African Journal of Advanced Pure and Applied Sciences (AJAPAS)
Online ISSN: 2957-644X
Volume 4, Issue 3, 2025
Page No: 71-78
Website: https://aaasjournals.com/index.php/ajapas/index
ISI 2025: 1.126 SJIFactor 2024: 6.752 معامل التأثير العربي: 1.62

Multi-Agent system Adaptation for Feature Driven Development Methodology

Khamisa A. Yousef [1*], Huweida A. Darbi [2*]
[1] Computer Department, Faculty of Science, University of Tobruk, Tobruk, Libya
[2] Computer Department, Faculty of Science, University of Derna, Derna, Libya

تكيف نظام متعدد الوكلاء لمنهجية التطوير المعتمدة على الميزات

خميسه عوض يوسف [1*]، هويدة عبد العزيز دربي [2]
[1] قسم الحاسوب، كلية العلوم، جامعة طبرق، طبرق، ليبيا
[2] قسم الحاسوب، كلية العلوم، جامعة درنة، درنة، ليبيا

[*]*Corresponding author: khamisa.yousef@tu.edu.ly*

Received: May 21, 2025	Accepted: July 07, 2025	Published: July 19, 2025

Abstract:
This paper presents the integration of Multi-Agent Systems (MAS) with Feature-Driven Development (FDD) methodology, which is an agile methodology that emphasizes client-valued features through well-defined stages. Despite its modularity and scalability, traditional FDD implementations often come down to flexibility and autonomous coordination in complex environments. The paper offers a conceptual model that illustrates how agents interact and align with FDD processes, presenting a new approach to improving flexibility and productivity in software development. The synergy between MAS and FDD can improve prototyping, automate jobs, and expedite requirement collection, leading to higher-quality software and better stakeholder alignment, and ultimately, more successful software projects.

Keywords: Multi-agents system (MAS), Feature Driven Development (FDD), Agile methodology, Adaptive Software Development, Software Process Improvement.

الملخص
تُقدم هذه الورقة تصورًا لدمج أنظمة الوكلاء المتعددين مع منهجية التطوير المعتمد على الميزات، وهي منهجية رشيقة تركز على الميزات ذات القيمة بالنسبة للعميل من خلال مراحل محددة وواضحة. وعلى الرغم من أن منهجية التطوير المعتمد على الميزات تتميز بطبيعتها النمطية وقابليتها للتوسع، إلا أن تطبيقاتها التقليدية غالبًا ما تواجه تحديات تتعلق بالمرونة والتنسيق الذاتي ضمن البيئات المعقدة. تقترح هذه الدراسة نموذجًا مفاهيميًا يوضح كيفية تفاعل الوكلاء وتوافقهم مع عمليات التطوير المعتمد على الميزات، مما يفتح آفاقًا جديدة لتعزيز المرونة والإنتاجية في تطوير البرمجيات. كما أن التآزر بين أنظمة الوكلاء المتعددين ومنهجية التطوير المعتمد على الميزات من شأنه تحسين عمليات النمذجة الأولية، وأتمتة المهام، وتسريع جمع المتطلبات، مما يسهم في إنتاج برمجيات عالية الجودة وتحقيق توافق أكبر مع أصحاب المصلحة، وبالتالي إنجاح مشاريع البرمجيات بشكل أفضل.

الكلمات المفتاحية: أنظمة الوكلاء المتعددين، التطوير المعتمد على الميزات، المنهجيات الرشيقة، تطوير البرمجيات التكيفية، تحسين عمليات تطوير البرمجيات.

Introduction

Software development methodologies have appeared to address the growing complexity and scale of modern software systems. Among these, Feature-Driven Development (FDD) has gained recognition because its structured, iterative approach focuses on the delivery of tangible, client-valued features [1]. FDD is typical for medium to large-scale projects for its modularity, progress, and clear documentation. However, it faces challenges in dynamic environments where coordination, adaptability, real-time feedback, and collaboration are critical to the success of the development process [2]. To be successful, teams need to be able to adapt, receive real-time feedback, and work together effectively. In the ever-incremental world of software development today, it is more important than ever to have methods that can adapt to changing needs and enhance teamwork. This study used a feature-driven development approach, which aims to quickly plan and build software features [3] [4]. At the same time, Multi-Agent Systems (MAS) have become a promising way to create smart, independent, and interactive parts that can make decisions on their own [5]. MAS has been used successfully in many fields, including robotics, distributed computing, and intelligent systems. This paper discusses how adding MAS to the FDD methodology enhances its flexibility and collaboration capabilities. We have a new way to use MAS to improve feature planning, task delegation, and collaborative development in software engineering by giving intelligent agents specific roles that are in line with FDD processes.

The remainder of the paper is organized as follows. In Section 2, we introduce the FDD process in more detail, and we introduce multi-agent systems and related works. In Section 3, we state a motivation and problem statement. In Section 4, we describe the proposed approach to improve the efficiency of FDD. Finally, in Section 5, we summarize our approach and outline the future work.

Background and Related Work

Feature-Driven Development (FDD)

Feature-Driven Development (FDD) is a highly iterative and collaborative agile development methodology, introduced by Jeff De Luca and Peter Coad, that was published in 1999 after its application at an international bank in Singapore in 1997 [6]. This methodology focuses on building and delivering software by incrementally developing small, client-valued features. that is composed of five processes, features [1][3][4]. The figure illustrates the overall five (5) primary FDD procedures.

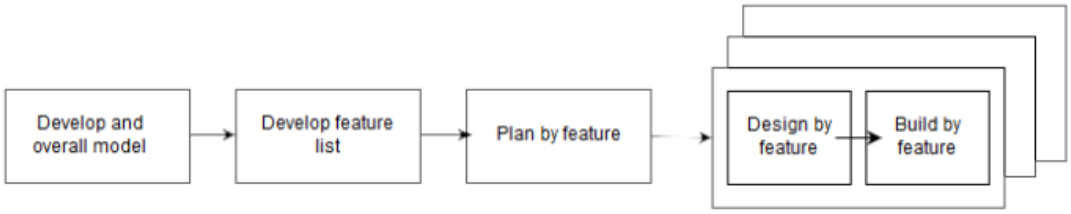

Figure 1: Feature Driven Development Process [6]

The FDD process contains five sequential activities [6]:

1. Develop an Overall Model: Collaboratively create a high-level domain model to understand the system's scope and context.
2. Build a Features List: Decompose the domain model into a comprehensive list of small, client-valued features.
3. Plan by Feature: Determine features and specify them to development teams, creating a feature-based plan.
4. Design by Feature: Design each feature in detail, ensuring it merges seamlessly into the general system architecture.
5. Build by Feature: Develop, test, and combine each feature, ensuring it faces predefined specifications and quality standards.

This methodology supports several best practices, including domain object modeling, feature-based development, individual class ownership, and regular builds, all aimed at improving both the quality and efficiency of the development process [6]. The assurance on that enhances communication among team members and ensures better code quality[7]. FDD's strengths include its scalability and modular approach, but it faces challenges in flexibility, adaptability, and real-time coordination in distributed environments [8].

Multi-Agent Systems (MAS)

A multi-agent system (MAS) is a computational model consisting of a set of independent agents that interact and cooperate to achieve individual or collective goals. These agents are capable of observation, reasoning, and decision-making in changing environments [5]. Each agent works autonomously within the system, understanding its environment and making decisions based on reasoning mechanisms to perform tasks. MAS are particularly effective in changing, complex, and distributed environments, where centralized control is impractical or ineffective [5]. In software engineering, MAS have been consolidated to be effective in managing complex tasks such as project planning, task scheduling, and resource division [9]. Their independence, adaptability, and ability to work concurrently make them ideal for enhancing traditional software processes. Additionally, MAS facilitate decentralized problem-solving and the parallel execution of tasks, making them suitable for applications such as smart grids, autonomous vehicles, distributed robotics, and adaptive software systems [10]. MAS also facilitates the modeling of complex interactions between heterogeneous components, providing a high degree of organization and adaptability in changing environments.

Integration of MAS in Software Engineering

The use of MAS in software development processes automation, and agile methodologies has been a growing area of interest. Several studies have sought the use of agents in software development environments [11], agent-based modeling for requirement engineering [12], and multi-agent-based DevOps automation [13]. However, little research has directly addressed how MAS can be embedded within the FDD methodology to improve its responsiveness and efficiency—an issue this paper aims to explore. This section will present six related works that highlight how multi-agent systems are used to enhance the software development methodologies:

In [14], Shanawar Ali et al. proposed a multi-agent system (MAS) to support the Scrum methodology in managing software development processes. The team presented a model that organizes and simulates software development tasks through autonomous agents for the following Scrum roles: The product owner agent extracts key concepts (nouns and verbs) from the System Requirements Document (SRS), while the Scrum Master agents organize the sprint backlog, assign tasks, and conduct daily meetings and retrospectives. The review agents then conduct quality control. Evaluating the model using a single SRS document confirmed its ability to automatically analyze user requirements, form teams based on specialization, estimate development costs, and track task allocation during sprint cycles. The study highlights the role of agent systems in describing software methodologies in greater detail by decomposing process roles into independent agents. This model not only mimics the steps of Scrum but also provides an architecture that can be generalized to other methodologies, such as FDD. In this case, each feature can be represented as an independent processing unit, handled by a specific agent.

W. Wysocki and C. Orłowski [15] emphasized the importance of using Multi-Agent Systems (MAS) when planning and supporting hybrid software development methodologies, proposing a model known as AGOMO (Agent-Object Model). The AGOMO model simulates the behaviors of different roles within the software development lifecycle using intelligent agents, helping organizations to choose the most appropriate methodology for a given project. AGOMO consists of a set of agents, each one responsible for specific activities such as analysis, planning, execution, or monitoring. These agents collaborate in an environment built using the JADE platform, simulating real-life interactions between development teams. The model was applied to two popular methodologies, Scrum and RUP, enabling the suitability of each to be analyzed according to the characteristics of the projects studied. The study's significance is as follows:

- It demonstrates how agent systems can characterize the components of any development methodology dynamically and clearly, representing each activity as an independent agent.
- MAS enables the analysis of the strengths and weaknesses of each stage of the methodology, allowing it to be improved or adapted to the project context.
- The model supports decision-making when selecting or integrating development methodologies, such as FDD or Scrum, and enhances intelligent planning and the automated customization of methodologies according to project requirements.

While the study did not directly address the FDD methodology, the analytical structure and task segmentation of the model align with the sequential and systematic FDD approach to building and developing features, making it adaptable to support FDD-like processes for analyzing and building features.

In [16], M. H. Nguyen et al. introduce AgileCoder, a new framework that integrates Agile software development concepts with multi-agent system designs to mimic how human development teams operate together. In this system, different agent roles, such as product manager, developer, reviewer, and tester, are allocated and work together in iterative sprint cycles that follow the Scrum methodology. The Dynamic Code Graph Generator (DCGG) is a key part of the framework. It dynamically maps code dependencies, which helps agents better grasp

how the project is put together and make more accurate changes while reducing duplication. The model also uses a structured communication protocol based on the instructor-assistant model. This makes sure that the clarity of interactions and the context are protected during the development process. Comparative tests show that AgileCoder routinely does better than well-known models like ChatDev and MetaGPT on coding benchmarks like HumanEval and MBPP, as well as a unique benchmark called ProjectDev that is meant to reflect the needs of real-world projects. The results show how useful it is to use agile principles in multi-agent contexts, and they show that AgileCoder is a relevant addition to the field of intelligent, behavior-oriented software engineering. Y. Wautelet et al. in [17] presented a framework for developing multi-agent systems using Agile methodologies such as FDD by analyzing user stories, designing an agent for each story, and transforming it into a directed agent model using JADE, which promotes accurate feature tracking and efficient task allocation. J. He et al. [18] discussed how to combine LLMs with MAS. They state that agents with linguistic intelligence can automate the representation of FDD feature lists, design features on their own, and run integration tests, which speeds up implementation and reduces the need for manual coordination. F. Feyzi [19] introduced a development methodology for self-adaptive multi-agent systems that begins with constructing a context-aware model and then decomposing it into smaller, executable components. These components are assigned to individual agents capable of adjusting their behavior in response to environmental changes. While the approach does not explicitly reference Feature-Driven Development (FDD), its reliance on modular task segmentation and feature allocation reflects core ideas of FDD. This perspective offers useful insights for researchers exploring how behavioral development strategies, such as BDD, might enhance adaptability and coordination in agent-based systems.

Motivation and Problem Statement

Motivation

In modern software development environments, there are increasing limitations for producing high-quality products quickly and with the ability to adapt continuously to changing requirements and stakeholder needs. Although the Feature-Driven Development (FDD) methodology offers a structured, feature-oriented framework, it has some limitations, including

- Dependence on centralized coordination, which limits its effectiveness in distributed environments.
- Pauper's ability to dynamically handle changing requirements.
- Dependence on manual task allocation and planning, which can lead to delays or variation in the quality of execution.

Recent studies have indicated that limited adaptability and a lack of automation and intelligent orchestration are among the most prominent challenges of traditional methodologies [2][13]. These challenges can be handled by incorporating multi-agent systems (MAS), which provide intelligent, autonomous, and collaborative behaviors that are well-suited to dynamic and distributed environments.

Research problem

Although the FDD methodology provides a framework for feature development, it lacks flexibility, decentralized coordination, and the ability to make real-time decisions in changing environments. Therefore, the main research question is:

- How can multi-agent systems be integrated into a feature-driven development methodology to enhance coordination, increase adaptability, and improve the efficiency of modern software development?

Method Overview

The underlying principle contributing to the efficacy of this research lies in our proposition to employ the paradigm of multiple agents, which assumes a pivotal role due to its recognition as a well-established software construct. Furthermore, the system operates within an environment characterized by autonomy, responsiveness to environmental fluctuations, proactivity in the pursuit of objectives, and sociability, all of which represent advantageous attributes for the system being developed in this context. For the following reasons, multi-agent systems exhibit a notable adaptability to the design of the FDD system:

1. The FDD process is inherently flexible and dynamic.
2. Agents serve as a prevalent metaphor for human actions.
3. Control, expertise, and data distribution are all self-regulated.
4. The agent possesses the capability to provide a high-level graphical representation of behavior.

Additionally, because of their many advantages, agent systems are utilized in a wide range of applications. The following are some of their main traits, which are mentioned in [20]:

1. Autonomy: This characteristic suggests that the agent is capable of acting on its own, meaning that it needs little outside assistance or input from people or other agents and that it has internal states and actions.
2. Sensing and acting, or reactivity to situations: This trait describes the agent's capacity for rapid reaction as it monitors its surroundings and responds to any alterations.
3. Proactiveness or goal-directed behavior: This attribute characterizes the agent's capacity to not only respond quickly to environmental changes but also actively start events and look for chances to generate activity.
4. Social Ability: Agent's social ability describes system agents' ability to communicate with one another.

Given that the key benefit of multi-agent systems is their ability to facilitate distributed problem-solving, the agents included in the model must coordinate their actions. All-in-one agent communication facilitates agent coordination and cooperation by allowing individual agents to interact, which includes information sharing. Our proposed work outlines a model for managing and organizing the features-driven development workflow across multi-agent systems. The general structure of this model is depicted in Figure 2. In this suggested model, we show the nature of communication in MAS at all levels and define the mechanism that provides the FDD team with all the tools they need to exchange work-related incentives.

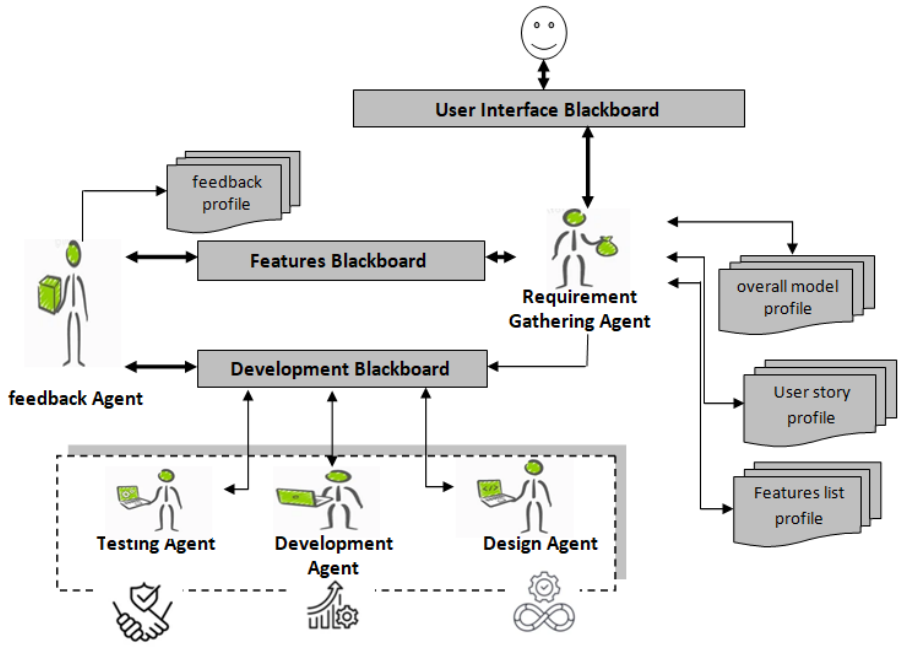

Figure 2: overview of the proposed model.

Before delving into the enumeration of the specifics of our research, it is imperative to elucidate the foundational structures that underpin our proposed model, which are articulated as follows:

- **Multi agent system roles**

This proposed model contains of five main agents with their various responsibility, are succinctly described below in table 1.

Table1.Proposed FDD Agents.

No	Agent role	Responsibility
1	Requirement Gathering Agent	is responsible for finding and defining features in response to client requirements. This entails involving stakeholders in gathering final requirements and ensuring that features are prioritized and matched with business objectives. and employs Collecting user stories to capture functional requirements from the user's point of view, with an emphasis on their needs and desired outcomes.
2	Design Agent	is responsible for design each feature in detail, ensuring conformity with the overall concept.
3	Development Agent	is responsible for implementing features in accordance with design criteria and working code for features.
4	Testing Agent	is responsible for tests every developed feature automatically to make sure it satisfies the criteria.
5	Feedback Agent	is responsible for Gathers feedback from stakeholders regarding completed features and Summary of feedback and recommendations for improvement

- **FDD profile**

There are three main profiles that make up the MAS-based model for the FDD : the overall model profile , User story profile, Features list profile, and feedback profile. These profiles are briefly discussed in table2.

Table2. Proposed Main Profiles.

No	Profile name	Description
1	User Story Profile	contains a brief description of a feature from the user's point of view.
2	Features List Profile	the features list profile is containing a vital component that defines the precise functionalities that will be implemented. it acts as a roadmap for the project, guaranteeing alignment with the user's needs.
3	Feedback Profile	contains the following items: assessment of features, quality insights, stakeholder input, team reflections, progress evaluation, and suggestions for future features

- **FDD blackboard**

This proposed model has three primary blackboards, each with distinct properties, which are concisely outlined in Table 3 below.

Table3. Proposed Main blackboards.

No	Blackboard name	Description
1	User Interface Blackboard Agent	starts with the FDD members and is triggered by an agent action. It then provides each participant with the outcomes of every action taken throughout the session.
2	Features Blackboard	It mediates communication of the user agent, product owner agent, and scrum master agent including their corresponding databases. In more detail. It collects data from the agents and submits it to the appropriate database. It also executes the opposite procedure, retrieving pertinent data from the databases and sending it to the various agents in the product blackboard.
3	Development Blackboard	It mediates communication of the scrum master agent and team agents. In this blackboard, the scrum master agent is responsible of coordinating work among team members and controlling time sprint.

to illustrate the features list process, a sequence diagram is given in Figure 3, which demonstrates the process implicit in user story as indicated, when the user agent submits user story to Requirement Gathering Agent, Priorities are set based on the customer's desire and update features list by using Requirement Gathering Agent**.**

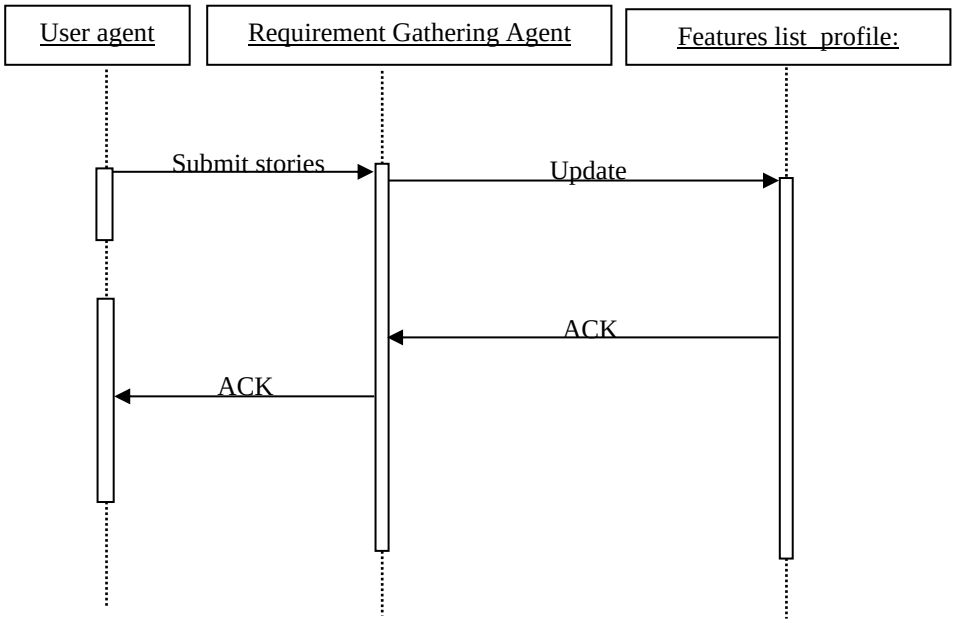

Figure 3. Sequence diagram for making features list priority

In the conclusion of the work, the agent that responsible of the development team will document the output of the last Features list and put it in one of the standard system requirements specification templates. Since, the agent structure differs from the other normal systems structure; its documentation differs in order to suit it. Therefore, we will use Table 4 in [10].

Table 4. sample requirement table.

Name	--
description	--
Cause	--
Information used	--
Outputs	--
Required effect	--
Identifier	--

Conclusion

Feature-Driven Development, when combined with Multi-Agent Systems, produces a solid architecture that improves the software development lifecycle. Previously, we produced a study that was conducted on the multi-agent system as a conceptual model for managing the Scrum process [21]. This method increases responsiveness and harmonizes development efforts with stakeholder expectations by utilizing autonomous agents to streamline communication, automate tasks, and collect user input. This paper has reported the proposed model to construct a support for the Feature-Driven Development (FDD) process predicated upon Multi-Agent Systems (MAS). In our proposed framework grounded in MAS, scalability is posited as a desirable attribute of a network, system, or process, denoting the capability of a system to adapt to an increasing number of components or entities while effectively managing the escalating workloads. At present, the majority of design paradigms do not accommodate this principle, and those that do are often susceptible to limitations in expansion. The scalability of the proposed model has been scrutinized in relation to the number of users within the team. The incorporation of agents enhances the proposed model, rendering it dynamic, as agent technology inherently possesses this capability. These agents are proficient in establishing and severing connections within an FDD process system with ease. Moreover, flexibility constitutes a pivotal consideration in FDD development systems, since each requirement-gathering agent must possess the capacity to comprehend the processes while acquiring the features list and user stories from the end user. Additionally, the interchange of information among team agents is facilitated through an intelligent blackboard that enables all functionalities, thereby allowing an agent to access all requisite information at any given moment. Subsequent investigations could go deeper into the empirical verification of this design in actual projects, offering valuable perspectives on its pragmatic uses and advantages. As well as conducting empirical studies to assess the efficacy of these integrations in many development contexts, future research ought to concentrate on creating standardized frameworks for integrating MAS into FDD.

References

[1] S. R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002.

[2] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2016.

[3] R. Kumari and Heena, "Study Of Feature Driven Development Using The Concepts Of Object Oriented," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 2, no. 3, 2013.

[4] S. Ozercan, "Adapting Feature-Driven Software Development Methodology to Design and Develop," M.S. thesis, The Russ College of Engineering and Technology, Ohio Univ., 2010.

[5] [5] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. Wiley, 2009.

[6] [6] M. Rychlý and P. Tichá, "A Tool for Supporting Feature-Driven Development," in *Balancing Agility and Formalism in Software Engineering*, B. Meyer, J. R. Nawrocki, and B. Walter, Eds. Springer, Germany, 2008, pp. 196–207. doi: 10.1007/978-3-540-85279-7_16.

[7] R. Palacios, M. E. Morales-Trujillo, and F. J. Pino, "Agile software development methodologies in highly regulated environments: A systematic literature review," *Comput. Stand. Interfaces*, vol. 62, pp. 54–67, 2019. doi: 10.1016/j.csi.2018.10.005.

[8] S. Ambler, "Agile Modeling and Feature-Driven Development," *Agile Modeling*, 2005. [Online]. Available: www.agilemodeling.com.

[9] N. R. Jennings, "An agent-based approach for building complex software systems," *Commun. ACM*, vol. 44, no. 4, pp. 35–41, 2001.

[10] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Auton. Agents Multi-Agent Syst.*, vol. 1, no. 1, pp. 7–38, 2014. doi: 10.1023/A:1010090405266.

[11] J. P. Müller, M. Pischel, and M. Thiel, "Modelling reactive behavior in vertically layered agent architectures," *Int. J. Hum.-Comput. Stud.*, vol. 48, no. 1, pp. 29–49, 2003.

[12] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, “Tropos: An agent-oriented software development methodology,” *Auton. Agents Multi-Agent Syst.*, vol. 8, no. 3, pp. 203–236, 2004.

[13] A.-U.-H. Yasar, M. Tahir, and M. Asim, “Agent-based DevOps pipeline automation: A simulation-based study,” *J. Syst. Softw.*, vol. 162, p. 110489, 2020.

[14] S. Ali, N. Ahmad, and J. Iqbal, “Multi-agent system using Scrum methodology for software process management,” in *Proc. Int. Conf. Intell. Technol. Appl. (INTAP)*, 2018, pp. 141–154. doi: 10.1007/978-3-030-03840-3_12.

[15] W. Wysocki and C. Orłowski, “A multi-agent model for planning hybrid software processes,” *Procedia Comput. Sci.*, vol. 159, pp. 1084–1093, 2019. doi: 10.1016/j.procs.2019.09.253.

[16] M. H. Nguyen, T. P. Chau, P. X. Nguyen, and N. D. Q. Bui, “AgileCoder: Dynamic Collaborative Agents for Software Development based on Agile Methodology,” *arXiv preprint* arXiv:2406.11912, Jun. 16, 2024.

[17] Y. Wautelet, G. Haillot, and V. Yerramalla, “User Story Driven Development of Multi Agent Systems: A Process Fragment for Agile Methods,” in *Proc. 11th Int. Symp. Intell. Distributed Comput.*, 2017.

[18] J. He, C. Treude, and D. Lo, “LLM Based Multi Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead,” *arXiv preprint* arXiv:2404.04834, Apr. 7, 2024.

[19] F. Feyzi, “Model driven development of self adaptive multi agent systems with context awareness,” *Int. J. Comput. Aided Eng. Technol.*, vol. 12, no. 2, pp. 131–156, May 2020.

[20] T. Wickenberg and P. Davidsson, “On multi agent based simulation of software development processes,” in *Int. Workshop Multi-Agent Syst. Agent-Based Simulation*, Springer, Berlin, Heidelberg, 2002, pp. 171–180.

[21] Khamisa A. Yousef, and Fatma H. Fazzani. "Multi Agent System as Conceptual Model for Managing Scrum Process." *Journal of Bani Waleed University for humanities and Applied Science* 8, no. 2 (2023): 524-539.