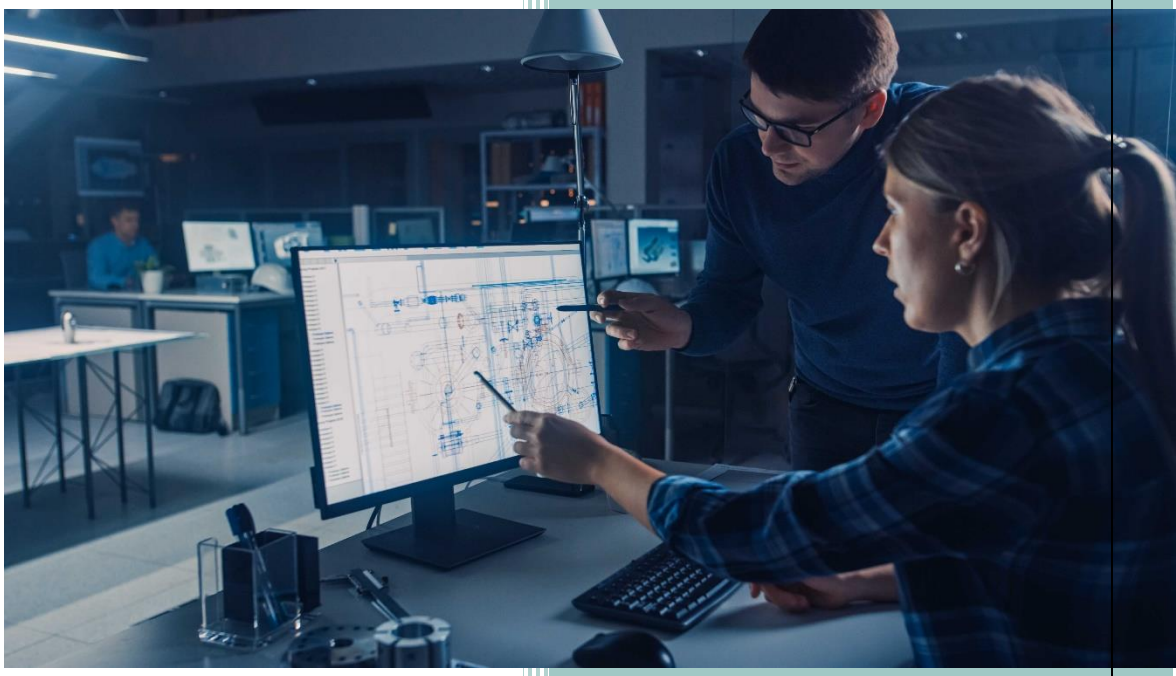
2024

[OUbs033214] Software Engineering



Doosan PAGOOAH

Learner ID: 202306292

11/4/2024

Name: Doosan Pagooah | Learner **ID:** 202306292

Table of Contents

Question 1 (a)

Identify three functional and three non-functional requirements for the above computer monitoring system.

1. **Functional Requirements:**

   a) **Monitoring of Animal Movement:** The system must track the movements of all animals in the park by receiving signals from radio transmitters affixed to the animals and determining their whereabouts within 0.1 seconds.

   b) **Perimeter Surveillance and Alert System**: The system must identify any damage to the electric fences around the animal cages and activate an alert as required.

   c) **Veterinary Notification**: Technology must automatically warn the veterinary team (via a beep or notification) whenever any animal sustains an injury.

2. **Non-Functional Requirements:**

   a) **Performance**: The system must calculate the location of each animal within 0.1 seconds to provide prompt monitoring and reaction.

   b) **Usability**: The system's interface must be menu-driven to provide user-friendliness for personnel in the control centre.

   c) **Reliability**: The system must undergo comprehensive testing prior to delivery to guarantee safety, since it monitors hazardous animals. It must operate flawlessly, particularly during emergencies such as animal escapes or fence breaches.

Question 1 (b)

Given the above scenario, recommend a software development life-cycle that you would adopt for the above system. State and explain the steps involved in the proposed life-cycle and justify your answer by explaining why you have chosen this software process model.

The incremental model is the best appropriate software development life cycle for this project.

What is the rationale behind the incremental model?

The system encompasses several intricate and essential tasks, such as tracking animal movements, alarm systems, and veterinarian notifications. Constructing the complete system simultaneously, as per the Waterfall Model, may result in risks and challenges in fulfilling requirements. According to *Sommerville (2016)*, the incremental Model allows for the gradual development of different parts of the system, helping to reduce the risks associated with large-scale system failures during development.

Iterative Testing and Feedback: The Incremental Model facilitates testing and feedback collection for each increment, which is essential for a safety-critical system where ongoing enhancements are required to maintain the efficacy of the monitoring system for hazardous animals. As *Boehm (1988)* highlights, incremental approaches enable continuous refinement based on user feedback and testing, which is vital for a system that must ensure public safety.

Flexibility: It permits modifications throughout the development process. Improvements may be implemented based on feedback from preliminary increments prior to the complete system deployment. Such flexibility is particularly important in this case, where improvements based on early increments - such as user interface adjustments or veterinary alerts - can be made before full

deployment *(Pfleeger & Atlee, 2010)*. This continuous refinement leads to a more reliable final system.

Time Constraints and Gradual Delivery: The park is set to open in December, necessitating the system's readiness by that time. Incremental system development facilitates the prompt delivery of essential functionality, such as fundamental animal monitoring, while enabling the ongoing enhancement of other components, such the interface, alerts, and veterinary notifications, in later increments. *Pressman (2019)* notes that the incremental delivery approach ensures that essential features are available early in the development life cycle, making it well-suited to projects with tight deadlines.

Question 1 (c)

Identify four difficulties you might encounter during requirements elicitation and analysis for the above project.

During the requirements elicitation and analysis for the computer monitoring system designed for the park housing hazardous animals, many challenges may emerge.

1. Incomplete or Ambiguous Requirements

Stakeholders may lack a precise understanding of their system needs. For instance, they may provide ambiguous definitions of the operational parameters for the monitoring or alarm systems. This may result in incomplete requirements necessitating further clarification, hence causing delays in the development process. According to *Sommerville (2016)*, incomplete requirements are one of the most common sources of project failure, as they often require additional clarification later in the process, causing delays in the development phase.

2. Varied Stakeholder Requirements

The project encompasses several stakeholders, including park administrators, wildlife wardens, veterinarians, safety authorities, and maybe visitors. Each group may possess distinct demands and objectives, complicating the reconciliation of all elements into a unified system design. *Robertson and Robertson (2012)* emphasize that conflicting stakeholder requirements can lead to significant challenges during the design phase, as it becomes difficult to create a system that satisfies everyone.

3. Insufficient Technical Knowledge

Among Stakeholders: Numerous stakeholders may lack familiarity with the system's technical components (e.g., radio signal transmission, real-time data processing, or software interface design). Consequently, they may find it challenging to articulate their requirements in practical terms for the development team, resulting in communication gaps and misunderstandings. *Davis et al. (2006)* points out that requirements elicitation is often hampered by communication gaps, especially when stakeholders struggle to express their needs in technical terms.

4. Intricate Safety Specifications:

Given that the system entails monitoring hazardous animals, the safety specifications are important and must be very accurate. Establishing precise specifications for alarms, fence surveillance, and emergency procedures may be challenging due to the little margin for error. A little misinterpretation of safety rules may result in substantial system failures, jeopardising safety. According to *Leveson (2011)*, safety-critical systems require exhaustive analysis and specification because even minor misinterpretations can have catastrophic consequences.

5. Evolving Requirements

As the park approaches completion or throughout system development, stakeholders may recognise the need for modifications to particular system components. For instance, they may choose to include more species or establish new tourist zones, therefore impacting the monitoring system. The changing needs might complicate the finalisation of the system specs. Pohl (2010) suggests that requirements are rarely static, and as stakeholders learn more about the system during its development, they often wish to incorporate new features or make adjustments.

6. Feasibility Constraints

Reconciling the needs with the project's budget (Rs 6 million) and temporal limitations (deadline in December) may prove to be arduous. Stakeholders may possess elevated expectations regarding the system's functionalities (such as intricate animal monitoring or sophisticated alerts); yet, some objectives may not be attainable inside the specified budget and timeframe. Identifying which criteria to prioritise within these limitations might be challenging. As *Boehm and Turner (2004)* note, reconciling high stakeholder expectations with budgetary and time constraints is a persistent issue in software development projects.

These problems need the maintenance of transparent and continuous communication with stakeholders, the engagement in iterative requirements gathering, and the use of tools like as prototypes and user stories to improve and elucidate the system's requirements.

Question 1 (d)

The Software Requirements Specification (SRS) is the official document that states in precise and explicit language functions and capabilities that the system must provide, as well as states any required constraints by which the system must abide. As the project manager you have been requested to list and explain the desired characteristics of the SRS document.

As the project manager, the Software Requirements Specification (SRS) document must exhibit certain attributes to guarantee clarity, accuracy, and comprehensiveness in delineating the system's capability and limitations. The following are the necessary attributes of the SRS document, along with elucidations:

1. Accuracy

The SRS must precisely represent the genuine needs and requirements of the stakeholders. All specified requirements must be essential, and no erroneous or obsolete information should be included. The accuracy of the SRS guarantees that the system will fulfil stakeholder expectations and provide the desired functionality. As *Sommerville (2016)* points out, accuracy in an SRS is vital to ensure that the final system aligns with stakeholder expectations and delivers the required functionality without unnecessary features.

2. Comprehensiveness

The SRS must include all required information on the system, including both functional and non-functional needs. This indicates that all system behaviours, constraints, and assumptions are well documented, ensuring that no critical information is omitted. According to *Pressman (2019),* an exhaustive SRS helps the development team fully understand the project scope, leading to better project planning and implementation.

3. Clarity

Every provision in the SRS must be articulated clearly and unequivocally. There must be no ambiguity in the interpretation of a single requirement. A statement such as "The system should be user-friendly" is unclear due to the subjective nature of "user-friendly." Instead, specific, quantifiable criteria should be used (e.g., "The system shall enable a trained user to complete task A within 120 minutes"). According to *Wiegers and Beatty (2013),* clear requirements help ensure that stakeholders, developers, and testers all have a consistent understanding of the system's goals.

4. Uniformity

The SRS must guarantee the absence of competing obligations. If one section of the document mandates mobile device accessibility while another restricts access to desktop computers, it results in inconsistency. Consistency entails that all criteria, vocabulary, and assumptions within the text must be congruent. As *Pohl (2010)* highlights, consistency within an SRS prevents confusion and ensures that all requirements align harmoniously with each other.

5. Confirmability

Each requirement in the SRS must be articulated in a manner that facilitates testing or verification. A verifiable requirement establishes explicit criteria for assessing whether the system fulfils that need. For instance, "The system shall determine the position of each animal within 0.1 seconds" is verifiable since it can be quantified and assessed. *Boehm (1988)* emphasizes that verifiable requirements are essential to ensure the system's functionality can be tested effectively.

6. Alterability

The SRS must be organised to facilitate modifications without necessitating significant redesign. This is significant since needs may change over the development process. An adaptable SRS must include distinct parts and subsections, facilitating the identification and modification of particular criteria. *Sommerville (2016)* suggests that an adaptable SRS with well-organized sections

facilitates quick updates and ensures that modifications are contained to specific parts of the document.

7. Traceability

The SRS must provide seamless traceability of requirements, enabling the tracking of each requirement from its source (e.g., a stakeholder needs) through its design and implementation phases, culminating in the testing stage. As explained by *Pohl (2010),* traceability supports the verification of requirements by linking them to their corresponding test cases.

8. Prioritisation

The SRS must specify the priority of each demand, particularly in the context of time and money restrictions. High-priority needs are essential for the system's effective functionality, but low-priority requirements may be postponed or eliminated if needed. This facilitates the management of stakeholder expectations and directs the emphasis of growth. *Wiegers and Beatty (2013)* argue that prioritization helps manage stakeholder expectations and focuses the development team on the most critical features.

Comprehensibility

The SRS must be comprehensible and accessible to all stakeholders, including non-technical individuals such as park management. It should eschew excessive technical terminology and jargon wherever feasible and should include explanations or illustrations as needed to facilitate comprehension. *Pressman (2019)* stresses that clear communication in requirements is critical for successful stakeholder involvement.

10. Viability

The SRS must only include needs that are technically and practically achievable given the project's limitations (budget, timeline, and available resources). Requirements that are impractical or too expensive to implement should be excluded. This mitigates unreasonable expectations and scope expansion. According to *Boehm (1988)*, managing feasibility ensures that stakeholders do not have unrealistic expectations, allowing the development team to focus on achievable goals.

11. Brevity

The SRS must be thorough but succinct and direct. Superfluous information must be eliminated, and specifications should be articulated clearly and concisely. Conciseness minimises ambiguity and facilitates the evaluation and maintenance of the material. *Sommerville (2016)* suggests that brevity in an SRS helps minimize misunderstandings and makes the document easier to manage and modify.

12. Maintainability

The SRS must be structured and articulated to facilitate updates during the project's advancement. Maintainability guarantees that the SRS continues to be an active document, accurately representing any changes in scope, requirements, or system limitations over time. Well-organised papers with version control enable this. According to *Pressman (2019)*, maintainable documentation helps ensure that the SRS continues to accurately reflect the project's objectives.

13. Scalability

The SRS must consider possible future growth or scaling of the system. For instance, if the park intends to include more animals in the future, the system must be constructed with scalability as a priority. *Robertson and Robertson (2012)* emphasize that designing with scalability in mind reduces the need for extensive system redesigns later on.

Conclusion:

An SRS document serves as the cornerstone for effective software development and must be unequivocal, accurate, and comprehensive. An adequately formulated SRS not only directs developers but also guarantees that all stakeholders possess a unified comprehension of the system's functionalities, hence reducing the chance of project failure stemming from misinterpretations or overlooked requirements. By ensuring the SRS conforms to these specified qualities, the project manager can guarantee the system is constructed effectively, within budget, and satisfies the park’s requirements for safety and usefulness.

Question 2

Create a use case diagram for the system required by the book shop, as described above. Also, describe the meaning of the relationships in your use case diagram.

Relationship Type:

1. **Association**: This represents a direct interaction between an actor and a use case. As depicted in Figure 2.1, the Customer is associated with Purchase Book and Order Book, indicating that they perform these actions.

2. **Include**: This relationship indicates that the base use case relies on another use case to be completed. As depicted in Figure 2.1, Process Payment includes Record Transaction, meaning processing a payment always involves recording the transaction.

3. **Extend**: This relationship shows optional functionality that extends the base use case. As depicted in Figure 2.1, Order Book is extended by Assist with ISBN Search, meaning that searching for the ISBN is an optional, additional feature when ordering.

4. **Generalisation**: This relationship shows a hierarchy where one actor or use case is a specialized version of another. As depicted in Figure 2.1, In-store Customer and Ordering Customer are specialized types of Customer, each with slightly different behaviours.

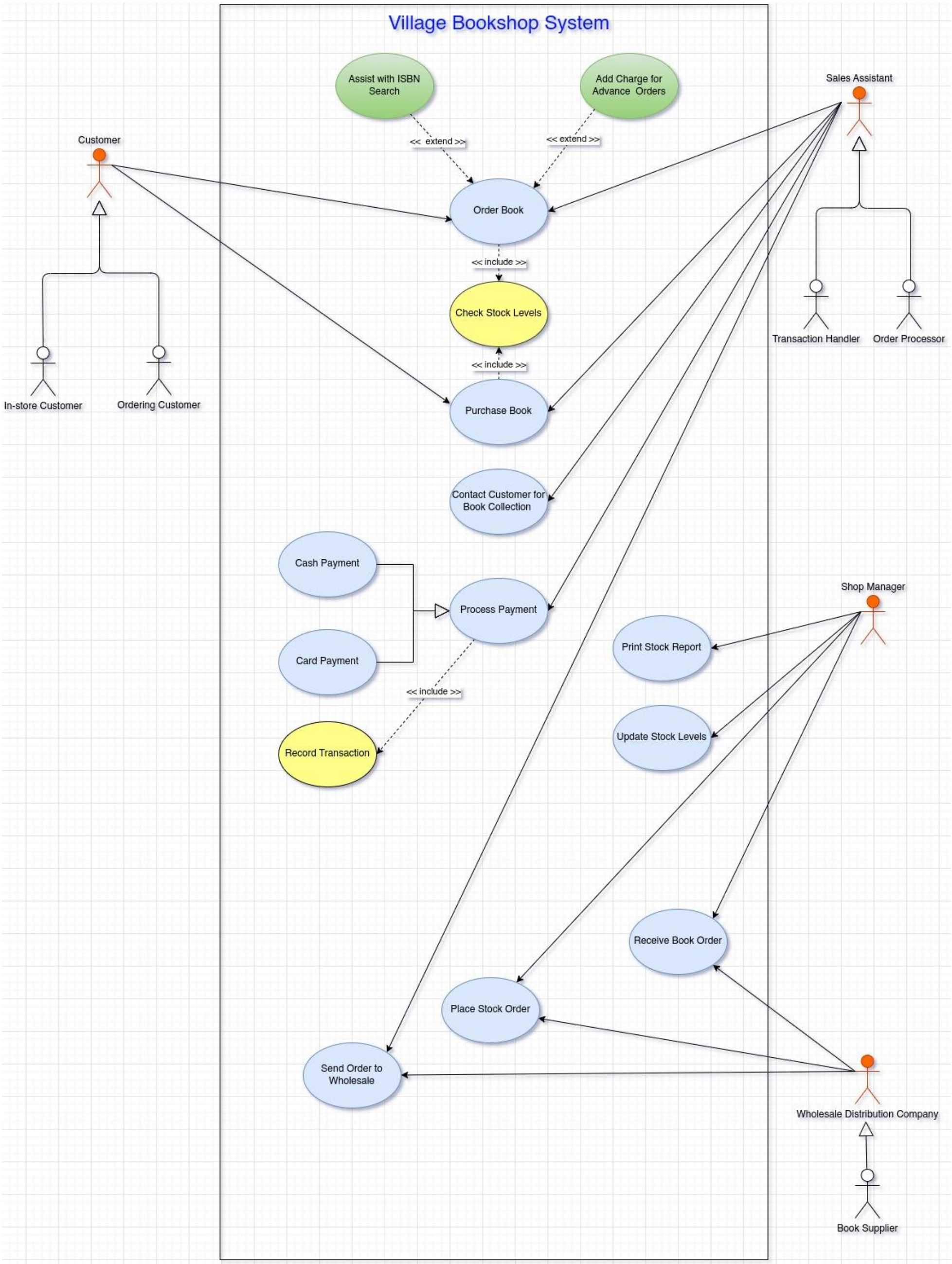


Figure 2.1: Use case diagram for Village Bookshop System

Reference

1. Boehm, B. and Turner, R. (2004). *Balancing agility and discipline: a guide for the perplexed*. Boston, Mass.: Addison-Wesley.

2. Boehm, B.W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), pp.61–72. doi:https://doi.org/10.1109/2.59.

3. Davis, A., Dieste, O., Hickey, A., Juristo, N. and Moreno, A.M. (2006). Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. *14th IEEE International Requirements Engineering Conference (RE'06)*. doi:https://doi.org/10.1109/re.2006.17.

4. Karl Eugene Wiegers and Beatty, J. (2013). *Software requirements*. Redmond, Washington: Microsoft Press.

5. Leveson, N. (2012). *Engineering a safer world: systems thinking applied to safety*. Cambridge, Massachusetts: Mit Press, [Piscataqay, New Jersey.

6. Pohl, K. (2010). *Requirements engineering fundamentals, principles, and techniques*. [online] Berlin; Heidelberg: Springer, Cop. Available at: https://dl.acm.org/citation.cfm?id=1869735.

7. Pressman, R.S. and Maxim, B.R. (2019). *Software engineering: a practitioner's approach*. New York, Ny: Mcgraw-Hill Education.

8. Robertson, S. and Robertson, J. (2012). *Mastering the requirements process*. Upper Saddle River, N.J: Addison-Wesley.

9. Shari Lawrence Pfleeger and Atlee, J.M. (2010). *Software engineering: theory and practice*. Upper Saddle River N.J.: Prentice Hall.

10. Sommerville, I. (2016). *Software engineering*. 10th ed. Harlow: Pearson Education.