Analysis on Intelligent Agent based Approach for Software Engineering

[1]Anandakumar Haldorai
[1]Department of Computer Science and Engineering, Sri Eshwar College of Engineering, Coimbatore, India.
[1]anandakumar.psgtech@gmail.com

Correspondence should be addressed to Anandakumar Haldorai: anandakumar.psgtech@gmail.com.

Article Info
Journal of Computing and Natural Science (http://anapub.co.ke/journals/jcns/jcns.html)
Doi: https://doi.org/10.53759/181X/JCNS202202020
Received 25 May 2022; Revised form 22 June 2022; Accepted 28 July 2022.
Available online 05 October 2022.

Abstract – A broad area of research, known as "Agent-Based Computing", focuses on developing "agent-based" intelligent software using agent-based techniques. However, there is a scarcity of research focusing on providing enough evidence of the superiority of agent-based techniques in creating complex software systems. This article has attempted to provide evidence for why agent-based techniques are superior to traditional methods for creating complex software systems, such as control systems. A case of a distinct agent-based control system (the power transportation management system used by Iber-26 drola) is used to illustrate these broader principles. This line of reasoning allows advocates of complex software engineering paradigms to accurately assert that their method can replicate the essential ideas behind agent-based computing. When broken down to their most basic components, agent-based systems are just computer programmes, and every programme has the same set of computable functionalities. The value of a paradigm lies in the way of thinking and the tools it teaches to programmers. As such, agent-based ideas and approaches are not just an extension of those now accessible within existing paradigms, but also well suited to the development of large, networked systems.

Keywords – Agents, Agent-Based Software Engineering, Agent-Based Systems, Agent-Based Computing, Agent-Based Approach.

I. INTRODUCTION

There is no agreed-upon definition of an "agent" since it is difficult to extract consistent and simple characteristics from the wide variety of applications and environments. The root of the agent-based approach, however, is that its fundamental unit, the "agent", may stand in for any kind of intelligent entity, functioning autonomously within an environment to achieve its agenda or objectives and actively engaging with other agents if required. According to Lopez-Merino and Rouchier [1], the following characteristics are ideal for a typical agent in an agent-based model: (i) autonomy, or the ability to function independently of outside control; (ii) sociability, or the ability to work with the environment and/or other agents to complete its task(s); (iii) reactivity/perceptibility, or the ability to detect and react to changes in its environment; (iv) proactivity, or the display of goal-directed behavior on its own initiative; and (v) adaptation/learning, or the capacity to gain knowledge through experience and adjust to the environment and the simulation's goals.

Note that the characteristics given above are neither complete nor exclusive; agent modeling is adaptable. It is possible for several kinds of agents to coexist in a simulation experiment, each with its own unique set of characteristics and their own unique set of perceived significance. Many kinds of agents may be used in a simulation of a demand-responsive transportation system. Clients may be seen as independent actors who perpetually seek transportation services. They will also be proactive and adaptable in design, learning to monitor and control itinerary iteration options and itineraries. In order to keep up with the ever-increasing demands placed on today's control systems, complex software is required. The need to adapt to a more unpredictable world, manage in more volatile settings, and provide more versatility drives these expectations. Because software always consists of numerous interconnected pieces, it is inherently difficult to understand and debug. Because of this intricacy, cutting-edge software engineering practices are required.

This article will argue that software engineers might benefit greatly by seeing such massive software applications as a set of interconnected, independently movable building blocks (i.e., as agents). To convince others that the agent-based software engineering technique works, it is essential to provide hard data showing how using it sped up the development of a variety of (control systems) projects. Although there are now multiple deployed applications, such information is currently unavailable (the same cannot be said for other modern software engineering methods such as design patterns, frameworks for building applications, or modules). Due to this, a quantitative explanation of the benefits of agent-based methods to developing complicated control systems is out of the question. These generalizations are supported by a case study assessing the performance of an agent-based strategy in the realm of industrial process control (with an emphasis on the administration of electrical transmission systems). Before making a more broad case analysis for the engineering of

agent-base software, however, we analyze the complex software systems' features (where control systems are considered an example). We next go into the methods developers have developed to tackle this complexity. Every new software development paradigm should aim to simplify the process of dealing with complexity by introducing new frameworks and tools. Designers may breathe a sigh of relief, however, since this complexity displays a number of crucial regularities.

First, a hierarchy is a common representation of complexity. The term "hierarchical system" refers to a system in which the lowest level of elemental sub-systems is comprised of even lower levels of sub-systems that are themselves hierarchical [2]. However, several common types of these organizational links may be identified (client-server, peer, etc.), despite the fact that their exact nature differs between sub-systems. As time goes on, these connections tend to shift and change. Second, certain parts of the system are considered "primitive" depends on the perspective of the observer and their specific goals. Finally, hierarchical systems mature more swiftly than their flat counterparts of the same size (i.e., In cases where stable intermediate types can be identified, complex systems will emerge from typical ones much more quickly than would be the case otherwise). Moreover, the interactions between and within sub-systems may be distinguished. The latter are not only more common, but also easier to anticipate, by at least an order of magnitude. It's because of this that experts believe complex systems are virtually decomposable, meaning their constituent parts may be analysed and manipulated as if they were completely autonomous entities—but not quite, because of their interdependencies. Even if many of these interactions may be anticipated during the design phase, some cannot.

According to Morales-Silva and Gao [3], these understandings allow for the definition of a canonical perspective of a complex system. The "related to" connections reveal the system's hierarchical structure, "frequent interaction" connections reveal the system's sub-systems, and "infrequent interaction" connections reveal the system's interactions amongst its components. In light of these facts, software engineers have developed many standard tools or method (e.g., those highlighted in **Table 1**) for their craft to aid in the management of this complexity.

Table 1. Tools/methods for the management of a complex system

Tools and Methods	Brief Description of the Tools and Method	Sources
Decomposition	The most fundamental method for solving complex issues is to break them down into smaller; more manageable pieces that may be tackled independently. Because it restricts the designer's horizons, decomposition is useful for tackling complexity.	Fox and Furmanski in [4]
Abstraction	The action of developing a system model that accentuates some aspects or attributes while minimizing others. This is effective once again because it confines the designer's attention to a manageable set of problems.	Cazzola, Ghosh, Al-Refai, and Maurina in [5]; Tsagkani and Tsalgatidou in [6]; and Khazaaleh, Samarasinghe, and Kulasiri in [7]
Organization	An activity involving the articulation and management of connections among diverse elements of problem-solving. Included in this category are phenomena like subroutines and inheritance (in object-oriented architectures) (in procedural languages). As a result of being able to specify and implement institutional relationships, designers are better able to deal with complexity because (i) they can group together a number of basic components and analyse them as a single, unified unit, and (ii) they have a language for describing the relationships between the different units in the system.	Rapuano and Valickas in [8] and Scaglione, Meyer Jr, and Mamédio in [9]

By describing complex software systems and identifying the core software engineering methodologies that help in managing this complexity, we have shown the relevance of agent-based software engineering in this work. Below is the outline for the rest of the paper: Section II presents a critical review of agent-based software engineering where (i) an overview of agents, (ii) software engineering credentials of agent-based approach, and (iii) adoption of agent-based techniques is provided. Section III uses a case-based application scenario to discuss the concept of agent-based software engineering. In the section, a case study of electricity transportation management is used; (i) defining the relevance of the application, (ii) defining the relevance of agent-based techniques for this application, (iii) specification of the agents, and (iv) cooperative diagnosis and restoration. Section IV presents concluding remarks regarding the research as well as future research directions.

II. AGENT-BASED SOFTWARE ENGINEERING

To argue for a more agent-based strategy to the engineering of software, it is essential to start by identifying the main ideas of agent-oriented computing, e.g., agents, software engineering accreditation, and the agent-based strategy [10]. The first concept is that of an agent, which is (i) a computer system, which exists in isolation, (ii) is aware of its surroundings, and (iii) may take whatever action necessary to accomplish its predetermined objectives within that environment.

Overview of Agents

There are many aspects of this concept that could be clarified. Agents are (i) discrete, self-contained entities that can solve problems; (ii) ingrained in surroundings over which they have some degree of control; they accept input related to the status of their surroundings via sensors and respond to the surroundings via signaling pathways; and (iii) designed to perform a specific task and have clear objectives. (iv) Self-reliant, in that they decide what to do and how to do it.

When using an agent-based viewpoint by Ransikarbum, Kim, Ha, Wysk, and Rothrock [11], it is important to have several agents so that the problem's decentralized character, different loci of control, diverse views, or opposing interests may be correctly portrayed. It will be necessary for the agents to coordinate their actions in order to accomplish their goals and/or deal with the dependencies that arise from sharing a physical environment. From straightforward semantic interaction (the exchange of data) to intricate social exchanges and even classical client-server exchanges, interactions can take many forms (capacity to work together, communicate, and reach an agreement on a plan of action). Agent interactions, unlike those in other software development paradigms, are distinguished by two primary properties that are independent of the underlying social process. Agents often use a high-level (declarative) agent communications language when starting a discussion (often using the principles of speech act theory). Thus, conversations take place at the level of information: (i) which aims, when, and by who ought to be pursued (Syntactically-only method or function invocation). (ii) agents are considered adaptive solvers of problems operational in a world with minimal control, therefore, interactions have to be malleable. To interact and react with what was not projected at the design time; agents need to be able to make context-dependent judgments regarding the type and extent of their interactions.

According to Zato, de Luis, De Paz, and López [12], Individuals and organizations often employ agents to carry out certain tasks on their behalf. As a result, the interactions of the agents are often grounded in an underlying organisational environment. The context establishes the rules that must be followed and the nature of the connection between the actors. The agents may be coworkers on the same team under the direction of a manager, or they may be independent contractors subject to the guidelines of a certain auction house while negotiating. Agent systems have built-in mechanisms for modelling institutional ties of this kind (e.g. auctioneer, team member, or manager). In various contexts, these connections are dynamic, constantly shifting as a result of social interaction, which causes both current connections to alter (for example, peers on a team could pick a leader) and new connections to be forged (e.g. groups of unconnected agents work together to provide a service that no one could provide alone). These connections may last for as little as a single transaction or as long as a lifetime. To manage this diversity and flux, agent researchers have created procedures for the formation and dissolution of organisational groupings, strategies to guarantee that these groups function in concert, and structures to define the macro behaviors of cooperatives.

According to Achter, Borit, Chattoe-Brown, and Siebers [13], software engineers using an agent-based methodology divide a complex issue into manageable portions, each of which may take its own initiative and collaborate with others to get the intended outcome. Secondly, an agent-based approach is characterized by a focus on agents, interactions, and organizations as major abstraction models, and thirdly, the resultant complex systems are often described and managed using explicit structures and processes. Comparing this viewpoint on software systems to research on heterochical systems in distributed control reveals a number of commonalities. By delegating authority to a number of autonomous nodes, the work of Sharma et al. in [14] mitigates some of the problems associated with traditional hierarchical management structures. Despite this, most research into heterarchical control has focused on the distributed processing aspect of these controllers and, to some extent, on the autonomy of the individual elements, rather than the malleable, high-level definition of the interplay and the explicit portrayal of the organisational context.

Basic Functionalities of Agents

According to Wang, Jing, and Goel [15], an agent may be thought of as a smart synthesis with distinct, individually-executed pieces. Effectors, sensors, communication, intentions, motivations, and cognitive processes are the six components essential to an agent's fundamental functions. The features may be classified into four categories: (a) Intention (b): the agent's short- and long-term goals; Motivation (c): putting those goals into action; Cognition (d): the agent's knowledge base, reasoning and deciding component, and protocol execution element. The evaluations of the existing state of affairs, as well as the selection, execution, and monitoring of the agent's concluding acts, are all handled by the final module. **Fig. 1** depicts agent functionality, including interactions between agents and with the environment.

Background and Concept of the Agent-Based Model

The concept of the agent-based model [16], which should not be confused with the more general activity-based model, is to evaluate the effects of autonomous agents by simulating their behaviors and interactions. Different from other approaches, it can study the emergent and collective consequences of complex systems made up of autonomous entities, making it one of a kind. As time has progressed, agent-based models have become more commonplace in transportation research. There are three main factors contributing to this: first, the transportation system is an extremely complex (and adaptable) system. Key features of driver behavior are too difficult to predict effectively, including multimodality, heterogeneity, complexity, and adaptability. According to Sulis and Taveter [17], agent-oriented modeling allows one to create a more accurate model by taking into account the diversity and interdependence of road users and the transportation network. Second, the introduction of novel technologies (such as autonomous vehicles (AV), vehicle-to-everything (V2X)

and vehicle-to-vehicle (V2V) communications) broadens the focus of study, increases the complexity of the transportation system, and poses new difficulties for the modeling process. The growth of technology also yields a plethora of specific data, which is crucial for adjusting agent-based models. Finally, as computing power has increased according to Moore's law, it has become possible to investigate transport system operations by dissecting their individual parts.

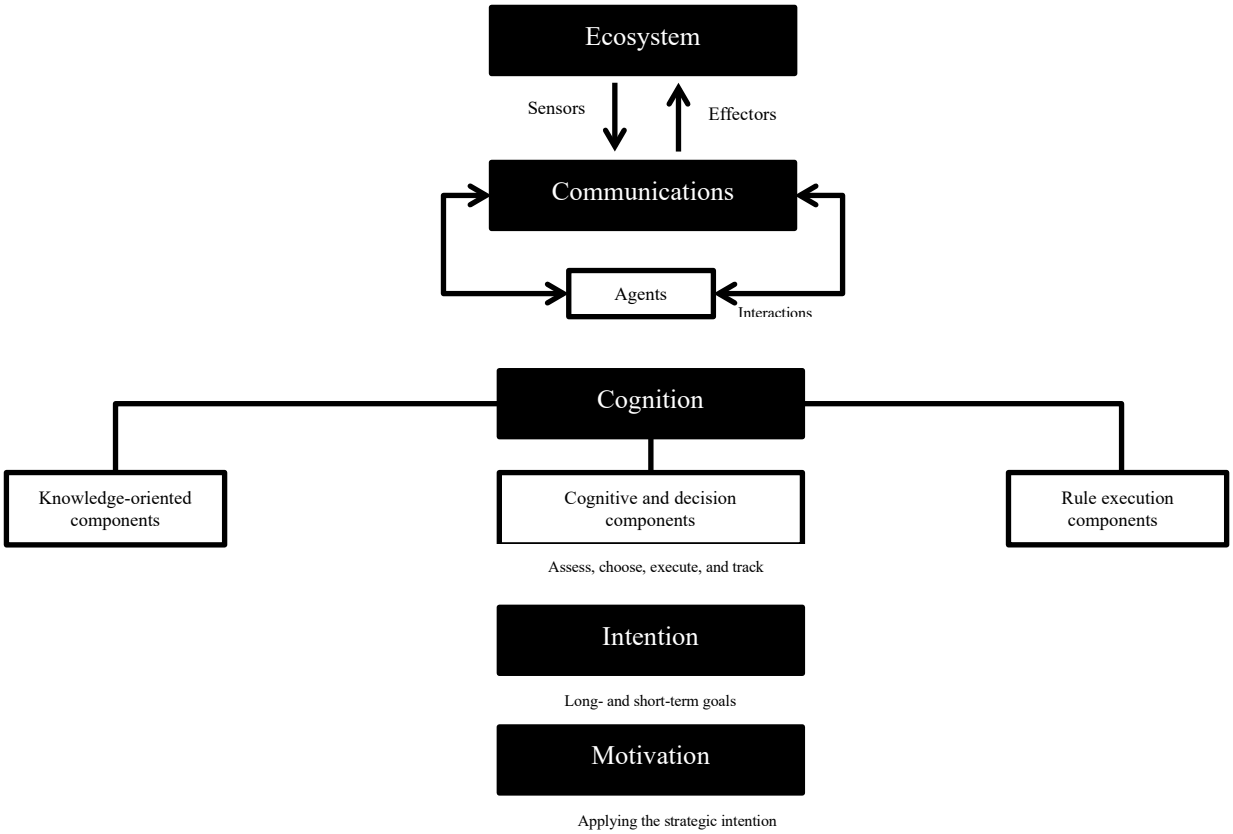

Fig 1. Agent functionality and interactions between agents and with the environment

An agent-based model (ABM) [18] is a model that mimics the behaviors and interconnections of intelligent entities as a means of investigating the mechanics of a network and the variables that impact its behavior (which might be groups or individuals). Game theory, evolutionary programming, multi-agent systems, computational sociology, emergence and complex systems are all included in this approach. The stochastic nature of these models is explored using Monte Carlo techniques. Individual-based models are another name for ABMs, especially in the field of ecology (IBMs). A survey of recent works in the fields of biology, ecology, and social science reveals the widespread use of agent-based models (ABMs). In contrast to multi-agent simulations of multi-agent frameworks, the concept of agent-based modeling (ABM) is not to create agents or to solve particular practical or technical challenges; rather, ABM seeks explanatory insights in a collective action of agents following basic rules, generally in natural systems.

According to Conrad, Köppl, and Djurdjevac [19], microscale models of the kind known as "agent-based models" seek to recreate and anticipate the occurrence of complex events by simulating the activities and interactions of several actors in real time. Some have described this as an emerging process, with the phrase "the whole is larger than the sum of its parts" capturing the idea. In other words, the attributes of a system at a higher level develop as a result of the interactions of subsystems at a low level. To rephrase, changes in the state at a macro level result from actions at the individual level. In other words, complex actions result from agents' basic behaviors (meaning the state tend to change at a whole systems' level). Typically, individuals are portrayed as bounded rational, with the assumption that they are following simple decision-making rules or heuristics in pursuit of their self-interests (such as economic gain, social status or reproduction). It is possible that ABM agents can "learn," changes, and even have offspring. In general, agent-based models have the following components: (1) a large number of agents described at different levels of detail (or "agent-granularity"); (2) a set of decision-making approaches; (3) a collection of learning protocols and adaptive strategies; (4) a collection of interactive topologies; and (5) the ecosystem. To investigate the effect that changing certain behaviors would have on the system's ultimate general behavior, researchers often employ computational models of ABMs, which may take the form of either custom software or ABM toolkits.

Theory and Framework

Most studies (e.g., [20] and [21]) that use computer modeling focus on equilibrium or transitions between equilibria for the systems being studied. However, agent-based modeling using even the most basic of principles may provide surprising complexity and novelty. Aspects of emergence and complexity are also crucial to agent-based models, along with the concept of agents as objects. Rule-based agents communicate with one another in real time in agent-based models. Their interplay may generate complexity on par with the actual world because of the systems they inhabit. Agents are often located in time and space and have neighbors that are either networks or lattices. Algorithms in computer programs save information on the agents' whereabouts and the ways in which they react to their surroundings. It's possible to attribute intelligence and motivation to the agents sometimes, but not always. Objects like trees in a forest might represent agents in environmental ABM (also called "individual-based systems" in ecology) that are not conscious but yet have "intent" in the sense that they're striving to improve their odds of obtaining resources (such as water). Inductive is the best term to explain how the models are created. The modeler assumes what they believe to be the most important facts about the scenario, and then observes new occurrences as a result of the agents' interactions with one another. When it happens, equilibrium is the consequence. Emergent patterns may occur at times. However, there are instances when it becomes an incomprehensible mash-up.

Traditional techniques of analysis may be supplemented by using ABM. ABM provide the option of establishing equilibria, while analytic tools help humans define them. Perhaps the most widely applicable feature of agent-based modeling is this creative input. Social segregation persists despite accommodating demographics, traffic congestion follow a power law dispersion, and warfare and crashes in stock markets follow similar distributions. All of these are examples of higher-order patterns that may be described by agent-based systems. Lever points, or periods in time at which actions have disproportionate effects, are one sort of route dependence that may be distinguished using agent-based models. Robustness, the ability of complex systems to endure and even thrive in the face of perturbations both internal and external, is often prioritized above stability in models. The agents' variety, interconnectedness, and interaction degree must all be taken into account in order to harness this complexity.

Recent efforts by AL-Qutami, Ibrahim, Ismail, and Ishak [22] to describe and simulate massive adaptive systems have highlighted the value of combining agent-based and complicated network-based models. Several interdisciplinary case studies are used to illustrate [23]'s four-tiered approach for modeling complex adaptive systems. Complex network modeling is a technique for creating models by making use of information about interactions between parts of a system. Agent-based modeling at the exploratory level is used to determine whether or not a study is even feasible. Examples of where this would be helpful include creating proof-of-concept models for use in grant applications without subjecting researchers to a steep learning curve. The use of templates and advanced network models form the basis of DREAM, which is a technique for developing descriptive accounts of agent-based models. The creation of DREAM models paves the way for inter-disciplinary model comparison. Agent-based modeling with formal verification and validation is through Virtual Overlay Multi-agent System (VOMAS). Besides using pre-made code snippets, a frequent method for describing agent-based models is to use textual methods like the Overview, Design Ideas, and Design Details (ODD) protocol. The role of the agents' micro and macro surroundings is increasingly being considered in agent-based simulation and modeling. A basic setting allows for simple agents, whereas a complicated setting provides a wide range of behaviors.

Multi-Scale Modelling

Agent-based modeling's strength lies in its capacity to moderate communication at different levels of analysis. When more information about an agent is required, researchers may include models explaining the necessary information. To better understand how emergent behaviors evolve in a community of agents, it is helpful to combine the agent-based paradigm with a spectrum model describing population dynamics. CD4+ T cells are an important part of the adaptive immune system, and in a recent study, Zimmermann et al. [24] modeled the interplay between intracellular, cellular, and systemic levels of biological events (signal transduction, metabolism, gene regulation, cytokine transport, and cellular behaviors). The resultant modular framework incorporates many distinct types of representations, including a logical model for signaling pathways and gene regulation, constraint-based frameworks for metabolism, an agent-based model for cell population demographics, and ordinary mathematical models for systemic cytokine levels. The agent-based model acts as the hub of this multi-scale framework, coordinating the many levels of data exchange between them.

The Agent-Based Software Engineering Credentials

In this article, the argument for an agent-based method in software development is deconstructed: First, by showing how useful agent-based decompositions are for breaking down the complex problem of a complex system, then by showing how useful agent-based key abstractions (i.e., suitability factor) are for modeling such systems, and finally by showing how useful the agent-based philosophy is for modeling and managing organizational relationships. This set of methods provides a mapping (cross product) between complicated system properties and the fundamental software engineering concepts for handling complexity in agent-based networks. Now, we'll address each individual stage.

Usefulness of Agent-Based Decompositions

To accomplish their goals or handle the dependencies that arise from sharing a space, agents will have to communicate with one another. The Adequacy and Establishment hypotheses suggest that agent-oriented approaches have the potential to vastly increase the capacity to develop, design and define massive, distributed models, and will definitely succeed as the mainstream software engineering model. These hypotheses argue that distributed systems' deficiencies (rigorously specified interactions between computational agents and inadequate approaches available for defining the system's intrinsic organizational structure) are overcome by the advantages of using such an approach. There are three primary ideas around which the various approaches to complexity are built: (i) Decomposition, which is vital for tracking larger issues. This can be sub-divided into manageable segments each of which could be handled by relative isolations. (ii) Abstraction, which is a process of considering simplified framework of systems, which focus on the properties and details which suppressing the others, and (iii) Organization, which acts on the identity and management of the relationships between different problem-solving elements.

Subsystems inside complex systems are often organised hierarchically. Subsystems at any level cooperate to perform the tasks of the higher level system. Also, the parts of a subsystem cooperate to provide the whole functionality. Thus, the system follows the same fundamental concept of cooperating components working together to accomplish certain goals. Consequently, it makes perfect sense to divide the components into modules based on the tasks they do. Function/action/process-based decompositions are easier and intuitive compared to object- or data-based decompositions, and this is a consensus held even by the object-oriented community. Each component may be conceived as serving a specific purpose, or perhaps many. Furthermore, it is important to understand that "actual systems have no top," which means that authority may be exerted at different points across a complex system. Each subcomponent in a decomposition that achieves its aims should localize and internalize its own control according to this notion. As a result, entities need to be able to make decisions and carry them out on their own (they need to be active) (i.e. they should be autonomous). In order to succeed, both the active and autonomous components must work together (recall complex model are decomposable). However, it is challenging to anticipate in advance all potential linkages between the various pieces of the system due to the system's inherent complexity. Interactions will take place at random, for unknown causes, and include a wide variety of entities. This is why it is pointless to attempt to foresee or analyse every possible scenario during the design phase. More realistically, components should be given the freedom to choose the kind and scope of their interactions on the fly during execution. In light of this, it follows that parts must be able to begin (and react to) interactions with relative ease.

The idea of postponing decisions about component interconnections until run time may be useful in two ways for the development of complex systems. The first major benefit is a drastic decrease in component coupling issues (by responding to them with pliability and declaration). Various components are tailor-made to handle unforeseen requirements and may make their own requests for help if they get stuck. When agents engage using a sophisticated language designed for their communications, the amount of coupling between them increases to that of true understanding. As a result, grammatical considerations are no longer a factor in the categories of mistakes brought on by unanticipated interactions. Second, classic objective-based decompositions no longer have to deal with the formidable challenge of maintaining control connections between software components. Continuously operating agents manage any necessary bottom-up coordination and synchronisation through mutual interactions among themselves.

Based on the above discussion, it is clear that a complex system is best modularized into a set of independent modules that may act and interact in a wide variety of ways to accomplish their goals. The agent-based method conforms most closely to this goal; hence it is the most appropriate.

Suitability Factor of Agent-Based Abstractions

The design process relies heavily on selecting appropriate models through which to examine the issue at hand. In most cases, there will be several qualified individuals to choose from. In software design, the most effective abstractions are those that narrow the semantic chasm between the problems intuitively conceived atomic units of analysis and the solution paradigm's prevailing heuristics and constructs. Characterizing the problem in complex systems involves looking at sub-system components, sub-systems, organisational relationships, and interactions.

Considering the above, (i) there is a natural mapping between agent structures and the divisions of a system. Many different parts play important roles in the whole, and these parts all have their own distinct behaviours and interactions. (ii) It has been argued above that subsystem components should be regarded as agents. (iii) Since "in a complex system at any given level of abstraction, we uncover meaningful groupings of objects that collaborate to accomplish some higher level vision," framing the interaction between the different sub-systems and their constituent pieces on the basis of high-level social engagements makes the most sense. This perspective is in perfect harmony with the agent-based approach, which treats interaction from the perspective of accumulated knowledge. When describing agent systems, phrases like "cooperating to accomplish shared goals," "coordinating their behaviours," and "negotiating to resolve disputes" are often used, and (iv) Interconnections between the parts of a complex system are dynamic webs. Furthermore, they demand that, when evaluated at a higher level of abstraction, collections of pieces be regarded as indivisible entities. Again, the agent-based view provides helpful abstractions. There is a plethora of structures at your disposal for graphically depicting interdepartmental ties. There are established interaction protocols for establishing new coalitions and dissolving existing

ones. Finally, we have the frameworks we need to model groups as a whole. The latter is especially relevant when it comes to representing sub-systems, which are essentially just a collection of parts doing their part to accomplish a larger whole.

The Urgency of Adaptive Management for Emerging Organizational Structures

Unlike traditional models, agent systems see organizational structures and connections as independent entities (in the programming language aspect) [25]. Agent-based systems also include the computational methods for dynamically establishing, sustaining, and disbanding organizations. Because of their superior capacity for representation, agent systems can profit from two features of complex systems. First, one's definition of a "primitive component" may shift depending on the context. Subsystems as a whole can be thought of as singletons, while groups of agents can be seen as more basic building blocks, and so on down the hierarchy until the system collapses. Second, these architectures provide the steady transitional forms necessary for the rapid development of complex systems. Having access to them allows for the development of individual agents or organisational structures in isolation, before being progressively integrated into the system. This, in turn, guarantees steady development of features.

The Philosophy of Agent-Based Techniques

Whether or whether agent-based approaches become the dominant software engineering paradigm depends on two factors: (i) how much of a departure agents represent from conventional software engineering ideas, and (ii) how easily existing software could be incorporate with an agent.. The following two paragraphs will address each of those points individually. By tracking the development of different types of code, we can see a few recurring tendencies.

First, there has been a consistent move away from languages with fundamental abstractions based on the fundamental hardware and toward those with abstractions rooted in the problem domain. In this context, agent-based ideas may provide the most intuitive means of defining various forms of issues. The world is full of objects that undergo operations, and it is also full of dynamic, resolute agents, which engage in complex interactions to accomplish their goals. This is the starting point for many object-oriented analyses: "we consider the world as a collection of autonomous agents who interact to fulfill some higher level purpose". Second, there is a trend toward greater localization and encapsulation among the programming models' basic building blocks. In keeping with this pattern, agents encapsulate action selection and localize purpose within individual agents. Third, more and more options for encouraging reuse are becoming available. As a result, this is a pinnacle moment for the agent perspective as well. Agents allow for the re-use of entire sub-systems as well as flexible interactions, rather than just individual components (component-ware and design patterns) or rigidly pre-ordained interactions (application models). The former describes the process of reusing agent ideas and deployments in many contexts. For instance, think about the group of agent architectures that are predicated on the agent's knowledge, desires, and actions. Such architectures have been deployed in numerous sectors, including aviation, manufacturing, transportation, and fault diagnosis. When it comes to the latter, adaptable interaction patterns e.g., the Con-contract Net Protocol (whereby a job-possessing agency advertises its availability to other agents it deems competent of completing the task, those interested in doing so make bids, and the job is awarded to agents, which submitted higher bids), and other types of resource-assignment auctions (such as the First- and Second-, English-, Dutch- Price Sealed Bid). To sum up, agent-based techniques are the logical next step from where software engineers are at conceptually, so its central ideas and tenets should be well received by the software engineering community.

Second, agent use does not necessitate a complete integration of an organization's current software system. Since older (non-agent) software may be easily integrated into agent-based systems, these architectures are evolutionary and gradual (see the case study on Administration of Electrical Transmission Systems). Wrapping software is employed to protect the old programmes from being modified. The software components are exposed via an agent interface provided by the wrapper. So, it blends in with the scenery as any other agent would. The wrapper's inner workings are a translations function that works in both directions, translating queries from external agents into operations in the legacy systems and calls from the legacy systems into the correct set of asynchronous communication commands. The fact that agents can be used to "wrap" preexisting systems suggests they might serve initially as an integration technique. However, custom agents may be built and added to the system when new needs are imposed upon it. This allows a complicated system to evolve over time (through stable intermediate metrics), based on the critical concepts, which ought to be functional applications of the system.

III. CASED-BASED APPLICATION SCENARIO

Following a discussion of the usefulness, suitability factor and philosophy of agent-based systems for complex systems in general, we turn our attention to a control system application scenario, namely the Administration of Electrical Transmission Systems. The purpose of providing this case study is twofold: (i) to provide concrete examples of how agent-based computing might be used; and (ii) to emphasise the real-world benefits that can result from adopting this approach. Additionally, a control system is discussed to emphasise the breadth and usefulness of agent-based solutions: the electricity transport management case study at the level of managing an entire network.

Administration of Electrical Transmission Systems

Spanish Iberdrola, a renewable energy company, created and released this application (the enabling agent technology is described in further depth in [14], and further information about its use in this area is available in [7]). Energy management, in its broadest sense, is keeping tabs on and managing the process through which electrical energy is created, transmitted, and distributed to commercial and residential consumers. Hydraulic, thermal, nuclear, and solar power is all examples of raw energy that must be converted into a more usable form before being transmitted from the point of production to the point of consumption. Transmission losses may be minimized by increasing the voltage of electricity transmissions to 132 kilovolts (kV) or higher before putting them on a transportation system and extending their length by many kilometers.

Lastly, the voltages are reduced, and power is sent to customers through a vast, dispersed distribution system consisting of several kilometers of wire (all at voltages lower than 132 kilovolts). The transportation network is equipped with a state-of-the-art data collection system (SCADA) and different traditional application programs that assist an operator (control engineer) in analyzing it to guarantee it stays within the required safety and economic limitations (these systems are meant to run under typical settings).

According to Wang et al. [26], the SCADA system sends numerous warnings to the Dispatching Control Room (DCR), where the network is maintained, whenever an anomaly is identified. Because of this, the operator must depend on prior experience to interpret the data, make a correct diagnosis, and implement effective corrective measures to restore the network's security. In order to assist operators make better judgments in such situations and to do so more quickly, Iberdrola initially built a variety of (detached) decision-support frameworks (e.g., database that records network data in real-time and intelligent systems for analyzing warnings to determine the cause of problems in the network). For this reason, it was decided that additional capabilities should be implemented and the systems involved should be made to interoperate with one another in order to provide a unified viewpoint (in order to take use of the additional sources of data, such as chronological data and quicker rate snapshots, made accessible as a result of enhancements to the SCADA network and to allow the control engineer to actively conduct and dynamically analyze the service recovery process). The following section elaborates on the role that agent technology played in facilitating this outcome.

Relevance of this Application

There were many advantages to using this application. First, the agent system outperforms stand-alone alternatives due to its ability to incorporate and make sense of a wide variety of sources of information and knowledge. Because of the overlapping functionalities present in the agent system, it is more resilient to the failure of individual components (agents). Thirdly, cooperation allows for a quicker delivery of some results. Moreover, the various domain systems can have their capabilities expanded separately, which reduce the burden of upkeep. Fifthly, an aggregated view of relevant outcomes is made available to the control engineer. Finally, the system was built to be extensible, allowing for the addition of new agents over time. Fault diagnosis in this multi-agent system is handled by incorporating data from secondary sources (the non-chronological alarm systems utilized by the Alarms Analysis Agent (AAA) and the chronological alarm systems utilized by the Breakers and Relays Supervisor (BRS)) and two perspectives (the normal diagnosis methodology of theory verification and generation employed by BRS and AAA, as well as the Black-out Area Identifier (BAI) monitoring strategy, which offers a high level of fault isolation).

When implemented in this way, the optimal solution approach can be chosen on-the-fly. If both the BRS and AAA are functioning, the control engineer will get the consensus solution; otherwise, the control experts will obtain a workable solution established by BRS. Also, if it's desired to avoid duplicating efforts, you may take use of the fact that numerous agents are working toward the same goals. If, for instance, the BRS supplies initial hypotheses and makes them accessible to the AAA before the AAA begins its generation work, then the AAA may forego generating its own hypotheses and instead utilize the ones supplied by the BRS. The overall system's robustness is greatly increased by the agents' ability to dynamically manage multiple data sources and problem-solving perspectives during execution. This is so even if one agent fails to provide a solution, the others can.

Relevance of Agent Techniques for this Application

For this purpose, established decision-support technologies needed to be combined with brand-new features [27]. Either the present systems may be extended to include the new capabilities, or a distributed strategy might be used, enabling the additional functionality to be described as independent computational units that can communicate with the current systems through a shared distribution platform. In this case, we opted for the second choice because we believe it offers the most potential for success in terms of:

Permitting Reasoning with Intermittently Granular Data

Both the time-based and asynchronous warnings must be dealt with. In the first situation, the time stamp corresponds with the time of reception by the controller (thus, it is controlled by the polling operation of the system), but in the latter case, it corresponds with the moment when the event really occurred. Despite its benefits—a more complete picture of network activities, and hence more rapid diagnosis—chronological data has a low rating in Iberdrola's channels of communication. Therefore, their arrival time is unpredicted when the channels are overloaded, as is commonly the case during a

disturbance. It was chosen to construct a new alert assessment intelligent system that utilized chronological data and might blend its results with those extracted from the previous system, instead of designing a single process, which permitted in both types of data and had to integrate both types of diagnostic knowledge.

When thinking about restoring service, a similar dilemma arises. The two sorts of data that are useful for this task are snapshots (which represent the overall status of the network at the moment) and alert messages (that indicates how the components' state has transformed over time). The former may be created quickly and provide a full image of the system's state, whereas the latter could take some minutes in the event of a major interruption but are important to determine the nature of the issue from which the systems should be recovered. Instead of trying to combine the two kinds of information and reasoning in a single system, it was practice to have a subsystem for service restoration, which majorly handled images and gathered the require high-level knowledge on defective equipment from diagnostics subsystem (instead of trying to process the unprocessed alarm messages).

Accepting Network Frameworks in a Similar Framework

In order to tackle certain problems, analysts will need to examine the network's SCADA model, while analysts working on other problems will need the applications system framework (model that may be used to solve differential equations and which also takes into consideration the physical properties of all of its constituent parts). At design phase, it was agreed that rather than attempting to merge and harmonise these complicated and different models, each component should function on whatever model was most suited for its mission. Then, during runtime, the different parts may collaborate to fix any discrepancies caused by their usage of incompatible network models.

Accessing Different Problem-Solving Methods

Due to the wide variety of tasks that must be completed, there is no single, optimal approach to problem-solving. For diagnosis, symbolic thinking based on heuristic search excels, while procedural techniques are needed for algorithmic computations such connectivity (to identify the essential components that are related to specific components) and the load-flow assessment (solution will differential computations). Due to the dispersed nature of this strategy, we were able to encode each individual part using the most effective technique.

Acceptable Performance Based on the Application

Time is of the essence in transportation management, and because many different kinds of data may be handled in parallel with just a minor synchronization cost, using several linked devices helps enhance the reaction time of the total system. After settling on a decentralized method, it was necessary to select whether to use traditional distributed processing methods or agent-based approaches. The following factors in **Table 2** led to the selection of the latter in this case.

Table 2. Factors that influenced the selection of agent-based approaches

Factors	Description of the selected factors	Sources
Robustness	Since the subsystems' areas of competence overlap, the absence of an answer from any one of them does not rule out the possibility of finding one (because another system has the potential to provide some kind of answer). This redundant feature, nonetheless, is beyond the capabilities of current-generation distributed processing systems since it requires the intelligent coordination of several problem-solving components in a context-sensitive approach (for more information, see the cooperative scenario).	Liu, Polukarov, Ventre, Li, and Kanthan in [28]
Reliability	The operator may be provided with more trustworthy data by cross-referencing the solutions of the overlapping frameworks. Again, this cross-referencing capability must be managed appropriately in light of the current conditions, and this in turn necessitates the use of flexible and dynamic reasoning.	Guo et al. in [29]
Natural domain representation	When a major disruption occurs, control engineers use a strategy that is faithfully represented by an agent-based method. They each take on a specific role in the process of restoring service, with one person overseeing the restoration while another attempts to analyse the issue using many data points and arriving at a conclusion as fast as possible.	Jorquera, Georgé, Gleizes, and Régis in [30]

Specification of the Agents

The operator in the DCR is tasked with mostly mundane and easy tasks when managing the network under normal conditions. The multiple restrictions that must be considered and the low quality of the data provided make management more difficult in emergency circumstances. A line, bus bar, or transformer short is usually the cause of an emergency. Equipment failure (like a tripped breaker) or subsequent overloads can amplify the problem (If one line fails due to overload, it might have a domino effect on the surrounding lines, causing them to overheat and fail). To prevent a relatively minor problem from becoming a major disaster, swift and precise action to restore service is required. Loss of electricity from disconnected power plants may exacerbate the problem by throwing the electrical grid into disarray. The

operator's options in these situations are limited to operating breakers, altering the network topology, and activating and deactivating automations and protective relays.

Larger disruptions, however, could also need action on power plants. Based on this work description, a top-down analysis determined the following responsibilities should be included in any good decision support system for a control engineer: (i) find the source of the disturbances; it is important to distinguish between normal maintenance and true disturbances, since both might trigger the functioning of protective relays and breakers. (ii) Identify the origin, nature, and extent of the disturbance, as well as the extent of any equipment failure. (iii) Evaluate the network's status after it reaches a stable state; (iv) create a restoration strategy to bring the network back to its operating baseline.

It was determined to include the alarms assessment intelligent framework and the interface to the management platform as agents after bridging this top-down evaluation with the bottom-up viewpoint of studying the existing systems. Finally, it has long been known that knowing the initial region that is out of service (the "blackout region") can help limit the search for the faulty equipment, but a dedicated stand-alone structure for this purpose has never been considered financially viable due to the poor performance of the original alarm evaluation intelligent system (if somewhat slow). However, a lot of the essential infrastructure required to provide this functionality was suddenly made available from other agents, owing to agent technology, enabling the creation of a system which might produce this information cheaply conceivable. The operating system is made up of agents (highlighted in **Table 3**) that are running on five separate computers.

The Iberdrola network, which consists of four substations, is shown in this diagram as a tiny section (Sestao, Erandio Achuri and Sodupe). A remote transmission unit (RTU) [31] is connected to each of these substations and sends information to the DCR in Bilbao on the condition of the breakers, substation's bus bars, and other electrical components. The front end computer within the DCR gathers this data, and the cooperating agents may access it using management system's interface features.

Table 3. Description of agents running on separate computers

Agents	Description of the agents
Black-out Area Identifier (BAI)	The goal of the Black-out Area Identifier (BAI) is to determine which network components are first down since this is where the problematic component must be. It uses non-chronological alert messages as its source of information and works in tandem with the BRS as well as the AAA to enhance the general effectiveness of the analysis and the process of diagnosis. When a malfunction occurs in the system, the relays and breakers strive to isolate just the faulty component, or as few pieces of equipment as feasible.
Control System Interface (CSI)	The Computer System Interface (CSI) is the point of contact between the computer-controlled system and the rest of the world. Its purposes are to gather and disseminate network information to other agents, to communicate with the application software of conventional management systems, and to keep a watch on the maintenance procedure to detect any anomalies. The CSI-D component tracks disruptions and essential pre-processes the asynchronous and chronological alerts utilizes for BAI, AAA and BRS; CSI-R component tracks for and rectifies any changes within the image data files within the network, provides estimations of power flow in it, and ensures the accessibility of data to the User Interface Agent (UIA) and the Service Restoration Agent (SRA).
Breakers and Relays Supervisor (BRS)	The new alarms analysis expert model can now identify disturbances, ascertain fault types and severity, develop a prioritized list of problem hypotheses, test those hypotheses, and finally pinpoint the faulty piece of machinery. Time-stamped alarm messages and network snapshots detailing the operation of each circuit breaker and switch are required for the analysis to proceed.
Alarms Analysis Agent (AAA)	This agent, like the BRS, is motivated by a need for knowledge, but it gets lower-quality data. The alarm signals received by the AAA and the BRS both pertain to the identical physical actions, however the BRS's alarm messages are accurate to within a few microseconds. This suggests that the BRS provides a more accurate diagnosis than the AAA does when the data is devoid of errors. However, the BRS may underperform the AAA if part of the temporal data is lost (risky when the SCADA network is overloaded). Thus, collaboration between the two systems is required to make the whole system more resilient and trustworthy where partial or incorrect information—which is the case in most interesting cases—exists.
Service Restoration Agent (SRA)	After a network outage, this agent will create a strategy for restoring service and getting things back to normal. The diagnostic agents identify the limitations imposed by the damaged machinery, which are then used to guide the repair process.
User Interface Agent (UIA)	To connect the user community to the agent ecosystem, this agent is responsible for implementing the interface. The user is able to observe the findings of the diagnosing agents, listen to the notifications, and peruse the log of problems that were examined. From a restoration standpoint, the user may go through the created plan, make adjustments, put it through its paces in a simulated setting, and request a new plan be prepared that includes the tasks deemed important. The correct information is displayed on the screens of all the control engineers working on the UIA's operation at any given time thanks to a distributed windowing system.

With this system architecture, every duty uncovered by the top-down assessment will be handled by a least a single agent. Having a number of agents that all produce similar outcomes (or at least some overlap) is how you get robustness. Parallel task activation is the key to efficiency. The system's reliability is enhanced by the fact that in the event of a failure of a single component, the remaining components may frequently provide a result that is still of some value to the operator.

Cooperative Restoration and Diagnosis

The agents in the AAA, BAI, and BRS all work together by exchanging information with one another. All three systems should be able to make sense of the same data, but the BAI adds fresh information to the mix to arrive at a result that is consistent with the conclusions reached by the AAA and BRS. Consider that the SCADA system has sent over a batch of alarm signals that are not in any particular order, and that the CSI has determined that these alerts are all connected to the same disruption. Inferring this information from its framework of the other agents, CSI will send the alerts out as unsolicited data to the AAA and the BAI. At a later point in time, the BRS will again receive the same sequence of alarms based on time. Currently, the AAA, BRS and BAI are all running in tandem. With the warning signals in hand, the AAA may begin its diagnosing process and generate first hypotheses. This is also when the BAI would have gotten the alert signals and begun attempting to pinpoint the source of the blackout. The BAI provides the AAA with this data once again based on its simulations of the other agents. At the same time, albeit with a little lag, the BRS agent begins processing the chronological alarm messages. This will also lead to the development of some preliminary hypotheses. Once again, the BRS notates the AAA and sends the hypotheses to it to see if any agents are interested in the data. The BRS then moves on with its diagnostic in an effort to confirm the root of the problem.

IV. CONCLUSION AND FUTURE RESEARCH

This article has set out to defend the relevance of agent-based techniques for creating large software systems, with a focus on control systems. By approaching such complex software structures (i.e., as agents) as a set of interactive, autonomous, flexible components, analysts, developers, and implementers may reap several advantages. After introducing these broad concepts, this article illustrates their use in the context of a concrete agent-based control system by looking at the transportation administration system used by Iber-power drola. Advocates of complex approaches to software engineering might use these kinds of justifications to assert that their method can replicate the essential ideas behind agent-based computing. This research comes to the conclusion that agent-based systems provide a number of advantages for the next generation of controllers. These systems offer a locally determined, decentralized substitute that increases the system's capacity for adaptation and resilience. However, it becomes progressively more difficult to forecast the overall behavior of the system when decision-making is delegated to autonomous components. In order to achieve this goal, research on agent-based techniques tailored for control applications is required. A control system manages the agents in the systems detailed in this work, making it easier to steer their behaviors in a manner that results in the development of desirable system attributes. However, producing predictable system-wide behavior in the more general scenario of several organizations remains a topic for future research.

Data Availability

No data were used to support this study.

Conflicts of Interest

The author(s) declare(s) that they have no conflicts of interest.

References

[1]. P. Lopez-Merino and J. Rouchier, “The diffusion of goods with multiple characteristics and price premiums: an agent-based model,” Appl. Netw. Sci., vol. 7, no. 1, 2022.

[2]. Y.-J. Kang, Y. Noh, M.-S. Jang, S. Park, and J.-T. Kim, “Hierarchical level fault detection and diagnosis of ship engine systems,” Expert Syst. Appl., vol. 213, no. 118814, p. 118814, 2023.

[3]. D. Morales-Silva and D. Gao, “Canonical duality theory and triality for solving general global optimization problems in complex systems,” Math. Mech. Complex Syst., vol. 3, no. 2, pp. 139–161, 2015.

[4]. G. C. Fox and W. Furmanski, “A string theory for time dependent complex systems and its application to automatic decomposition,” in Proceedings of the third conference on Hypercube concurrent computers and applications Architecture, software, computer systems, and general issues -, 1988.

[5]. W. Cazzola, S. Ghosh, M. Al-Refai, and G. Maurina, “Bridging the model-to-code abstraction gap with fuzzy logic in model-based regression test selection,” Softw. Syst. Model., vol. 21, no. 1, pp. 207–224, 2022.

[6]. C. Tsagkani and A. Tsalgatidou, “Process model abstraction for rapid comprehension of complex business processes,” Inf. Syst., vol. 103, no. 101818, p. 101818, 2022.

[7]. M. Khazaaleh, S. Samarasinghe, and D. Kulasiri, “A new hierarchical approach to multi-level model abstraction for simplifying ODE models of biological networks and a case study: The G1/S Checkpoint/DNA damage signalling pathways of mammalian cell cycle,” Biosystems., vol. 203, no. 104374, p. 104374, 2021.

[8]. V. Rapuano and A. Valickas, “Application of complexity theory to organizational career management system’s development,” Manag. Organ. Syst. Res., vol. 85, no. 1, pp. 47–64, 2021.

[9]. V. L. T. Scaglione, V. Meyer Jr, and D. F. Mamédio, "Improvisation in higher education management: Coping with complexity and organizational dynamics," Glob. J. Flex. Syst. Manag., vol. 20, no. 4, pp. 291–302, 2019.
[10]. G.-F. Deng and W.-T. Lin, "Notice of Retraction: Agent-based modeling of supply chain network for adaptive pricing strategy," in 2011 IEEE 2nd International Conference on Software Engineering and Service Science, 2011.
[11]. K. Ransikarbum, N. Kim, S. Ha, R. A. Wysk, and L. Rothrock, "A highway-driving system design viewpoint using an agent-based modeling of an affordance-based finite state automata," IEEE Access, vol. 6, pp. 2193–2205, 2018.
[12]. C. Zato, A. de Luis, J. F. De Paz, and V. F. López, "Dynamic assignation of roles and tasks in virtual organizations of agents," in Advances in Intelligent and Soft Computing, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 59–68.
[13]. S. Achter, M. Borit, E. Chattoe-Brown, and P.-O. Siebers, "RAT-RS: a reporting standard for improving the documentation of data use in agent-based modelling," Int. J. Soc. Res. Methodol., vol. 25, no. 4, pp. 517–540, 2022.
[14]. V. Sharma et al., "OGAS: Omni-directional Glider Assisted Scheme for autonomous deployment of sensor nodes in open area wireless sensor network," ISA Trans., 2022.
[15]. Q. Wang, S. Jing, and A. K. Goel, "Co-designing AI agents to support social connectedness among online learners: Functionalities, social characteristics, and ethical challenges," in Designing Interactive Systems Conference, 2022.
[16]. J. Watts, R. E. Morss, C. M. Barton, and J. L. Demuth, "Conceptualizing and implementing an agent-based model of information flow and decision making during hurricane threats," Environ. Model. Softw., vol. 122, no. 104524, p. 104524, 2019.
[17]. E. Sulis and K. Taveter, "Agent-Oriented Modeling," in Agent-Based Business Process Simulation, Cham: Springer International Publishing, 2022, pp. 77–104.
[18]. C.-Y. Lin, Y. C. E. Yang, K. Malek, and J. C. Adam, "An investigation of coupled natural human systems using a two-way coupled agent-based modeling framework," Environ. Model. Softw., vol. 155, no. 105451, p. 105451, 2022.
[19]. N. D. Conrad, J. Köppl, and A. Djurdjevac, "Feedback loops in opinion dynamics of agent-based models with multiplicative noise," arXiv [math.DS], 2022.
[20]. W. Zhao, Y. Li, B. Xu, and H. Yang, "Experimental investigation and modeling of vapor-liquid equilibria for Bi–Zn and Bi–Sn–Zn systems at 10 Pa," CALPHAD, vol. 74, no. 102287, p. 102287, 2021.
[21]. I. A. Bashkirtseva, "Analysis of stochastically forced equilibria and noise-induced transitions in nonlinear discrete systems," Comput. Res. Model., vol. 5, no. 4, pp. 559–571, 2013.
[22]. T. A. AL-Qutami, R. Ibrahim, I. Ismail, and M. A. Ishak, "Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing," Expert Syst. Appl., vol. 93, pp. 72–85, 2018.
[23]. "Call for papers: Special issue on intelligent optimization, modeling, and simulation with knowledge for complex systems," Complex Syst. Model. Simul., vol. 1, no. 3, pp. 253–254, 2021.
[24]. P. Zimmermann et al., "The interplay of familial depression liability and adverse events in predicting the first onset of depression during a 10-year follow-up," Biol. Psychiatry, vol. 63, no. 4, pp. 406–414, 2008.
[25]. D. Isern, D. Sánchez, and A. Moreno, "Organizational structures supported by agent-oriented methodologies," J. Syst. Softw., vol. 84, no. 2, pp. 169–184, 2011.
[26]. Z. Wang, Y. Tang, X. Chen, X. Men, J. Cao, and H. Wang, "Optimized daily dispatching strategy of building- integrated energy systems considering vehicle to grid technology and room temperature control," Energies, vol. 11, no. 5, p. 1287, 2018.
[27]. G. Beltrao, I. Paramonova, and S. Sousa, "User interface design for AI-based clinical decision-support system : Preliminary study," in 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), 2022.
[28]. B. Liu, M. Polukarov, C. Ventre, L. Li, and L. Kanthan, "Agent-based markets: Equilibrium strategies and robustness," in Proceedings of the Second ACM International Conference on AI in Finance, 2021.
[29]. X. Guo et al., "An agent-based dynamic reliability modeling method for multistate systems considering fault propagation: A case study on subsea Christmas trees," Process Saf. Environ. Prot., vol. 158, pp. 20–33, 2022.
[30]. T. Jorquera, J.-P. Georgé, M.-P. Gleizes, and C. Régis, "Agent-based natural domain modeling for cooperative continuous optimization," in Computational Collective Intelligence. Technologies and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 437–446.
[31]. S. Pacheco-Gutierrez, UK Atomic Energy Authority, Remote Applications in Challenging Environments, Culham Science Centre, Abingdon, Oxfordshire OX14 3DB, United Kingdom., I. Caliskanelli, and R. Skilton, "Point cloud compression and transmission for remote handling applications," J. Softw., pp. 14–23, 2021.