**Functional Requirements:**

1. **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.
2. **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.
3. **Veterinary Notification**: Technology must automatically warn the veterinary team (via a beep or notification) whenever any animal sustains an injury.

**Non-Functional Requirements:**

1. **Performance**: The system must calculate the location of each animal within 0.1 seconds to provide prompt monitoring and reaction.
2. **Usability**: The system's interface must be menu-driven to provide user-friendliness for personnel in the control centre.
3. **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.

The Incremental Model is the best appropriate software development life cycle for this project. Here are the reasons, along with the associated steps:

**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.

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.

Flexibility: It permits modifications throughout the development process. Improvements may be implemented based on feedback from preliminary increments prior to the complete system deployment.

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.

**Phases Involved in the Incremental Model**

1. **Requirements Analysis**

* Comprehend and record all functional and non-functional needs of the system.
* Segment the requirements into distinct modules or components that may be built progressively.

2. **Design**

* For each increment, build the architectural design and detailed design of the individual modules under development. Commence with the design of the animal movement tracking and location computation module first.

3. **Implementation (Initial Increment)**

* Establish the fundamental functionality, including the reception of signals from transmitters, computation of animal locations, and their visualization.
* Rigorously evaluate this increase for precision and efficiency.

4. **Testing**

* Following the implementation of each increment, do thorough testing to verify compliance with the requirements.
* Given that this system oversees hazardous fauna, it is important to evaluate each component to guarantee the absence of malfunctions, such as fence breaches or alarm activations.

**Delivery of Initial Increment:**

* Upon the successful implementation of the system's foundational functionality (animal tracking, fence monitoring), present this to the customer or stakeholders to solicit feedback.
* Deliver incomplete but operational solutions to address urgent requirements while more enhancements are being created.

**Reiterate Procedures for Following Increments**

* Incorporate supplementary functions such as the veterinarian warning system, the personnel management menu, and the alarm system for fence breaches in succeeding phases.
* Evaluate, distribute, and collect feedback for each increment.

**Integration and Final Testing**

* Upon the completion and individual testing of all increments, amalgamate them into the final system.
* Conduct thorough system-level testing to verify the integration of all components, with particular emphasis on evaluating the fail-safes and alarms, since they are essential for a system handling hazardous animals.

**Deployment and Maintenance**

* Upon the complete integration of the system, implement it for utilisation inside the park.
* Persist in maintaining and updating the system in accordance with operational feedback or required corrections.

**Rationale for Selecting the Incremental Model**

• **Mitigated Risk**

Incremental system development mitigates the possibility of identifying significant difficulties late in the development phase, which is crucial for safety-critical systems such as monitoring hazardous animals.

• **Prompt Provision of Essential Features**

The Incremental Model facilitates the early delivery of system components, such as basic animal monitoring, so assuring that essential functions are operational throughout the ongoing development of the system.

• **Adaptability for Modifications**

The model permits alterations in consecutive increments informed by feedback, which is advantageous for a system necessitating comprehensive evaluation. Evaluation may need modifications after the observation of real-time outcomes.

• **Temporal Organisation**

Given the park's tight timeline for opening, the delivery of functional increments enables some components of the system to be operational while further elements are still under development.

In summary, the Incremental Model provides adaptability, mitigates risk, and facilitates ongoing development, testing, and refining of the system, which are vital for a project that prioritises safety and criticality.

**Overview of the Incompatibility of Waterfall and Reuse-Oriented Models:**

1. **Waterfall Model:**

* The inflexible framework of the Waterfall model makes it difficult and expensive to rectify any modifications or faults that arise during the creation of sophisticated safety-critical components. This strategy lacks flexibility, and testing occurs just at the conclusion, posing excessive risk for such a significant undertaking.
* Delayed integration: The animal monitoring system must operate flawlessly, since complications during late-stage integration may result in delays that threaten the opening date.
* Inadequate adaptability to shifting needs: Should requirements change during development (for example, due to feedback from preliminary testing), the Waterfall Model would struggle to integrate these modifications effectively.

1. **Model Focused on Reusability:**

* Reliance on existing components: This paradigm is significantly dependent on the availability of reusable components, which may be lacking for such a specialised system (e.g., monitoring hazardous wildlife). Modifying reusable components may need considerable work, undermining the time and cost advantages of reuse.
* Integration challenges: Utilising components from diverse sources may result in integration difficulties. In a system necessitating real-time monitoring and safety protocols, these issues may result in significant delays.
* Restricted adaptability: Although this approach may save time and expenses if suitable components are identified, it lacks the necessary flexibility to accommodate the developing and bespoke demands of the park's monitoring system.

**Reasons the Incremental Model is Optimal:**

The Incremental Model offers an optimal equilibrium of adaptability, risk mitigation, and prompt delivery. The iterative methodology facilitates the prompt delivery of essential features, ongoing testing, and flexibility, making it the optimal selection for this project.

Section C

During the requirements elicitation and analysis for the computer monitoring system designed for the park housing hazardous animals, many challenges may emerge. Presented here are six possible challenges:

**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.

**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.

**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.

**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.

**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.

**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.

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.

Section D

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.

**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. An exhaustive SRS mitigates the danger of overlooking features and guarantees that the development team comprehends the whole scope of the project.

**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 X within Y minutes”).

**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.

**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.

**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.

**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. Traceability guarantees that no need is neglected during development and testing, ensuring comprehensive coverage of functionality by 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.

**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.

**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.

**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.

**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.

**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. This indicates that forthcoming needs may be integrated without substantial system redesign or overhaul.

**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.

# **Reference list**

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

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.

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.

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

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

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

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

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

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

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