Summative Assessment 1: Development Team Project

Software Engineering Project Management July 2024

TEAM 2

Andrius Busillas, Thomas Gray, Seyun Hur, Zukiswa Tuso

Word count: 1000

Project Report

Executive Summary

This report presents the design and specifications of Synputer, an advanced desktop computer developed by Synful Computing, to address the evolving needs of household and business consumers in the rapidly changing computer industry. The Synputer development was in response to specific requirements of English Digital Computers (EDC), which emphasized the necessity for an industry-standard operating system, enhanced memory capacity, external networking capabilities, and compatibility with contemporary graphical user interfaces (GUIs). Synputer is designed to be an innovative product, effectively integrating advanced features that cater to both personal and professional computing requirements. This report details the comprehensive design, development, and testing processes undertaken to ensure that the Synputer meets high-performance standards. Additionally, the report examines the selection of materials and the pricing strategy that positions Synputer as a premium product in the market. Through innovative design and strategic planning, Synputer was poised to establish a new standard in the computing industry.

This report describes the design and specs of the Synputer, a desktop computer created by Synful Computing in response to English Digital Computers (EDC) needs. The product aims to meet the changing needs of household and business customers in the computer sector. The Synputer's initial specifications comprised a 68008 CPU, minimal RAM, ROM configurations, and restricted networking choices. However, a full assessment was carried out when EDC submitted a formal protest about the insufficiency of these standards. The EDC's requirements highlighted the need for an industry-standard operating system, increased memory capacity, external networking possibilities, and compatibility with current graphical user interfaces (GUIs).

1. Design Approach/Methodology

Selected Methodology: Agile Development

Selecting an appropriate methodology is crucial in the development of software and hardware. Three methodologies Agile, Waterfall, and Spiral - each offering unique approaches, were considered and evaluated for their potential application.

- **Agile:** This method is well suited for projects where requirements are expected to change or are not fully defined at the outset. It allows for quick iterations and ongoing feedback, potentially leading to a product that better meets the user's needs. Agile is particularly beneficial in fast-paced environments, where rapid market entry is essential.
- **Waterfall:** This approach is ideal for projects with clearly defined requirements and scope. It works well for small-scale, low-risk projects and offers a sequential process with clear milestones. However, their lack of flexibility may be problematic in dynamic settings.

- **Spiral:** This model is beneficial for large and complex projects in which risk management is crucial. It combines Agile's iterative nature with Waterfall's structured approach, allowing continuous improvement and risk evaluation. The Spiral method is especially effective for high-stakes projects, where failure can be costly.

The Agile methodology was chosen for the Synputer project because of its adaptability and iterative nature, which are essential for accommodating the dynamic requirements of the system. Agile allows for the constant integration of user feedback and iterative changes, thanks to EDC input and the requirement to make quick modifications based on user expectations. This approach is particularly advantageous in hardware and software development, where requirements may change based on technological advancements and market demands. Agile encourages cooperation among cross-functional teams, which is vital for effectively integrating hardware and software components. Furthermore, short development cycles (sprints) allow the team to deliver prototypes quickly, enabling early user testing and validation, which aligns with the project's timeline constraints.

Table 1. Comparison and justification for methodologies.

Criteria	Agile	Waterfall	Spiral
Definition	An iterative and incremental approach that emphasizes flexibility and customer collaboration (Foley et al., 2024; GeeksforGeeks, 2024).	A linear and sequential approach where each phase must be completed before the next begins (Yu Stepanov, 2021).	A risk-driven model that combines iterative development with the systematic aspects of the waterfall model (Biswas et al., 2024; Malikov et al., 2023).
Flexibility	Highly flexible; changes can be made at any stage based on feedback (Foley et al., 2024).	Rigid; changes are difficult to implement once a phase is completed (Biswas et al., 2024; Rahman et al., 2024).	Moderate flexibility; allows for changes at the end of each iteration but is more structured than Agile (Biswas et al.,

			2024; Malikov et al., 2023).
Customer Involvement	Continuous customer involvement throughout the project lifecycle (Diansyah et al., 2023).	Limited customer involvement; feedback is typically gathered at the end of the project (Saravanos & Curinga, 2023).	Customer involvement is essential at the end of each iteration to assess risks and gather feedback (Yu Stepanov, 2021).
Risk Management	Risks are managed through regular iterations and feedback loops (Guerrero-Ulloa et al., 2023).	Risks are identified initially but only revisited at the end, which can lead to unforeseen issues (Rahman et al., 2024).	Emphasises risk assessment and management at each iteration, allowing for proactive adjustments (Biswas et al., 2024).
Project Size Suitability	Best suited for small to medium-sized projects with evolving requirements ((Diansyah et al., 2023; Leong et al., 2023).	Ideal for small projects with well-defined requirements and scope (Diansyah et al., 2023).	Suitable for large, complex projects where risk management is critical (Yu Stepanov, 2021).
Documentation	Minimal documentation; focuses on working software and collaboration (Mishra & Alzoubi, 2023).	Extensive documentation; each phase requires detailed documentation before moving on (Mishra & Alzoubi, 2023; (Yu Stepanov, 2021).	Balanced documentation; documentation is created as needed, focusing on risk management (GeeksforGeeks, 2024).
Time to Market	Faster time to market due to iterative releases and continuous feedback (Pressman, 2015).	Longer time to market as all phases must be completed sequentially (Saravanos & Curinga, 2023).	Moderate time to market; iterations allow for partial releases but require thorough risk assessments (GeeksforGeeks, 2024).
Team Collaboration	High collaboration among team members; encourages cross-functional teams (GeeksforGeeks, 2024).	Limited collaboration; team members work in silos based on their phase responsibilities (Saravanos & Curinga, 2023).	Encourages collaboration, especially during risk assessment phases, but can still be siloed in execution (Yu Stepanov, 2021).

2. Requirements Gathering

Based on the discussion transcript provided in the case study, the requirements table presents a comprehensive outline of the necessary specifications and features for the development of the system. It highlights critical requirements such as battery life, bundled applications, operating system capabilities, expandability, documentation, and performance specifications. Each requirement meets specific user needs and expectations, guaranteeing that the final product is functional, user-friendly, and competitive in the marketplace. A clear delineation of these requirements in the table facilitates the efficient planning, development, and assessment of the project, enabling the identification of potential gaps and assumptions that could affect user satisfaction and system performance.

Table 2. Requirements list.

Category	Requirement	Description	Critique	Priority Level	Priority Justification
Hardware Requirements	**R001** Industry Standard Operating System: Support for an industry-standard OS.	The system must be compatible with widely used operating systems to ensure interoperability with existing software and hardware, facilitating user adoption.	**Assumption:** It assumes that users will prefer an industry-standard OS without considering niche markets that may require specialized systems. **Gap:** No specific OS options are mentioned, which could lead to compatibility issues.	High	Essential for compatibility with existing software and systems, ensuring user acceptance and marketability.
	R002 External Keyboard/Connecto: It is capable of connecting an external keyboard.	The design should include ports or connectors that allow users to attach an external keyboard, enhancing usability and flexibility for various user preferences.	**Assumption:** It assumes all users will want an external keyboard. **Gap:** No mention of the type of connectors (USB, PS/2) or compatibility with existing peripherals, which could limit user options.	High	Critical for user experience and functionality; many users expect an external keyboard for productivity.
	R003 Memory: Minimum of 512KB of RAM.	The system should have at least 512KB of RAM to support multitasking and run modern applications efficiently, ensuring a smooth user experience.	**Assumption:** It assumes that 512KB is sufficient for all applications. **Gap:** As software demands increase, this may quickly become inadequate, and no scalability options are provided.	High	Necessary for running modern applications and multitasking; aligns with user expectations for performance.
	R004 Removable Drive: At least one industry-standard removable drive.	Including a removable drive is essential for data transfer, backup, and software installation,	**Assumption:** It assumes that users will require a removable drive. **Gap:** No specification on the type of removable drive, which could affect	High	Important for data transfer and storage flexibility; users require the ability to easily manage data.

		adhering to industry standards.	compatibility with modern data transfer methods.		
	R005 Small Computer System Interface (SCSI) Expansion Capability: Support for SCSI for future expansion options.	The system should have SCSI ports to allow for the connection of additional storage devices and peripherals, providing scalability and flexibility for future upgrades.	**Assumption:** It assumes that SCSI will be a preferred standard. **Gap:** SCSI is less common in modern systems; alternatives like SATA or USB might be more relevant for users.	Medium	Provides future-proofing and scalability; while not immediately necessary, it enhances the system's longevity.
	R006 CPU: Minimum of a 68000 CPU, with upgrade options.	The system must be powered by at least a 68000 CPU to ensure adequate processing power, with the potential for future CPU upgrades to enhance performance.	**Assumption:** It assumes that the 68000 CPU will meet future performance needs. **Gap:** No clear upgrade path or options for newer CPUs are provided, which could limit the system's longevity.	Medium	Important for performance; an upgradable CPU allows users to extend the life of their investment.
	R007 Serial Ports: At least two serial ports supporting RS 422/485 standards.	The inclusion of two serial ports that comply with RS 422/485 standards is necessary for connecting various peripherals, such as printers and modems, ensuring compatibility.	**Assumption:** It assumes that users will need serial ports. **Gap:** Many modern devices use USB or wireless connections; this requirement may be outdated and not align with current technology trends.	Medium	Relevant for business applications that require multiple connections; enhances the system's utility in professional environments.
	R008 GUI Support: The Board is ready to support a GUI system and mouse if required.	The hardware should be designed to accommodate a graphical user interface (GUI) and support mouse input, aligning with user	**Assumption:** It assumes that all users will want a GUI. **Gap:** No details on the specific GUI capabilities or requirements, it could lead to confusion about what is supported.	Medium	Increasingly expected by users; a GUI enhances usability and aligns with modern computing trends.

		expectations for modern computing experiences.			
	R009 Battery Life: Minimum of 2 hours of continuous operation under typical usage conditions, including moderate application use, screen brightness settings, and wireless connectivity.	The system should provide at least 2 hours of battery time to ensure portability and usability.	**Assumption:** The requirement presumes that users will engage in typical office tasks, which may not reflect the full range of potential use cases for the device. **Gap:** The lack of detail regarding high-performance applications and other factors affecting battery life could result in unmet user expectations and a negative experience if the system does not perform as anticipated under varied conditions.	High	Essential for user satisfaction, especially for portable systems; users expect at least 2 hours of battery life for practical use.
Software Requirements	**R010** Bundled Applications: Complete office suite including word processor, spreadsheet, database, and graphics tools.	The system should come pre-installed with essential productivity applications to provide users with a comprehensive software environment for various tasks.	**Assumption:** It assumes that users will need a complete office suite. **Gap:** Any mentions of the specific applications or their compatibility with existing file formats, could limit usability.	High	Critical for immediate usability; providing a complete office suite enhances the system's value and appeal to users.
	R011 Multi-tasking OS: Development of a multi-tasking, Unix-like operating system.	The operating system should support multi-tasking capabilities, allowing users to run multiple applications simultaneously, enhancing productivity and user experience.	**Assumption:** It assumes that a Unix-like OS will be suitable for all users. **Gap:** No consideration for users who may prefer other OS types (e.g., Windows, macOS) or the learning curve associated with Unix systems.	High	Important for modern computing needs; it allows users to run multiple applications simultaneously, improving productivity.

	R012 Programming Language: Inclusion of a structured, modular superset of BASIC (HyperBasic).	The system should support HyperBasic, a more advanced version of BASIC, enabling users to develop applications easily and efficiently, catering to beginners and experienced programmers.	**Assumption:** It assumes that users will want to program in HyperBasic. **Gap:** No mention of support for other programming languages, which could limit the appeal to a broader audience.	Medium	While important for developers, it is less critical for end-users; a robust programming language can enhance the system's capabilities.
Expansion and Compatibility Requirements	**R013** Expandable System: Allow for the addition of native and third-party expansion packs.	The system architecture should be designed to support both native and third-party expansion packs, enabling users to enhance their systems with additional features and capabilities.	**Assumption:** It assumes that users will want to expand their systems. **Gap:** No details on the types of expansion packs or how they will be integrated, which could lead to confusion about compatibility.	Medium	Provides future-proofing and adaptability; users appreciate the ability to upgrade and expand their systems as technology evolves.
	R014 Documentation and Support: Comprehensive documentation covering basic operation and HyperBasic.	Detailed user manuals and technical documentation should be provided to assist users in understanding system operations, programming in HyperBasic, and troubleshooting issues.	**Assumption:** It assumes that users will read and utilize the documentation. **Gap:** No mention of support channels (e.g., forums, customer service) for ongoing assistance, which could impact user satisfaction.	Medium	Necessary for user onboarding and troubleshooting; good documentation and support can significantly enhance user experience.
Compliance and Performance Requirements	**R015** Performance Specifications: Meet performance needs of >80% of user requirements.	The system must be designed to fulfill the majority of user requirements (over 80%) as identified in market research, ensuring it	**Assumption:** It assumes that market research accurately reflects user needs. **Gap:** No specific metrics or criteria for measuring performance against user requirements, it could lead to subjective evaluations.	High	Directly impacts user experience and satisfaction; meeting performance expectations is crucial for the system's success in the market.

		meets the expectations of the target audience.			
	R016 Response to Complaints: Official response detailing how requirements will be addressed.	An official communication must be prepared to address any complaints or concerns raised by stakeholders, outlining how the system will be modified to meet their requirements and the timeline for implementation.	**Assumption:** It assumes that all complaints can be addressed satisfactorily. **Gap:** No clear plan for prioritizing and addressing complaints, it could lead to unresolved issues and dissatisfaction among users.	High	Essential for maintaining customer relationships and trust; addressing complaints promptly can prevent potential legal issues and enhance reputation.

Table 3. Missing Requirements list.

Category	Requirement	Description	Critique	Priority Level	Priority Justification
User Interface	**MR001** Detailed GUI Specifications	Define specific design elements, layout, and user interaction guidelines.	**Assumption:** It assumes that the development team understands user needs without explicit guidelines. **Gap:** in clarity.	High	A well-defined GUI is critical for user satisfaction and usability, impacting overall product success.
Performance	**MR002** Specific Performance Benchmarks	Establish benchmarks for processing speed, memory usage, and response times.	**Assumption:** It assumes that performance will meet user expectations without defined metrics. **Gap:** gap in measurable.	High	Clear performance benchmarks are essential to ensure the system meets user demands and industry standards.
Security	**MR003** Data Protection Features	Outline required security measures, including encryption and authentication.	**Assumption:** It assumes that basic security will be sufficient. **Gap:** the gap in addressing potential vulnerabilities.	High	Security is paramount, especially for business applications; a lack of measures could lead to data breaches.
Connectivity	**RM004** Connectivity Options	Specify required ports and wireless capabilities for peripheral compatibility.	**Assumption:** It assumes standard connectivity will suffice. **Gap:** the gap in addressing specific user needs for connectivity.	Medium	Connectivity is important for user experience; however, it may not be as critical as performance or security.
Customization	**MR005** User Customization Options	Allow users to adjust settings for performance, display, and interface.	**Assumption:** It assumes users will be satisfied with default settings. **Gap: the** gap in personalization needs.	Medium	Customization enhances user experience and satisfaction, but may not be essential for all users.
Environmental Impact	**MR006** Energy Efficiency Requirements	Define energy consumption limits and eco-friendly materials.	**Assumption:** It assumes users do not prioritize sustainability. **Gap: the** gap in addressing growing environmental concerns.	Low	While important, it may not be a top priority compared to functionality and performance in the short term

Documentation	**MR007** Comprehensive User Documentation	Provide detailed user manuals and technical documentation.	**Assumption:** It assumes users will figure out the system without guidance. **Gap:** the gap in support for less tech-savvy users.	Medium	Comprehensive documentation is important for user support and training, enhancing overall usability.
Testing	**MR008** Comprehensive Testing Strategy	Outline a strategy for unit, integration, and user acceptance testing.	**Assumption:** It assumes that existing testing processes will suffice. **Gap:** the gap in ensuring thorough validation of the system.	High	A robust testing strategy is crucial to identify and resolve issues before launch, ensuring product quality.

Missing Requirements

- **High-Performance Application Support:** The requirement table does not address the need for the system to handle high-performance applications, such as gaming or video editing. This gap may alienate a segment of users who require robust performance for demanding tasks, potentially leading to dissatisfaction if the system cannot meet their needs.
- **Connectivity Options:** There is no mention of specific connectivity options (e.g., USB ports, HDMI output, or wireless capabilities) that are crucial for modern devices. This lack of detail could hinder the system's compatibility with peripherals and external displays, limiting its functionality.
- **User Customization:** The absence of requirements for user customization options (e.g., adjustable settings for performance, display, and user interface) may restrict user experience and personalization, which are increasingly important in consumer technology.
- **Security Features:** The requirement table does not include security features such as encryption, user authentication, or data protection measures. This oversight could expose users to risks, especially in business environments where data security is paramount.
- **Environmental Considerations:** There is no mention of energy efficiency or eco-friendly materials in the requirements. As sustainability becomes a growing concern for consumers, this gap could affect the product's marketability and acceptance.
- **Detailed Specifications for the GUI and User Interface:** The requirement table lacks specific guidelines for the graphical user interface (GUI) and user experience design. These specifications are necessary for

the development team to create an intuitive and visually appealing interface, which is critical for user satisfaction and adoption.

- **Specific Performance Benchmarks for the System:** There are no defined performance benchmarks to evaluate the system's capabilities, such as processing speed, memory usage, or response times. This absence makes it difficult to assess whether the system meets user expectations and industry standards, potentially leading to performance-related issues post-launch.
- **Security Features for Data Protection:** While security features were previously mentioned, the requirement table does not specify the types of security measures needed. Detailed requirements for data encryption, secure boot processes, and user authentication protocols are essential to protect sensitive information and build user trust, especially in business applications.

Assumptions

- The project will have access to necessary hardware components within the budget.
- The team has the required expertise in both hardware and software development.
- User acceptance testing will be conducted with a representative sample of end-users.
- All required resources are full-time available with no unforseen absence or delays.
- The project will be completed on time and within budget.

3. Key Requirements and Gherkin Specifications

Key Requirements

- Industry-standard operating system
- At least 512KB of RAM
- At least a 68000 CPU – preferably upgradable
- High-resolution graphics (512 colors, 1024 x 768 resolution)
- Ability to run games from older machines

Gherkin Specifications

These Gherkin specifications demonstrate several parts of the Synputer case design, ensuring they fulfill the user needs and expectations indicated in the case study.

```
Scenario: Operating System
    Given the system is powered on
    When the user checks the operating system
    Then it should display "Industry Standard OS"
```

```
Scenario: RAM Capacity
    Given the system is assembled
    When the user checks the RAM
    Then it should be at least 512KB
```

```
Scenario: CPU Upgradeability
    Given the system is powered on
    When the user checks the CPU model
    Then it should be a 68000 CPU and should be upgradeable
```

```gherkin
Scenario: Successful login with valid credentials
    Given I am on the login page
    When I enter a valid username and password
    And I click the "Login" button
    Then I should be redirected to my dashboard
    And I should see a welcome message
```

```gherkin
Scenario: Port accessibility
    Given the case is designed for user interaction
    When the user looks for ports
    Then they should find ports for a mouse and two joysticks
    easily accessible
```

```gherkin
Scenario: Cooling system
    Given the internal components generate heat
    When the case is designed
    Then it should include ventilation to prevent overheating
```

```gherkin
Scenario: Aesthetic appeal
    Given the market research indicates a preference for
    modern
    designs
    When the case is completed
    Then it should have a visually appealing design suitable
    for business and home users
```

```gherkin
Scenario: Documentation inclusion
    Given the case is finalised
    When the user receives their Synputer
    Then it should include a printed manual covering basic
   operation and an introduction to Hyperbasic
```

```
Scenario: External Keyboard Connection
    Given the system is on
    And an external keyboard is connected
    When the user types
    Then the system responds
    And the keyboard layout is configurable
```

```
Scenario: Storage Compatibility
    Given the system is on
    When a removable drive is inserted
    Then it is recognized
    And allows read/write access
```

As part of Behavior-Driven Development (BDD), Gherkin scenarios are used to specify system behaviour in a clear and structured way (Smart, 2014).

4. Costed Project Plan

4.1. Budget Overview

- Total Budget: £500,000
- Cost Price Target per Machine: £250
- Total Machines: 2,000
- Total Revenue from EDC: £500,000

4.2. Cost Breakdown

Text

Item	Cost (£)
Hardware Components	200,000
Software Development	150,000
Testing and Quality Assurance	50,000
Project Management	50,000

Contingency Reserve	50,000
Total	**500,000**

Timeline

Text

Phase	Duration (Weeks)	Start Date	End Date
Design	8	Week 1	Week 8
Hardware Development	12	Week 9	Week 20
Software Development	16	Week 9	Week 24
Testing	8	Week 21	Week 28
User Acceptance Testing	4	Week 29	Week 32
Total Time	**32 weeks**		

Key Milestones

- Prototypes Ready: Week 20
- Models Available for Sale: January 1984
- Expected Income Generation: February 1984

5. Testing Justification

- **Unit Testing:** 2 weeks for hardware and software components to ensure individual parts function correctly.
- **Integration Testing:** 2 weeks to verify that hardware and software components work together seamlessly.
- **System Testing:** 2 weeks to validate the complete system against requirements.
- **User Acceptance Testing:** 4 weeks to gather feedback from end-users and make necessary adjustments.

Conclusion

The information drawn from the transcript of the conversation between EDC and Synful Computing provides a solid foundation for the development of deliverables in line with user and organisational needs. By utilising Agile methodology, the most amount of flexibility is allowed for the development to go ahead. Given the complexity of this kind of project and the needs to ensure that strict timelines and financial constraints are adhered to, utilising a solid developmental methodology is essential to ensuring that success can be achieved.

By utlising a number of different tools to aid with shaping the developmental documents for the project, such as Gantt Charts, Gherkin Statements and assumptions, a fully fleshed out plan can be formed that allows all members of a developmental team to feel confident in what they are doing, the overall aims of the project and the timeline for this to be completed.

Reference list

Biswas, T. et al. (2024) ‘ScrumSpiral: An improved hybrid software development model’, International Journal of Information Technology and Computer Science, 16(2), pp. 57–65. doi:10.5815/ijitcs.2024.02.05.

Cohn, M. (2004) User Stories Applied: For Agile Software Development. Boston: Addison-Wesley Professional.

Diansyah, A.F., Rahman, R.M., Handayani, R., Nur Cahyo, D.D. & Utami. E. (2023) ‘Comparative analysis of software development lifecycle methods in software development: A systematic literature review’, International Journal of Advances in Data and Information Systems, 4(2), pp. 97–106. doi:10.25008/ijadis.v4i2.1295.

Foley, J., McEwan, C. & Foster, T. (2024) What is Agile Software Development? Agile Alliance. Available at: https://www.agilealliance.org/agile101/ [Accessed: 25 August 2024].

GeeksforGeeks (2024) Comparison between agile model and other models in software engineering, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/software-engineering-comparison-between-agile-model-and-other-models/ [Accessed: 28 August 2024].

Guerrero-Ulloa, G., Rodríguez-Domínguez, C. & Hornos, M.J. (2023) ‘Agile methodologies applied to the development of internet of things (iot)-based systems: A Review’, Sensors, 23(2), p. 790. doi:10.3390/s23020790.

Ismail, R., Othmani, I.M., Said, I. & Dan, W. (2023) ‘The effective project management approach towards minimising environmental impacts in the construction industry’, Journal of Technology and Operations Management, 18(2), pp. 60–72. doi:10.32890/jtom2023.18.2.5.

Leong, J., May Yee, K., Baitsegi, O., Palanisamy, L. & Ramasamy, R. (2023) ‘Hybrid project management between Traditional Software Development Lifecycle and agile based product development for future sustainability’, Sustainability, 15(2), p. 1121. doi:10.3390/su15021121.

Malikov, M., Aloraini, F.A., Kavak, H., Kennedy, W.G. & Crooks, A. (2023) ‘Developing A Large-Scale Agent-Based Model Using The Spiral Software Development Process‘, 2023 Annual Modeling and Simulation Conference (ANNSIM), Hamilton, ON, Canada, pp. 282-293.

Mishra, A. & Alzoubi, Y.I. (2023) ‘Structured software development versus agile software development: A comparative analysis’, International Journal of System Assurance Engineering and Management, 14(4), pp. 1504–1522. doi:10.1007/s13198-023-01958-5.

Pressman, R.S. (2015) Software engineering: A practitioner’s approach. New York, NY: McGraw-Hill Education.

Rahman, A., Cysneiros, L.M. and Berry, D.M. (2024) ‘An empirical study of the impact of waterfall and agile methods on numbers of requirements-related defects’, Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing [Preprint]. doi:10.1145/3605098.3635901.

Saravanos, A. & Curinga, M.X. (2023) ‘Simulating the software development lifecycle: The waterfall model’, Applied System Innovation, 6(6), p. 108. doi:10.3390/asi6060108.

Smart, J.F. (2014) BDD in Action: Behaviour-Driven Development for the Whole Software Lifecycle. Shelter Island: Manning Publications.

Sommerville, I. (2016) Software engineering. Boston: Pearson.

Yu Stepanov, D. (2021) 'Using waterfall, iterative and spiral models in ERP-system implementation projects under uncertainty', Journal of Physics: Conference Series, 2142(1), p. 012016. doi:10.1088/1742-6596/2142/1/012016.

(Foley et al., 2024)

(Sommerville, 2016)

(Pressman, 2015)

(GeeksforGeeks, 2024)

(Mishra & Alzoubi, 2023)

(Diansyah et al., 2023)

(Yu Stepanov, 2021)

(Leong et al., 2023)

(Biswas et al., 2024)

(Ismail et al., 2023)

(Guerrero-Ulloa et al., 2023)

(Saravanos & Curinga, 2023)

(Rahman et al., 2024)

(Malikov et al., 2023)