http://doi.org/10.35784/iapgos.7810 *received: 25.10.2025 | revised: 19.12.2025 | accepted: 07.01.2026 | available online: 30.06.2026*

APPLICATION OF UML IN THE DEVELOPMENT PROCESS OF COMPUTER GAMES

Lyudmila Samchuk[1], Yuliia Povstiana[2], Yaroslav Tymoshchuk[2]
[1]Lutsk National Technical University, Faculty of Transport and Mechanical Engineering, Department of Applied Mechanics and Mechatronics Lutsk, Ukraine,
[2]Lutsk National Technical University, Faculty of Computer and Information Technologies, Department of Software Engineering, Lutsk, Ukraine

Abstract. *The paper explores the use of UML diagrams in the process of developing computer games as an effective tool for organizing team interaction and planning project architecture. Particular attention is paid to the use of class, use case, sequence, and activity diagrams for modelling game mechanics, code structure, and user interaction scenarios with the game. The paper examines how UML diagrams assist development teams – including programmers, designers, testers, and managers – in aligning on requirements, identifying logical inconsistencies at early stages, distributing tasks effectively, and maintaining development transparency. The results presented demonstrate the importance of using UML to improve game quality, speed up the development cycle, and reduce development costs.*

Keywords: UML diagrams, class diagrams, sequence diagrams, state diagram, computer games, game architecture, game algorithms

ZASTOSOWANIE JĘZYKA UML W PROCESIE TWORZENIA GIER KOMPUTEROWYCH

Streszczenie. *Artykuł analizuje wykorzystanie diagramów UML w procesie tworzenia gier komputerowych jako skutecznego narzędzia do organizowania współpracy zespołu i planowania architektury projektu. Szczególną uwagę poświęcono wykorzystaniu diagramów klas, przypadków użycia, sekwencji i aktywności do modelowania mechaniki gry, struktury kodu i scenariuszy interakcji użytkownika z grą. W artykule przeanalizowano, w jaki sposób diagramy UML pomagają zespołom programistycznym – w tym programistom, projektantom, testerom i menedżerom – w uzgadnianiu wymagań, identyfikowaniu niespójności logicznych na wczesnych etapach, efektywnym rozdzielaniu zadań i utrzymywaniu przejrzystości procesu tworzenia. Przedstawione wyniki pokazują znaczenie wykorzystania UML dla poprawy jakości gier, przyspieszenia cyklu tworzenia i obniżenia kosztów rozwoju.*

Słowa kluczowe: diagramy UML, diagramy klas, diagramy sekwencji, diagramy stanów, gry komputerowe, architektura gier, algorytmy gier

Introduction

Video game development is a complex and multifaceted process that combines creativity, technical knowledge and strategic planning. A successful game requires not only the ability to create exciting gameplay and a unique visual style, but also the correct organization of all stages of development, from the formation of the concept to the release of post-release updates. Each stage is crucial, and the coordinated efforts of all stakeholders throughout the development process are essential to the game's success. An important part of this process is team management, organizing funding, creating and testing game mechanics, as well as effective promotion in the market. In this context, UML diagrams, such as class, sequence and state diagrams, are powerful tools for modelling and organizing this process, as they allow you to clearly visualize the interactions between different roles and stages in game development.

However, despite the widespread use of UML in software engineering, its systematic application to the coordination and structuring of the computer game development process remains insufficiently formalized in existing studies.

1. Literature review

Analysis of domestic scientific sources for the period 2019–2023 indicates the active use of UML (Unified Modelling Language) in the process of developing computer games by Ukrainian researchers. These studies cover various aspects of using UML, from automaton programming to designing game scenarios and information systems.

Instrumental features of automaton programming for computer games. The article [4] considers the use of UML diagrams to formalize technological processes in game design. They emphasize that the use of the automaton programming paradigm based on finite state machines facilitates the analysis and testing of program code. However, the authors note that the transition from UML modelling to the implementation of a finite state machine in a game scenario may be complicated by certain nuances that need to be taken into account to reduce the risk of errors.

The article [3] examines the application of UML modelling for the architectural design of a mobile game-oriented information system. UML diagrams are used to formalize system structure, functional components, and interaction logic between users and software modules. The study demonstrates that UML-based models support systematic representation of gamification mechanisms within the overall system architecture. This work is relevant to the present study as it illustrates the applicability of UML for modelling interactive systems that combine software architecture with game-related functionality.

A UML-based methodology has been proposed for modelling adventure-based educational games, where a graphical notation derived from UML is used to represent structural and semantic elements of game design [5]. The authors outline how UML can be extended and adapted to accommodate specific characteristics of game systems and to support collaborative design processes between technical and non-technical participants. This approach demonstrates a broader applicability of UML beyond standard software modelling toward structured design of interactive game environments.

One of the recent studies by researchers in the field of requirements engineering proposes a use case–driven methodology for modelling computer games using UML diagrams [2]. The article emphasizes the central role of use case diagrams in defining user interaction scenarios and aligning them with internal system behaviour. The authors argue that a structured UML-based approach improves the traceability of requirements and supports the iterative nature of game development. This is particularly relevant for adventure and story-driven games, where branching logic and complex interactions must be accurately specified and tested early in the design phase.

In contrast to purely technical implementations, the study integrates behavioural modelling with requirement elicitation, offering a clear framework for identifying, refining, and validating game features through UML. The proposed methodology addresses not only software architecture but also gameplay logic and user expectations. As a result, it supports consistency across development stages and reduces miscommunication between stakeholders. This aligns well with the objectives of the current article, further supporting the value of UML as a versatile tool in game development.

The article [2] is devoted to the use of UML modelling in the development of a logic game. The authors consider how UML can help in creating an effective game structure, defining work algorithms and optimizing the development process.

The study [6] focuses on the application of UML as a formal modelling tool in the computer game design process. The authors analyse the use of UML diagrams – specifically use case, class,

activity, and sequence diagrams – to represent system structure, behaviour, and interactions between software components. UML modelling is employed to formalize game logic, object relationships, and system states, providing a structured representation of the development process. The article demonstrates that UML supports systematic organization of software architecture and facilitates consistency between design and implementation stages, thereby improving coordination within the development team. The analysis is limited to UML-based modelling aspects and does not address gameplay mechanics or implementation-specific details.

The paper explores the use of UML modelling to develop the QueensChallengeGame puzzle game, which spans levels from 1×1 to 25×25. The authors emphasize that UML as a standard modelling language provides efficient visualization and documentation of object-oriented systems [1]. However, traditional UML diagrams have limited functionality for game modeling, as they cannot clearly express specific requirements for game mechanics. As a result, an extension of UML is proposed that includes use case, sequence, activity, and class diagrams.

The research focuses on analysing the problem of placing n queens on a chessboard so that they do not threaten each other. Each level of the game is a separate task, which creates difficulties for players, as it requires downloading new levels. This increases time consumption, reduces user engagement, and limits the commercial potential of the game. To solve this problem, a single system is proposed that contains all levels in a single application.

The UML extension includes a context diagram for a general representation of the system, as well as additions to traditional UML diagrams. The use case diagram illustrates how the user interacts with the system, encompassing actions such as selecting a level, starting the game, and viewing the solution. The sequence diagram details the communication between objects, showing how the player selects a level, obtains a solution, and interacts with the game. The activity diagram illustrates the workflows, including the level selection mechanism, queen placement verification, and game completion logic.

The class diagram demonstrates the game structure, including classes for level management, user interface, and queen placement mechanics. The conclusions of the article emphasize the importance of extending UML for modelling the behaviour of game systems. The proposed approach allows for a clear definition of game requirements, reduces the complexity of analysis and development, and improves consistency between the specification and implementation of the software. The use of UML in combination with adaptation to the specifics of the game helps to optimize the puzzle creation process and increase user convenience.

The article is devoted to the use of UML modelling in the process of designing computer games. The authors emphasize that UML diagrams allow not only to formalize the game process, but also to ensure its balancing. The study considers the main types of balances in video games, such as the balance of power, time, space, and economy.

The main idea of the article is that UML modelling contributes to the effective display of logical relationships between game objects and events [7]. Unlike other visualization methods, UML provides flexible tools for structural analysis of game mechanics. For example, case diagrams help to understand the interaction between characters and game events, and state diagrams allow to assess the dynamics of changes in game situations.

Special attention is paid to the issues of game balancing. The authors give examples when an unbalanced game process leads to a loss of interest in players. To prevent this, UML diagrams are used, which allow you to visualize weak points in the game mechanics and make adjustments to them. For example, when developing combat systems, a game designer can use a conceptual class diagram to determine the interaction between weapons, characters, and their characteristics.

In addition, the article demonstrates the practical application of UML in the development of games of various genres, including strategy, role-playing, and simulation games. The work contains an analysis of the UML methodology in terms of its capabilities for modelling gameplay and analysing game mechanics.

The conclusions of the article underscore the critical role of UML diagrams in ensuring game balance at the development stage, as they contribute to minimizing the potential risks of subsequent modifications and enhancing the efficiency of the game development process.

UML modelling has previously been applied to the structural representation of management systems in technical domains [8]. This study extends the application of UML diagrams to the modelling of interactive software systems, focusing on computer game development.

2. Research methodology

UML (Unified Modelling Language) plays an important role in coordinating the work between the various departments of computer game development, including design, programming, testing, and production. UML diagrams, such as use case, activity, class, and sequence diagrams, provide each team with a clear understanding of their responsibilities and how they interact with other system components. For example, a use case diagram will help game designers and producers describe the scenarios of game use, which are then implemented by programs in the form of classes and methods. This provides a single view of the game's functionality at all stages of development.

With UML, the testing team can focus on the specific parts described in the diagrams and create test cases to verify the interaction between modules. At the same time, programs execute class and component diagrams to understand the code structure and dependencies between parts of the game, which reduces the risk of errors. Thus, UML acts as a "common language" that facilitates communication between departments and ensures the convenience of a game project from concept to implementation.

3. Results

This section presents the results of applying UML diagrams to model the computer game development process across its main organizational and technical stages. From idea to final release, a game goes through many stages, each of which plays a key role in its success. It is important not only to create exciting gameplay and a unique visual style, but also to ensure optimization, testing and effective promotion in the market.

This study examines all the main stages of game development: from market analysis and concept formation to the release of updates and post-release project support. Particular emphasis is placed on the significance of adequate funding, effective task distribution within the development team, and the incorporation of high-quality player feedback. UML provides a variety of diagram types that can be effectively utilized throughout the computer game development process.

Figure 1 shows a class diagram that models the architecture of object-oriented game development, defining the core classes and their relationships. It covers key roles in the game development process, from funding and project management to development, testing, and marketing. The Publisher class is responsible for funding and releasing the game, while the GameDirector forms the vision of the project and oversees its implementation. The Project contains general parameters, including genre, style, and status. The GameDesigner develops mechanics and balance, while the LeadProgrammer defines the code architecture and coordinates the team of programmers.

The QA Manager is responsible for testing and bug identification, while the Localization Manager oversees the translation and adaptation of the game for various regions. The ArtDirector defines the visual style, and the AudioDirector creates the sound atmosphere. The MarketingManager plays an important role, developing promotion and advertising strategies. This structure promotes an organized approach to game

development, optimizing interaction between teams and ensuring a high-quality result.

Figure 2 shows a sequence diagram showing the order of interaction of actors during the game development process, showing how information and actions are passed between key project participants. First, the Publisher provides the GameDirector with the game concept, after which the GameDesigner is tasked with developing game mechanics and balance, while the LeadProgrammer is responsible for creating the game architecture. Next, the GameDesigner and LeadProgrammer work on implementing the mechanics, after which the LeadProgrammer integrates the graphics with ArtDirector and adds sound design from AudioDirector. The finished build is passed to the QA_Manager for testing, who informs the LeadProgrammer about any bugs found. After the errors are fixed, a balance check is performed, and the final content is passed to the LocalizationManager for adaptation in different languages. The LocalizationManager agrees on the correctness of the translation with the GameDesigner, after which the versions are beta-tested in cooperation with the QA_Manager. The final stage includes final testing before release. At this time, the MarketingManager forms an advertising strategy and agrees it with the Publisher, who, in turn, approves the release date together with the GameDirector. After the game is released, the GameDirector, together with the QA_Manager, analyses feedback and works on updates, ensuring continued support for the project.

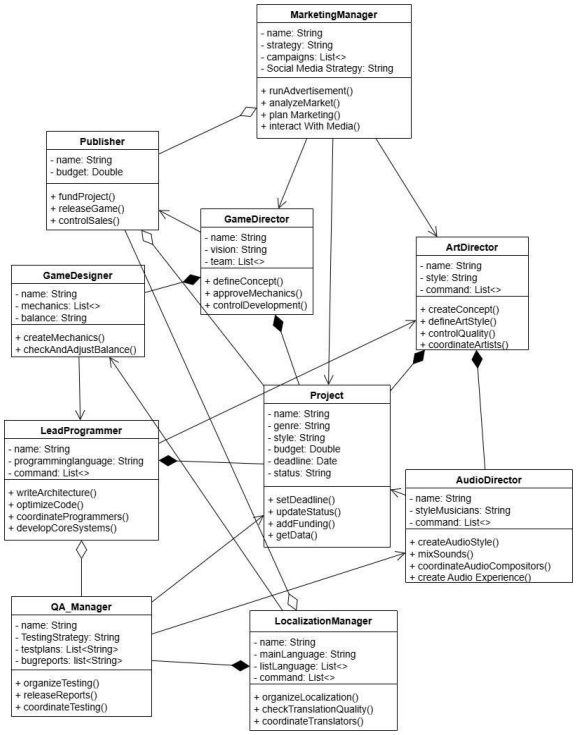


Fig. 1. Object-oriented game development architecture

Figure 3 shows a state diagram that shows the game development process, showing the sequence of tasks performed by different departments, from receiving the game document to the final release of the product. The process starts with defining the game concept and distributing tasks between departments. The graphics, development, and audio departments receive their tasks and begin working on the relevant aspects of the project.

After creating the basic materials, the information is passed to the localization department, where the content is translated and adapted for different markets. The tasks are broken down into smaller steps for more efficient implementation, and the localization process is coordinated depending on the languages. After the main stages of development are completed, the audio, graphics, and program code are integrated.

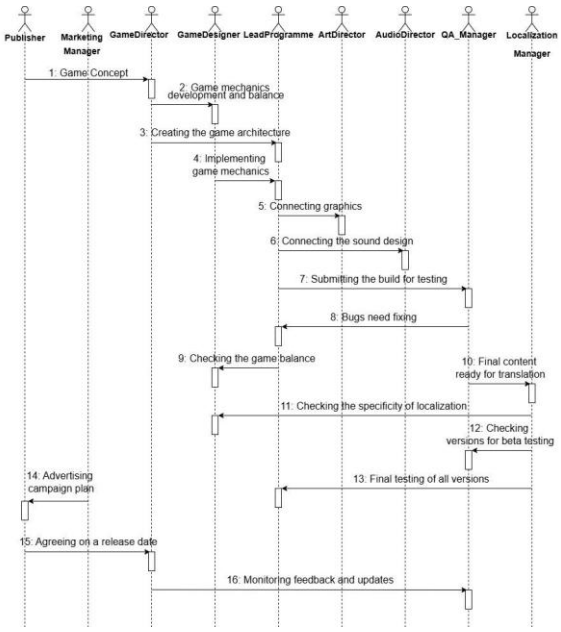


Fig. 2. Actor interactions during the game development process

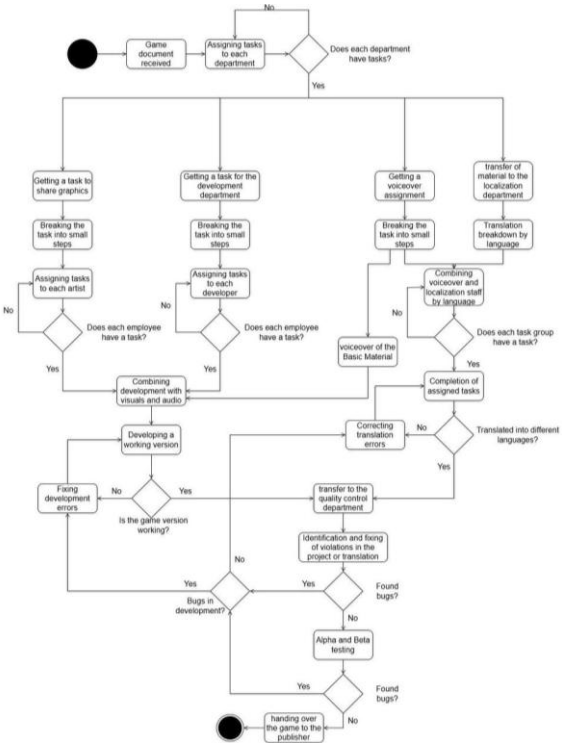


Fig. 3. The game development process, and the sequence of tasks performed by different departments

Next, the game is tested. First, bugs are identified and fixed, after which the team fixes the errors. This is followed by alpha and beta testing, which allows you to test the game before the final release. When the product is ready, it is handed over to the publisher, who organizes a marketing campaign and launches the game on the market.

The state diagram for the enemy shows how behaviour changes depending on the situation. Initially, the enemy is in a patrol state, moving along a defined route. If it notices the player, it switches to pursuit mode, trying to catch up with him. When it reaches a certain distance, the enemy activates an attack state, damaging the player. If it receives critical damage, it switches to a flight state, trying to avoid combat.

This structure allows you to model various aspects of the development and interaction of objects in the game, providing a clear understanding of all processes from the initial concept to the final release.

Figure 4 shows an activity diagram that shows the states and their execution sequence described in the state diagram. Game development begins with market research and audience analysis, which allows you to identify current trends and genres. After that, a basic concept is created, which includes the genre, plot and key mechanics, as well as an idea of the appearance of the game and how the player interacts with it.

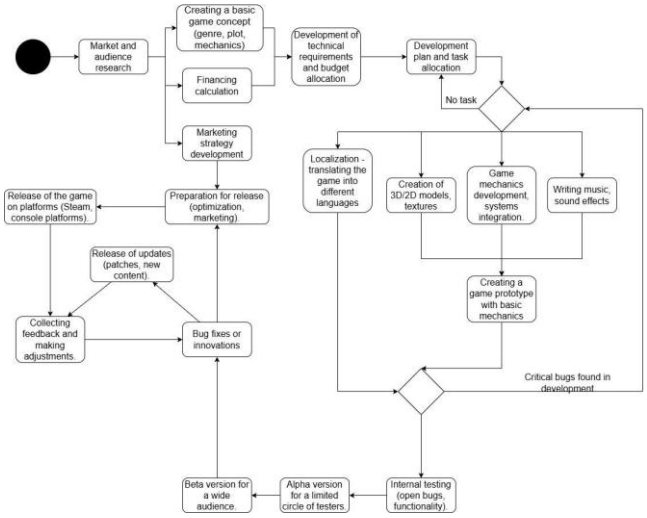


Fig. 4. An activity diagram that depicts the states and their execution sequence described in a state diagram

Next, a financing calculation is performed to determine the required budget. After that, technical requirements are drawn up and financing is distributed, taking into account the resources required for development. A detailed work plan is developed and tasks are distributed among team members.

At the visual design stage, 3D and 2D models are created, textures that define the style of the game. In parallel, game mechanics are developed and the necessary systems are integrated. Sound is also created, including music and effects.

The next stage is the creation of a prototype containing the main game mechanics. After that, internal testing is carried out, which allows you to identify critical errors. Next, an alpha version is developed, which undergoes closed testing, after which a beta version is released for a wide audience.

Based on the feedback received, bugs are fixed and improvements are implemented. Optimization and marketing preparation are carried out before the release. The game is released on selected platforms, such as Steam or consoles.

After the release, player feedback is collected, which helps in further adjusting the game. Updates are released in the form of patches or new content. A marketing strategy is developed, including an advertising campaign, to attract new players and maintain interest.

4. Conclusions

The development of UML diagrams for modelling the video game creation process was driven by the need to clearly structure all stages of this complex and multi-component process. Creating a game is not just programming or design, but the interaction of a large number of specialists from various fields, starting from game design and ending with marketing and post-release support. To better understand the roles of each participant, as well as the connections between them, it was advisable to use visual modelling tools.

UML diagrams made it possible to present these connections clearly, focusing on the logic of interaction between the publisher, game director, programmers, designers, testers and other project participants. This made it easier to assess how tasks are distributed, where bottlenecks may arise, and which communication links require special attention.

Also, modelling helped to reveal not only the technical, but also the organizational side of game creation. Without effective coordination among team members, even the most promising concept may fail to progress beyond the initial idea stage. That is why UML diagrams have become not just illustrative material, but a tool for understanding processes and finding ways to improve them. As a result, this approach allows not only to increase the efficiency of development, but also to increase the chances of product success in the competitive video game market.

References

[1] Abu-Dalbouh, H. M., AlJibreen, G., & AlDowighri, N. (2018). Generic modelling using UML extensions for queens challenge puzzle game from 1 to 25 levels system. *International Journal of Software Engineering & Applications*, *9*(6), 31–39. https://doi.org/10.5121/ijsea.2018.9603

[2] Albaghajati, A., & Hassine, J. (2022). A use case driven approach to game modeling. *Requirements Engineering*, *27*(1), 83–116. https://doi.org/10.1007/s00766-021-00362-4

[3] Basalkevych, O., & Hrybovskiy, O. (2023). Information System Concept for a Sports and Game Mobile Application with Elements of Artificial Motivation to Lead a More Active Lifestyle. *Journal of Lviv Polytechnic National University 'Information Systems and Networks'*, *14*, 126–141. https://doi.org/10.23939/sisn2023.14.126

[4] Blazhko, O., Antonyuk, V., & Troianovska, Y. (2018). Instrumental features of automata-based programming of computer games. *Management of Development of Complex Systems*, (35), 83–89.

[5] De Lope, R. P., Medina-Medina, N., Urbieta, M., Lliteras, A. B., & Mora García, A. (2021). A novel UML-based methodology for modeling adventure-based educational games. *Entertainment Computing*, *38*, 100429. https://doi.org/10.1016/j.entcom.2021.100429

[6] Kravcenko, S. M., Suhoniak, I. I., Marchuk, H. V., Gryshkun, Ye. O., & Venhlovska, Yu. M. (2023). UML modeling of the design process of a puzzle game. *Scientific Notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences*, *1*(3), 157–162. https://doi.org/10.32782/2663-5941/2023.3.1/25

[7] Luhova, T., & Lys, D. (2019). UML models as the basis of designing and balancing computer games scenarios. *ΛΟΓΟΣ. The Art Of Scientific Mind*, (7), 33–37. https://doi.org/10.36074/2617-7064.07.00.007

[8] Samchuk, L., & Povstiana, Y. (2024). UML diagrams of the management system of maintenance stations. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, *14*(4), 141–145. https://doi.org/10.35784/iapgos.6320

Ph.D. Lyudmila Samchuk
e-mail: samchuk204@gmail.com

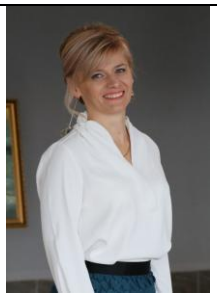
Candidate of Technical Sciences, associate professor, Faculty of Transport and Mechanical Engineering, Department of Applied Mechanics and Mechatronics, Lutsk National Technical University.

https://orcid.org/0000-0003-2516-045X

Ph.D. Yuliia Povstiana
e-mail: yuliapovstyana@ukr.net

Candidate of Technical Sciences, associate professor, Faculty of Computer and Information Technologies, Department of Software Engineering, Lutsk National Technical University.

https://orcid.org/0000-0001-5426-4157



Tymoshchuk Yaroslav
e-mail: jariksuperlol6@gmail.com

Bachelor's student at Lutsk National Technical University.

https://orcid.org/0009-0001-1129-5612