive processes in the software development process. Testing is crucial because the final product must meet all the requirements and display acceptable performance and functionalities before deployment. A software system deployed before an extensive testing procedure devastates the client and the developer. Therefore, developers spend a lot of money on testing to ensure that the software system meets minimal operational standards.

Software testability is the measure of the system’s permissibility to support testing. Testability is the property of a software system to reveal faults as soon as possible without much effort and time. Testability assumes that there is at least one fault in the software system. If a defect exists in the software, it must be detected as early as possible.

Various parties, such as quality assurance personnel, developers, or users, perform the testing process continuously at different levels and with varying capacities. For example, a developer seeks to validate the source code, while quality assurance personnel are interested in the software's quality aspects. In addition, a user may achieve the system's usability, functionality, performance, and aesthetic elements.

As stated above, testability is not a quality that a single party can achieve. Hence, testability is usually carried out as a team. Therefore, development and business teams are collaboratively involved in the artifact's testability.

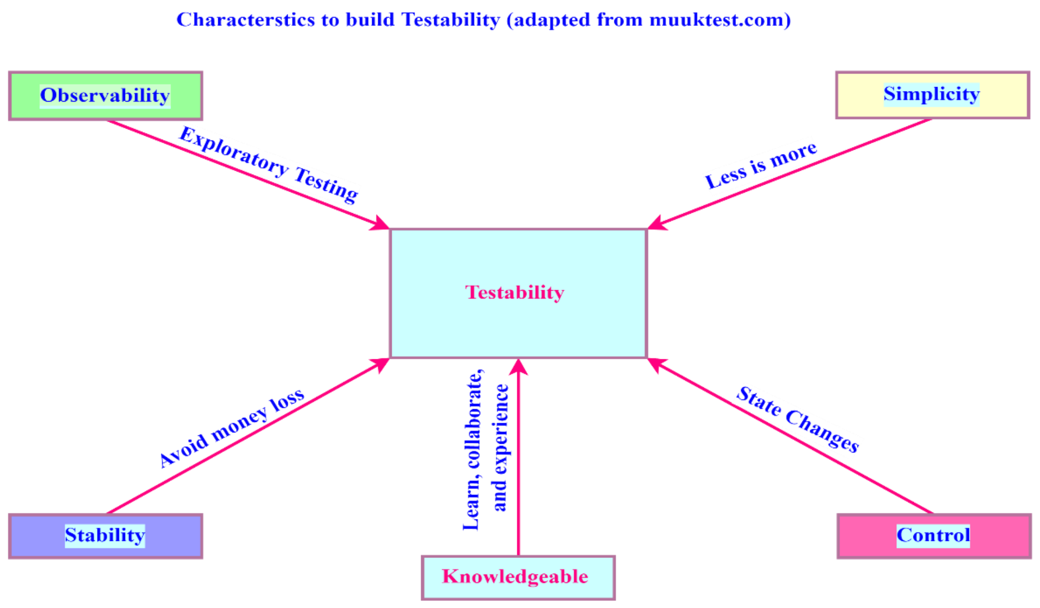

Fig. 41. Characteristics to build testability.

As indicated in fig. 41, incorporating the following five testable software characteristics [76] can improve the testability of a software system.

1. *Observability* – improving observability increases the chance of catching errors and resolving them early in the development process.
2. *Simplicity* – adding simplicity (i.e., creating the most straightforward possible solution [76]) eliminates complexity in the testing process.
3. *Control* – controlling the functionality help manage the state changes that can significantly alleviate the complexity of testing.
4. *Knowledgeable* – being an expert (involving an expert in the testing process) in the subject matter help simplify testing and reduce overlooking.
5. *Stability* **–** testing software stability is critical because if the system is unstable, it can create barriers to testing.

5.1.7. Integrability

One of the complex tasks in software development is the task of integration. Integration can mean performing several duties that involve risk and intimidation. For example, integration can mean deploying the system on a new platform or migrating it from its current platform to a new platform. Integration can also mean adding new features to the existing system or upgrading the components of the system to meet the current technological requirements.

Integrability generally involves the system's evolvability in content and size [52]. Suppose there is a system with $\{s_i\}$ subcomponent parts where $i \in \{0 \dots n\}$. Integrability tries to answer the question that, given this scenario, can the system S support add a new set of components $\{s_j\}$ such that $j \in \{0 \dots m\}$? Can S support removing some of the features of the system? Can S accommodate changing or upgrading the components?

System integration can change the software’s performance, availability, security, usability, etc. It can also affect the system's database and other resources, such as network protocol or topology. Integration difficulty involves two variables: size and distance [52]. Size is the number of potential dependencies between components to add to the system, whereas distance is the difficulty of resolving the dependency differences.

According to Bass, the integrability of software systems depends on the system’s dependency and is often measured in terms of semantic or syntactic dependency. Two or more methods are syntactically dependent if one method calls, inherits, or uses the other. On the other hand, two or more methods are semantically dependent if one method is coupled with the other and shares knowledge of the same protocol, file format, unit of measure, metadata, or some other aspect [52].

One strategy to reduce syntactic aspects of dependency is using services and microservices, known as decoupling. Decoupling will reduce the number and distance of the component’s dependency since services know each other only through their interface. If the integration requires a change to one service, the other services might not be affected.

5.1.8. Safety

With software taking control of more and more devices, safety features of the software (system) became a genuine concern. A software system is unsafe if it is not operationally reliable. The Boeing 737 system failure problem reported as the control system failure is an example of what a software safety concern implies. In addition, the 2009 Shushenskaya incident, the Therac 25 deadly radiations, the Ariane 5 explosion, and many non-fatal incidents are related to software safety [52].

Sometimes safety concerns emerge from the human, not the software. If the human-operated software is unsafe while the software is doing the right thing, it may cause a safety issue. For instance, the incident of Air France flight 447 was due to the human pilot's miscalculation and misoperation about the aircraft's airspeed while there was no problem from the software side. Software safety issues can arise from a variety of factors, including: [52]

1. *Omissions* – if an event or component is absent, this can result in wrong actions that lead to safety issues. In most cases, omission happens due to negligence, overlooking, or insufficient test scenarios.
2. *Commission* – sometimes, an undesirable event may spuriously occur and lead to the wrong conclusion. The incorrect decision can mislead the human operator or the software into taking improper action.
3. *Timing* – wrong timing (an early or late occurring event) can lead to a wrong decision. Out-of-time events are problematic because they lead to the right decision at bad timing, resulting in erroneous operations.
4. *Problems with system values* – there are two categories of incorrect values, coarse and subtle false. Coarse values are detectable wrong values, whereas subtle values are undetectable incorrect values.
5. *Sequence omission and commission* – when a sequence of events misses the necessary part of the sequence (sequence omission) or events contain an irrelevant section to the sequence (sequence commission), an erroneous operation can happen.
6. *Out-of-sequence* – if the order of the sequence of events is wrong, the execution result is incorrect, and the system can misbehave.

Stateful systems can use the advantage of their state to check if the operations are safe or unsafe. If the system detects a hazardous operation, one of the following should be implemented to restore it to a safe state.

- Continue operations after recovering from the unsafe state.
- Fail the system and shut it down.
- Switch the system to manual operation.

5.1.9. Deployability

The steps, processes, and activities required to ship the software to where it experiences life and performs its anticipated duties are what developers call software system deployment. Furthermore, software deployment involves resending the new version of the software after upgrading the software components, updating patches, and fixing bugs. Hence, deployability is a software property that measures the ease and speed with which software accommodates changes and prepares for deployment.

Today, software deployment is a rapid process of fixing bugs quickly and promptly upgrading the software with newer versions, resulting in fast maintenance. Frequent software deployment is desirable for several reasons. First, it will help keep the software up-to-date and productive, leading to customer satisfaction. Second, routine deployment creates reliable and technologically current software that can stay in the market for an extended time. Finally, it helps discover erroneous operations in the software and fix them before they cause irremediable incidents.

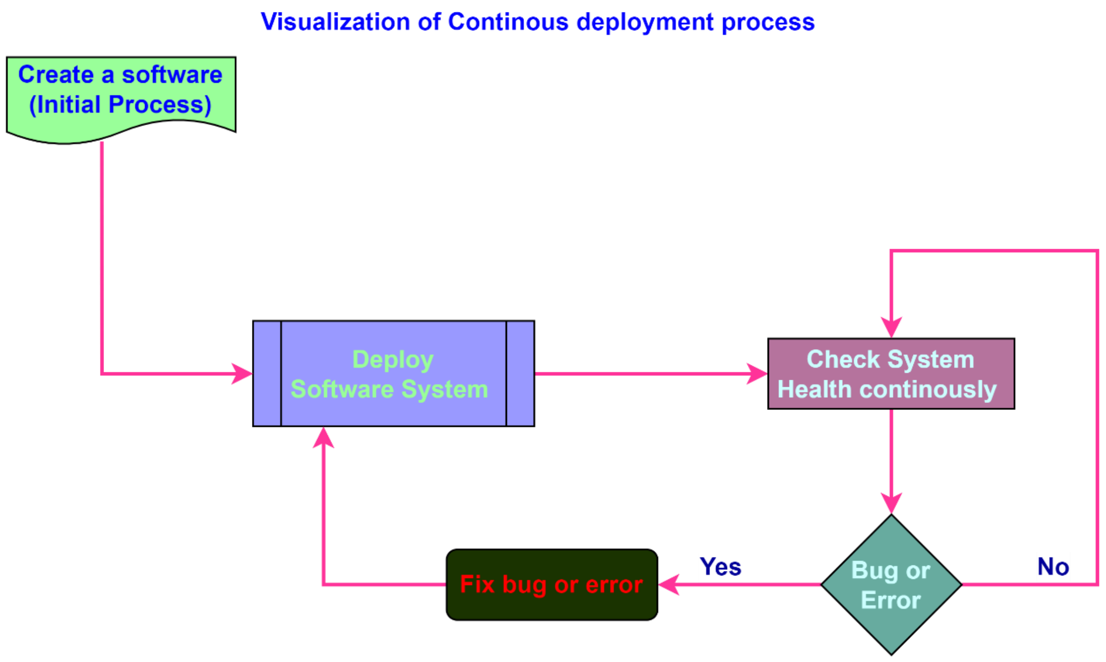
Visualization of Continous deployment process
Create a software (Initial Process)
Deploy Software System
Check System Health continously
Bug or Error
Yes
No
Fix bug or error

Fig. 42. Continuous Deployment.

Continuous deployment is the deployment process in which an automated system continually monitors the software system's health without human intervention. The continuous deployment process might require what is known as a deployment pipeline, “a sequence of tools and activities that begin when you check your code into a version control system and end when your application has been deployed for users to send it requests.” [96]

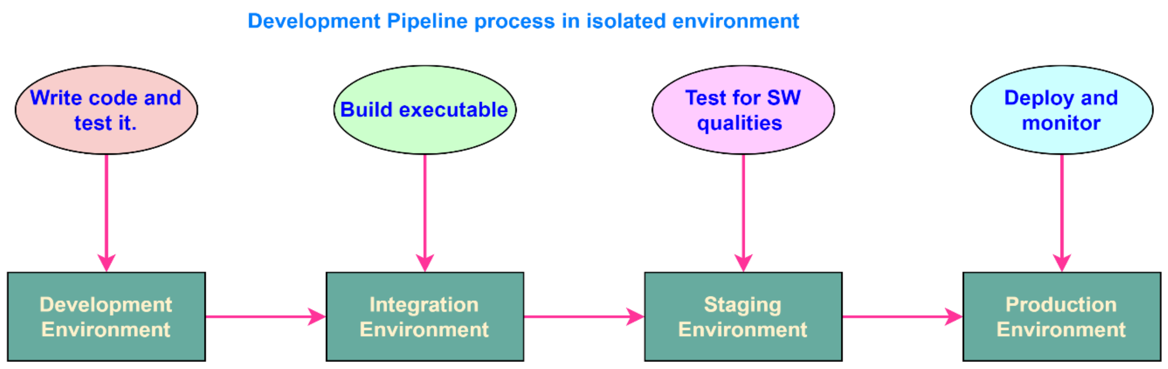

Fig. 43. Development Pipeline process.

The deployment pipeline process takes place in an isolated environment. The major components of the environment are:

- *Development environment* – the environment for code development. The developer conducts a standalone unit test to prepare the code for deployment; an expert will review it. Finally, the code will be committed to a version control system.
- *Integration environment* – the environment for building the executable. The new version (upgrade) is ready for production (use).
- *Staging environment* – the environment to test for various qualities. The staging environment will test software qualities for performance, security, license conformance, etc.
- *Production environment* – the environment that avails the system for use. In the production environment, either the developer or someone will continually monitor the system until confidence is gained in its quality.

5.1.10. Energy efficiency

In the past, energy used by computers was never an issue. However, with the development of mobile devices that use limited and replenishable energy, the energy efficiency of devices and software architectures has become an enormous concern. Moreover, the computing mode changed from running local small-scale servers to cloud-based large-scale servers in our contemporary era. In addition, IoT interconnects countless devices and handles continuous and fast communications that consume a considerable amount of energy.

In his book, *Software architecture in practice, 4th ed.,* Bass states four reasons for considering energy efficiency as a first-class quality attribute [52].

1. *Importance of energy* – if energy is not available to run a device or the software, both the device and the software are useless. The significance and usefulness of a software system depend on energy availability, and energy will be available for a longer time if the software system is energy efficient.
2. *Create awareness* – educating young engineers and software developers on energy efficiency's role is vital. Developers and architects should consider energy efficiency for every piece of engineering and coding they perform.
3. *Incorporate design pattern concepts* – designing for energy efficiency should become part of the design concepts.

Software system designers and architects should consider balancing energy efficiency and other quality parameters such as performance and availability. Performance and availability tradeoff for energy efficiency is an area that requires keen consideration in the cloud context, mobile devices, and IoT.

Chapter 6 – Examples of Multi-agent architecture

Computer scientists and many scholars in education have been studying the notion of a coherent, intelligent, and adaptive educational system since the inception of personal computing machines, particularly since the invention of the internet. However, the idea got immense attention only after the early 1990s. Since then, scholars have conducted tons of research on the topic, and thousands of research papers have been published. This section will present some works of scientists in adaptive multi-agent educational systems.

This chapter presents in detail a few articles that propose educational multi-agent architectures. The aim is to establish a comparative evaluation of the proposed architecture by offering the architectural details on the features of other educational multi-agent architectures. In addition, critically analyzing closely related software systems will assist the readers in understanding the proposed architecture more deeply.

6.1. The architecture by Lin

In the article "A Multi-Agent and Service-Oriented Architecture for Developing Integrated and Intelligent Web-based Education Systems," [40], [46] Lin et al. vividly presented a mechanism to incorporate intelligent agents into a web-based educational system. They argued that integrating agents into the web-based educational system will make the organization adaptable and personalized [59]. Furthermore, they proposed an architecture consisting of two categories of agents: personal agent (PA) and task agent (TA). They further subcategorized personal agents into user personal agents (UPAs) and instructor personal agents (IPAs).

6.1.1. The goal of the article

The article by Lin et al. aims to develop an integrated and intelligent web-based educational system (WBES) using multi-agent technology. The system proposed by Lin et al. can intelligently adapt and automatically identify task orders to reduce the workload. Moreover, the system is designed to support multiple topic areas with diverse users. Lin et al. claim that using web services as clients or supporters is a good fit for the system's modularization and makes it robust, flexible, autonomous, and intelligent.

6.1.2. The proposed architecture

The proposed architecture consists of three primary classes of agents, personal agents, task agents, and web services. This classification is plausible as it demarks the learner (user), the provider (system), and the platform (the environment). Fig. 44 shows the primary agents involved in the learning system proposed by Lin.

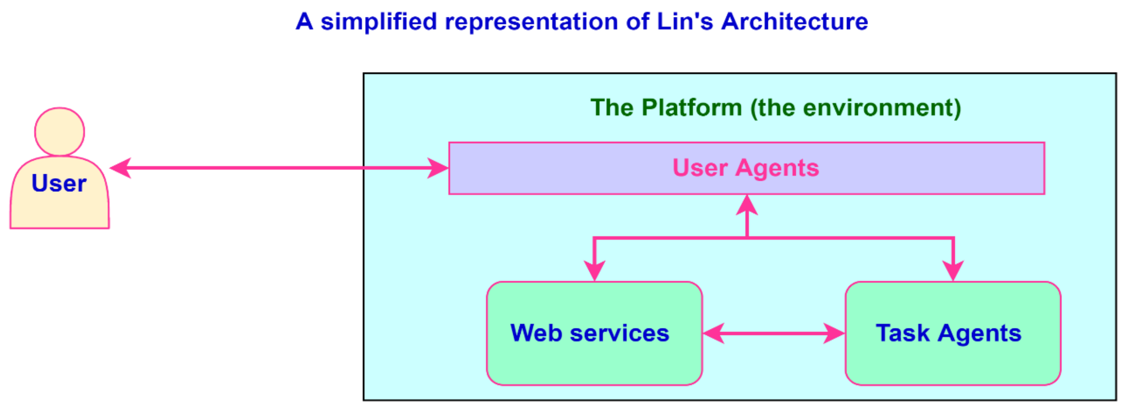
A simplified representation of Lin's Architecture
User
The Platform (the environment)
User Agents
Web services
Task Agents

Fig. 44. A simplified Lin's Architecture.

Lin’s architecture is service-based and uses the web as its platform. The architecture comprises personal agents that run on an agent platform that can interact with the learner or the instructor through web services. The users can log in to the system to download their favorite agents in an isolated environment. Fig 45 shows the general structure of Lin’s architecture.

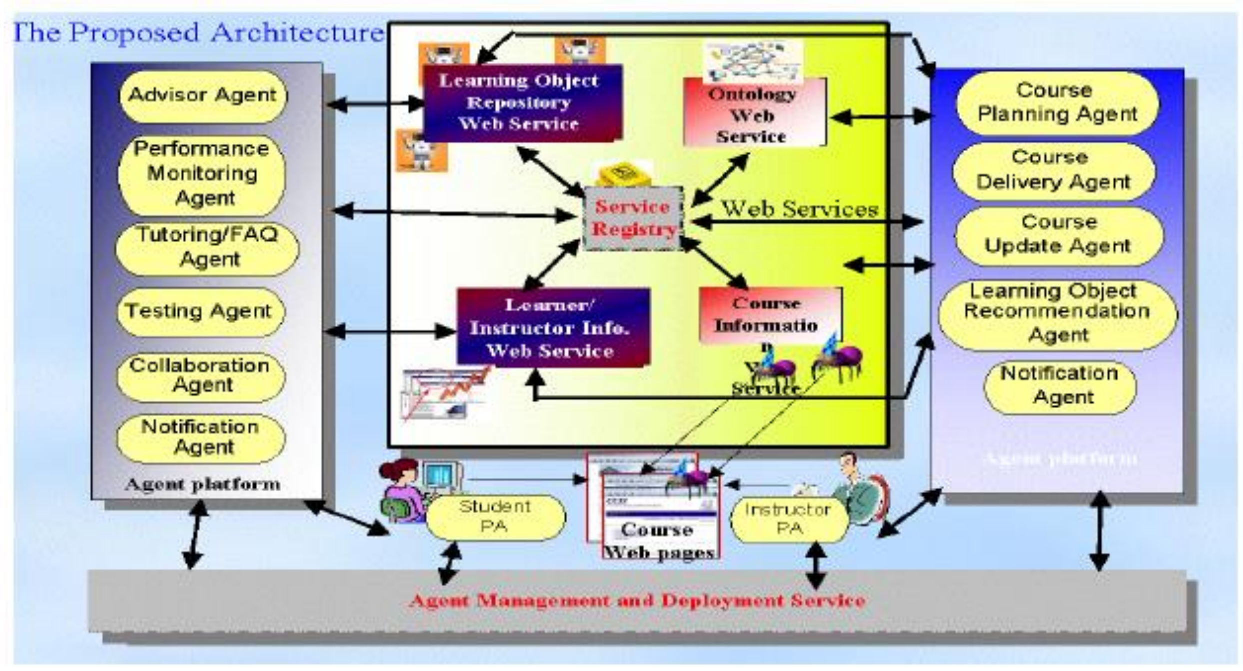

Fig. 45. Multi-agent Architecture Proposed by Lin.

6.1.2.1. The personal Agents

The personal agents (PA) provide learning support to the users (user interface). According to the article by Lin, a user can delegate rights to an agent, manage task agents with the environment, and configure options. Users can communicate with task agents by running a secured web service or an application. Lin identifies two categories of agents as personal agents, the Learner Personal Agent (LPA) and the Instructor Personal Agent (IPA).

The instructor's personal agent performs tasks that support the instructor by generating, delivering, and maintaining online courses. The IPA includes the curriculum planning agent, course delivery agent, course update agent, learning objective recommendation agent, and notification agent.

The LPA consists of an advising agent, tutoring agent, performance monitoring agent, collaboration agent, notification agent, and testing agent. The LPA simulates an instructor who can provide adaptive course materials and appropriately differentiated instruction.

6.1.2.2. Task agents

The task agent performs learning-related and teaching-related tasks such as providing services, knowledge, information, coordination, and communication obligations. The task agents play the role of a web service client or a supporter of web services. As a web services client, an agent can perform the following tasks.

- Searches of different entries.
- Include reasons about the semantics of web services.
- Mediate and compose web services.
- Send messages and RPC-style calls to a web service.

As a supporter of a web service, an agent can facilitate and enable web services by completing the task autonomously and intelligently.

6.1.2.3. Web Services

In a web-based education system, shared distributed resources are familiar. Web services provide knowledge management and information resource management services for web-based education. Knowledge management services include locating, analyzing, and retrieving resources. For example, domain knowledge and curriculum planning knowledge are part of knowledge management services, whereas learner information and curse information management are part of information resource management services.

6.1.3. Agent Management services

The agent management services provide support to the agents. The agent management services are implemented through web technology. It includes log-in benefits to gain access to personal agents. The service will assign a unique agent ID and record information for further reference. This report does not include the architectural implementation because it is not part of the purpose of this document.

6.2. The Architecture by Abdelhamid

Abdelhamid proposes an architecture that can serve a single purpose (advising services) as part of an educational system [44]. According to Abdelhamid, academic advising is tiresome and time-consuming, principally when the number of students is significantly large. Furthermore, the academic advising process has several features, such as long-term, iterative, and limited-time features, that make it difficult for advisors to provide optimal academic advising. Hence, developing a system that can provide this service with minimal human presence is necessary.

6.2.1. The goal of the article

Abdelhamid’s architecture proposes a framework of integrated environment based on multi-agent to automate the entire purpose of academic advising. The author lists several reasons for developing the system. First, the traditional academic advising process at Shaqra University is broken and needs repair. Second, the conventional advising process deprives the student of independent decision-making ability. Lastly, the traditional advising process uses one of the academic advising processes, namely, prescriptive, integrated, developmental, or engagement.

Therefore, an academic advising software system can provide quality advising service to students since it can incorporate a blend of the advising models by automatically selecting one based on the student’s needs.

6.2.2. The proposed Architecture

Abdelhamid proposed a multi-agent academic advising system that uses machine learning modules for centralized coordination and communication. Including machine learning algorithms makes an intelligent system capable of providing a remote advising service. The system uses complete student information such as student academic records, university regulations, previous advising information, semester schedule, feedback from professors, and institutional program information.

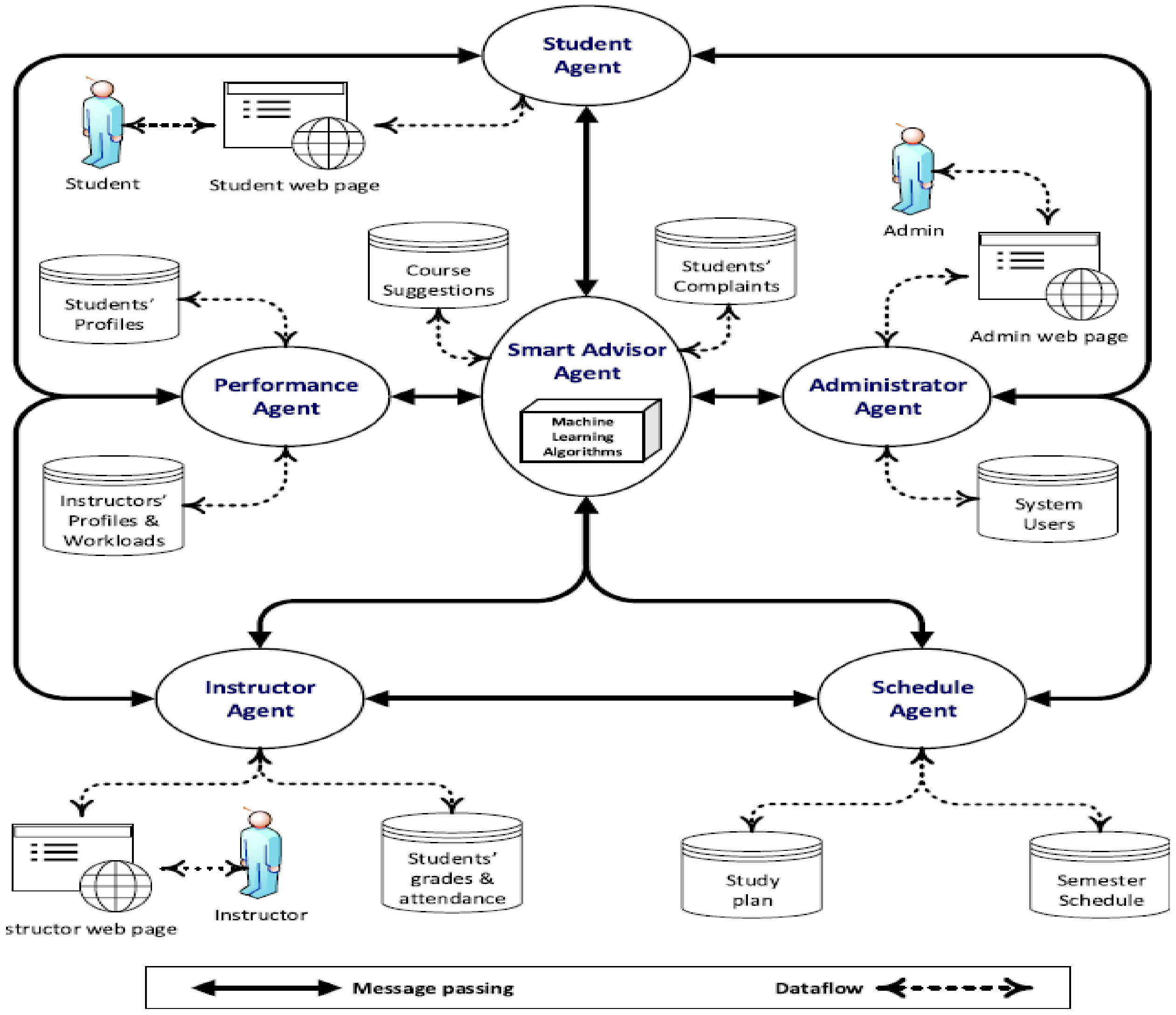

Fig. 46. Multi-agent Architecture proposed by Abdelhamid.

The advising system will keep track of the student's progress, identify the student's position on the academic ladder, plan the next set of courses the student must take, and provide helpful information on course completion and planning. In case of insufficient data, the system will refer the student to a human advisor by booking the student's meeting schedule with the advisor.

The architecture proposed by Abdelhamid comprises six agents; the student agent, the instructor agent, the administrator agent, the schedule agent, the performance monitoring agent, and the smart advisor agent. The agents perform the complete academic advising process collaboratively and coordinately.

6.2.2.1. The Instructor Agent

The instructor agent will perform the duties of a human advisor (an instructor). The tasks of the instructor agent include moderating the course topic, follow-up the course progress, and calculating the percentage of achievement in the intended learning outcome. The following use-case diagram shows the interaction between the instructor agent and other agents.

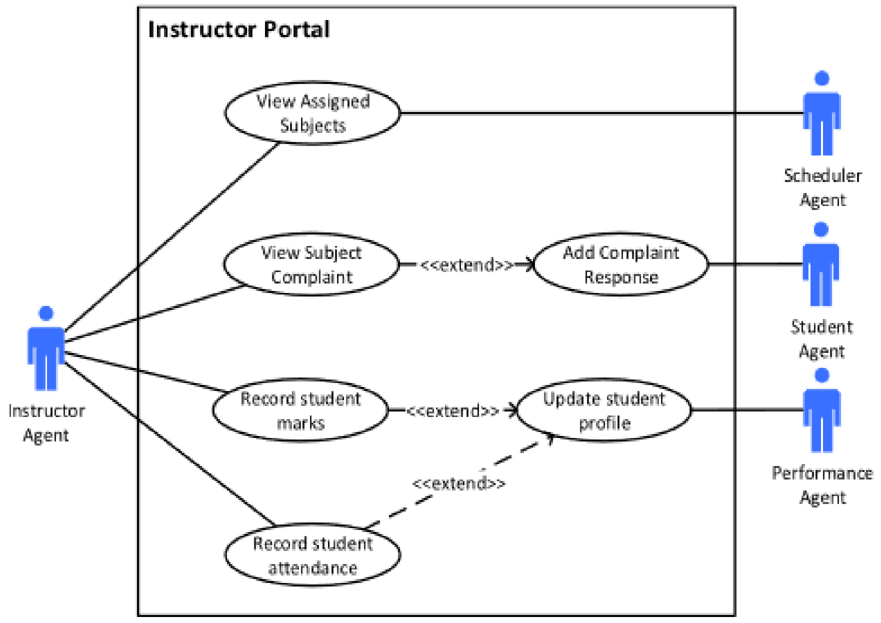

Fig. 47. Instructor agent use-case diagram.

6.2.2.2. The Student Agent

The student agent handles the academic advising requests that come from the students. One of its tasks is to identify each student's preferences and planning requirements. Initially, the student will select two career tracks from the available lists, the first being the most preferred and the second being the best alternative. To help the incoming students understand the different career tracks, the student agent will provide more information about the career path.

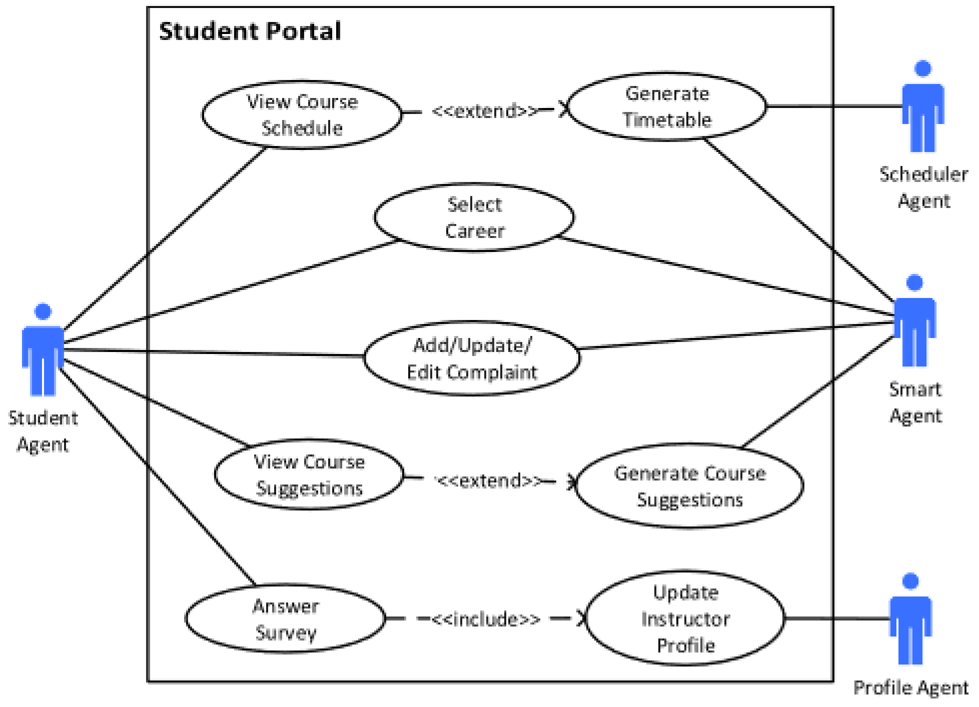

Fig. 48. Student agent use-case diagram.

6.2.2.3. The Administrator Agent

The administrator agent performs the duties of a course administrator. The agent will conduct course curriculum protection tasks and maintain the curriculum model. The administrator agent will have an interface to allow the human to add, remove or update courses.

The administrator agent will directly communicate with the schedule agent to request resource management and with the student agent to approve or disapprove student requests. The following use-case diagram shows the interaction between the administrator agent and the other agents.

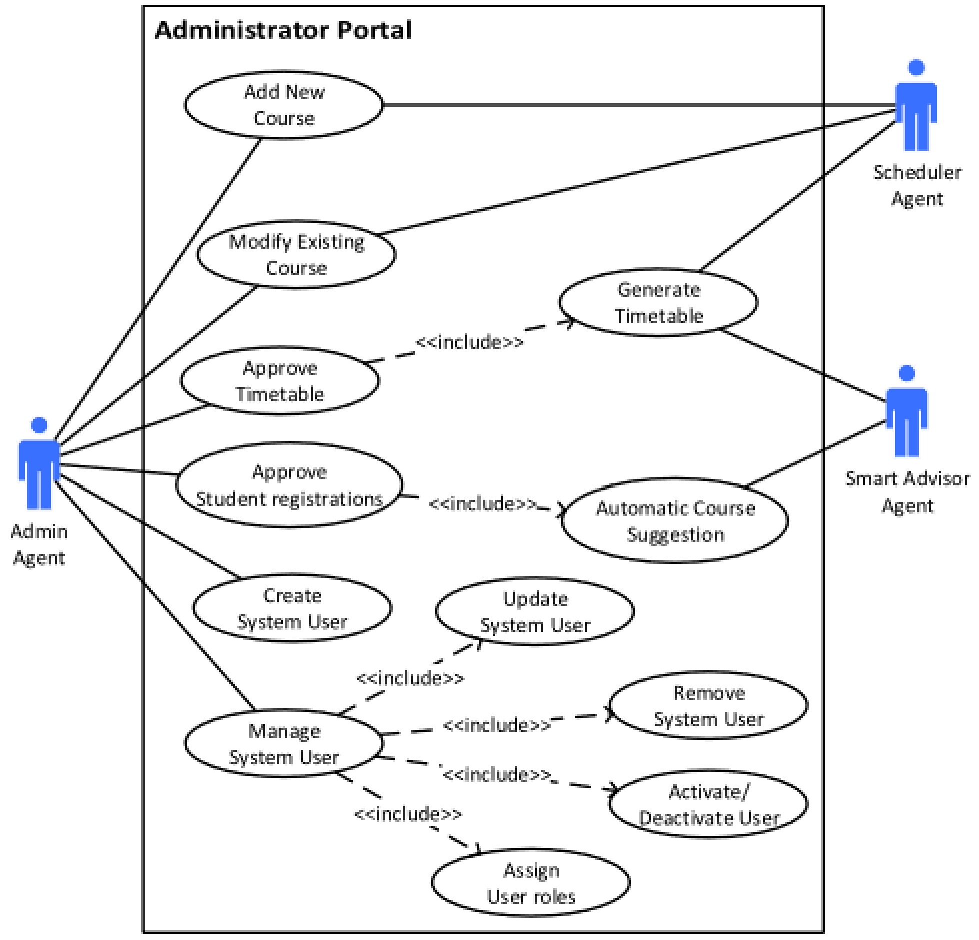

Fig. 49. The administrator portal use-case.

6.2.2.4. The Schedule Agent

The schedule agent automatically creates and maintains the course schedule starting each semester. The schedule agent depends on other resources such as study plans, instructors' workload, and previous semester course schedule to perform its tasks correctly. The figure below shows the overall communication between the scheduler agent and other agents in the smart advisor system.

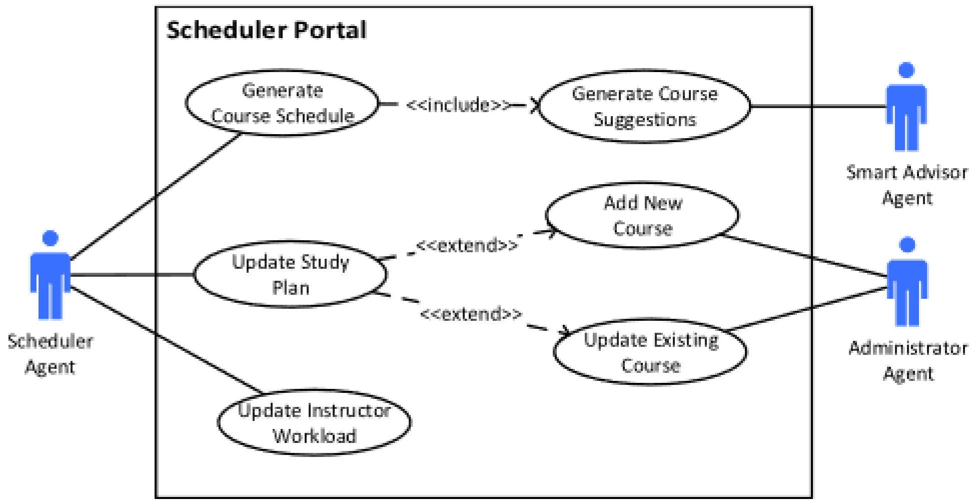

Fig. 50. Scheduler Agent use-case diagram.

6.2.2.5. The Performance agent

The performance agent creates and updates the students' and Instructors' profiles. Furthermore, it gathers preferences, feedback, and results to build and update the profiles. In addition, the performance agent will monitor the instructor's and students' performances concerning the course and update the instructor's profile accordingly.

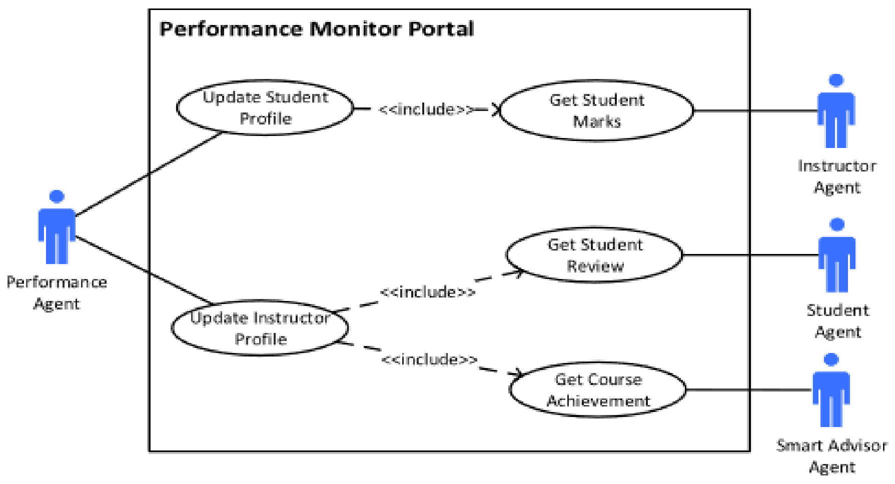

Fig. 51. Performance monitoring agent.

6.2.2.6. The Smart Advisor Agent

The smart advisor agent is the central agent in the proposed architecture. The significance of the smart advisor agent is due to the machine learning algorithms that it contains needed to deduce proper advice to the student. Moreover, the smart advisor agent coordinates the communications among other agents. The primary tasks of the smart agent are:

- To suggest courses to students based on their profile and previously determined career tracks.
- To contribute to the generation of the semester study plan, it analyses the data received from the performance agent, student agent, and administrator agent to provide the schedule agent with the set of best courses to offer in the semester plan.

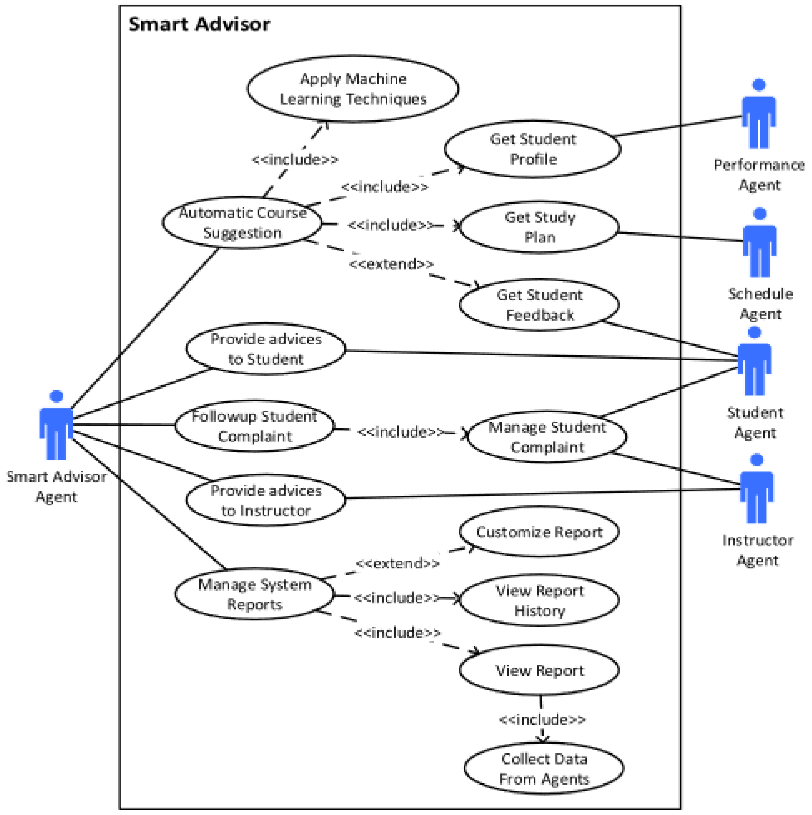

Fig. 52. Smart advisor agent.

6.3. The Architecture by Hammami

Hammami et al. [41] suggested a multi-agent intelligent educational system that uses a blackboard agent as a centralized communication platform. Hammami focuses on achieving adaptability through guiding and personalization. According to Hammami, regardless of the complex interaction patterns of agents in a multi-agent system, the agents are robust, flexible, appropriate for web-based services, and can simulate a specialist.

6.3.1. The goal of the article

The article by Hammami aims to propose an educational software architecture with several multi-agent levels interfaced with a blackboard. Architecture's primary goal is to foster technology in education to improve the quality of teaching and advance educational services. Furthermore, the author suggests a solution to the complexity of designing an e-learning system by using multi-agent technology.

6.3.2. The proposed Architecture

Hammami et al. recommended an architecture consisting of two distinct multi-agent system (MAS) components: the authoring multi-agent system (AMAS) and the learner multi-agent system (LMAS). The authoring MAS is responsible for generating and maintaining the course structure, whereas the learner MAS is responsible for presenting the course and evaluating students' progress.

The system stipulated by Hammami consists of pedagogical agents [20] working collaboratively to introduce adaptability and personalization to a web-based distributed educational system. The system uses an interface to facilitate communication among agents of a MAS and across the MASs. There are four primary MASs in Hammami’s architecture, the Authoring MAS, the Learning MAS, the interface MAS, and the Interaction MAS. Furthermore, the MASs have multiple databases attached to them.

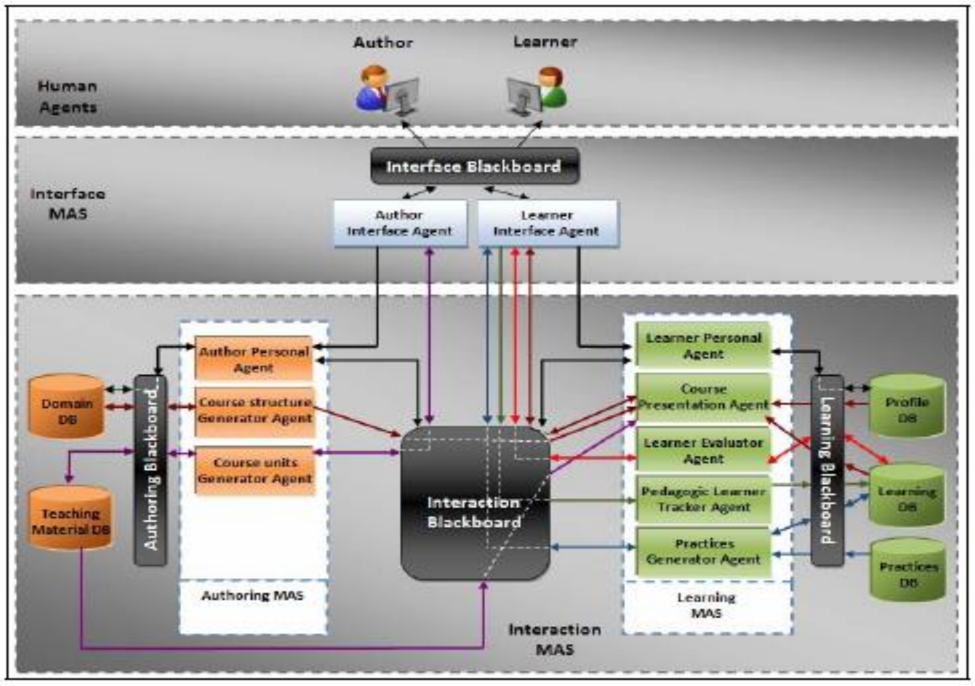

Fig. 53. Multi-agent Architecture proposed by Hammami.

6.3.2.1. The Authoring MAS

The authoring MAS consists of agents involved in preparing the learning contents and setting it up to be adaptively accessible to the students taking the related course. The agents adapt course presentations and modify contents to tailor to the student's personalized needs. In addition, the authoring MAS assists the course author in incorporating the required course contents.

6.3.2.2. The Learning MAS

The learning MAS contains agents interacting with the learner to facilitate learning. The learning MAS consists of three agents: the learner's personal agent, the course presentation agent, and the learner’s evaluator agent.

6.3.2.2.1. The learner personal agent

The purpose of this agent is to provide a personalized learning environment [49] to the learner. It communicates bi-directionally with the presentation agent and practice generator agent to coordinate the relevant course presentation mechanism and allow them to practice the course contents. Furthermore, it connects with the pedagogic learner tracker agent to record the learner's actions and performance.

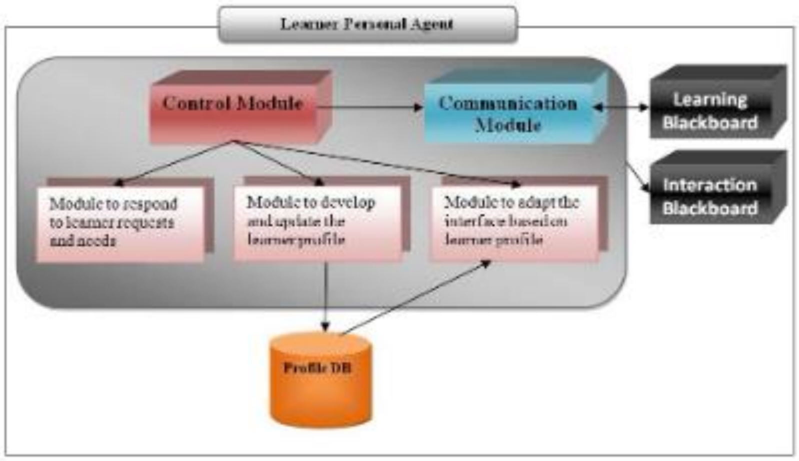
Learner Personal Agent
Control Module
Communication Module
Learning Blackboard
Interaction Blackboard
Module to respond to learner requests and needs
Module to develop and update the learner profile
Module to adapt the interface based on learner profile
Profile DB

Fig. 54. Structure of Learner Personal Agent.

6.3.2.2.2. Course Presentation Agent

The course presentation agent obtains the course materials from the course database and organizes them into a coherent structure. The agent will also customize the course contents to the student's needs. The structure and content of the course presentation agent are shown in fig. 55.

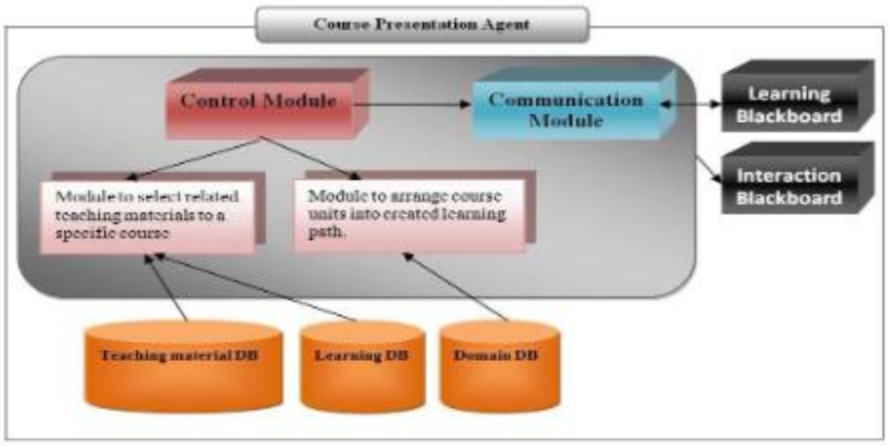
Course Presentation Agent
Control Module
Communication Module
Learning Blackboard
Interaction Blackboard
Module to select related teaching materials to a specific course
Module to arrange course units into created learning path.
Teaching material DB
Learning DB
Domain DB

Fig. 55. Structure of Course Presentation Agent.

6.3.2.2.3. *Pedagogic Learner's Tracker Agent*

The pedagogic learner tracker agent tracks the various activities and interactions of the learner with the system. This agent collects data about the behavior that is useful for evaluating the learner's knowledge state.

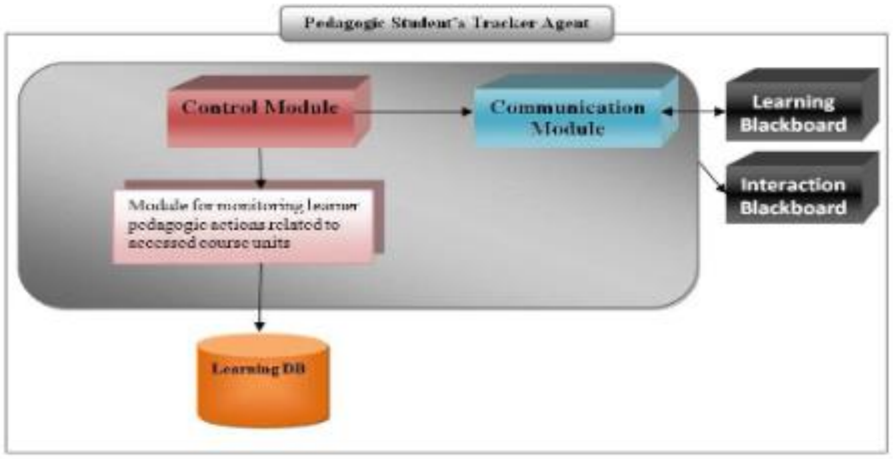
Pedagogic Student's Tracker Agent
Control Module
Communication Module
Learning Blackboard
Interaction Blackboard
Module for monitoring learner pedagogic actions related to accessed course units
Learning DB

Fig. 56. structure of Pedagogic Learner's Tracker Agent.

6.3.2.2.4. Practices Generator Agent

The practice generator agent tracks the learner's knowledge state or preferences. It adaptively presents the practices and the results to the learning database. Other tasks, such as organizing the order of the practice problems, are also the duty of the practice generator agent.

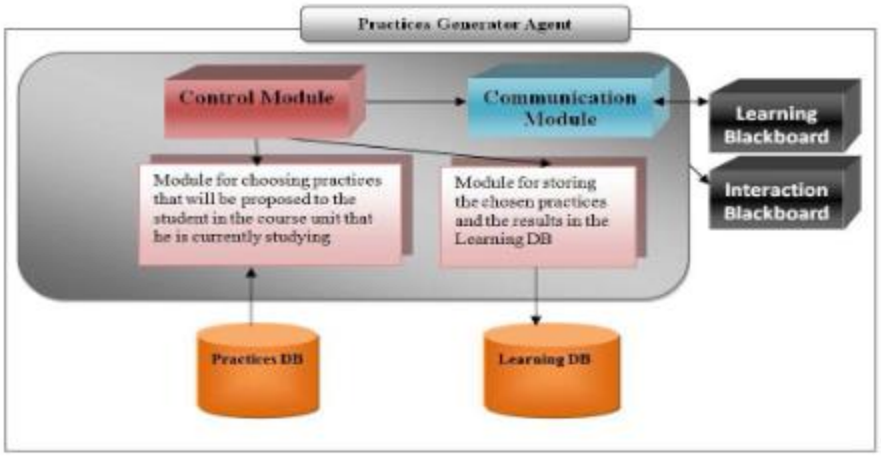
Practices Generator Agent
Control Module
Communication Module
Learning Blackboard
Interaction Blackboard
Module for choosing practices that will be proposed to the student in the course unit that he is currently studying
Module for storing the chosen practices and the results in the Learning DB
Practices DB
Learning DB

Fig. 57. Structure of Practice Generator Agent.

6.3.2.3. The Interface MAS

The interface MAS consists of agents that assist the human agent (the student or the course instructor) in interacting with the system. Hence, the interface agent act instead of the human agents to make transactions with the system. The interface agents consist of three agents that are discussed below.

6.3.2.3.1. The interface blackboard agent

The interface blackboard agent is an agent interface that coordinates and controls the interaction between the external agents (human agents) and the system agents. It receives a request from the user and returns the result to the user.

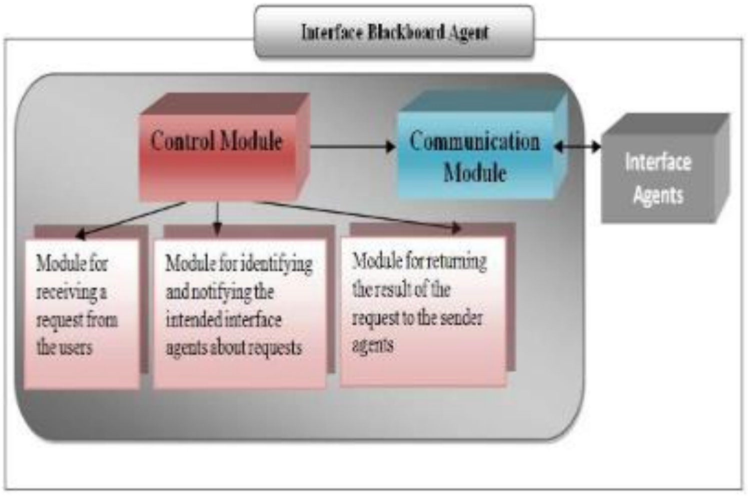
Interface Blackboard Agent
Control Module
Communication Module
Interface Agents
Module for receiving a request from the users
Module for identifying and notifying the intended interface agents about requests
Module for returning the result of the request to the sender agents

Fig. 58. Structure of Interface Blackboard Agent.

6.3.2.3.2. Learner Interface Agent

The learner interface agent interfaces between the learner MAS and the other MASs. It monitors the learner's actions and gives access to system resources. All communication and coordination between the learner and the rest of the system happen through the learner interface agent.

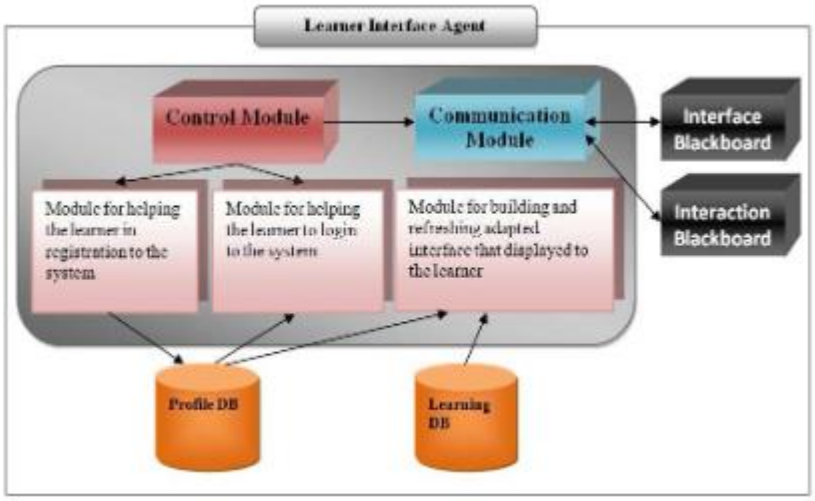
Learner Interface Agent
Control Module
Communication Module
Interface Blackboard
Interaction Blackboard
Module for helping the learner in registration to the system
Module for helping the learner to login to the system
Module for building and refreshing adapted interface that displayed to the learner
Profile DB
Learning DB

Fig. 59. Structure of Learner Interface Agent.

6.3.2.3.3. The Author Interface Agent

The author interface agent coordinates the interaction between the author and the system. The author interface agent avails the information it needs for the whole learning process. The course author can also interact with the system through the authoring MAS because the authoring interface agent can facilitate the interaction.

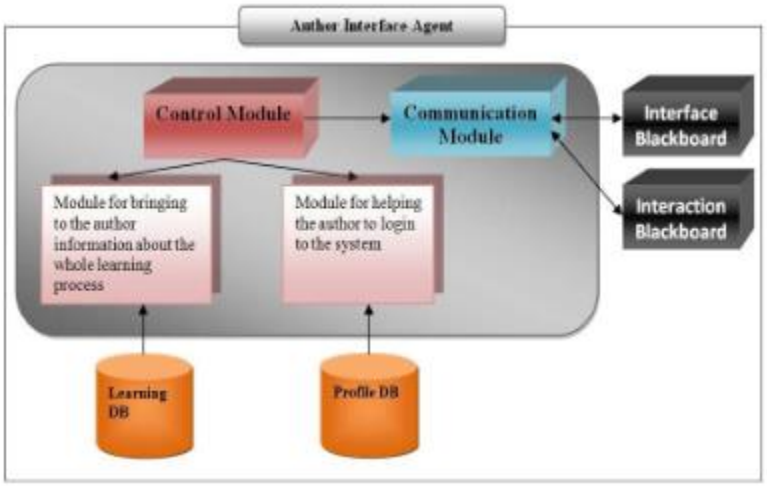
Author Interface Agent
Control Module
Communication Module
Interface Blackboard
Interaction Blackboard
Module for bringing to the author information about the whole learning process
Module for helping the author to login to the system
Learning DB
Profile DB

Fig. 60. Structure of Author Interface Agent.

6.3.2.4. The interaction MAS

The interaction MAS contains a collection of intelligent blackboards with various functionalities. The main task of the interaction MAS is to facilitate interaction among the MASs. Each MAS is connected to the interaction MAS and can only communicate with other MAS in the system.

Chapter 7: Applying Software Quality Attributes To The Proposed Architecture

The software evaluation process will recall the characteristics of the proposed adaptive multi-agent system discussed in *Chapter 2 – Multi-Agent Architecture,* and compare each feature to the standards described in *Chapter 4 – Quality Attributes of Software Architecture*. The goal is to evaluate the system's characteristics by comparing the stated features to the evaluation standards presented in Bass's book [52] and arrive at a sound conclusion based on the evaluation results.

7.1. Background of the Evaluation Criteria

Software development requires a step-by-step process called Software Development Life Cycle (SDLC). This process consists of phases of the software cycle and how those phases are executed [10]. The process starts with the requirements analysis. Once the requirements are met and the system's feasibility is determined, the design step follows. Finally, the design steps determine the system's structure and layout, also known as software architecture. Since the requirements analysis step includes software system evaluation, why do we need to re-evaluate the software architecture?

Several reasons dictate software architecture valuation. Clements states the following points as the principal reasons for evaluating a software architecture [12].

1. *Error detection or identification* – evaluating software architecture maximizes the chance to identify errors early before they incur a substantial cost to resolve them. Finding and fixing software faults at the architectural design stage is considered the best practice in software development.

2. *Software system quality factors heavily rely on system architecture* – the architecture of a software system affects the software quality factors such as performance, functionality, availability, security, reliability, etc. Therefore, designing software with the correct architecture is crucial for increasing customer satisfaction and retention.
3. *The project's overall structure depends on the software architecture* **–** the software architecture determines the team composition, team structure, budget, schedule, and other software development-related factors such as the configuration of control librarie**s** and selection of the development environment. Therefore, determining the correct architecture before the implementation phase can affect the pace and effectiveness of production.

In his book, *Software Architecture in Practice, 4th ed.,* Bass generalized software architecture quality attributes and boiled them down to ten criteria used as software evaluation standards in various research papers.

7.2. The Quality Attributes Displayed by the Proposed Architecture

7.2.1. Adaptability

The adaptability feature of a software system is often viewed as one of the software quality attributes [34] [35]. Today, software architects reasonably list adaptability with quality attributes, a common feature in many modern software systems. For example, the article by Tarvainen, *"Adaptive Evaluation at Software Architecture Level,"* listed adaptability with the well-known software quality attributes such as performance, reliability, availability, integrability, etc. in fact, the list of quality attributes is growing and changing from time to time that adaptability might also be included in the list soon [77].

The adaptability of a system can improve *usability*. For example, Wesson et al. proposed an adaptive user interface for mobile applications to increase adaptability [78]. Furthermore, Faisal et al. argued that an adaptive system is for its *modifiability* [79]. They state, "… it is necessary to engineer adaptability into software systems to meet various future requirements." Moreover, according to Galster et al., the simulation result of adaptive systems demonstrates the shortest response time over a range of time, implying higher *performance* [35, 37].

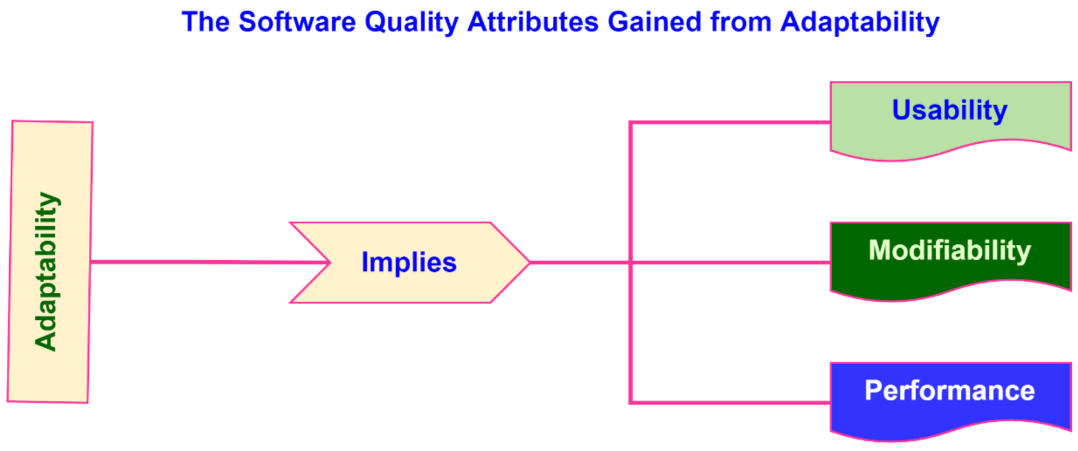

Fig. 61. Quality Attributes Derived from Adaptability.

Adaptive systems are easy to use. Consider the adaptability of the mobile systems as proposed by Janet et.al. The users of these devices are active and consistently engaged. They seek to do things as quickly as possible. For example, if the user wants to send a text message using the device, they want to accomplish it quickly. The user can easily and quickly send the text message if the system adapts to the user's typing speed and assists the user adaptively in spelling correction and word suggestions for the language they are using.

Adaptive systems are also modifiable. If the system is adaptive, it can accommodate changes required to make it robust and up-to-date. For example, technology use may change over time, and the client may request the inclusion of new features. However, it is hard to anticipate all the future changes from a client at the production time. Nevertheless, the developer or the architect can make the system modifiable by making the system adaptable to unanticipated changes.

Adaptive systems demonstrate better performance. Adaptive systems (particularly self-adaptive software) have high performance because they can anticipate environmental changes and predict the next set of instructions. For instance, consider a software system that consecutively executes tasks A, B, C, and D. Suppose at a point of iteration of the tasks, the values of B and C are not changed, i.e., the system can reuse the values from the previous execution. If the system can anticipate the return values of Method B and C required to run D before running methods B and C, instead of running processes B and C, it can use the previous values to minimize the run time.

7.2.2. Interactivity

The cooperation and coordination of agents in a multi-agent educational system to effectively respond to dynamic situations can drive efficient *agent-to-agent* and *agent-to-learner communication* at the global level that requires either distributive negotiations or integrated negotiations that involve the learner. Therefore, the interactivity of a multi-agent software system provides a structure that supports multiple software quality attributes.

A multi-agent educational system that provides various agent-to-learner communication functionality furnishes an interactive learning environment that elevates the efficiency and interactivity of agent-to-learner and vice-versa. For example, the user (learner) can either ask a text-based or a voice-based question. The agents collaborate to figure out a comprehensive response to the questions, substantiate the response, and present the answer progressively and interactively.

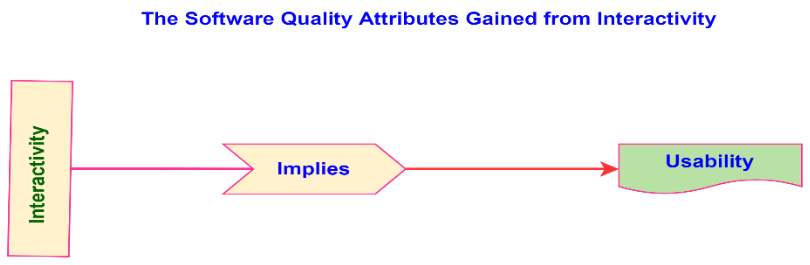

Fig. 62. Quality Attributes derived from Interactivity.

According to Sandnes et al., interactivity can improve the *usability* of a software system [80]. It is self-evident that interactive systems are user-friendly, engaging, and easy to use. Interactivity is one of the valuable qualities of the proposed educational software less affiliated with software quality attributes. Sandnes et al. indicated that focusing on quality alone may deteriorate user experience and result in an ineffective system [80]. Hence, although interactivity is an indispensable feature of the proposed software system, it is less associated with the software quality attributes.

7.2.3. Autonomy

As discussed in Chapter Two, agents in an adaptive multi-agent system can act and react independently to the dynamic changes in their environment if they are autonomous. Autonomous agents can learn the changes in the background, store the learned behavior, and respond to the changes according to the stored data. In addition, agents can adjust their response(s) to the learned behaviors based on new circumstances or data.

In a study on autonomous (self-driving) marine vehicles, Dalpe et al. concluded that autonomous systems have superior *performance* [82]. In the article *"Survey on Characteristics of Autonomous Systems,"* Singh et al. summarized autonomous systems' characteristics into four major categories: self-configuring, self-healing, self-optimizing, and self-protecting, also called the self CHOP [81].

According to Singh, the *self-configuring* property of a software system refers to the system's ability to add new components, remove old or faulty parts and incorporate the configuration into the system with no or little human interference. The self-configurable property is closely related to the *integrability* quality attribute of a software system. Chapter four of this document states that integrability involves changing the software system's *performance*, *availability*, *security*, *modifiability*, and *usability*.

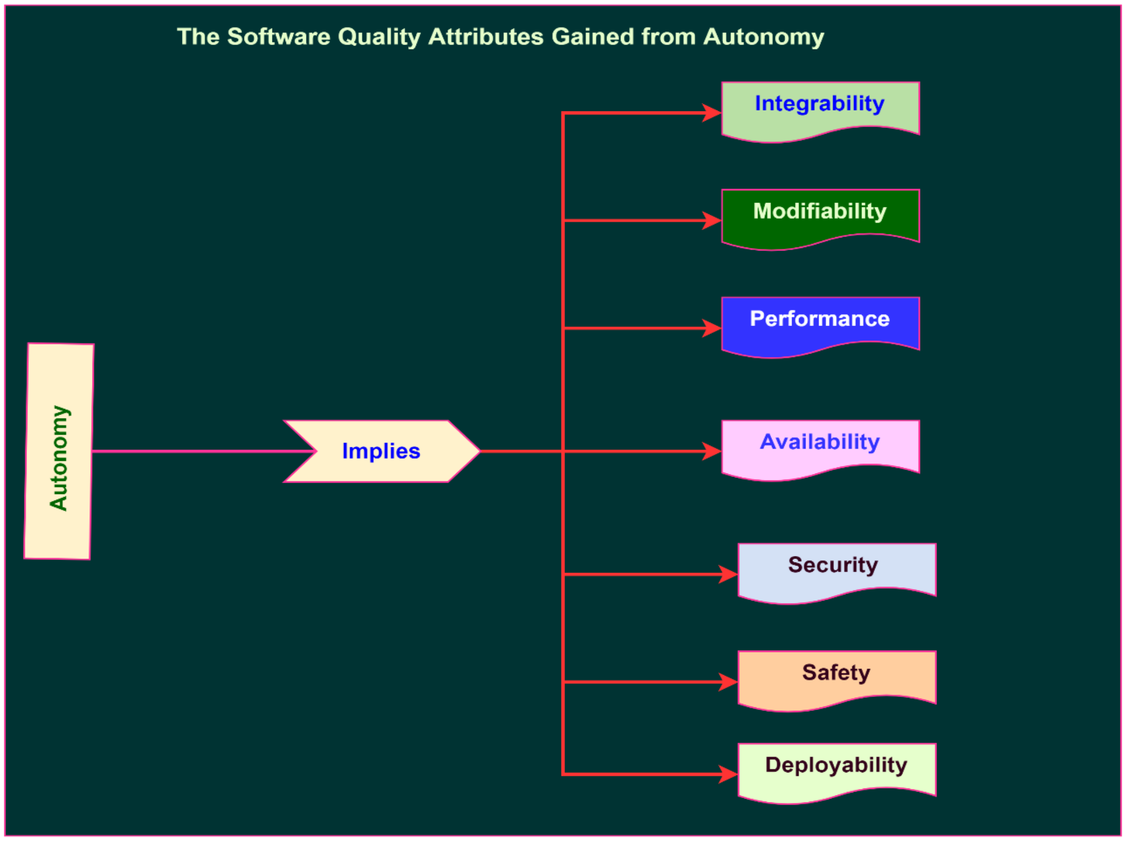

Fig. 63. Quality Attributes derived from Autonomy.

Furthermore, *self-healing* is a property related to the system's self-diagnostic maintenance. If the system encounters a problem, it captures the error, analyzes its cause, and resolves it. Self-diagnostic maintenance behavior of a system requires a process state that can continually capture and store the past conditions of the system and retrieve the most current healthy state from the stored process states. Self-healing is closely aligned with the *safety* and *maintainability* quality attributes.

The *self-optimizing* is the property of the software system to achieve improved efficiency in resource utilization and workload management dynamically. Hence, self-optimization is closely related to *performance*, *usability*, *modifiability*, *integrability*, and *deployability*. Today, self-optimizing software systems are the center of focus for cloud-based computing and embedded software systems with limited storage and memory.

Lastly, the *self-protecting* property of the software system has to do with the ability of the system to guard itself against malicious attacks by detecting hostile behaviors and taking corrective actions. The self-protecting characteristic of an autonomous system aligns with the system's *safety*, *security*, and *reliability* quality attributes.

7.2.4. Decentralization

*Decentralization***,** one of the characteristics of an adaptive multi-agent educational system described in chapter one of this research paper, is crucial for improving the system's *performance*. Schuff states that decentralization maximizes network performance as the system does not experience a performance bottleneck [28]. Furthermore, a decentralized system guarantees better performance as each node can reroute a user's request. In addition to localization of decisions, a decentralized system will provide enhanced performance by sharing workload among the available nodes.

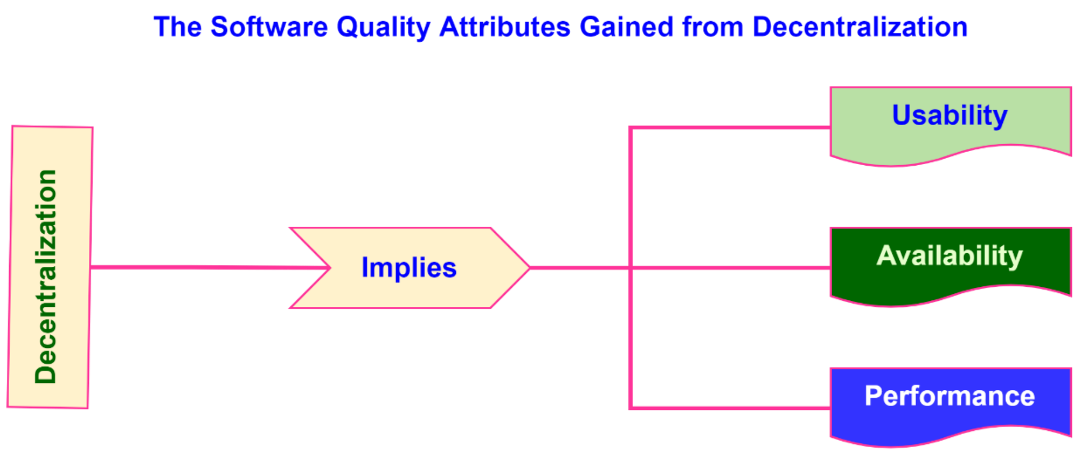
The Software Quality Attributes Gained from Decentralization
Decentralization
Implies
Usability
Availability
Performance

Fig. 64. Quality Attributes Derived from Decentralization.

One notable characteristic of decentralized systems is their high *availability*. Decentralized systems do not have a single point of failure, meaning that if the system fails at a node, the other nodes will continue to provide the services with the breakdown unnoticed anywhere in the network. Furthermore, high availability leads to *reliability* and *usability* quality attributes.

Decentralization is a key feature that can improve the *reliability* of a software system. Data access in a large-scale distributed system involves dispersed access, i.e., the data can be accessed from any node in the distributed network. For example, suppose there are a hundred servers in the link that can provide access to data. If two or three servers fail to operate, the user can continue accessing the remaining servers' data without sensing the failure. On the other hand, if data is accessed from a single server (centralized system), the user cannot access the data until the server is maintained and running again.

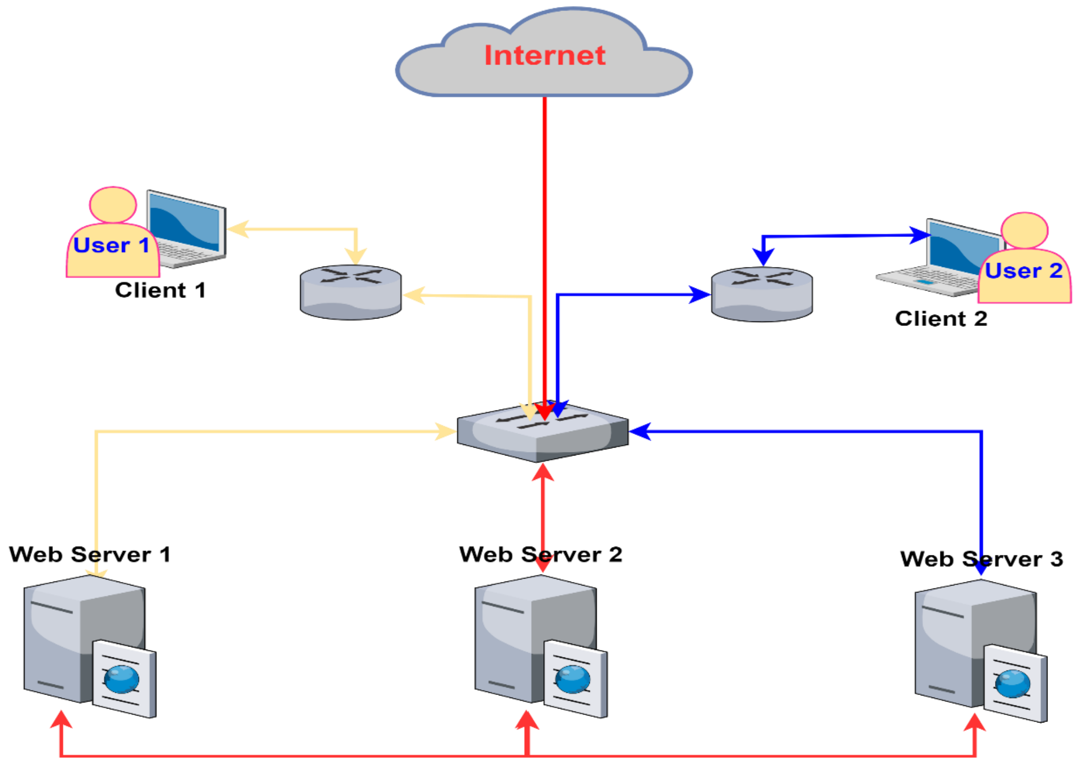

Fig. 65. Decentralized Systems reliability.

User 1 is connected to Web Server 1 through client 1, whereas Web Server 2 has no established connections. User 2 is accessing Web Server 3 through client 2. Web Servers 1, 2, and 3 are connected to form a network of servers. If Web Server 3 fails, user 2 will switch and connect to Web Server 1 or 2 without losing connection.

A software system is reliable if it can produce accurate results within acceptable error margins. However, measuring accuracy is not easy, and its implementation may differ from system to system based on the assigned job. Moreover, accuracy is relative as one may need an "exact" result to compare it with the output.

Another characteristic of a decentralized system is *scalability*. Scalability is a software system's characteristic to accommodate a size change. A size change can involve growing the system volume by adding more components or shrinking it by removing some parts. Referring to the discussion in chapter four, we can easily relate scalability to the *modifiability* quality attribute of the software system. Achieving modifiability is relatively straightforward in a decentralized system since decentralization reduces subsystem coupling significantly.

As elaborated in chapter one, decentralized systems have multiple central units independent of each other. The independency of the subsystems simplifies the system’s *deployability*. For example, suppose a decentralized system contains two autonomous subsystems, A and B. if a component in subsystem A requires modification, it is easy to modify and deploy the subsystem without affecting the ongoing operation of subsystem B.

7.2.5. Continuous and dynamic improvement

Dynamic improvement is the term used to signify a process that improves over time to adapt to its environment. According to Chen, systems with dynamic improvement characterization maintain the best performance (high throughput and low latency) [83]. The *software architecture's continuous and dynamic improvement* feature can enhance the software system’s *performance*, *availability*, *security*, *modifiability*, *testability*, and *integrability* quality attributes.

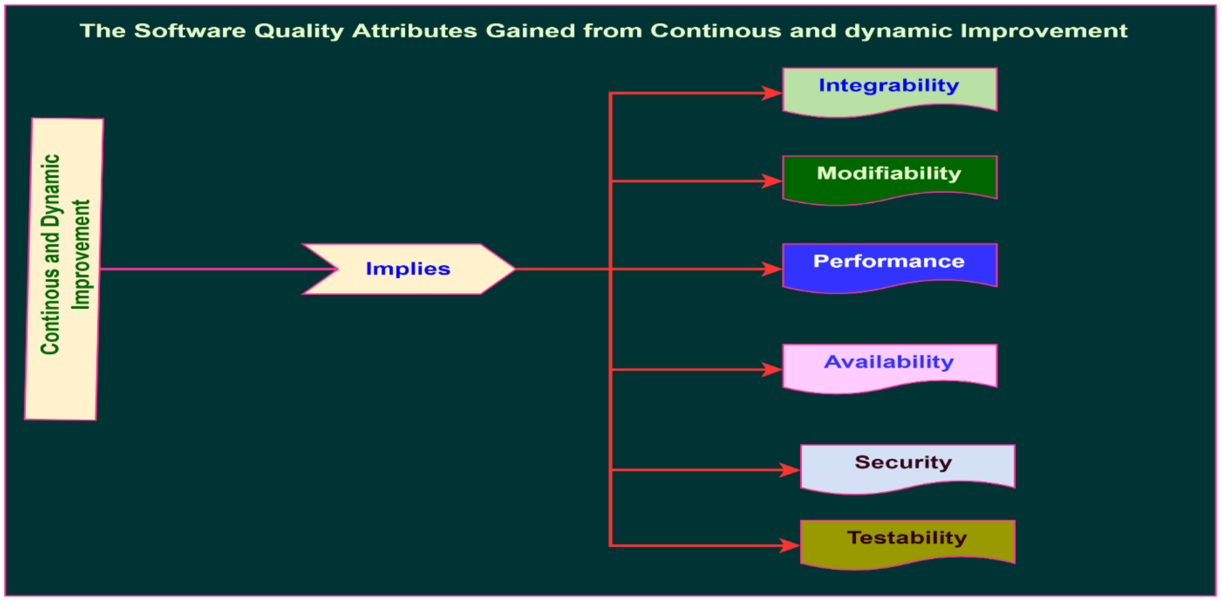
The Software Quality Attributes Gained from Continous and dynamic Improvement
Continous and Dynamic Improvement
Implies
Integrability
Modifiability
Performance
Availability
Security
Testability

Fig. 66. Quality Attributes Derived from Continuous and Dynamic Improvement.

One of the enhancement targets characterized by the continuous and dynamic improvement process is the system's *performance*. The system's performance will get better as the system undergoes a constant and dynamic improvement process. For example, consider a system that runs multiple tasks (multitasked process). The system can improve its performance in a dynamically improving environment by improving its process scheduling after each process run, i.e., an efficient scheduling process leads to better performance.

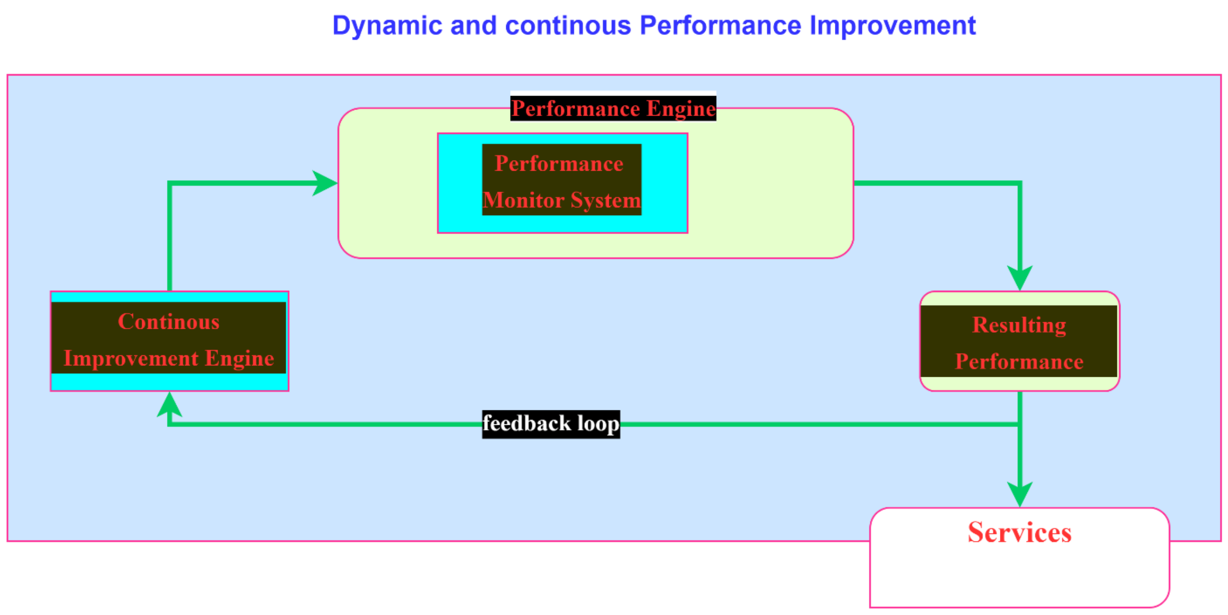
Dynamic and continous Performance Improvement
Performance Engine
Performance Monitor System
Continous Improvement Engine
Resulting Performance
feedback loop
Services

Fig. 67. Performance Improvement Infrastructure.

Dynamically improving systems can also fix bugs and other software problems before preventing the system from running. For instance, the system can dynamically heal itself by continuously running a self-diagnostic process. The self-healing behavior results in better system *availability* and *reliability* features. Fig. 64 shows typical self-healing system components infrastructure.

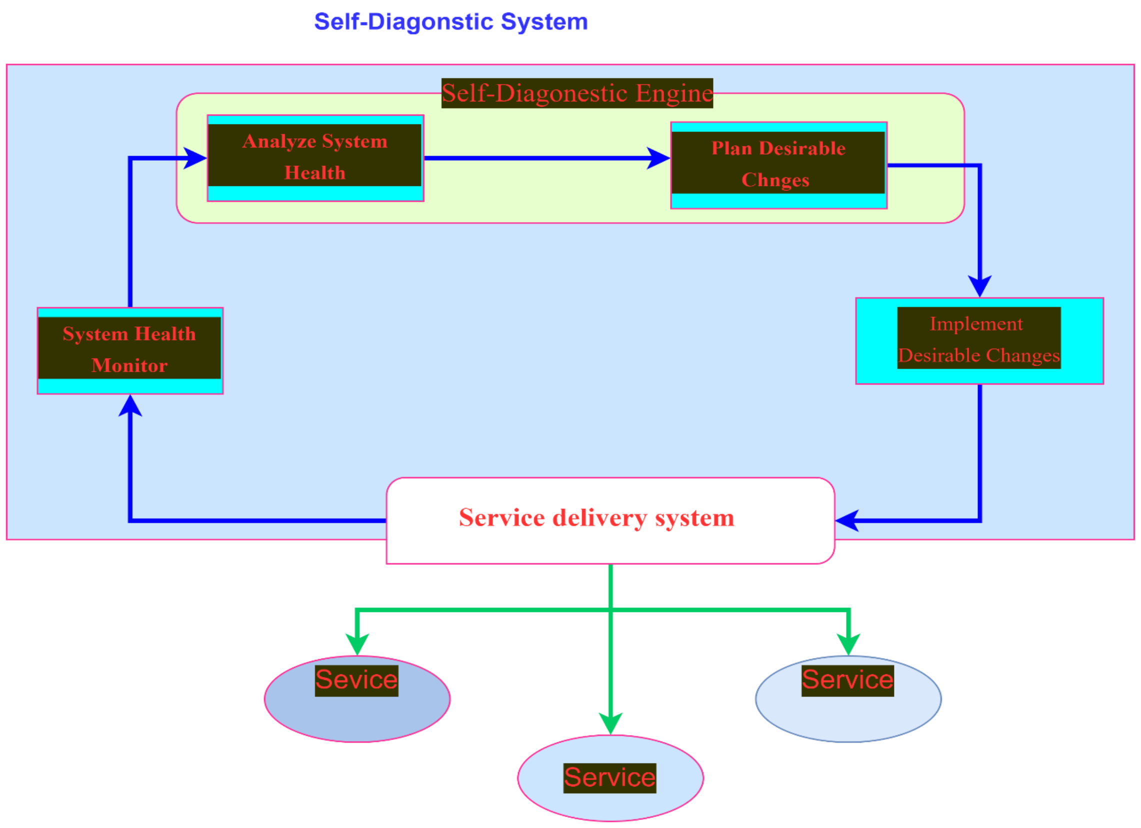

Fig. 68. Self-Healing Process Infrastructure.

A system's continuous and dynamic improvement features can help *secure* the software. Steady and dynamically improving systems can promptly detect malicious software threats and unauthorized and insecure system accesses and take the necessary corrective actions before they pose a danger to the system. A restorative measure can block a suspicious user from gaining access to the system or request additional authentication to ensure the system is not exposed to hackers. On the other hand, the system can remove the malicious software and clean the affected components if malicious software is detected.

Dynamically improving systems are *modifiable* since they cannot change unless adjustable. Therefore, modifiability is an inherent property of dynamically enhancing systems. A software system’s modifiability is not limited to changes in the volume of the software components and can apply to any observable change, such as the system's performance. Hence, the above discussion on the system’s ability to accommodate performance, security, and availability changes is sufficient to say that a software system's continuous and dynamic improvement feature embraces modifiability quality attributes.

Continuously and dynamically improving systems are easy to *test* because the systems display simplicity, observability, stability, and controlled state changes. A system’s improvement should always lead to simplicity, as complexity is not a desirable property. If a system is transparent (clarity), it is also observable. A dynamically and continuously improving system is more stable (dependable) than a static system. Furthermore, it is arguable that a dynamically and constantly improving system has a mechanism to control the state change of the system. Hence, we can conclude that a system with continuous and dynamic improvement features is also testable.

Software improvement can involve adding, subtracting, or modifying components. A software system’s component addition, removal, or modification process involves structural changes requiring software *integration*. Therefore, a continuously and dynamically improving software system must accommodate the integrability quality attribute to incorporate the desired changes to the system.

7.2.6. Content filtering

Content filtering is a mechanism to selectively allow content to pass while blocking irrelevant or inappropriate content. The organization that owns the local network determines the contents to be accessible and that need to be stopped. The main target of content filtering is the organization's *security*. However, content filtering can also enhance the system's *usability* and *performance***.**

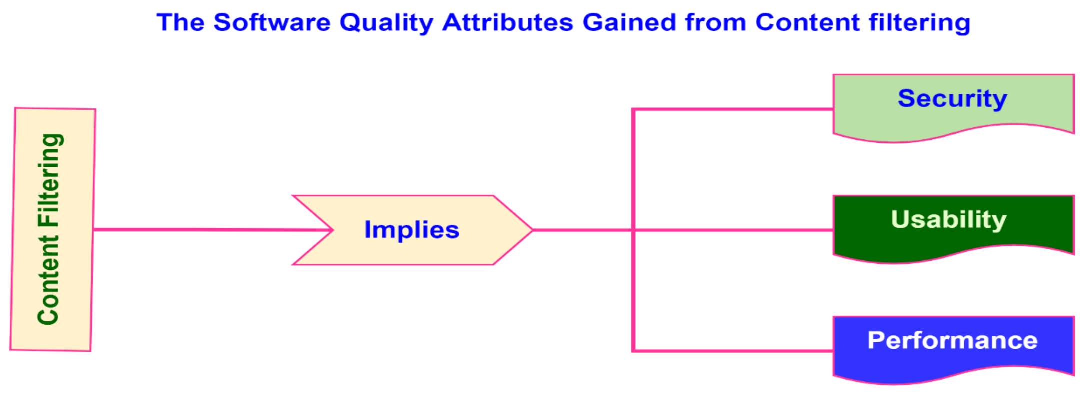
The Software Quality Attributes Gained from Content filtering
Content Filtering
Implies
Security
Usability
Performance

Fig. 69. Quality Attributes Derived from Content Filtering.

Content filtering enhances the performance of the system by increasing bandwidth efficiency. The bandwidth limits the amount of data traffic that can get through the network per unit. Limiting the amount of data that can get through the network improves bandwidth efficiency. Efficient bandwidth leads to better system performance, making data traffic faster and more reliable.

Content filtering can also improve the usability of the system. Users will see and interact with appropriate and valuable data only. Data suitability eliminates worries and builds confidence making the system trustworthy. Users tend to engage and utilize the system if they feel safe and trust it. System trustworthiness is essential in the usability of an education system as it significantly boosts the family's confidence.

Lastly, content filtering enhances the system's security by blocking malicious software from entering the system. Enhanced security is the backbone of system *availability*, *usability*, and *deployability***.** An effective content filtering mechanism is necessary for lasting system health and security.

Chapter 8: Comparing the Proposed and Example architectures

Building software systems cost tremendous resources and money from its inception to its production as it involves many developers, lengthy time, and substantial budgetary. Such labor, money, time, and other resource-intensive product must be undoubtedly valuable. However, measuring the usefulness of a software system at the beginning of its production phase is challenging. Nevertheless, the manufacturer must ensure the product's accuracy and effectiveness by measuring the final product's benefits since remaking after failure is not an option. Hence, the architect must assess the final product's usefulness before launching its production.

8.1. Comparison Method

The process of measuring the usefulness of a software product is known as architectural evaluation. Agren et al. defined architectural evaluation as "…the activity of evaluating the architectural design decisions of an (envisioned) system to build confidence that the system can fulfill the stakeholder concerns" [84]. Chauhan and Babar offer a more precise definition of architectural evaluation as "… the analysis of the capability of the system to satisfy the most important stakeholder concerns based on its large-scale design …" [85]. Furthermore, Bass described architectural evaluation as "... the process of determining the degree to which an architecture is fit for the purpose for which it is intended" [52].

Architectural evaluation is a complex and tiresome multi-stage process that requires expert-level knowledge, stakeholder involvement, budget, and years of preparation [52]. Therefore, evaluating the proposed architecture according to the well-known **Architectural Evaluation** *Methods* (Base et al. Chapter 19) [52] is farfetched. Nevertheless, for this research paper, we can assess the architecture for software quality attributes in terms of its *components* and *functionalities*. The chapter will decompose and analyze the two software characteristics and compare the proposed and example architectures to see how well they satisfy the criteria.

8.1.1. Software component

A software component is “A unit of composition with contractually specified interfaces and explicit context dependencies only” [86]. Furthermore, Lau and Cola described a software system component as the part of the software deployable independently and is subject to composition by a third party. Lau and Cola typified components as a software system with the following characteristics:

- It has none or some clients (a software element).
- It has an official usage description (used by a client author)
- It is independent (not tied to any fixed set of clients).

Components are used for managing the complexity of an extensive system application by breaking down the system into smaller pieces. Moreover, components are a good way of abstracting away the complexity of their implementation behind an interface. In addition to abstraction, software developers use components to swap in and out interchangeable software parts to control features, versions, or software upgrades [87].

As discussed in previous chapters, the proposed architecture has multiple essential software systems features that make it indispensable for the learning environment. For example, the adaptability feature is vital to customized learning. The interactive feature promotes student engagement and creates a cooperative atmosphere for student participation. Autonomy enables students’ independent learning by establishing an individualized learning path. Decentralization enhances localization by dispersing control and fostering modification of contents and other resources. Continuous and dynamic improvement dynamically applies changes to the learning environment to accommodate variations in learning differences. Content filtering makes the learning environment safe and secure. Concurrency enhances the performance of the system.

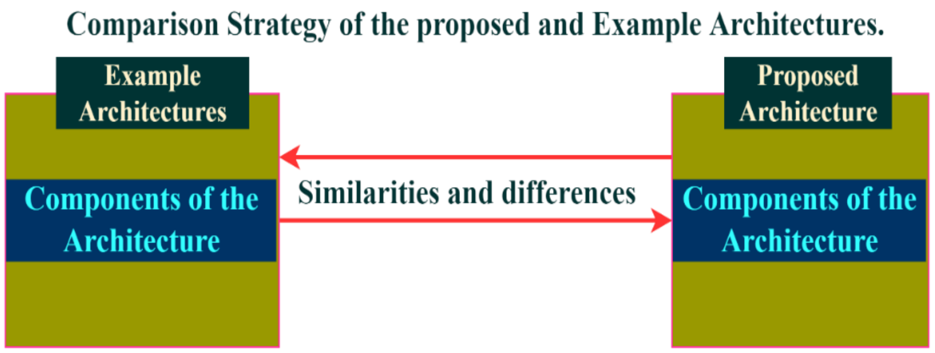
Comparison Strategy of the proposed and Example Architectures.
Example Architectures
Proposed Architecture
Components of the Architecture
Similarities and differences
Components of the Architecture

Fig. 70. Comparison Strategy.

8.1.2. Organization of Software Components

A software component is a part of the system or an application that breaks down the system's complexity into manageable pieces. A software system's componentization (breaking down into parts) provides an additional security layer since it hides the implementation of each component behind an interface. In addition, it is suitable for software development, maintenance, and operation [87].

A software system will generally have the view, the controller, the model, and the data access object components categorized under the architectural composition. The components talk, maintain coordination, and perform different tasks.

There are various benefits to using MVC architecture. First, the MVC organization provides easy and secure access to the data users need by providing an interface between the business logic and the UI. The UI does not have direct access to the data nor assume complete control over the presentation of information. Data is accessed through what is known as the *Controller*, and the *View* presents the requested information.

Second, the MVC organization is suitable for developing large-scale web applications. The development of a large-scale application, such as an educational system, requires the involvement of a large team of developers. The MVC architecture makes the work of developers easy by allowing them to handle different pieces of the structure independently. Since developers can work on the part of the software separately, fast development is guaranteed.

Other benefits of MVC structure include easy modifiability, asynchronous method invocation (AMI), easy maintainability, and easy testability. Fig. 71 depicts the overall representation of the MVC organization.

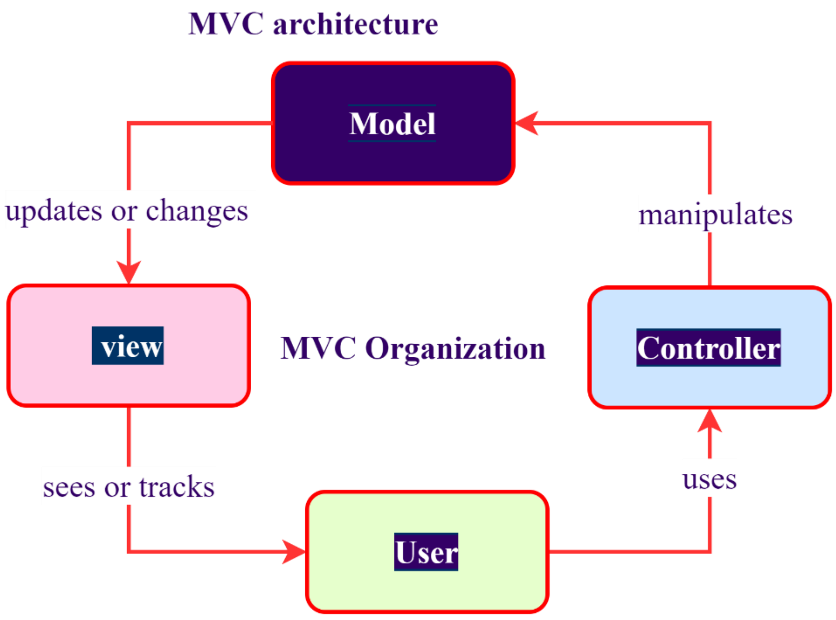

Fig. 71. Structure of MVC.

The proposed architecture follows the MVC organization. Nevertheless, it is not easy to identify and categorize each element of the whole structure into these three pieces. The difficulty is due to the complexity of the application inherent in its complex tiers with sophisticated communications and interactions. One approach would be to consider each MAS as standalone software and organize each software into an MVC structure. Nevertheless, this is still not easy as the MASs maintain complex interactions among themselves.

Consider the Adaptive MAS architecture. The adaptive MAS consists of nine software agents and an AMAS facilitator. When the user interacts with the AMAS, the UI facilitator sends a message to the AMAS facilitator. The AMAS facilitator would redirect the message to the right agent. The agent sends the response back to the AMAS facilitator and the AMAS facilitator to the UI facilitator. In this case, the facilitators play the role of both the controller and view. The individual agents play the role of a model. The following figure depicts the scenario described here.

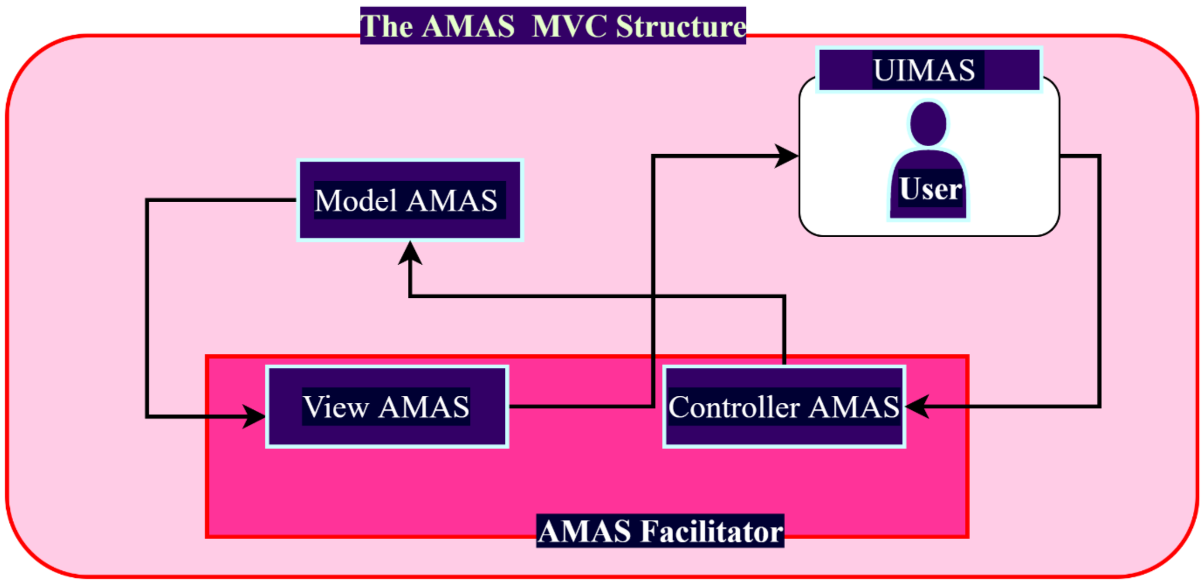

Fig. 72. The AMAS MVC Structure.

In Lin's architecture, the web services act as controllers and views. It receives the information (message) from the user and determines which agent to contact (controller) to get the correct response. When a reply is available, the web service determines how to render the information (view). Due to this, we can think of the services registry as the facilitator in the proposed architecture. In the case of Lin's architecture, each web service (Learning Object Repository, Learner/Instructor Information, Course Information, etc.) behaves as the local facilitator. The individual agents play the role of a model.

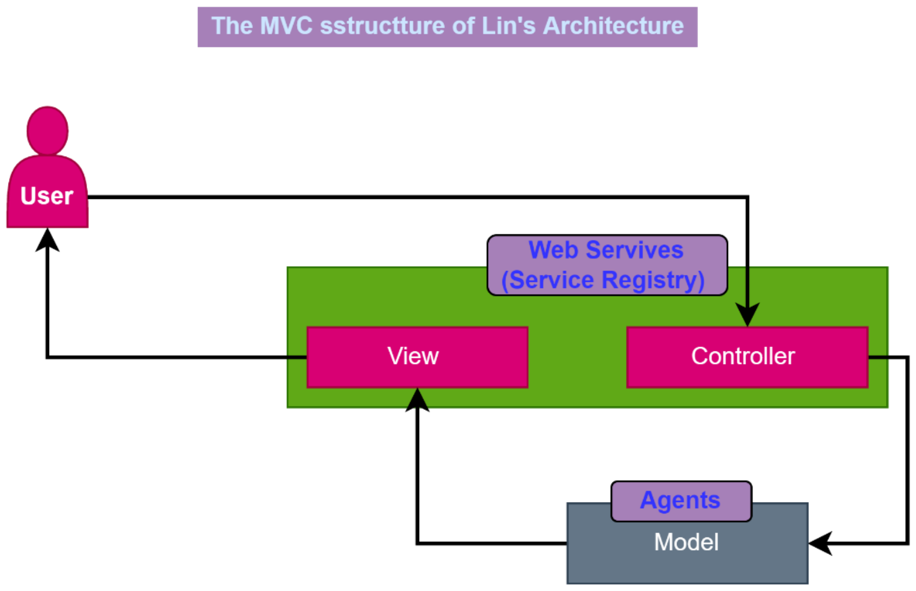
The MVC sstructture of Lin's Architecture
User
Web Servives (Service Registry)
View
Controller
Agents
Model

Fig. 73. MVC Structure of Lin's Architecture.

Abdelhamid's architecture does not clearly depict the MVC structure. The user conveys a message to the user's webpage that communicates directly with the user's agent. The user agent exchanges messages with the smart advisor agent that uses machine learning algorisms to manage other agents. Considering the direction of information flow, one can think of the web page as the view, the user agent as a controller, and the smart advisor agent as the model. However, the smart advisor agent manages other agents (Performance Agent (PA), Scheduling Agent (SA), and other user agents), which makes it a control component of the MVC model. In the forward direction, the web page acts as the passage medium only, while in the reverse route, the user agent acts as a passage medium only. This scenario is shown in the diagram below.

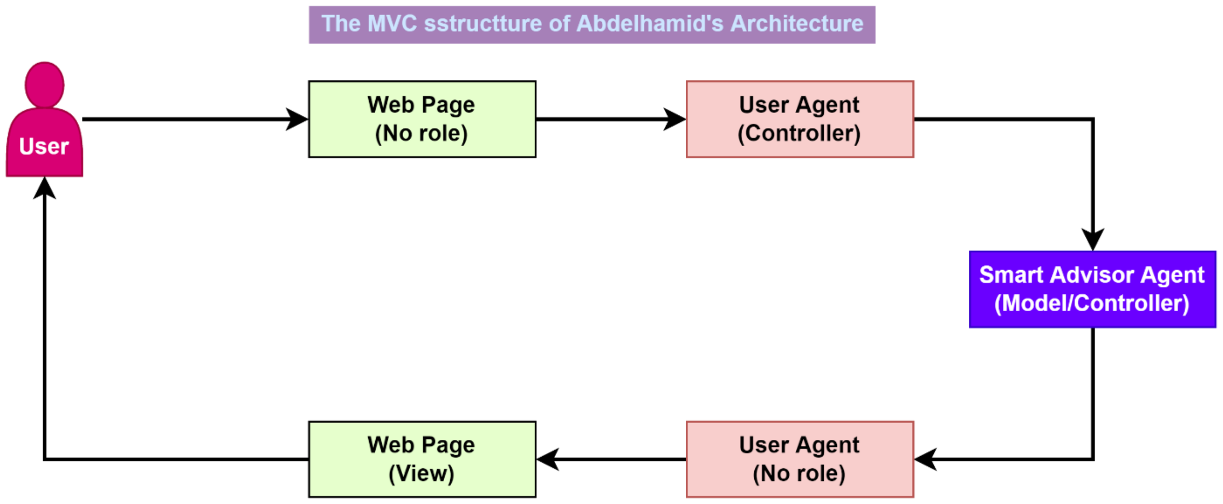

Fig. 74. MVC Structure of Abdelhamid Architecture.

Considering MVC frameworks, Hammami's architecture is the most versatile architecture with complex communication patterns. The Interface Blackboard acts as the UI facilitator in the proposed architecture. The Authoring and Learning Blackboards work the same way as local facilitators in the proposed architecture. Furthermore, similar to the proposed architecture, one can view a MAS as a component with its own MVC structure since it consists of independent agents. Consider the Interface MAS. The MAS consists of the Interface Blackboard, the authoring interface agent, and the learning interface agent. Suppose a learner is interacting with the Interface Blackboard (IB). In this case, the IB acts as the controller and the view, while the Learning Interface Agent (LIA) acts as a model.

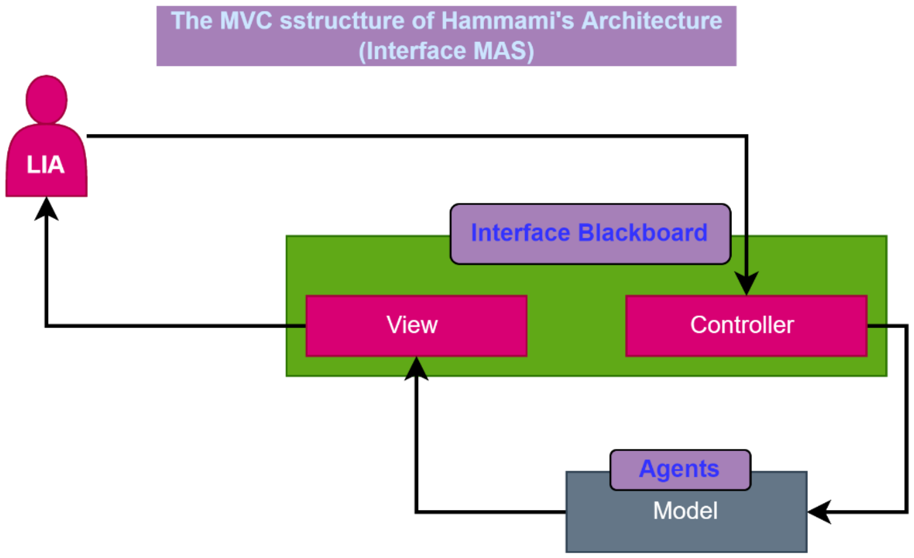
The MVC sstructture of Hammami's Architecture (Interface MAS)
LIA
Interface Blackboard
View
Controller
Agents
Model

Fig. 75. MVC Structure of Hammami's Architecture.

8.1.2.1. The View Component

The view is the part that defines data rendering methods and processes. The followings are the characteristics or actions of the view component [88].

- The information displayed by view can have any form: tables, graphs, diagrams, charts, texts, etc., or a combination of these data.
- Data can have multiple views based on the user preferences and use of the displayed data.
- The view ultimately reflects the User's actions which the controller controls.

In the proposed architecture, the View AMAS is the subcomponent of the AMAS facilitator that processes the agents' responses and determines how to render them. Each facilitator will decide on the rendering method locally so that the UI facilitator will not be overwhelmed by the number and complexity of the decision-making needed for each agent response. In this case, the UI facilitator view component will act as a liaison between the user and the AMAS view facilitator.

Similarly, in Lin’s architecture, each web service plays the role of a view. However, whether the rendering is done locally within each web service or coordinated in the service registry is unclear. Nevertheless, each web service will render the response from the agents on the course web page. In Abdelhamid’s architecture, the users’ web pages take the role of a view. The system's processed information will render on the user web page. Abdelhamid’s architecture uses separate web pages for each user category. Hammami’s architecture has a similar view structure to the proposed architecture.

8.1.2.2. The Controller

The controller monitors the components to run upon a request or an event. When an internal process or a user request fires an event, the controller will handle the event and decide what to do based on the event’s nature. The controller’s action is to update the model. The controller displays the following characteristics [88].

- The controller receives input from the user or an internal process, optionally validates the input, and performs interactions on the data model object.
- The controller may not change the view directly; it interacts with the view only through the model.

The controller of the proposed architecture resides inside the facilitator, as is the view. The controller is responsible for processing the user request and directing it to the proper agent to respond. For instance, if the user wants to continue working on a course and invokes the same request in the UIMAS, the UI facilitator will redirect the request to the AMAS facilitator. Next, the controller in the AMAS facilitator will process the request, identify it, and turn it to the Course Presentation Agent (CPA). The CPA will then send the next part of the course to the AMAS facilitator view to determine the best rendering for the part of the course. Finally, the piece goes back to the UIMAS facilitator to display.

The controllers in the example architectures have similar structures with few differences. For example, the controller in Lin’s architecture uses a web registry (web services) instead of complex facilitators. The controller in Abdelhamid’s architecture is not that clear. The architecture may employ the user agent as a controller or the smart advisor agent as a controller. It is important to note that Abdelhamid’s architecture is too simple to implement the complex MVC architecture for large and complex software systems [88]. Hammami’s architecture uses a similar structure and approach to the proposed architecture to implement a controller

8.1.2.3. The Model

The model is the central component of the MVC structure. It contains the application’s dynamic data, manages it, applies logic and rules to it, and independently changes the view from the user interface. The controller moderates the model’s actions. The model passes its action to the view, and the view renders it. The model shows the following characteristics.

- The model does not decide what the view should display; instead, it executes the controller's decision and passes it to the view.
- The model hosts the data, logic, and rules for what to execute; it does not call a method or pass parameters used in the view or controller processes.

In the proposed architecture, the individual agents act as models. Consider again the AMAS discussed above. In the AMAS component, nine independent agents represent the model collectively. However, since the models belong to independent agents, their actions do not interfere with each other. Nevertheless, the same controller manages different models, and the models use the same view to render the executed command.

In the example architectures, different entities or components of the software system represent the model. For instance, in Lin's architecture, the agents' platform that holds numerous agents plays the role of a model. In Abdelhamid's architecture, the smart advisory agent or the user agent represents the model. Again there is unclarity here. Due to the application's simplicity and the vague representation of the components' communication protocol in Abdelhamid's architecture, it is hard to say if a component represents any part of MVC with certainty. Finally, in Hammami's architecture, each MAS agent represents the models that closely resemble the proposed architecture.

Conclusively, the example architectures do not don't mention the MVC structure. However, some example architectures, such as Hammami's architecture, follow the layered (n-tier) architecture framework built around the MVC structure. Hence, we can safely anticipate that Hammami's architecture has the MVC structure, although not explicitly mentioned in the article. The other architectures do not mention the MVC structure at all. Nevertheless, one can safely approximate the structures to an MVC structure for comparison only.

8.1.3. The functionality of the software

In an online blog, Paget compares features and functionality and describes them as: "Features are the 'tools' you use within a system to complete a set of tasks or actions. Functionality is how those features actually work to provide you with a desired outcome" [89]. Furthermore, Bass defines functionality as: " … the ability of the system to do the work for which it was intended" [52]. According to Paget, features are merely vital if they do not carry the desired functionality [89]. For instance, a car may have tires [feature] that are not good for all seasons [functionality]. We barely depend on software features to accomplish a task; a software feature's functionality makes it desirable.

According to various online blogs, the desired operations in an educational software system are *data storage*, *content organization*, *content management* and *delivery mechanisms*, and *communication services*. The following sections will briefly discuss each functionality and how the two architectures exhibit them.

8.1.3.1. Data Storage

Educational systems' data storage is a vital part of the LMS and yet a challenging task [90]. Educational institutions store classified data on students, faculty, and staff. In addition, schools collect a vast array of data from different sources and keep it for reference, analysis, and use. Data storage and analysis are critical for learning enhancement and educational innovation.

Data storage involves two separate steps. The first is the data collection mechanism, where data is securely and correctly collected and redirected to its storage or data warehouse. This step will also categorize or group the collected data. The classified data is transported to its correct destination and stored in the proper location.

The second step involves data storage. The data storage step requires a storage space (Usually a data warehouse) to store the data. Once the data storage mechanism identifies the storage location, it must select the correct destination and deposit it correctly.

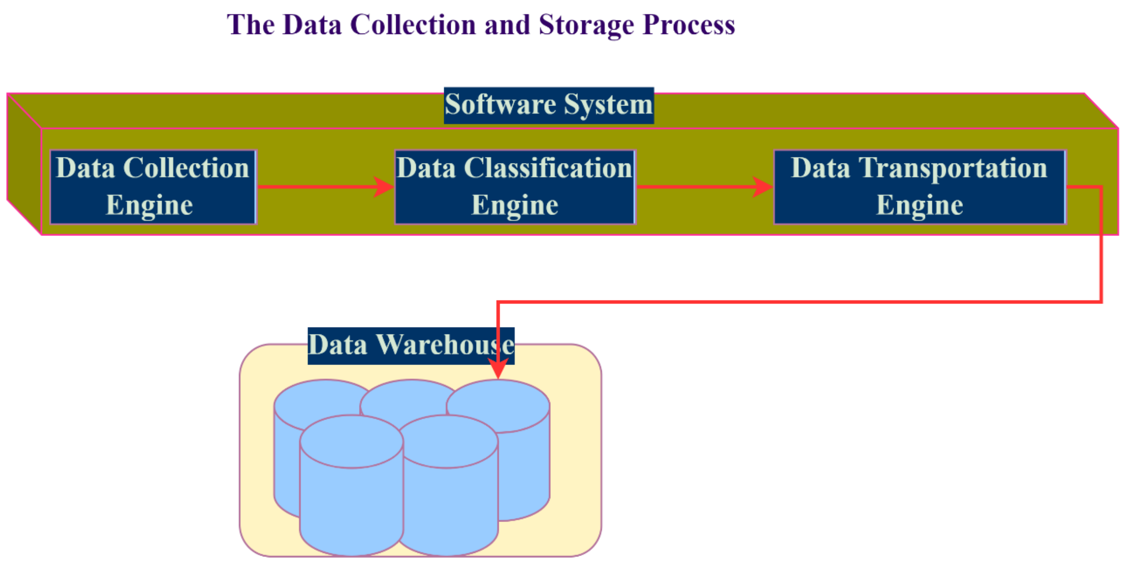
The Data Collection and Storage Process
Software System
Data Collection Engine
Data Classification Engine
Data Transportation Engine
Data Warehouse

Fig. 76. Data Collection and Storage Process.

The proposed educational system has data collection and storage functionality. In the proposed software architecture, the MAS facilitators provide the data collection engines that initiate data collection upon request from its agent. The MAS facilitator also contains a data transportation engine to send data to the warehouse. Once the information arrives at the data warehouse, the database interface will pass it through a data classification engine to determine its correct destination and deposit it.

The example architectures also provide similar data collection mechanisms to collect and store data. The example architectures do not specify whether data is stored locally on the system machine or transported to the warehouse for storage. Lin does not explicitly specify the data collection and storage methods. In Abdelhamid's architecture, the different agents are directly connected to the databases, implying agents' involvement in data collection and storage. Hammami’s architecture provides an interfacing mechanism (blackboard) that connects various agents to the databases. Although not mentioned explicitly, it is evident that Hammami's architecture provides a means to collect, categorize, transport, and store data.

8.1.3.2. Content Organization

Organizing information to make it easily accessible is known as a content organization [91]. In the context of this research paper, content organization refers to how the system handles the arrangement of educational materials. The content organization makes data accessible and minimizes the response time to its access request. According to an online article by Malamed, there are ten ways to organize instructional materials [91]. However, I will discuss only the three methods most related to the proposed architecture.

8.1.3.2.1. Categorical organization

If contents have a flat structure with no hierarchical, sequential, or prerequisite relationships, the contents can be organized categorically [Malamed]. Contents in different subject areas have a flat structure. For example, mathematics and history subjects are unrelated hierarchically, sequentially, or incrementally. Therefore, the two disciplines can follow the categorical organization of contents.

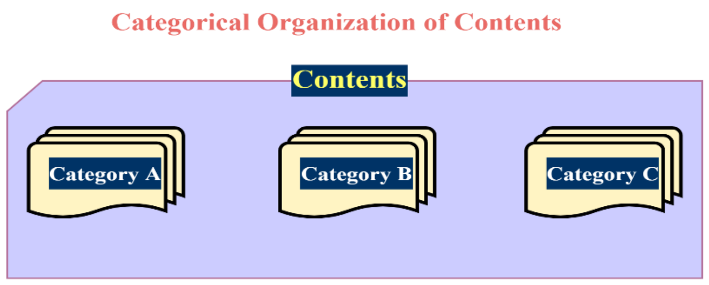

Fig. 77. Categorical Organization.

The sub-contents in each category can follow one or more content organization logics that we will discuss in the following subsections. For example, the contents in category A can follow other organizational methods. Evidently, the content organization in an educational system is more effortless if the contents are first organized categorically

8.1.3.2.2. Organization by Complexity level

The most prominent way of organizing instructional content, as evident from academic materials and academic institutions' course structures, is by using the level of complexity. Academic institutions, textbook authors, and educational policymakers tend to organize educational content from simple to complex even when the simpler content is not subordinate or a prerequisite to the complex content.

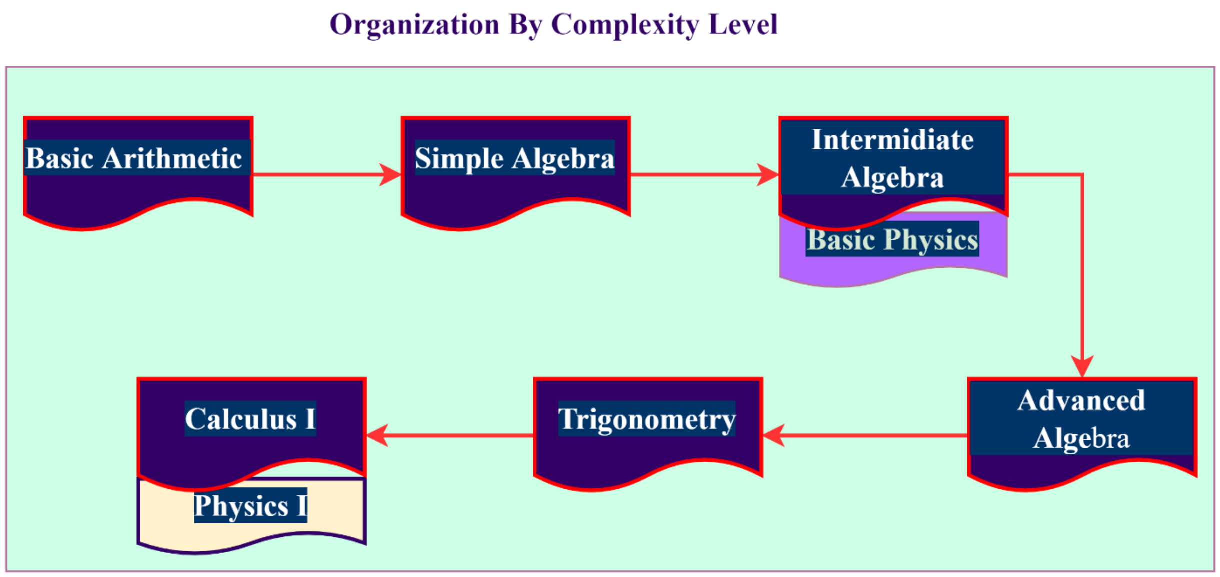

Fig. 78. Simple to Complex Organization.

Organization by complexity levels provides a slow initiation into a subject that gradually builds the learner’s confidence and knowledge [92]. Fig. 73 shows a simple to complex content organization structure that contains basic skills required for the next higher level. Some levels contain categorical organizations, such as the intermediate algebra and calculus I levels.

8.1.3.2.3. Hierarchical Organization

Hierarchical organization comes to play when the content at a higher level requires the mastery of the knowledge domains at a lower level. Hierarchical organization is one of the content organization strategies used widely [92]. Hierarchical content organization is effective as human learning efficiency depends on the existing knowledge. We learn by stratifying one piece of knowledge on top of another knowledge.

It is self-evident for architecture to display one or more of the content organization mechanisms discussed above. The proposed educational architecture uses a pool of resource databases to pull out target contents and organize the contents based on age, grade level, existing and acquired skills and knowledge, and learners' learning preferences. The Resource Database keeps the resources corresponding to the educational level. The Performance Database stores information related to the student's performance. The User Database stores user-related information. Finally, the system adapts the resources to the learner's needs and stores them in the personalized contents database.

On the other hand, the example architectures use databases, but the organization of resources in the database is unclear. For instance, the architecture by Lin does not provide a clue about how the architecture uses databases at all. The system identifies databases corresponding to the various agents in the Abdelhamid architecture. For example, the students' and Instructors' profiles & Workload databases are associated with the performance agent. However, the organization of the contents is not clearly defined. Hammami's architecture displays similar characteristics to Abdelhamid's architecture concerning content organization.

8.1.4. Content Delivery Mechanisms

Content delivery mechanisms are processes and methods used by the instructor to effectively and efficiently transfer knowledge to students. The teacher employs various techniques to transmit knowledge properly. For example, lecturing is the most common technique of conveying information in a classroom setting. Educators use discussions to challenge the students with the end goal of developing the student's critical thinking. Techniques such as laboratory activities, field visits, practical and participatory activities, etc., also known as active learning, effectively engage students.

In online education, using content delivery mechanisms such as lecturing, discussions, practical and participatory activities, field trips, etc., is challenging and sometimes infeasible due to their inherent nature. Therefore, it is crucial to look for alternative approaches to make them efficient or substitute them. Next, we will discuss some delivery mechanisms suitable for online education.

8.1.4.1. Media Options

Content delivery methods such as audio recordings, video presentations, and images are more suitable than other forms of delivery for online teaching. An online article, "*Why Use Media to Enhance Teaching and Learning,*" states that using media options enhances learning by simplifying abstract, new, and novel concepts [96]. Moreover, media options are more accessible than other forms of presentation.

Learning resource accessibility enhances the students' engagement and performance. For example, audio recording is more accessible because the student doesn't need a PC or laptop to listen to the audio. Moreover, the student can access the audio and listen to it on the go. The audio files are usually easily repeatable if the need arises. Hence, the availability of audio options increases accessibility and encourages students' engagement in the learning process. The same applies to the other media options, video presentation, and use of images.

8.1.4.2. Virtual Classroom

A virtual classroom is an online learning and teaching environment where students and their teachers connect virtually, interact with each other, and engage in the process of a course presentation. Virtual classrooms use video conferencing tools such as zoom and Microsoft Teams, providing easy interaction and engagement features. Most virtual classroom tools have added features to simulate the classroom situation and moderate student participation.

A virtual classroom is desirable in an online learning environment because it provides features beyond the in-class experiences besides the interactive and engagement features. The classroom environment and the course materials are accessible to the teacher before the class starts, and they will remain accessible to students and teachers after the class ends. Moreover, the classroom sessions are accessible globally using an internet-enabled device with a video streaming feature.

8.1.4.3. Interactive Packages

Interactive packages are tools used to enhance online course delivery. Today, most LMSs provide interactive packages as part of the software system. The packages are either developed and owned by the software vendor or are third-party tools incorporated into the LMS. For example, in the Schoology LMS, the teacher and students can access their google drive and tons of other third-party software.

The contents of an interactive package are used in course preparation, delivery, and management [56]. For instance, Visme, a design tool used to prepare presentations, infographics, concept maps, schedules, reports, etc., contains interactive packages with advanced features. The google classroom, an LMS part of the G Suite, is an interactive package that provides indispensable features such as lesson creation, quiz preparation, and documents all in one place.

The proposed architecture displays content delivery mechanisms supported by agents. As discussed in detail in chapter three, the Adaptive MAS (AMAS), the Interactive MAS (IMAS), the Decentralization MAS (DMAS), and Content Filtering MAS (CFMAS) agents are involved in content delivery. The reader can infer the degree of involvement of each MAS and its corresponding agents from the detailed explanation in chapter three.

Like the proposed architecture, all the example architectures contain a content delivery mechanism. For example, the Tutoring Agent (TA) and Course Delivery Agent (CDA) in Lin’s architecture, the Schedule Agent (SA) and Smart Advisory Agent (SAA) in Abdelhamid’s architecture, and the Course presentation Agent (CPA), Practice Generator Agent (PGA), and Course Unit Generator Agent (CUGA) are directly or indirectly in charge of course delivery.

8.1.5. Communication Services

An educational software system must provide a means of communication with easy-to-use features for information exchange between students and instructors and among the students. Communication services include video chat, text messaging, email, instant messaging, and video or audio calls. Modern LMSs [55] use built-in applications that facilitate communication among various software users.

Communication service-providing applications are a critical component of educational software systems as they facilitate communication and promote interactions between students and their teachers and among the students. An educational software system may use built-in applications to provide communication services or third-party applications integrated with the LMS. In a learning environment, built-in applications are preferred to third-party application integrations as they provide enhanced security.

Communication services can be automated or manual (user-initiated). Automated communication involves sending out messages or notifications as a batch to a list of recipients. A joint event will trigger the broadcasting of automated messaging. Automated messaging is an efficient means of communication when the message recipients of the notice constitute a group of people. Automated communications are standard in notifications, grade posting, assessment posting, or due date notifying.

Manual communications are communications that a sender initiates. An event does not trigger the messages in manual communications. The sender can send messages to a particular receiver or a group of receivers. Manual communications are the standards in emails, peer-to-peer communications, and private messaging services.

The Socialization Agent (SA) in the Interaction MAS (IMAS), the Learning Feedback Agent (LFA) in the Adaptive MAS(AMAS), the Feedback Agent (FA) in Autonomy MAS (AMAS), and the Resource Dispatching Agent (RDA) in Decentralization MAS (DMAS) are involved in communication one way or the other. Some agents, such as the FA, are used for internal communications, while others are engaged in external communications.

Similarly, the example architecture consists of agents involved in providing communication services. For example, the Advisor Agent (AA), Tutoring Agent (TA), and Notification Agent (NA) in Lin's architecture participate in communication services. Although not mentioned explicitly, the Smart Advisor Agent (SAA) in Abdelhamid's architecture might participate in communication services. Furthermore, although unclear from the structure of Hammami's architecture and Hammami didn't explicitly mention communication services, since communication modules are part of the software component, the architecture may involve at least internal communication services.

Table I: Functional Comparison of the Proposed Architecture and the Example Architectures

Functionalities	Proposed Architecture	Example Architecture
Data storage	Involves data collection, data classification, data transportation, and data warehouse.	All examples exhibit data collection and storage. Some involve data classification and transportation.
Content organization	Contents are organized categorically, at the complexity level, and/or hierarchically	The content organization strategy is unclear or was not described clearly.

Table I: Functional Comparison of the Proposed Architecture and the Example Architectures (Continued)

Functionalities	Proposed Architecture	Example Architecture
Content delivery mechanisms	The content delivery mechanism involves various agents: content delivery forms include media options, virtual classrooms, and interactive packages.	The content delivery mechanism involves various agents: content delivery forms are not known or included in the report.
Communication Services	The architecture supports both internal (among agents) communication services and external (between agents and the environment) communication services	The architectures support at least one type of communication service. Some example architectures support internal and external communication services, and others' involvement is unclear.

Chapter 9: Conclusion

Since its inception, the modern education system has undergone evolutionary transformations over the last five centuries [44]. During this long period, education became firmly structured and centrally controlled by the government or the school owners. Centralized education made learning less accessible by limiting educational freedom and obstructing independent learning. For instance, schools taught only the subjects they chose to teach; students lost control over what they should discover; instead, students were taught what they should know [1].

We must look back at our educational organization and design a more healthy and comprehensive education system that permits students to control their learning paths. This research paper recommended a modern education system that transforms education from traditional to computer-based by introducing a multi-agent architecture with intelligent software characteristics.

A computer-based educational system allows independent and personalized learning experiences [49]. However, an educational system can realize independent and customized learning if the system satisfies the smart educational system characteristics such as adaptivity, interactivity, autonomy, dynamic and continuous improvement, content filtering, and concurrency. Hence, the proposed educational system introduced an alternative approach to realizing an independent and adaptive education system.

The research paper presented the characteristics of an intelligent multi-agent educational system, discussed each feature in detail, and assessed the feasibility of the software system using a comprehensive comparative system evaluation. Furthermore, the research paper presented a few example architectures and compared the software features, functionalities, and components in the two architectures to understand the proposed system architecture characteristics.

9.1. Research Summary

- Many scientists and educational scholars believe the traditional education system is insufficient to meet society's educational needs. Therefore, they argued that the conventional delivery method must be transformed or supplemented with a flexible and decentralized system.
- Researchers consistently and constantly presented various architectures that can support flexibility and decentralization. As a result, they recommended an online intelligent, adaptive, and personalized learning platform as an alternative to the conventional classroom-based delivery method.
- A thorough observation, research exploration, and reasonable judgment revealed that a coherent online learning system should include all possible learning factors, such as social, interactional, cognitive, adaptive, personalization, and security.
- The researcher believes the system's design must support multi-agent architecture to realize a comprehensible online learning system architecture with all possible learning factors. Hence, the research focused on an intelligent multi-agent educational system with various characteristics that reflect the necessary functionalities.
- The research thoroughly evaluated the proposed architecture based on recognized software quality attributes. The evaluation results confirm that the prescribed architecture demonstrates practical feasibility.

9.2. Recommendations and Future Works

Based on these conclusions, future researchers should consider exploring the prescribed architectural characteristics with the possibility of incorporating new ideas and insights or excluding intangible concepts. Although the research recommends five architectural characteristics as the basis of the software system, the researcher refrains from concluding that these are the only possible features.

Furthermore, future studies should consider ground-breaking research work applying software quality attributes to the proposed architecture (Chapter 6). The research attempts to draw software quality attributes by analyzing the software features and interpreting the analysis. It is also possible to gauge each software feature in light of the software quality attributes to establish consistency.

References

[1] A. Arenas, “The Intellectual Development of Modern Schooling: An Epistemological Analysis,” *Universitas Humanística*, no. 64, pp. 165–194, Dec. 2007.

[2] X. Basogain, M. Á. Olabe, and J. C. Olabe, “Transition to a Modern Education System through e-Learning,” in *Proceedings of the 2017 International Conference on Education and E-Learning*, New York, NY, USA, Nov. 2017, pp. 41–46. doi: 10.1145/3160908.3160924.

[3] H. Peng, S. Ma, and J. M. Spector, “Personalized adaptive learning: An emerging pedagogical approach enabled by a smart learning environment,” *Smart Learning Environments*, vol. 6, no. 1, p. 9, Sep. 2019, doi: 10.1186/s40561-019-0089-y.

[4] S. Mitra, “Using UML Modeling to Facilitate Three-Tier Architecture Projects in Software Engineering Courses,” *ACM Trans. Comput. Educ.*, vol. 14, no. 3, p. 17:1-17:31, Oct. 2014, doi: 10.1145/2635831.

[5] O. Zine, A. Derouich, and A. Talbi, *Hypermedia multi-agent modeling: a proposition of a learner model agent based on ontologies*. 2016.

[6] C. Xie, “Topic Modeling-based Web Architecture Redesign,” in *Proceedings of the 38th ACM International Conference on Design of Communication*, New York, NY, USA, Oct. 2020, pp. 1–2. doi: 10.1145/3380851.3418623.

[7] P. J. Durlach and A. M. Lesgold, Eds., *Adaptive Technologies for Training and Education*. Cambridge: Cambridge University Press, 2012. doi: 10.1017/CBO9781139049580.

[8] J. Okpo, “Adaptive Exercise Selection for an Intelligent Tutoring System,” in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, New York, NY, USA, Jul. 2016, pp. 313–316. doi: 10.1145/2930238.2930369.

[9] C. Raibulet, F. Arcelli, S. Mussino, M. Riva, F. Tisato, and L. Ubezio, “Components in an adaptive and QoS-based architecture,” in *Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems*, New York, NY, USA, May 2006, pp. 65–71. doi: 10.1145/1137677.1137690.

[10] The Design Structure Matrix (DSM). "The Design Structure Matrix (DSM)." (Accessed February 21, 2022). https://dsmweb.org/.

[11] A. MacCormack and D. J. Sturtevant, “Technical debt and system architecture: The impact of coupling on defect-related activity,” *Journal of Systems and Software*, vol. 120, pp. 170–182, Oct. 2016, doi: 10.1016/j.jss.2016.06.007.

[12] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, 1st ed. 2001. Accessed: Feb. 12, 2022. [Online].

[13] N. Subramanian, and L. Chung. "Metrics for Software Adaptability," n.d., 14.

[14] “Understanding The Learning Process To Effectively Differentiate Instruction,” The Center for Literacy & Learning, Oct. 26, 2010. https://mycll.org/understanding-the-learning-process-to-effectively-differentiate-instruction/ (accessed Feb. 21, 2022).

[15] A. Bennane, “Tutoring and Multi-Agent Systems: Modeling from Experiences,” *Informatics in Education*, vol. 9, Oct. 2010, doi: 10.15388/infedu.2010.11.

[16] S. Ahamed., “The Science of Cliques” SiOWfa15: Science in Our World: Certainty and Controversy, Sep. 08, 2015. https://sites.psu.edu/siowfa15/2015/09/08/the-science-of-cliques/ (accessed Feb. 21, 2022).

[17] A. Bork, “Stand-Alone Computer Systems — Our Educational Future,” *Journal of Educational Technology Systems*, vol. 7, no. 3, pp. 201–207, Mar. 1979, doi: 10.2190/6QPR-U1TR-V9CT-JNN6.

[18] “Traditional vs. Progressive Education: Benefits and Challenges.” School of Education Online Programs, Jul. 14, 2020. https://soeonline.american.edu/blog/traditional-vs-progressive-education/ (accessed Feb. 21, 2022).

[19] K. Letrud, “A rebuttal of NTL Institute’s learning pyramid,” *Education*, vol. 133, pp. 117–124, Jan. 2012.

[20] H. C. Lane, “Pedagogical Agent - An overview.” ScienceDirect Topics, 2016. https://www.sciencedirect.com/topics/computer-science/pedagogical-agent (accessed Feb. 21, 2022).

[21] Federal Communications Commission, “Children’s Internet Protection Act (CIPA),” Dec. 30, 2019. https://www.fcc.gov/consumers/guides/childrens-internet-protection-act (accessed Feb. 21, 2022).

[22] “Internet Content Filtering.” Fairfax County Public Schools. https://www.fcps.edu/resources/technology/digital-citizenship-internet-safety/internet-content-filtering (accessed Feb. 21, 2022).

[23] “Scaffolding Definition.” The Glossary of Education Reform. https://www.edglossary.org/scaffolding/ (accessed Feb. 21, 2022).

[24] M. F. Khan and R. A. Paul, “Dependable System - An overview.” ScienceDirect Topics. 2012. https://www.sciencedirect.com/topics/computer-science/dependable-system (accessed Feb. 21, 2022).

[25] G. Maheshwari, “Factors affecting students’ intentions to undertake online learning: An empirical study in Vietnam,” *Educ Inf Technol*, vol. 26, no. 6, pp. 6629–6649, Nov. 2021, doi: 10.1007/s10639-021-10465-8.

[26] C. Fisher, D. Berliner, N. Filby, R. Marliave, L. Cahen, and M. Dishaw, “Teaching Behaviors, Academic Learning Time, and Student Achievement: An Overview,” *Journal of Classroom Interaction*, vol. 50, p. 19.

[27] L. Pugliese, “Adaptive Learning Systems: Surviving the Storm,” Educause Review, Oct. 17, 2016. https://er.educause.edu/articles/2016/10/adaptive-learning-systems-surviving-the-storm (accessed Feb. 21, 2022).

[28] D. Schuff and R. St. Louis, “Centralization vs. decentralization of application software,” *Commun. ACM*, vol. 44, no. 6, pp. 88–94, Jun. 2001, doi: 10.1145/376134.376177.

[29] B. Cantrill and J. Bonwick, “Real-world concurrency,” *Commun. ACM*, vol. 51, no. 11, pp. 34–39, Nov. 2008, doi: 10.1145/1400214.1400227.

[30] L. Atchison, “5 tips for improving availability,” O’Reilly Media, Aug. 16, 2016. https://www.oreilly.com/content/5-tips-for-improving-availability/ (accessed Feb. 21, 2022).

[31] L. Atchison, “5 tips for improving availability,” O’Reilly Media, Aug. 16, 2016. https://www.oreilly.com/content/5-tips-for-improving-availability/ (accessed Feb. 21, 2022).

[32] F. Bachmann, L. Bass, and R. Nord, “Modifiability Tactics,” Defense Technical Information Center, Fort Belvoir, VA, Sep. 2007. doi: 10.21236/ADA472581.

[33] J. D. Mooney, “Developing Portable Software,” in *Information Technology*, vol. 157, R. Reis, Ed. Boston: Kluwer Academic Publishers, 2004, pp. 55–84. doi: 10.1007/1-4020-8159-6_3.

[34] L. Bass., P. Clements, R. Kazman, “Understanding Quality Attributes in Software Architecture” in *Software Architecture in Practice,* 3rd ed., Addison-Wesley Professional, 2012. Accessed: Jul. 19, 2022. [Online]. Available: https://www.informit.com/articles/article.aspx?p=1959673&seqNum=2

[35] M. Galster, P. Avgeriou, D. Weyns, and T. Männistö, “Variability in software architecture: Current practice and challenges,” *SIGSOFT Softw. Eng. Notes*, vol. 36, no. 5, pp. 30–32, Sep. 2011, doi: 10.1145/2020976.2020978.

[36] C. Stace, “The top five educational trends to look out for in 2021,” Pearson English, Dec. 08, 2020. https://blog.pearsoninternationalschools.com/the-top-five-educational-trends-to-look-out-for-in-2021/ (accessed Feb. 21, 2022).

[37] C. Armes, "2021 Education Trends." Scientific Learning, Jan. 8, 2021. https://www.scilearn.com/2021-education-trends/.

[38] A. Feins, M. Waterman, A. Peters, and M. Kim, “The Teaching Matrix,” Nov. 1996.

[39] P. Gray, “A Brief History of Education” Psychology Today, Aug. 20, 2008. https://www.psychologytoday.com/us/blog/freedom-learn/200808/brief-history-education (accessed Feb. 21, 2022).

[40] F. Lin, P. Holt, S. Leung, and Q. Li, “A multiagent and service-oriented architecture for developing adaptive e-learning systems,” *International Journal of Continuing Engineering Education and Lifelong Learning*, vol. 16, Jan. 2006, doi: 10.1504/IJCEELL.2006.008919.

[41] S. Hammami and H. Mathkour, “Web Application For An Adaptive Multi-Agent E-Learning System: A Continuous Improvement Of E-Content,” *Journal of Web Engineering*, pp. 067–086, Mar. 2014.

[42] P. Q. Dung and A. M. Florea, “An Architecture and a Domain Ontology for Personalized Multi-agent e-Learning Systems,” in *2011 Third International Conference on Knowledge and Systems Engineering*, Oct. 2011, pp. 181–185. doi: 10.1109/KSE.2011.35.

[43] A. Abdelhamid, “Adaptive multi-agent smart academic advising framework,” *IET Software*, vol. 15, Apr. 2021, doi: 10.1049/sfw2.12021.

[44] S. Ahmad and M. Bokhari, “A New Approach to Multi Agent Based Architecture for Secure and Effective E-learning,” *International Journal of Computer Science and Applications*, vol. 46, May 2012.

[45] M. Calisti, F. Dignum, R. Kowalczyk, F. Leymann, and R. Unland, “Service-Oriented Architecture and (Multi-)Agent Systems Technology,” *Dagstuhl News*, vol. 13, 2010.

[46] G. Sun, T. Cui, D. Xu, H. Chen, S. Chen, and J. Shen, “Assisting Open Education Resource Providers and Instructors to Deal with Cold Start Problem in Adaptive Micro Learning: A Service Oriented Solution,” in *2017 IEEE International Conference on Services Computing (SCC)*, Jun. 2017, pp. 196–203. doi: 10.1109/SCC.2017.32.

[47] M. Berna-Koes, I. Nourbakhsh, and K. Sycara, “Communication efficiency in multi-agent systems,” in *IEEE International Conference on Robotics and Automation, 2004 Proceedings. ICRA ’04.* Apr. 2004, vol. 3, pp. 2129-2134 doi: 10.1109/ROBOT.2004.1307377.

[48] H. Xie, H.-C. Chu, G.-J. Hwang, and C.-C. Wang, “Trends and development in technology-enhanced adaptive/personalized learning: A systematic review of journal publications from 2007 to 2017,” *Computers & Education*, vol. 140, p. 103599, Oct. 2019, doi: 10.1016/j.compedu.2019.103599.

[49] J. Rocha, I. Boavida-Portugal, and E. Gomes, *Multiagent System*. IntechOpen, 2017. doi: 10.5772/intechopen.70241.

[50] N. Viswanathan, “Adaptive Multi-Agent E-Learning Recommender Systems,” *arXiv:2012.09342 [cs]*, Dec. 2020, Accessed: Feb. 12, 2022. [Online]. Available: http://arxiv.org/abs/2012.09342

[51] L. Chung and N. Subramanian, “Adaptable system/software architectures,” *Journal of Systems Architecture*, vol. 50, no. 7, pp. 365–366, Jul. 2004, doi: 10.1016/j.sysarc.2003.10.002.

[52] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 4th ed. Addison-Wesley, 2021.

[53] “Intelligent Agents Characteristics,” Jun. 22, 2015. https://www.ukessays.com/essays/management/intelligent-agents-characteristics.php (accessed Feb. 12, 2022).

[54] H. E. Fazazi, M. Elgarej, M. Qbadou, and K. Mansouri, “Design of an Adaptive e-Learning System based on Multi-Agent Approach and Reinforcement Learning,” *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, Feb. 2021, doi: 10.48084/etasr.3905.

[55] A. Fardinpour, M. M. Pedram, and M. Burkle, “Intelligent Learning Management Systems: Deinition, Features and Measurement of Intelligence,” *International Journal of Distance Education Technologies*, Dec. 2014.

[56] P. De Bra *et al.*, “GRAPPLE: Learning Management Systems Meet Adaptive Learning Environments,” in *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends*, A. Peña-Ayala, Ed. Berlin, Heidelberg: Springer, 2013, pp. 133–160. doi: 10.1007/978-3-642-30171-1_6.

[57] L. Oneto, F. Abel, E. Herder, and D. Smits, “Making today’s Learning Management Systems adaptive,” *Learning Management Systems meet Adaptive Learning Environments Workshop at European Conference on Technology Enhanced Learning (ECTEL*, p. 10, 2009.

[58] B. Khosravifar, R. Azevedo, R. Feyzi Behnagh, M. Taub, G. Biswas, and J. Kinnebrew, “Adaptive multi-agent architecture to track students’ self-regulated learning,” *CEUR Workshop Proceedings*, vol. 1009, pp. 49–52, Jan. 2013.

[59] T. Wilichowski and C. COBO, “Considering an adaptive learning system? A roadmap for policymakers,” *World Bank Blogs*, Jan. 06, 2021. https://blogs.worldbank.org/education/considering-adaptive-learning-system-roadmap-policymakers (accessed Apr. 03, 2022).

[60] H. Khosravi, “Development of Cost-Effective Adaptive Educational Systems via Crowdsourcing,” EDUCAUSE Review Dec. 02, 2019. https://er.educause.edu/blogs/2019/12/development-of-cost-effective-adaptive-educational-systems-via-crowdsourcing (accessed Mar. 06, 2022).

[61] K. Kuang, “Interactivity - An overview,” *ScienceDirect Topics*, 2020. https://www.sciencedirect.com/topics/psychology/interactivity (accessed Apr. 04, 2022).

[62] W. Lawless, R. Mittu, and D. Sofge, *Human-Machine Shared Contexts - 1st Edition*, ELSEVIER, Jun. 09, 2020. https://www.elsevier.com/books/human-machine-shared-contexts/lawless/978-0-12-820543-3 (accessed Jun. 05, 2022).

[63] P. Angelov, *Autonomous Learning Systems: From Data Streams to Knowledge in Real-time*. John Wiley & Sons, 2012.

[64] R. Singh, “Moving away from monoliths towards self-contained systems,” Softobiz Technologies India, Mar. 04, 2020. https://softobiz.com/moving-away-from-monoliths-towards-self-contained-systems/ (accessed Apr. 13, 2022).

[65] R. Khare, “Decentralized Software Architecture,” California Univ Irvine Institute For Software Research, Dec. 2002. Accessed: Apr. 15, 2022. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA441133

[66] M. Anderson, “Exploring Decentralization: Blockchain Technology and Complex Coordination,” *Journal of Design and Science,* Feb. 2019, Accessed: Apr. 16, 2022. [Online]. Available: https://jods.mitpress.mit.edu/pub/7vxemtm3/release/2

[67] A. Hayes, J. R. Brown, and S. Kvilhaug, “Blockchain Explained,” Investopedia, Mar. 05, 2022. https://www.investopedia.com/terms/b/blockchain.asp (accessed Apr. 16, 2022).

[68] J. Hu, P. Bhowmick, and A. Lanzon, “Distributed Adaptive Time-Varying Group Formation Tracking for Multiagent Systems With Multiple Leaders on Directed Graphs,” *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 140–150, Mar. 2020, doi: 10.1109/TCNS.2019.2913619.

[69] A. S. Palau, M. H. Dhada, and A. K. Parlikad, “Multi-agent system architectures for collaborative prognostics,” *J Intell Manuf*, vol. 30, no. 8, pp. 2999–3013, Dec. 2019, doi: 10.1007/s10845-019-01478-9.

[70] Q. Mahmoud, “Agents : Characteristics and Classification.” https://www.semanticscholar.org/paper/Agents-%3A-Characteristics-and-Classification-Mahmoud/ead40c8efb0299f37f881cf8d90839bd9545f443 (accessed Jun. 05, 2022).

[71] H. A. Abbas, S. I. Shaheen, and M. H. Amin, “Organization of Multi-Agent Systems: An Overview,” *International Journal of Intelligent Information Systems*, vol. 4, no. 3, Jun. 2015, doi: 10.11648/j.ijiis.20150403.11.

[72] W. Kim, J. Park, and Y. Sung, “Communication in Multi-Agent Reinforcement Learning: Intention Sharing,” in *the International Conference on Learning Representations*, Sep. 2020. Accessed: Apr. 28, 2022. [Online]. Available: https://openreview.net/forum?id=qpsl2dR9twy

[73] O. Shehory, “Architectural Properties of Multi-Agent Systems,” Feb. 1999.

[74] X. Wei, Q. Li, Y. Liu, H. Yu, T. Chen, and Q. Yang, “Multi-Agent Visualization for Explaining Federated Learning,” in *Proceedings of the 28th International Joint Conference on Artifical Intelligence*, pp. 6572–6574, 2019.

[75] Coss, “5 Characteristics to Build Testability in Software Engineering,” MuukTest, Sep. 30, 2021. https://muuktest.com/blog/building-testability-in-software-engineering/ (accessed May 11, 2022).

[76] P. Tarvainen, “Adaptability Evaluation at Software Architecture Level,” *The Open Software Engineering Journal*, vol. 2, no. 1, pp. 1–30, 2008, doi: 10.2174/1874107X00802010001.

[77] J. Wesson, A. Singh, and B. Tonder, “Can Adaptive Interfaces Improve the Usability of Mobile Applications?,” in *IFIP Advances in Information and Communication Technology*, vol. 332, 2010, pp. 187–198. doi: 10.1007/978-3-642-15231-3_19.

[78] F. Akkawi, M. Lee, A. Bader, and T. Elrad, “Adaptability vs. reusability of software systems,” in *Commercial Applications for High-Performance Computing*, Jul. 2001, vol. 4528, pp. 118–126. doi: 10.1117/12.434864.

[79] F. E. Sandnes, E. Eika, and F. O. Medola, “Improving the Usability of Interactive Systems by Incorporating Design Thinking into the Engineering Process: Raising Computer Science Students’ Awareness of Quality versus Quantity in Ideation,” in *2019 5th Experiment International Conference (exp.at’19)*, Jun. 2019, pp. 172–176. doi: 10.1109/EXPAT.2019.8876490.

[80] T. Singh and A. Kumar, “Survey on Characteristics of Autonomous System,” *International Journal of Computer Science and Information Technology*, vol. 8, pp. 121–128, Apr. 2016, doi: 10.5121/ijcsit.2016.8210.

[81] A. J. Dalpe, M.-W. L. Thein, and M. Renken, “PERFORM: A Metric for Evaluating Autonomous System Performance in Marine Testbed Environments Using Interval Type-2 Fuzzy Logic,” *Applied Science*, vol. 11, no. 24, p. 27, Dec. 2021.

[82] T. Chen, “Lifelong Dynamic Optimization for Self-Adaptive Systems: Fact or Fiction?,” *arXiv*, arXiv:2201.07096, Jan. 2022. doi: 10.48550/arXiv.2201.07096.

[83] S. M. Ågren *et al.*, “Architecture evaluation in continuous development,” *Journal of Systems and Software*, vol. 184, p. 111111, Feb. 2022, doi: 10.1016/j.jss.2021.111111.

[84] M. A. Chauhan and M. A. Babar, “Architecture Evaluation,” *ScienceDirect Topics*, 2017. https://www.sciencedirect.com/topics/computer-science/architecture-evaluation (accessed Jun. 18, 2022).

[85] K. K. Lau and S. D. Cola, *An Introduction to Component-Based Software Development*, World Scientific, vol. 3, 2017, pp. 15–19. doi: 10.1142/9789813221888_0002.

[86] Z. Svirca, “Everything you need to know about MVC architecture,” Medium, May 30, 2020. https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1 (accessed Jul. 16, 2022).

[87] V. Paget, “Features vs Functionality: How to accurately compare software systems.” Orah Blog, Jun. 18, 2017. https://www.orah.com/blog/features-vs-functionality (accessed Jul. 19, 2022).

[88] R. Sheldon, “Data storage management advantages and challenges explained,” TechTarget, May 12, 2021. https://www.techtarget.com/searchstorage/tip/Data-storage-management-advantages-and-challenges-explained (accessed Jul. 20, 2022).

[89] M. Baker, “Three components of content organization.,” Every Page is Page One, Aug. 11, 2014. https://everypageispageone.com/2014/08/11/three-components-of-content-organization/#:~:text=Content%20organization%20is%20a%20response,find%20it%20will%20be%20successful. (accessed Jul. 20, 2022).

[90] “Using Common Organizing Patterns.” https://saylordotorg.github.io/text_stand-up-speak-out-the-practice-and-ethics-of-public-speaking/s13-02-using-common-organizing-patter.html (accessed Jul. 20, 2022).

[91] C. A. Tomlinson, “Page 1: Defining Differentiated Instruction,” IRIS Center, Mar. 23, 2018. https://iris.peabody.vanderbilt.edu/module/di/cresource/q1/p01/ (accessed Nov. 29, 2022).

[92] O. Borchert, K. Lee, K. Sriram, D. Montgomery, P. Gleichmann, and M. Adalier, “BGP Secure Routing Extension (BGP-SRx): Reference Implementation and Test Tools for Emerging BGP Security Standards,” *National Institute of Standards and Technology*, Sep. 2021. doi: 10.6028/NIST.TN.2060.

[93] C. A. Tomlinson, “Page 1: Defining Differentiated Instruction,” IRIS Center, Mar. 23, 2018. https://iris.peabody.vanderbilt.edu/module/di/cresource/q1/p01/ (accessed Nov. 29, 2022).

[94] W. A. Paterson, “From 1871 to 2021: A Short History of Education in the United States,” SUNY Buffalo State College, Dec. 08, 2021. https://suny.buffalostate.edu/news/1871-2021-short-history-education-united-states (accessed Nov. 25, 2022).

[95] S. Mysari and V. Bejgam, “Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible,” in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Feb. 2020, pp. 1–4. doi: 10.1109/ic-ETITE47903.2020.239.

[96] “Why Use Media to Enhance Teaching and Learning,” Pedagogy in Action. https://serc.carleton.edu/sp/library/media/why.html (accessed Jun. 01, 2022).